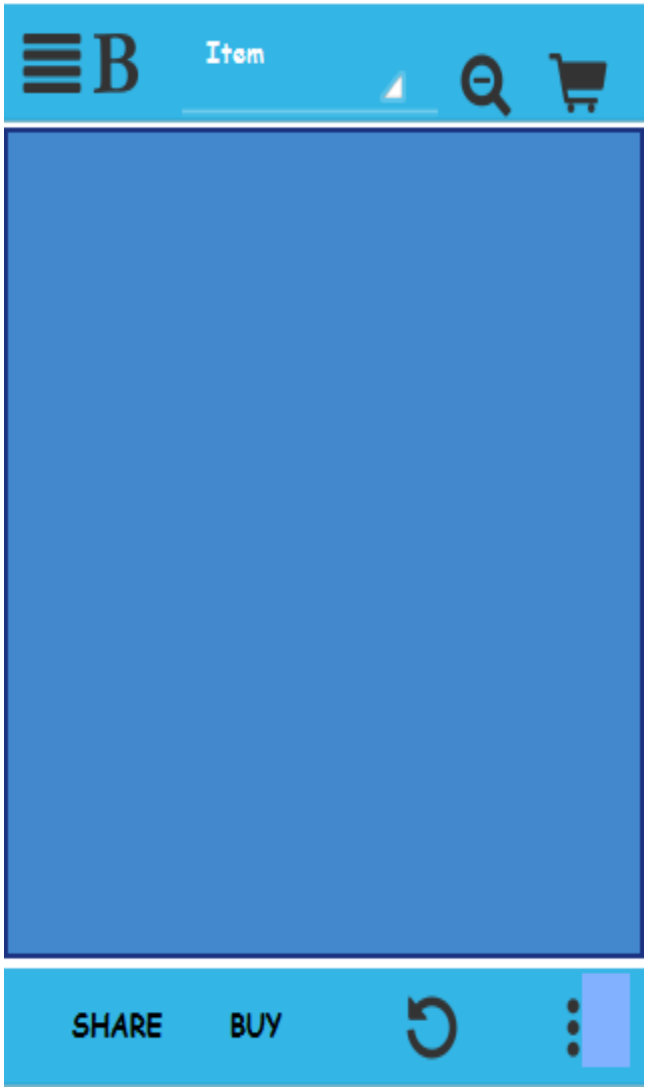
Programowanie urządzeń mobilnych

Wykorzystanie menu i ActionBar

Tłumaczenie i adaptacja materiałów: dr Tomasz Xięski.
Na podstawie prezentacji udostępnionych przez Victor Matos, Cleveland State University.

Tworzenie GUI

Wykorzystanie ActionBar



Nagłówek
ActionBar zajmuje miejsce na górze ekranu. Wyjątkiem jest sytuacja, w której brakuje miejsca do umieszczenia wszystkich elementów. Wówczas zawartość rozdzielana jest na dwa paski: jeden u góry, jeden na dole.

Główna część aplikacji

Stopka:
Paski narzędziowe również mogą zostać umieszczone na dole ekranu.

Tworzenie GUI

Wykorzystanie menu

- Menu jest elementem często wykorzystywanym podczas tworzenia interfejsu użytkownika w Androidzie.
- Dobre menu zapewnia prosty i jednolity interfejs dodający więcej możliwości do aplikacji, bez zajmowania znacząco większej przestrzeni roboczej.
- Menus może być ściśle powiązane z aktualnie prezentowanym widokiem. Każdy ekran może mieć swój własny zestaw opcji.
- Menu może być powiązane z dowolną liczbą widżetów umieszczonych na danej aktywności.
- **Aktualne rekomendacje w sprawie interfejsu użytkownika łączą wykorzystanie menu wraz z paskiem w nagłówku (ActionBar).**

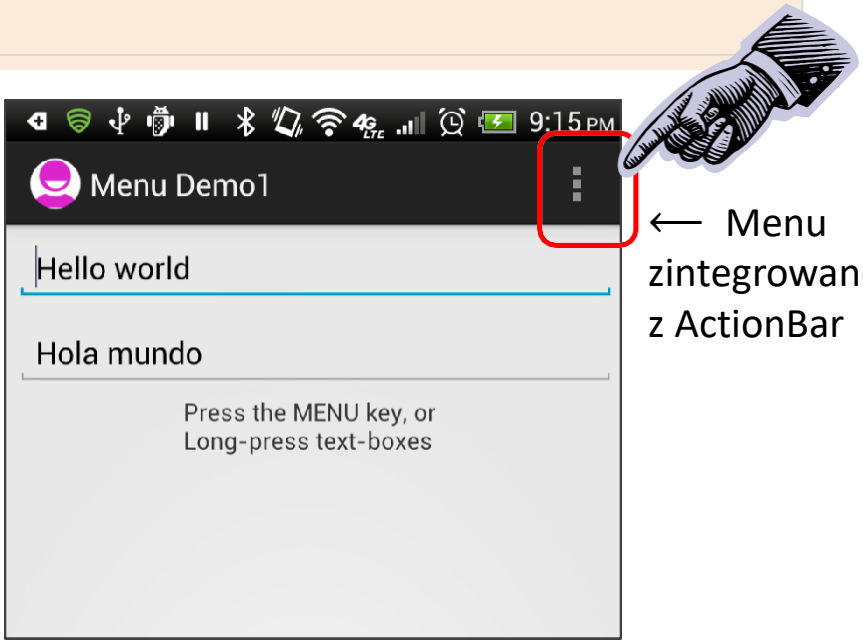
Wykorzystanie menu

Typy Menu

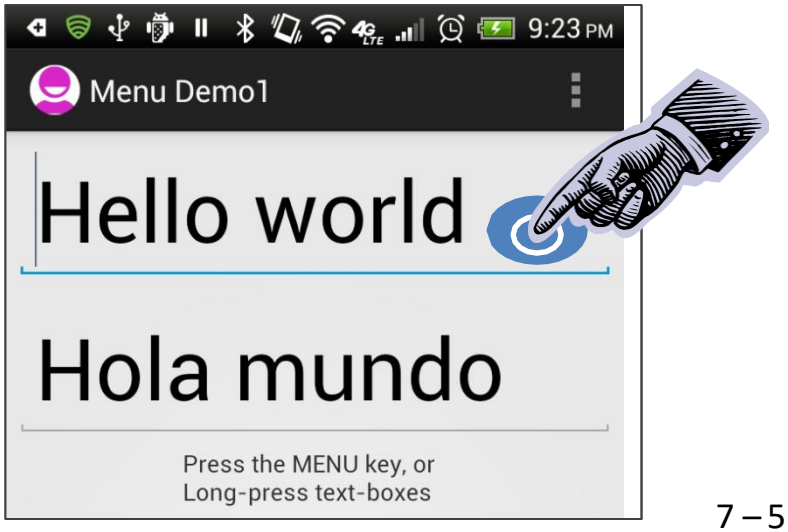
Android wspiera dwa typy menu: **Menu główne** oraz **Menu kontekstowe**.

1. Globalne **menu główne** (*options menu*) jest uaktywniane poprzez przyciśnięcie fizycznego bądź wirtualnego przycisku **Menu**. *Dla każdego ekranu dostępne jest tylko jedno menu główne*.

2. **Menu kontekstowe** (*context menu*) uaktywniane jest poprzez *dłuższe przytrzymanie* palca na widżecie powiązanym z danym menu. Każdy widżet może mieć swoje menu kontekstowe.



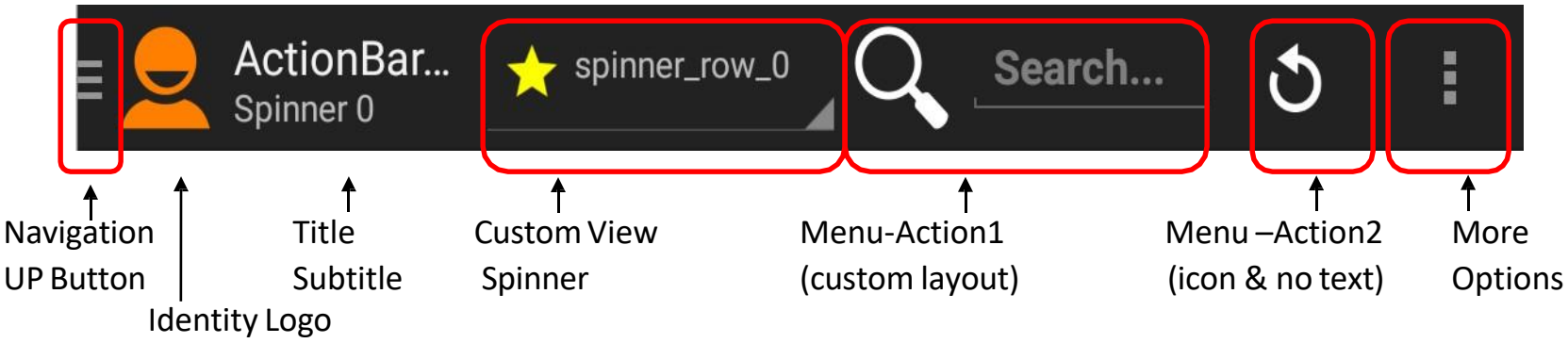
⟵ Menu zintegrowane z ActionBar



Wykorzystanie ActionBar

- Komponent **ActionBar** został wprowadzony wraz z SDK 3.0 i jest obecny w wielu aplikacjach mobilnych. Jest przedstawiony jako pasek narzędziowy znajdujący się na górze każdego ekranu i zwykle posiada stałe miejsce dla całej aplikacji.

- Zwykle składa się z następujących elementów:
  - Nawigacji – Przycisk wyżej (symbolizowany jako ikona strzałki)
  - Logo,
  - Tytułu i podtytułu,
  - Opcjonalnych widżetów użytkownika,
  - Kafelek (przycisków posiadających tekst i/lub grafikę),
  - Przycisku menu głównego
  - Zakładek (status *deprecated od SDK4.4*)



Wykorzystanie ActionBar

ActionBar jest bardzo istotnym elementem ponieważ:

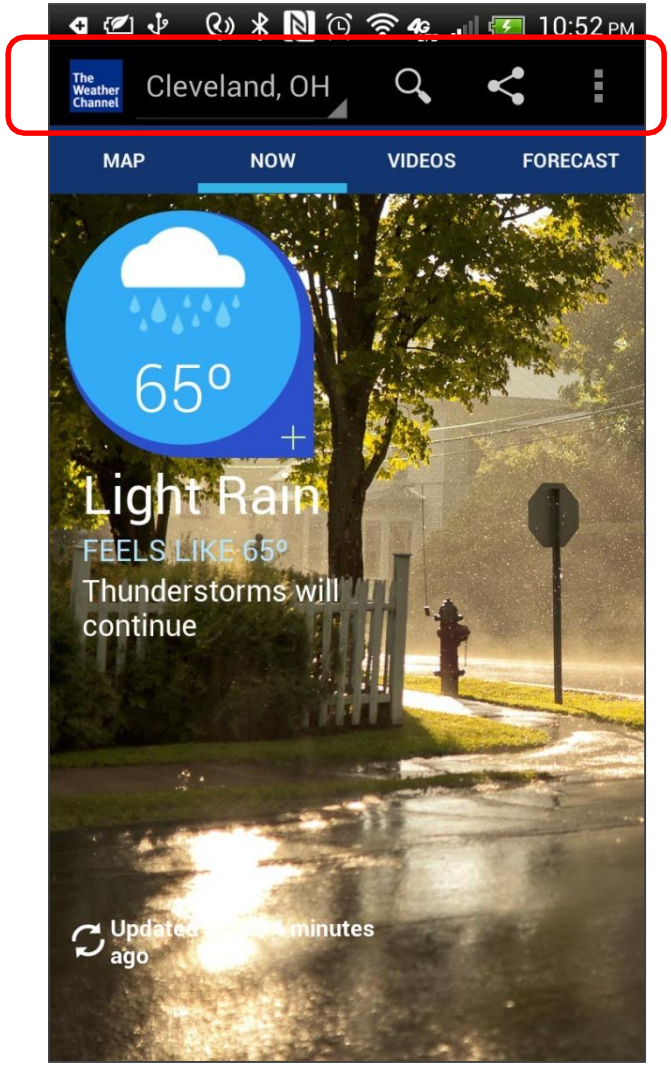
- Działa jako **skondensowany panel przełączający** pozwalający uzyskać łatwy dostęp do głównych funkcjonalności aplikacji.
- Zawiera przycisk wywołujący menu główne **‘ ⋮ ’**, którego funkcjonalność może się dostosować do każdego ekranu.
- Zwykle pojawia się wyłącznie na górze ekranu, budując wrażenie **spójnego wyglądu** gdzie "*wszystkie najważniejsze przyciski znajdują się w jednym miejscu*”.
- Może być wykorzystywany do implementacji **różnych typów nawigacji:**
  - (a) jako główny przycisk wyświetlający dodatkową listę opcji nad aktualnie wyświetlanym widokiem,
  - (b) jako zestaw zakładek do najważniejszych części aplikacji,
  - jako **przycisk wyżej**, który może być wykorzystany do cofania się o jedną pozycję w hierarchii stosu aktywności bądź do dowolnego innego miejsca hierarchii.

Przykład aplikacji wykorzystujących ActionBar

	Nazwa aplikacji	Liczba ściągnięć	Ocena	Kategoria
1	Google Search	1B	4.4	Tools
2	Gmail	1B	4.3	Communication
3	Google Maps	1B	4.3	Travel & Local
4	YouTube	1B	4.1	Media & Video
5	Facebook	1B	4.0	Social
6	WhatsApp	500M	4.4	Communications
7	Instagram	100M	4.5	Social
8	Pandora	100M	4.4	Music & Audio
9	Netflix	100M	4.4	Entertainment
10	Adobe Reader	100M	4.3	Productivity
11	Skype	100M	4.1	Communications
12	Twitter	100M	4.1	Social
13	eBay	50M	4.3	Shopping
14	Weather Channel	50M	4.2	Weather
15	Kindle	50M	4.1	Books & References
16	Wikipedia	10M	4.4	Books & References
17	Zillow	10M	4.4	Lifestyle
18	ESPN SportCenter	10M	4.2	Sports
19	BBC News	10M	4.2	News & Magazines
20	Amazon (Tablets)	10M	4.0	Shopping
21	Expedia	10M	4.0	Travel & Local

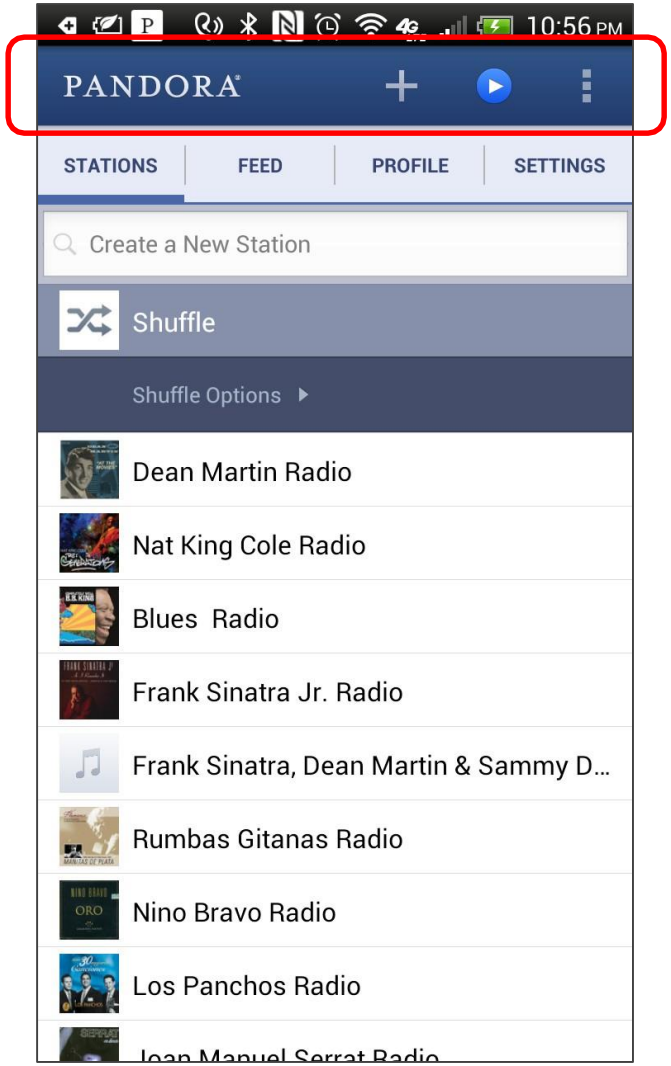
Źródło:
Sklep play. Dostęp: 16.02.2015. Link: https://play.google.com/store?hl=en

Przykład aplikacji wykorzystujących ActionBar



ActionBar

Dwie zupełnie różne aplikacje reprezentujące dość podobny widok i sposób nawigacji.



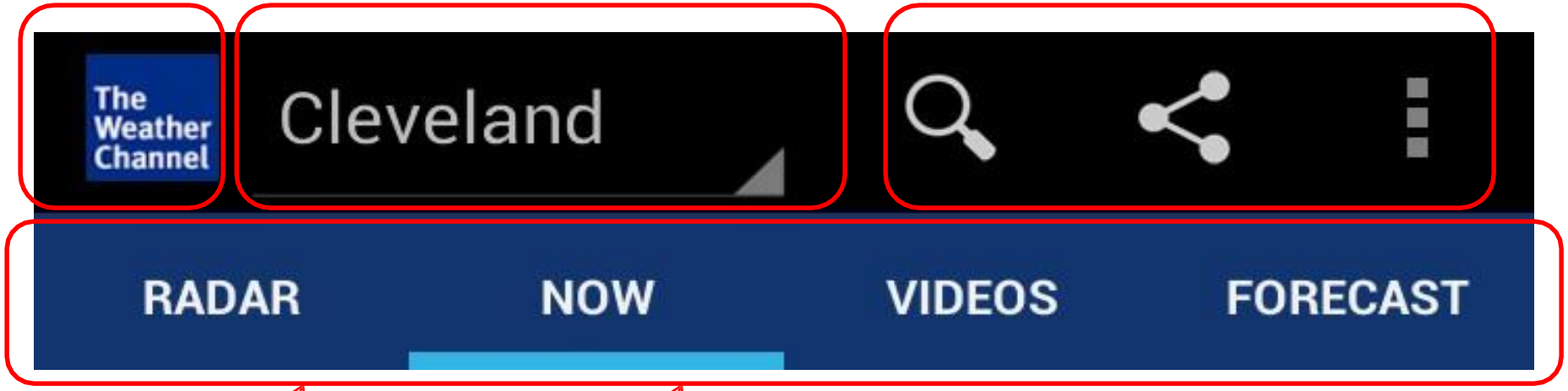
Identyfikacja elementów składowych ActionBar'a

Przykładowa interpretacja użytych komponentów dla aplikacji pogodowej:

Logo

ListView

Menu (Znajdź i Udostępnij) oraz pozostałe



PageViewer

Aktualnie wybrana zakładka

Kontrolka PageViewer stanowi alternatywę dla bezpośredniego użycia ActionBar do grupowania zakładek.

Architectura ActionBar

Klikalne *kafelki* udostępniane przez ActionBar zwykle są definiowane w pliku XML rezydującym w katalogu **res/menu**. Na jego podstawie tworzony jest obiekt odpowiedzialny za wyświetlenie menu głównego.

Standardowo każda dostępna opcja menu jest wizualizowana jako lista tekstowa aktywowana przez kliknięcie wirtualnego przycisku ⋮. Jednakże wybrane pozycje mogą być wyświetlone osobno jako przycisku (z tekstem i/lub grafiką) będące częścią komponentu ActionBar.

Lista opcji menu jest zazwyczaj stała przez cały cykl życia aplikacji, jednakże może być w sposób dynamiczny wyłączana czy zmieniana.

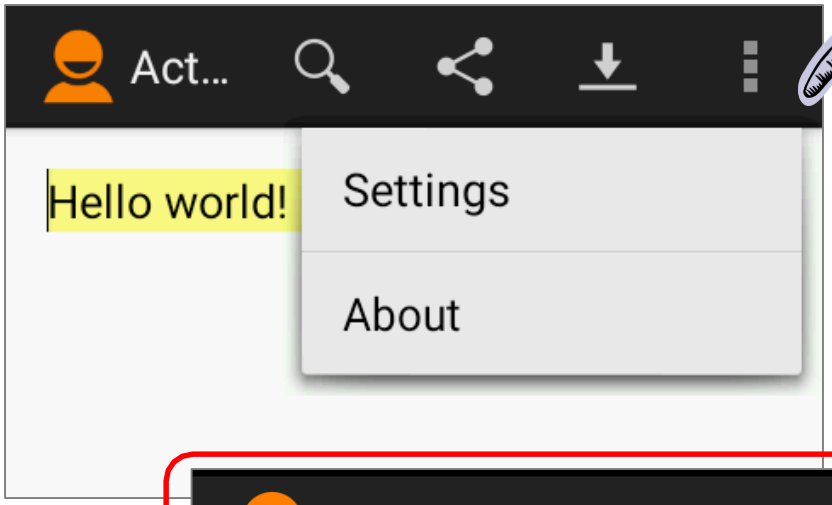
Dwie metody, których zadaniem jest manipulowanie zestawem opcji dostępnych w ramach komponentu ActionBar to:

onCreateOptionsMenu(…)
Tworzy menu na podstawie specyfikacji XML

onOptionsItemSelected(…)
Reaguje na zdarzenie kliknięcia poszczególnej opcji

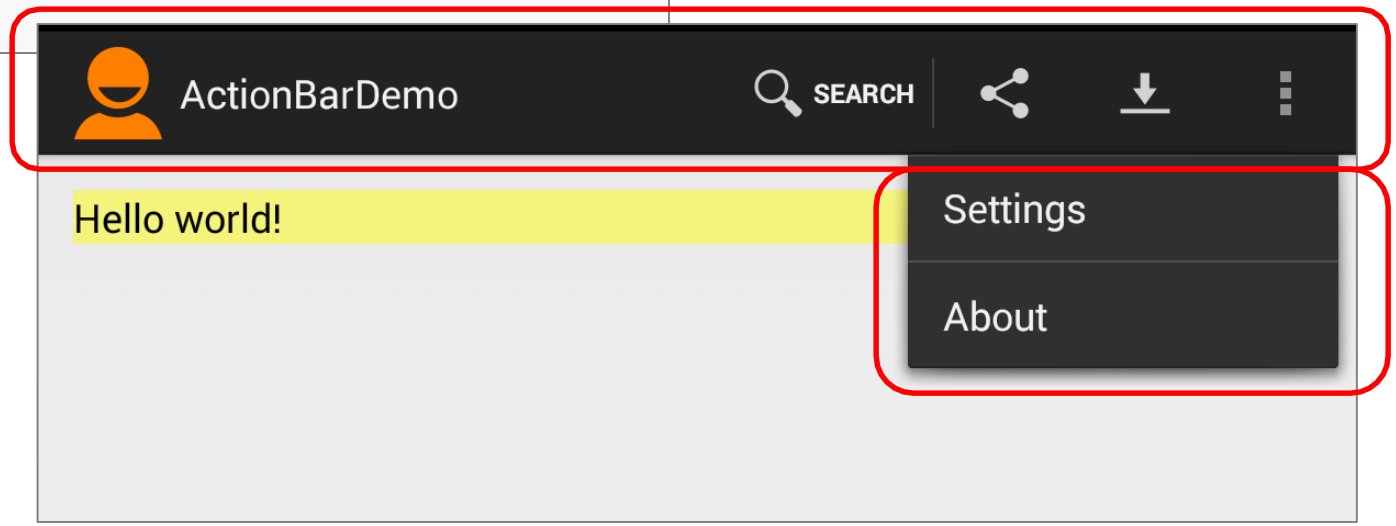
Przykład 1 – Proste wykorzystanie ActionBar

Aplikacja bazuje na szablonie typu Basic Activity. Należy zmodyfikować plik **res/menu/main.xml** by wygenerować odpowiednie opcje menu. Zdjęcia pochodzą z tabletu oraz telefonu komórkowego.



Ostateczna decyzja jak rozłożyć ActionBar jest podejmowana przez środowisko uruchomieniowe Androida bazując na tym ile przestrzeni roboczej jest dostępnej.

→ ActionBar

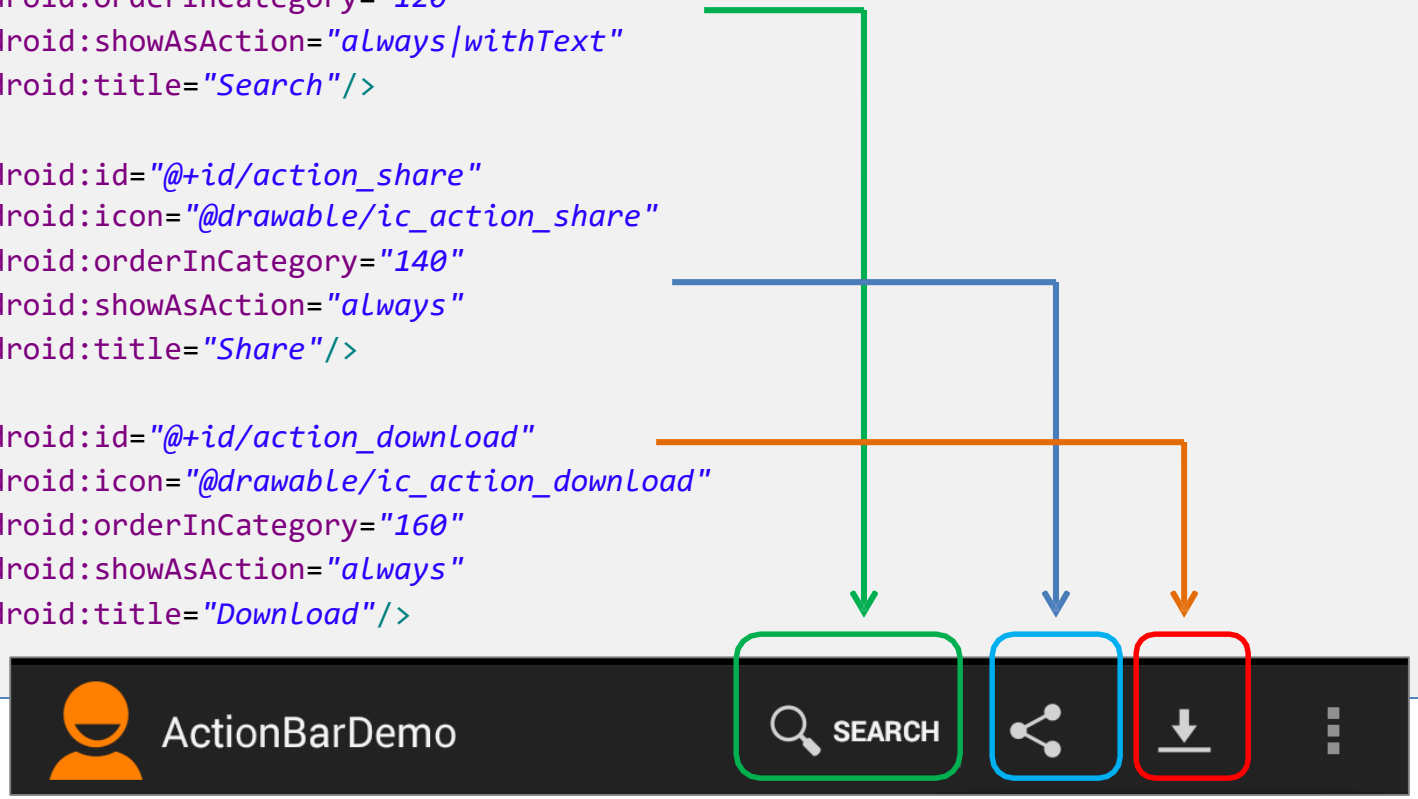


← Menu

Przykład 1 – Proste wykorzystanie ActionBar

UKŁAD res/menu/main.xml.

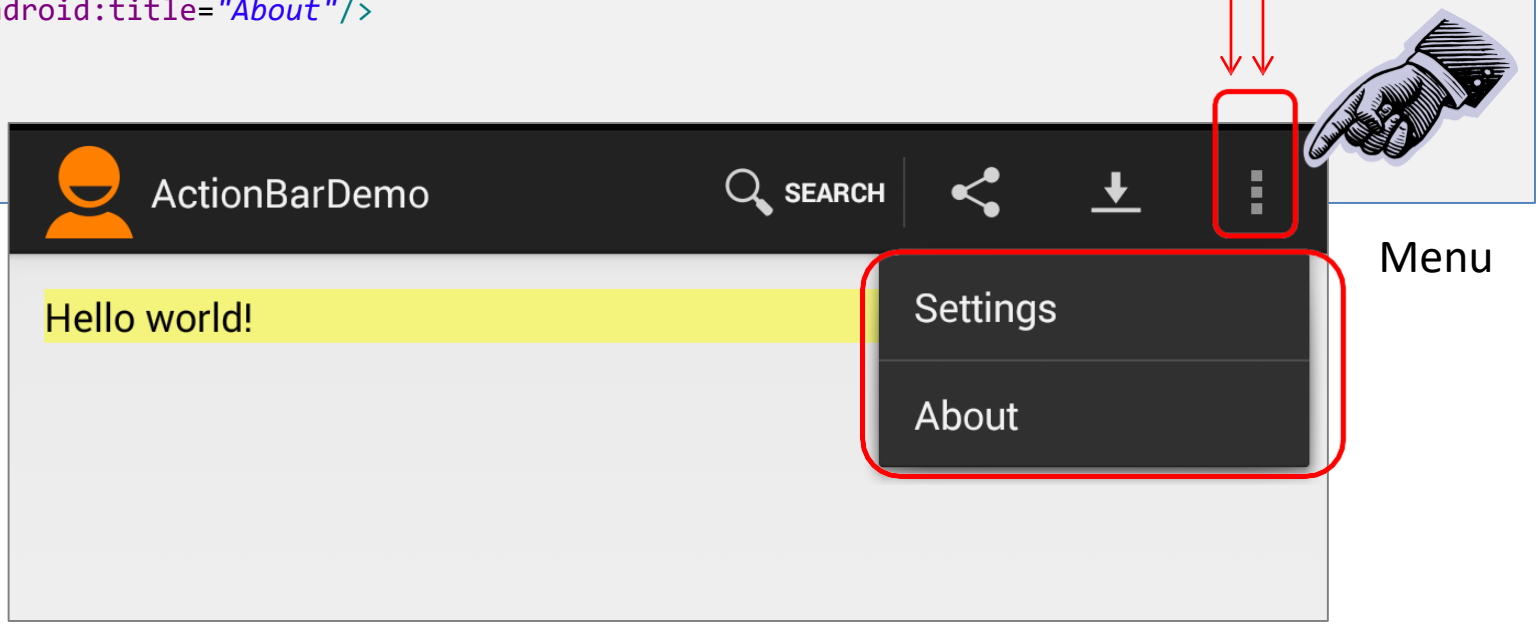
```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   tools:context="csu.matos.MainActivity" >

   <item
       android:id="@+id/action_search"
       android:icon="@drawable/ic_action_search"
       android:orderInCategory="120"
       android:showAsAction="always|withText"
       android:title="Search"/>
   <item
       android:id="@+id/action_share"
       android:icon="@drawable/ic_action_share"
       android:orderInCategory="140"
       android:showAsAction="always"
       android:title="Share"/>
   <item
       android:id="@+id/action_download"
       android:icon="@drawable/ic_action_download"
       android:orderInCategory="160"
       android:showAsAction="always"
       android:title="Download"/>
```



Przykład 1 – Proste wykorzystanie ActionBar

UKŁAD res/menu/main.xml.

```
<item
        android:id="@+id/action_settings
        "  android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about
        "
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>

</menu>
```

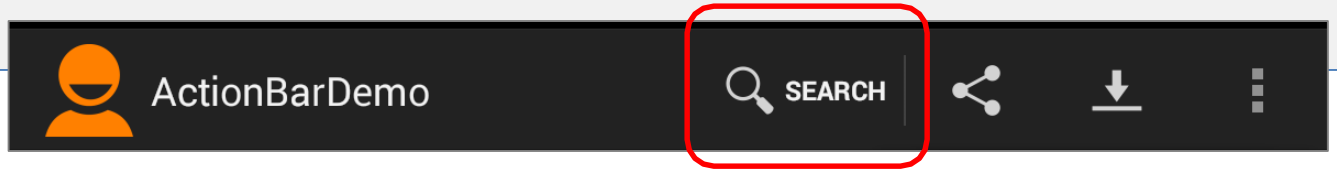


Menu

Przykład 1 – Proste wykorzystanie ActionBar

Każda pozycja menu <**item**> reprezentuje poszczególny kafelek:

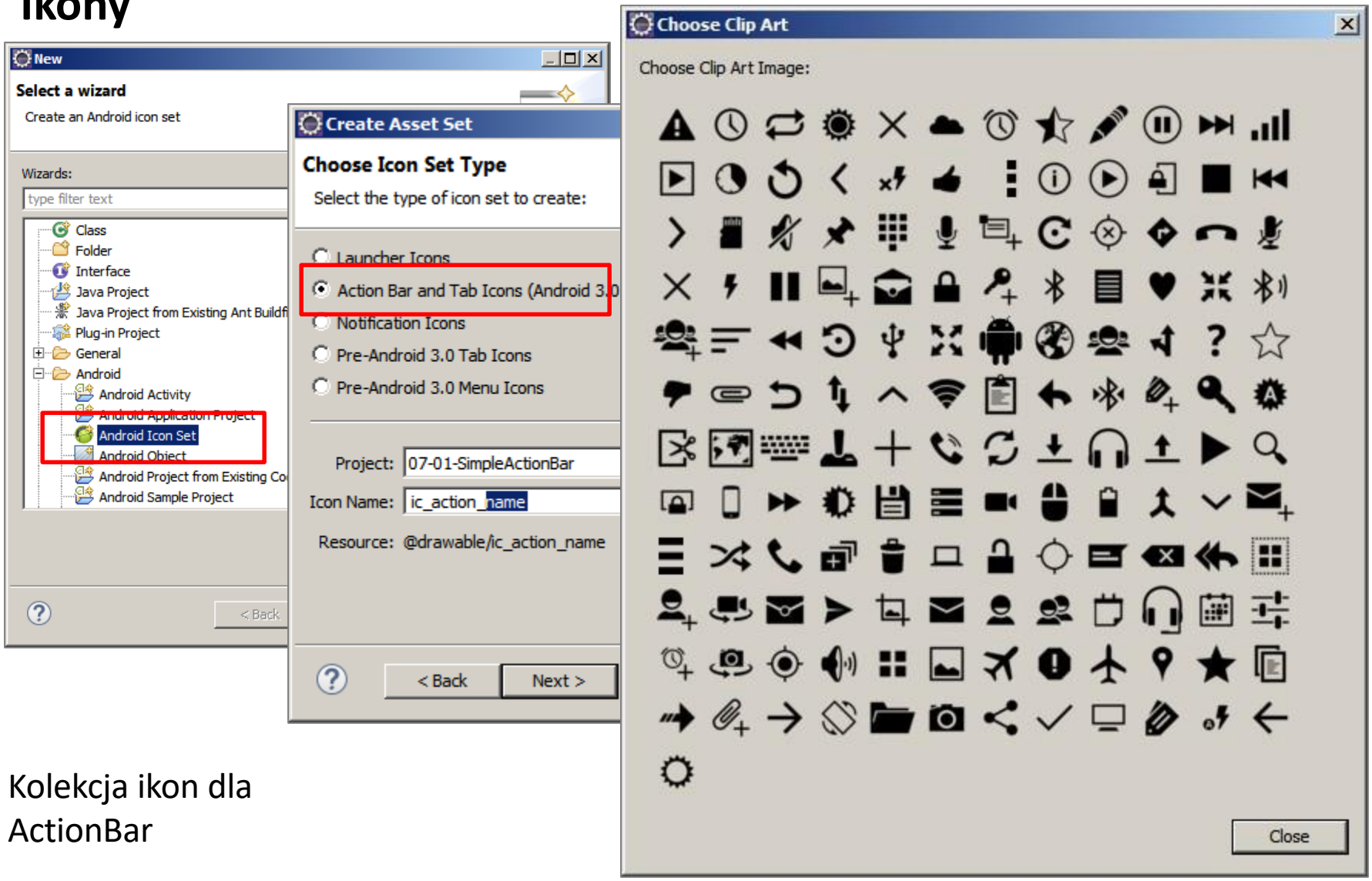
```
<item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
```



android:id	Identyfikator akcji **ID** (@+id/action_search), wymagany by widzieć która pozycja została wybrana
android:icon	Opcjonalna ikona powiązana z daną pozycją. Więcej informacji: **http://developer.android.com/design/style/iconography.html**
:orderInCategory	Relatywna pozycja tytułu na ActionBar (100, 120, 140, ...)
:showAsAction	Za pomocą tych klauzul możliwa jest kontrola położenia kafelek: “never”, “ifRoom”, “always”, “withText”, and “collapseActionView”.
:title	Opcjonalny tekst (‘SEARCH’) opisujący kafelek

Przykład 1 – Proste wykorzystanie ActionBar

Ikony



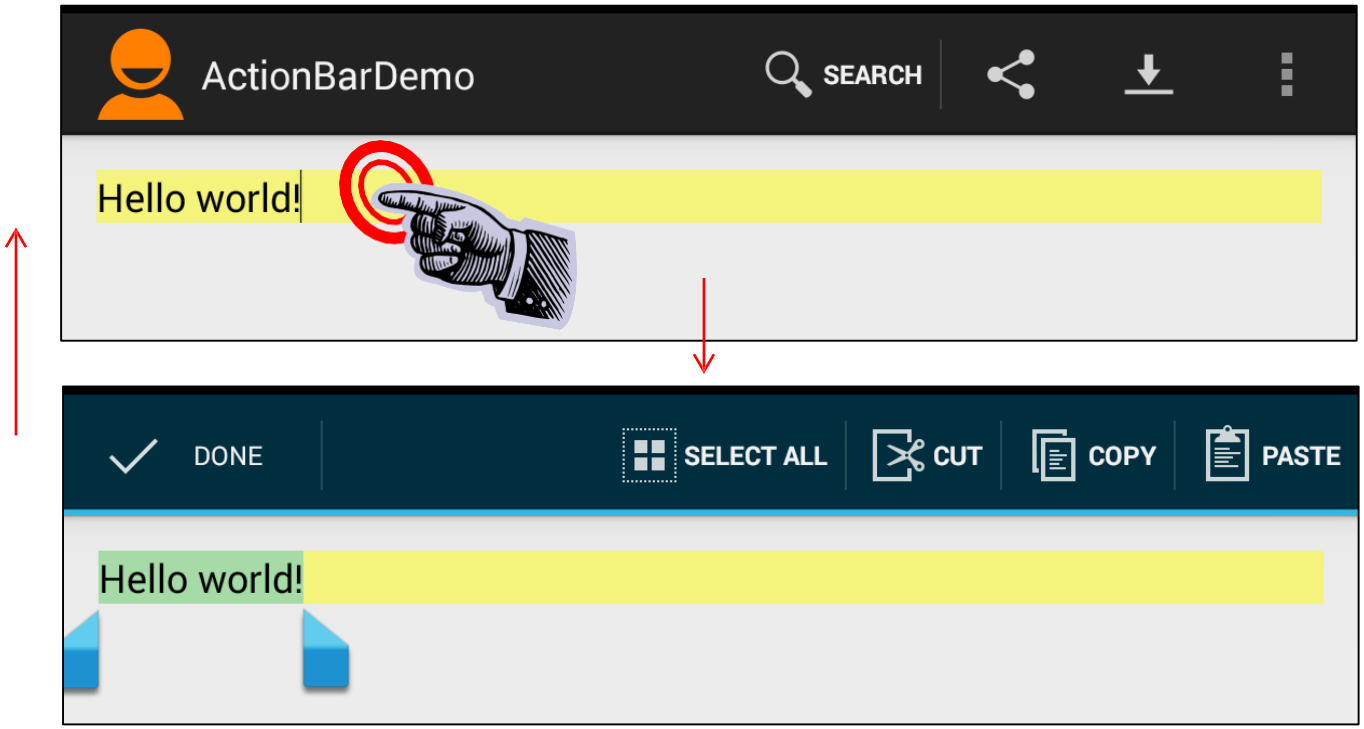
Kolekcja ikon dla ActionBar

Przykład 1 – Proste wykorzystanie ActionBar

OBSERWACJA: Zmień plik **activity_main.xml** – zmień typ komponentu "HelloWorld" z **TextView** na **EditText**. Uruchom aplikację i przytrzymaj bądź kliknij dwukrotnie na polu tekstowym. ActionBar zmienia się w tzw. **Contextual ActionBar** (CAB) podobnie jak na zaprezentowanym obrazku.

ActionBar zależny od kontekstu wprowadza szereg nowych możliwości interakcji z użytkownikiem.

Sposób tworzenia tego typu komponentów zostanie przedstawiony później.



⟵ **ActionBar**

⟵ **Contextual ActionBar**

Przykład 1 – Proste wykorzystanie ActionBar

ActivityMain.java

```
public class MainActivity extends Activity {
    EditText txtMsg;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
1       txtMsg = (EditText)findViewById(R.id.txtMsg);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; add items to the action bar
        getMenuInflater().inflate(R.menu.main, menu);
2       return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // user clicked a menu-item from ActionBar
        int id = item.getItemId();

3       if (id == R.id.action_search) {
            txtMsg.setText("Search...");
            // perform SEARCH operations...
            return true;
        }
```

Przykład 1 – Proste wykorzystanie ActionBar

ActivityMain.java

3

```
        else if (id == R.id.action_share) {
            txtMsg.setText("Share...");
            // perform SHARE operations...
            return true;
        }
        else if (id == R.id.action_download) {
            txtMsg.setText("Download...");
            // perform DOWNLOAD operations...
            return true;
        }
        else if (id == R.id.action_about) {
            txtMsg.setText("About...");
            // perform ABOUT operations...
            return true;
        }
        else if (id == R.id.action_settings) {
            txtMsg.setText("Settings...");
            // perform SETTING operations...
            return true;
        }

        return false;
    }
}
```

Przykład 1 – Proste wykorzystanie ActionBar

Komentarz: ActivityMain.java

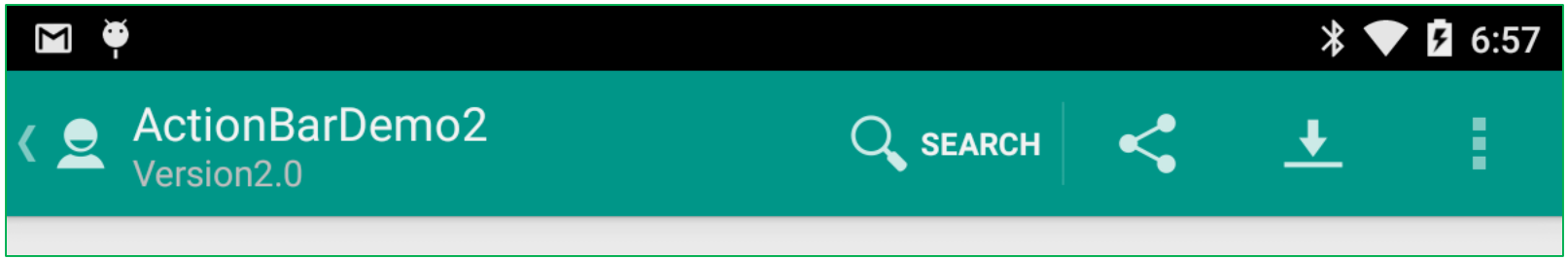
1. Metoda `onCreateOptionsMenu()` jest wywoływana by przygotować menu główne. Plik xml **res/memu/main.xml** zawierający specyfikację ActionBar, na podstawie której tworzone są obiekty (z wykorzystaniem MenuInflater). Niektóre elementy prezentowane będą w postaci kafelek na ActionBarze (posiadających Ikonę/Tekst) natomiast pozostałe opcje będą wyświetlane po wciśnięciu przycisku menu.

2. Gdy użytkownik kliknie na określone miejsce ActionBar, jego identyfikator jest przekazywany do metody `onOptionsItemSelected()`. Tam następuje obsługa danego zdarzenia. Następnie zwracana jest wartość **true** by zasygnalizować, że zdarzenie zostało w pełni obsłużone.

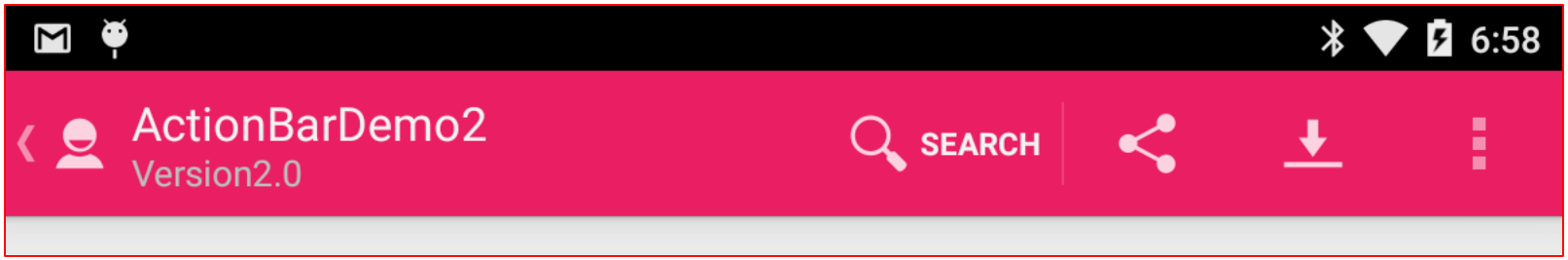
Przykład 2 – Modyfikacja ActionBar

Przykład jest modyfikacją poprzedniego. ActionBar został zmodyfikowany uwzględniając inny kolor tła, logo oraz strzałki wyżej. Kolory wybrano z: http://www.google.com/design/spec/style/color.html#color-color-palette

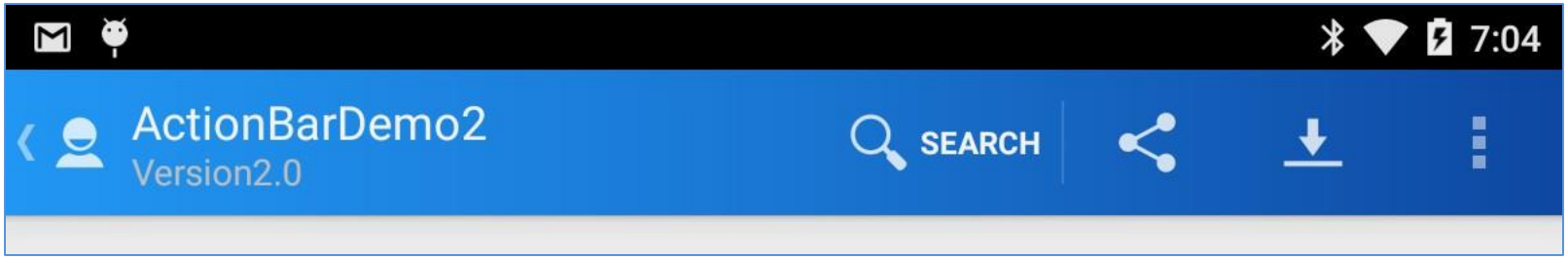
1



2



3



Przykład 2 – Modyfikacja ActionBar

ActionBar zmieniony został programowo jako:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    txtMsg = (EditText)findViewById(R.id.txtMsg);

    // setup ActionBar
    actionBar = getActionBar();

    actionBar.setTitle("ActionBarDemo2");
    actionBar.setSubtitle("Version2.0");
    actionBar.setLogo(R.drawable.ic_action_logo);

    // choose one type of background
    actionBar.setBackgroundDrawable(getResources().getDrawable(R.drawable.mybackground0));
    actionBar.setBackgroundDrawable(getResources().getDrawable(R.drawable.mybackground1));
    actionBar.setBackgroundDrawable(getResources().getDrawable(R.drawable.mybackground2));

    actionBar.setDisplayShowCustomEnabled(true);     // allow custom views to be shown
    actionBar.setDisplayHomeAsUpEnabled(true);       // show 'UP' affordance < button
    actionBar.setDisplayShowHomeEnabled(true);       // allow app icon – logo to be shown
    actionBar.setHomeButtonEnabled(true);            // needed for API14 or greater

 }
```

1 → `actionBar = getActionBar();`

2 → `actionBar.setSubtitle("Version2.0");`

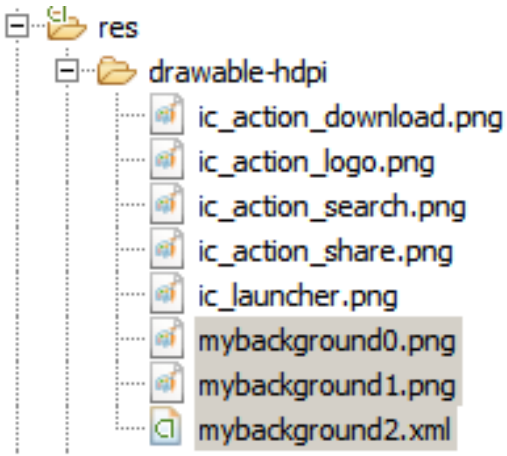
3 → `actionBar.setBackgroundDrawable(getResources().getDrawable(R.drawable.mybackground0));`

4 → `actionBar.setDisplayShowCustomEnabled(true);`

Przykład 2 – Modyfikacja ActionBar

Komentarz:

1. Wywołanie metody `getActionBar()` zwraca referencję do komponentu ActionBar. Poprzez referencję mamy dostęp do komponentów tam zawartych.
2. W tym przykładzie nowy tytuł i podtytuł jest przypisany do komponentu ActionBar. W przypadku skomplikowanych aplikacji z dużym zestawem ekranów jest to wręcz obowiązkowa czynność by ułatwić użytkownikowi nawigacje.
3. Logo może zostać zmienione poprzez metodę `.setLogo(drawable)`. Podobnie tło ActionBar może zostać zmienione na dowolny obrazek. Zwykle tego typu zasoby znajdują się w katalogu **res/drawable.**



mybackground0.png

mybackground1.png

Przykład 2 – Modyfikacja ActionBar

Komentarz:

3. Można również zdefiniować gradient jako tło dla ActionBar

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle"
    <gradient
        android:angle="0"
        android:centerColor="#1976D2"
        android:centerX="50%"
        android:endColor="#0D47A1"
        android:startColor="#2196F3"
        android:type="linear" />
</shape>
```

4. Można również zdefiniować widoczność niektórych elementów:

```
// set ActionBar options
actionBar.setDisplayShowCustomEnabled(true); // allow custom views to be shown
actionBar.setDisplayHomeAsUpEnabled(true);   // show ‘UP’ affordance < button
actionBar.setDisplayShowHomeEnabled(true);   // allow app icon – logo to be shown
actionBar.setHomeButtonEnabled(true);        // needed for API.14 or greater
```

Przykład 2 – Modyfikacja ActionBar

Komentarz:

4. Zwykle w celu ukrycia lub aktywacji pewnych opcji wykorzystywana jest pojedyncza instrukcja:

```
// set ActionBar options

actionBar.setDisplayOptions(  ActionBar.DISPLAY_SHOW_TITLE
                            | ActionBar.DISPLAY_SHOW_HOME
                            | ActionBar.DISPLAY_HOME_AS_UP
                            | ActionBar.DISPLAY_SHOW_CUSTOM );
```

Dodatkowe zasoby:

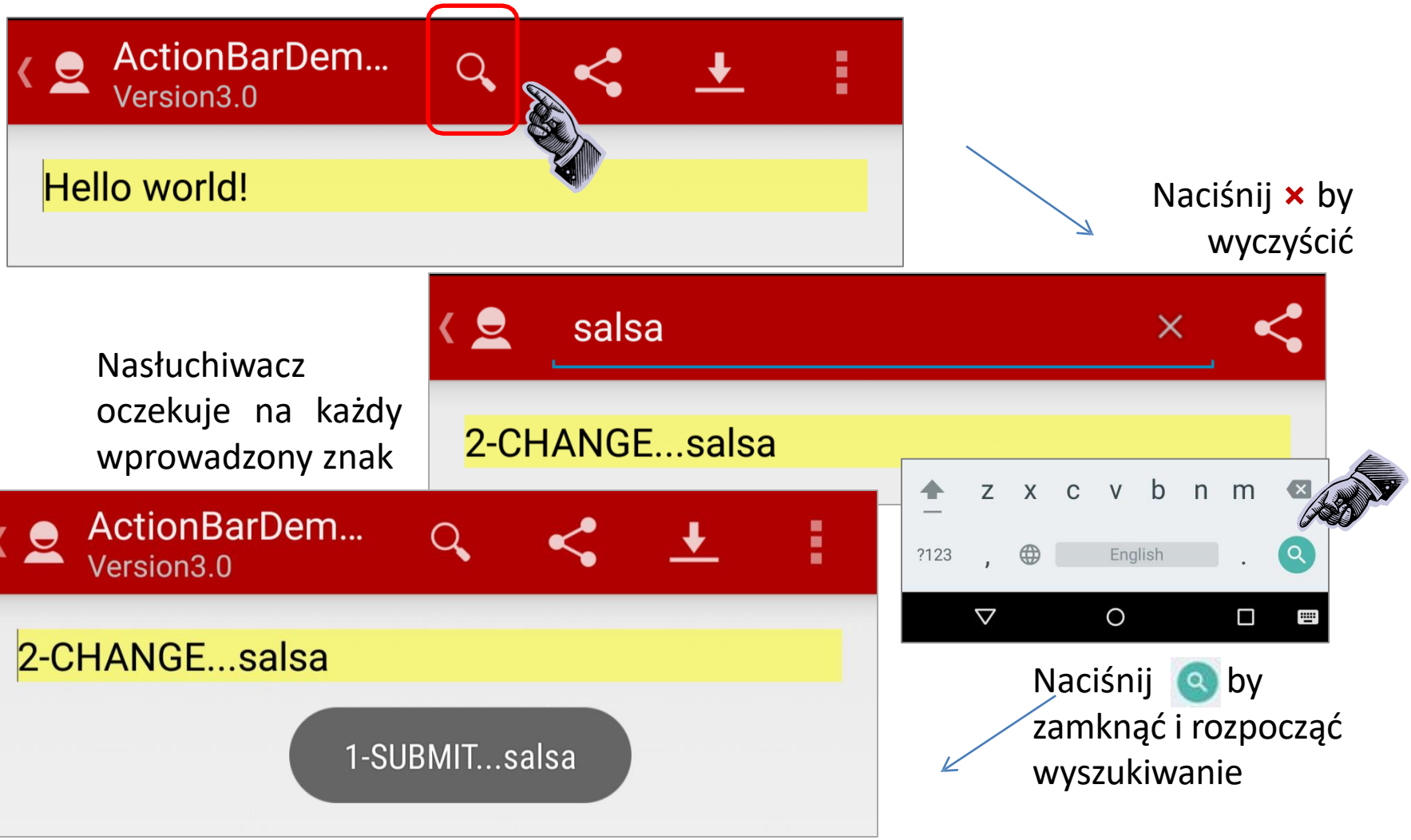
Gradienty http://angrytools.com/gradient/

Kolory http://www.google.com/design/spec/style/color.html#color-color-palette

Grafiki http://developer.android.com/guide/topics/resources/drawable-resource.html

Przykład 3 – Kafelek do wyszukiwania

Jest to wariant Przykładu 1. Opcja wyszukiwania została zaimplementowana przez widżet **SearchView** jako opcja menu. Gdy użytkownik kliknie na opcję wyszukiwania tekst wpisany w polu wyszukiwania jest wykorzystywany do uaktywniania tego procesu.



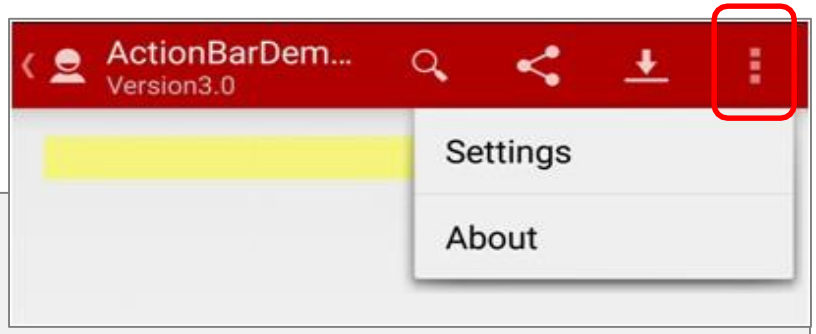
Przykład 3 – Kafelek do wyszukiwania

Pole typu **SearchView** jest zdefiniowane w pliku **XML** podobnie jak w zaprezentowanym przykładzie:


```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="csu.matos.MainActivity" >

    <item
        android:id="@+id/action_search"
        android:actionViewClass="android.widget.SearchView"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
```

1

Przykład 3 – Kafelek do wyszukiwania




```
<item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>

</menu>
```

Komentarz:

1. Element *action_search* nie zawiera klauzuli `:icon` – zamiast wykorzystuje klauzulę `android:actionViewClass="android.widget.SearchView"` by ustawić widok który rozwinięty umożliwia użytkownikowi wprowadzenie kwerendy oraz rozpoczęcie procesu wyszukiwania.

Przykład 3 – Kafelek do wyszukiwania

Kod wzorowany na przykładzie 2.

```
public class MainActivity extends Activity {
   EditText txtMsg;
   ActionBar actionBar;
   SearchView txtSearchValue;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      txtMsg = (EditText) findViewById(R.id.txtMsg);

      // setup the ActionBar
      actionBar = getActionBar();
      actionBar.setTitle("ActionBarDemo3");
      actionBar.setSubtitle("Version3.0");
      actionBar.setLogo(R.drawable.ic_action_logo);
      actionBar.setBackgroundDrawable(getResources().getDrawable(
                                       R.drawable.mybackground1));

      actionBar.setDisplayShowCustomEnabled(true);  // allow custom views to be shown
      actionBar.setDisplayHomeAsUpEnabled(true);    // show 'UP' affordance < button
      actionBar.setDisplayShowHomeEnabled(true);    // allow app icon – logo to be shown
      actionBar.setHomeButtonEnabled(true);         // needed for API14 or greater

   }
```

Przykład 3 – Kafelek do wyszukiwania

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   // Inflate the options menu to adds items from menu/main.xml into the ActionBar
   getMenuInflater().inflate(R.menu.main, menu);
   // get access to the collapsible SearchView
1  txtSearchValue = (SearchView) menu.findItem(R.id.action_search)
                                                    .getActionView();
   // set searchView listener (look for text changes, and submit event)
   txtSearchValue.setOnQueryTextListener(new OnQueryTextListener() {

      @Override
      public boolean onQueryTextSubmit(String query) {
2        Toast.makeText(getApplicationContext(), "1-SUBMIT..." + query,
                                 Toast.LENGTH_SHORT).show();
         // recreate the 'original' ActionBar (collapse the SearchBox)
         invalidateOptionsMenu();
         // clear searchView text
         txtSearchValue.setQuery("", false);
         return false;
      }

      @Override
3     public boolean onQueryTextChange(String newText) {
         // accept input one character at the time
         txtMsg.append("\n2-CHANGE..." + newText);
         return false;
      }
   });
   return true;
}
```

Przykład 3 – Kafelek do wyszukiwania

```
   @Override
   public boolean onOptionsItemSelected(MenuItem item) {
      // Handle ActionBar item clicks here.
      // NOTE: Observe that SEARCH menuItem is NOT processed in this
      // method (it has its own listener set by onCreateOptionsMenu)

      int id = item.getItemId();

      if (id == android.R.id.home) {
         txtMsg.setText("Home...");
         return true;
      } else if (id == R.id.action_share) {
         txtMsg.setText("Share...");

         return true;
      } else if (id == R.id.action_download) {
         txtMsg.setText("Download...");
         return true;
      } else if (id == R.id.action_about) {
         txtMsg.setText("About...");
         return true;
      } else if (id == R.id.action_settings) {
         txtMsg.setText("Settings...");
         return true;
      }

      return false;
   } //onOptionsItemSelected

} //MainActivity
```

4

Przykład 3 – Kafelek do wyszukiwania

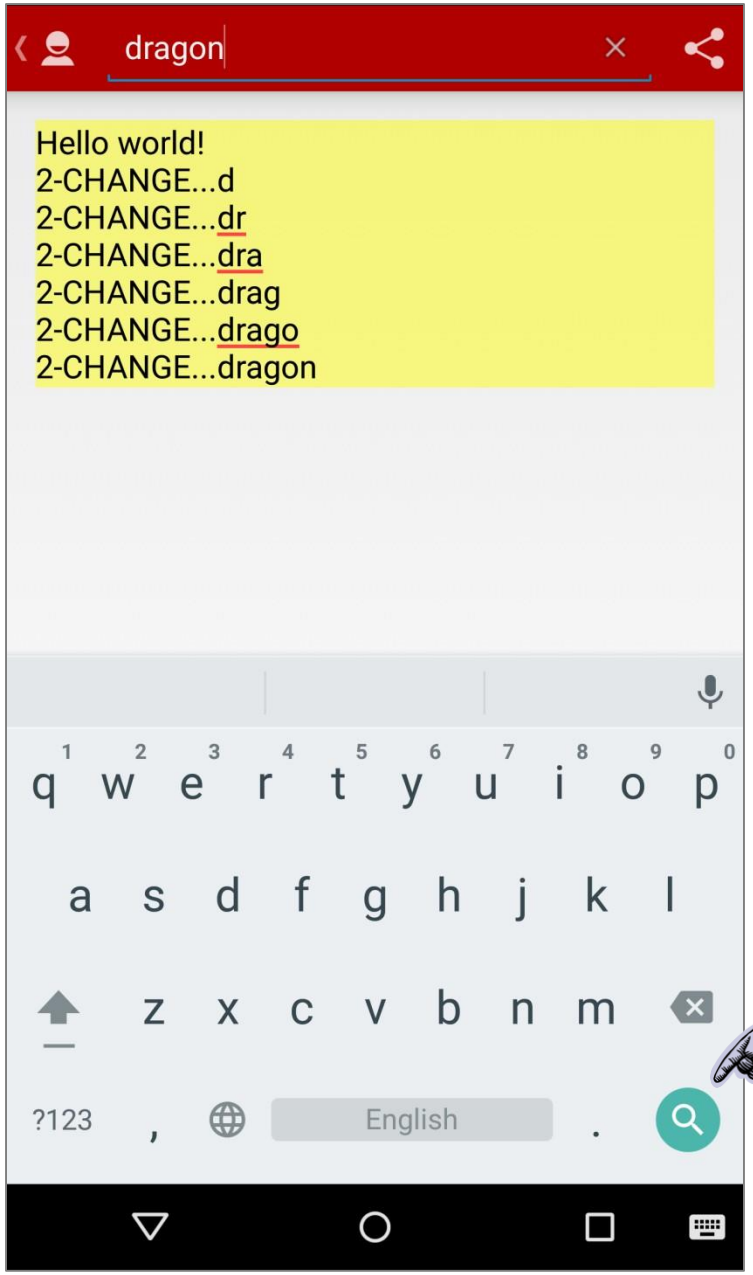
Komentarz

1. Podczas wywołania metody **onCreateOptionsMenu**() elementy definiowane w pliku zasobów *menu/main.xml* są dodane do ActionBar. Instrukcja
```
txtSearchValue = (SearchView) menu.findItem(R.id.action_search)
                                    .getActionView();
```
daje dostęp do elementu SearchView, reprezentowany jako ikona lupy.

2. Następny krok polega na zdefiniowaniu **QueryTextListener** powiązany z komponentem SearchView. Nasłuchiwacz ma dwie metody. Pierwsza – **onQueryTextSubmit**() – wywoływana jest kiedy użytkownik kliknie na ikonę wyszukiwania. W tym momencie, tekst zawarty wewnątrz SearchView może zostać wysłany do zdefiniowanego przez użytkownika *dostawcy wyszukiwania*. Instrukcja *invalidateOptionsMenu*() zamyka pasek wyszukiwania i odświeża widok ActionBara. Instrukcja *.setQuery*(*“”, false*) wykorzystywana jest by wyczyścić tekst nowo utworzonego komponentu SearchView (który nie jest jeszcze widoczny).

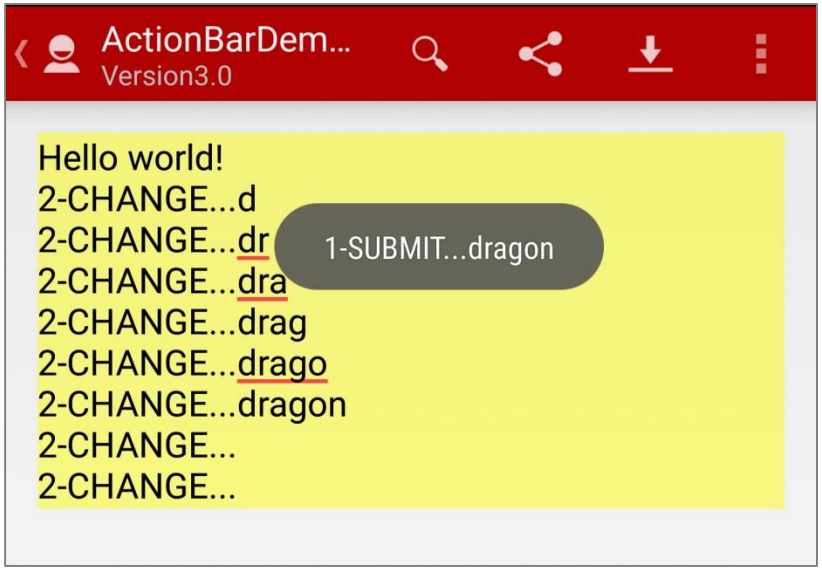
3. Druga metoda – **`onQueryTextChange`** – implementuje **`mechanizm text-watcher`** wywoływany kiedy nowy znak wprowadzony jest do SearchView. Metoda ta może być wykorzystana by pokazać użytkownikowi sugestie ciągów do wyszukiwania.

Przykład 3 – Kafelek do wyszukiwania

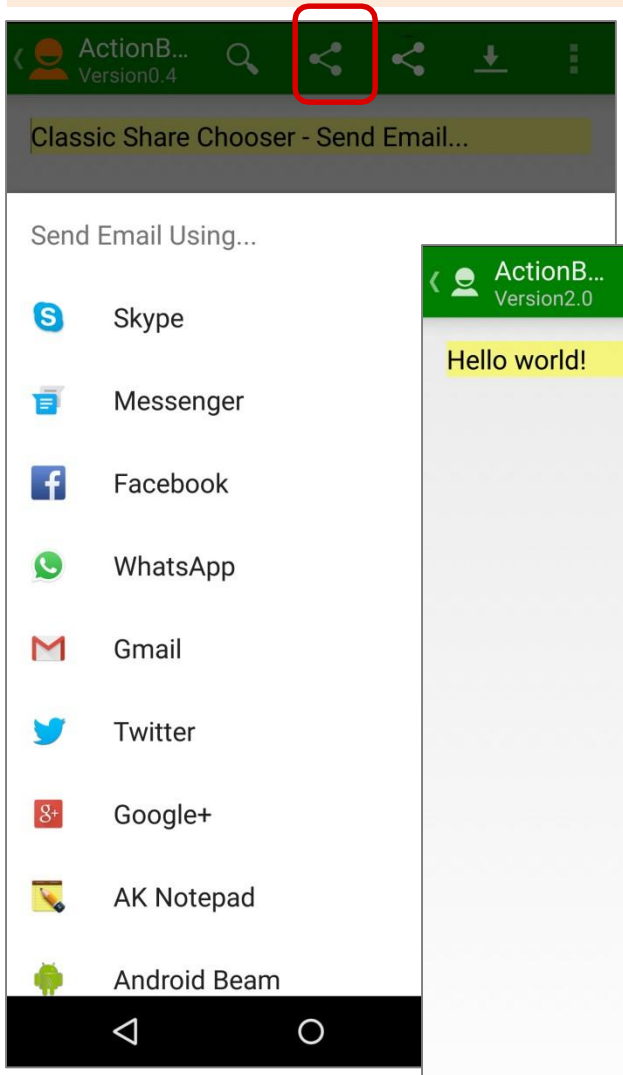


Rezultat działania przykładu 3. Nasłuchiwacz reaguje po każdorazowym wpisaniu nowego znaku.

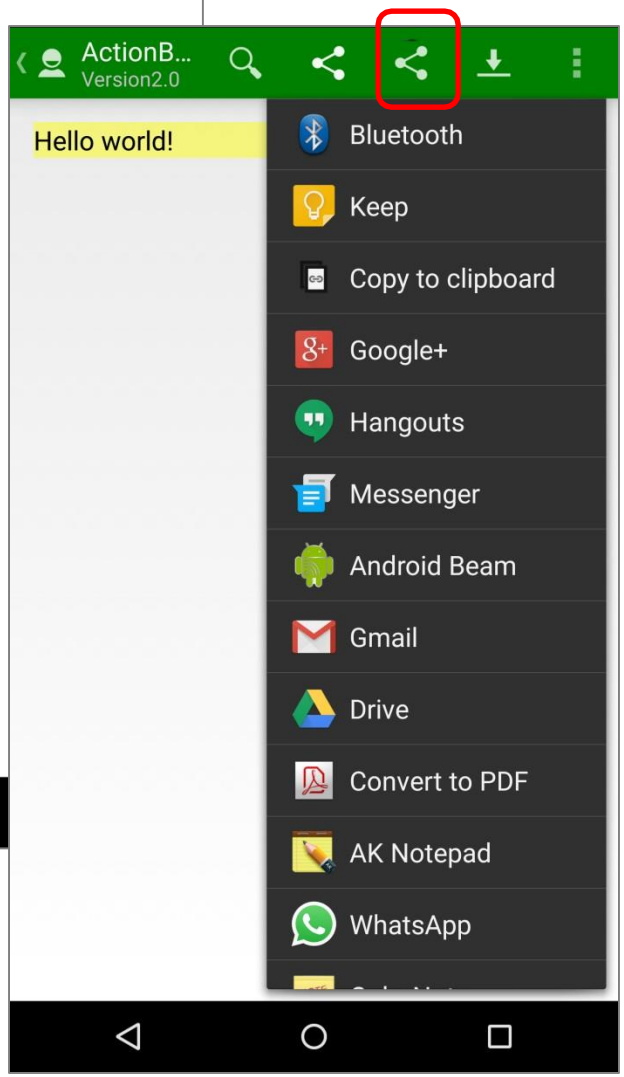
Komunikat typu toast pojawia się po zaakceptowaniu ciągu do wyszukiwania.



Przykład 4 – Przycisk do udostępniania



(a) Chooser Share



(b) Easy Share

W tym przykładzie aplikacja zostanie wzbogacona o możliwość udostępniania danych.

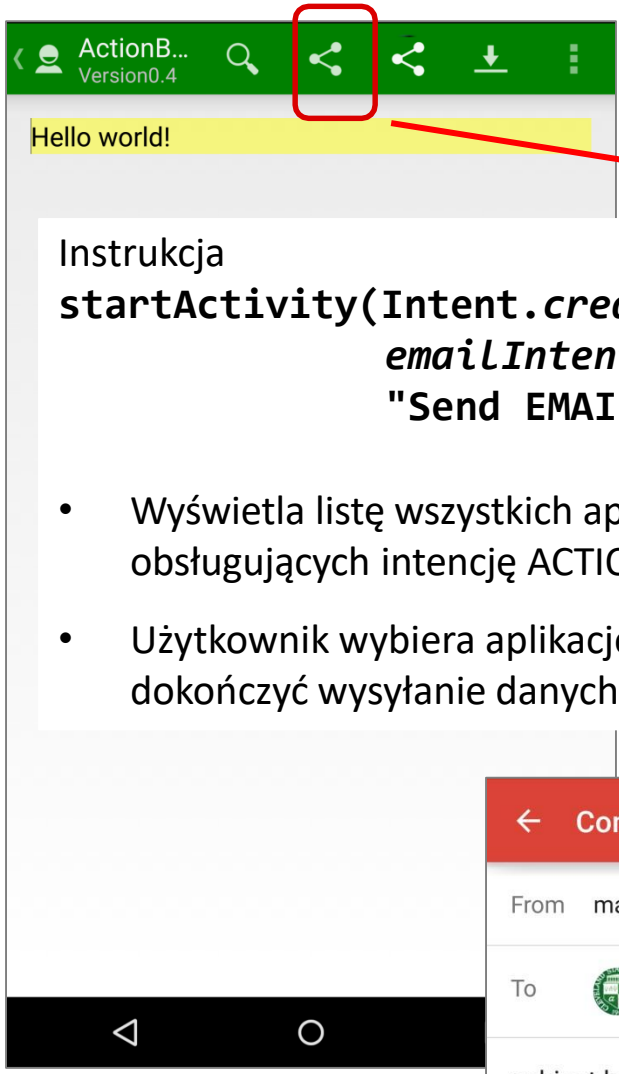
Użytkownik może wybrać jakie dane oraz przez jaki kanał komunikacji można wysłać.

Przykład prezentuje dwa warianty:

(a) klasyczny **dialog wyboru** z którego użytkownik wybiera sposób komunikacji

(b) z wykorzystaniem kafelka „**łatwego udostępniania**” bazującym na widżecie *SharedActionProvider* dostępnym od API-14.

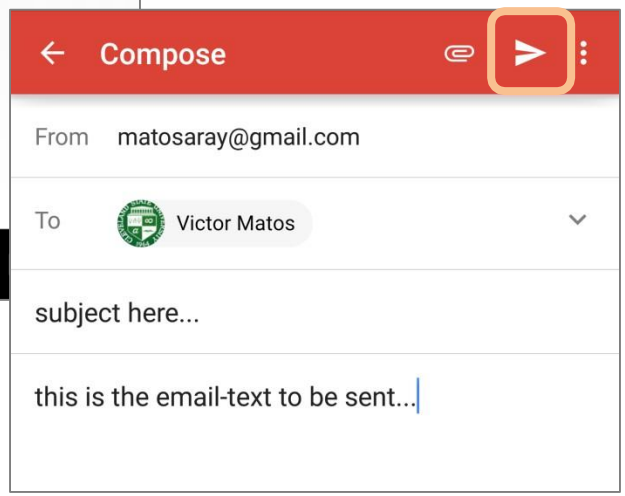
Przykład 4 – Przycisk do udostępniania

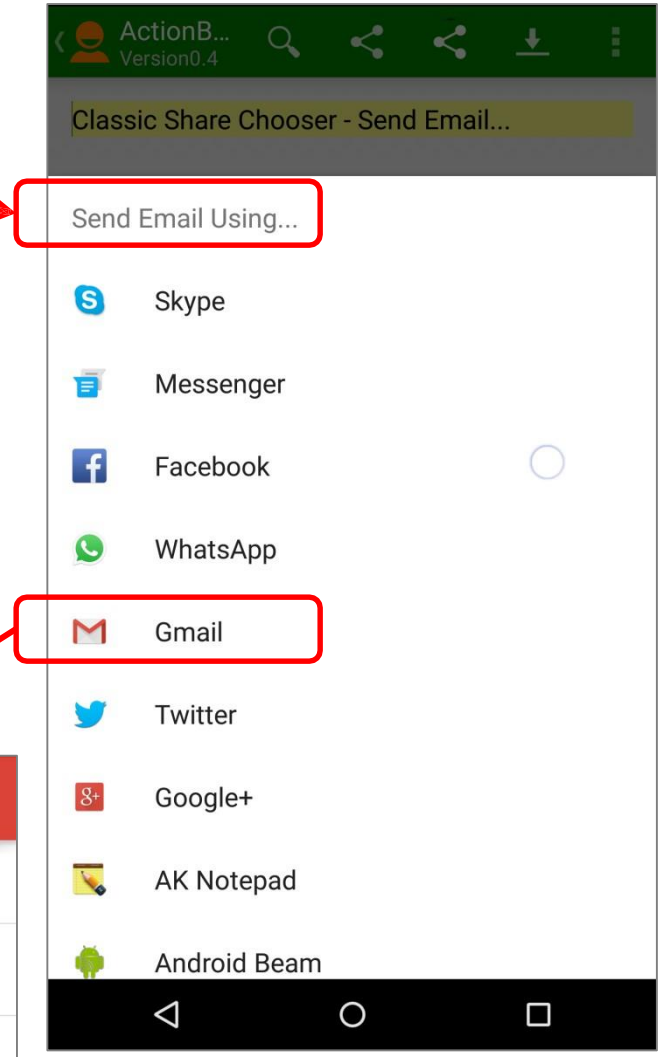


Instrukcja

```
startActivity(Intent.createChooser(
                emailIntent(),
                "Send EMAIL Using..."));
```

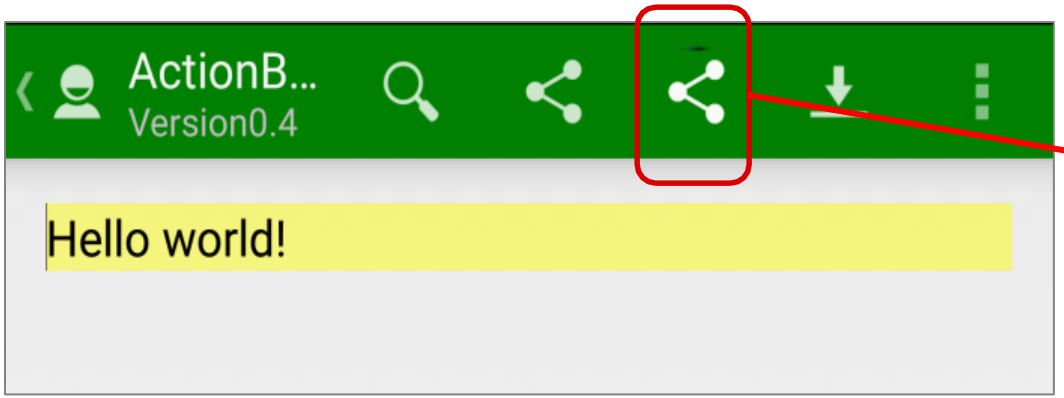
- Wyświetla listę wszystkich aplikacji obsługujących intencję ACTION_SENDTO
- Użytkownik wybiera aplikację z listy by dokończyć wysyłanie danych za jej pomocą.

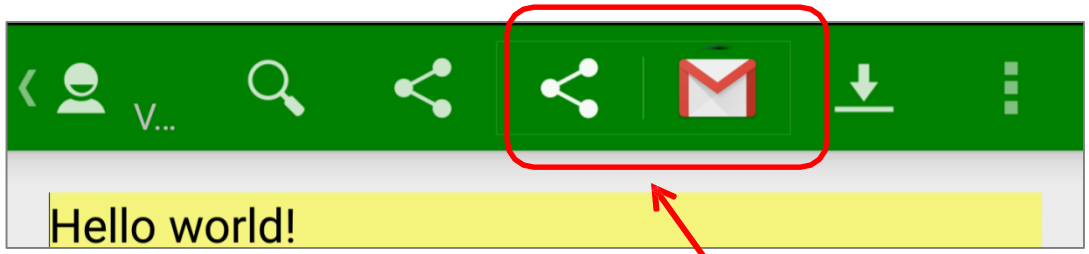


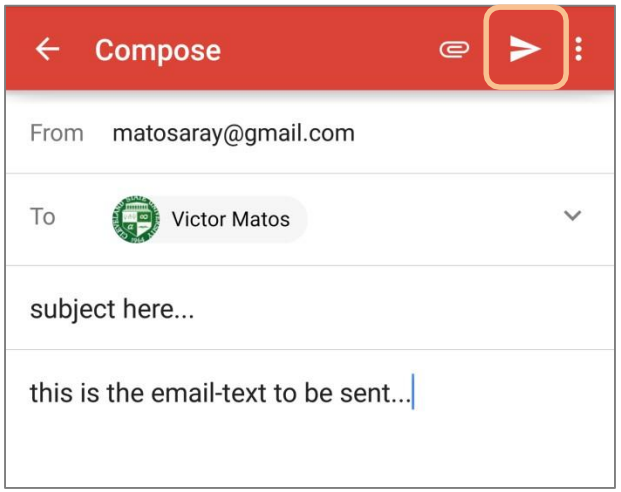


(a) Klasyczny dialog

Przykład 4 – Przycisk do udostępniania

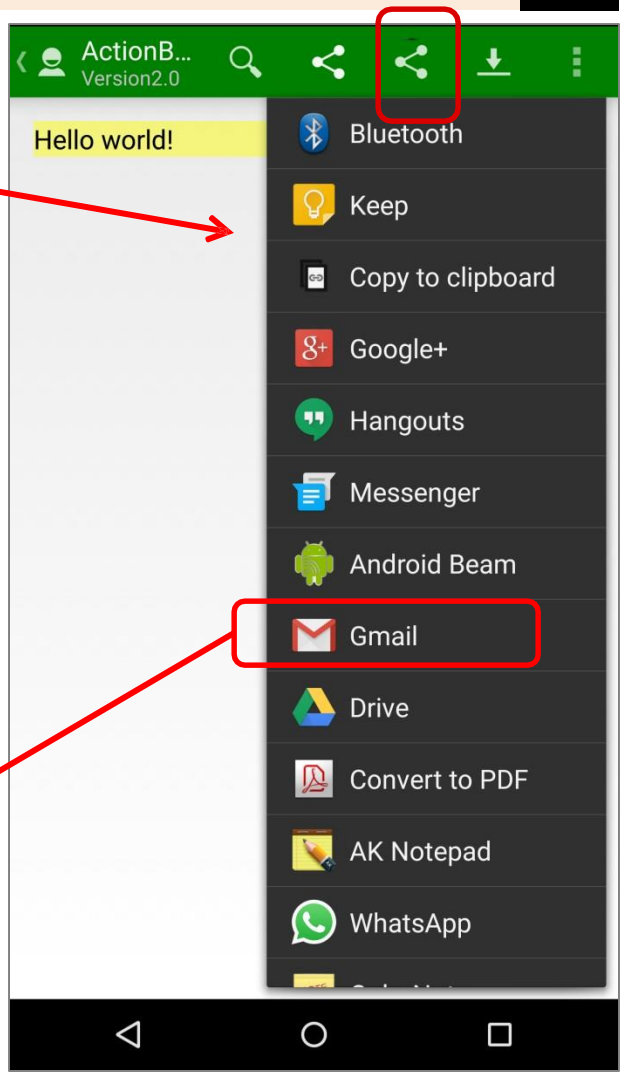






Po wyborze dostawcy ActionBar ulega zmianie.

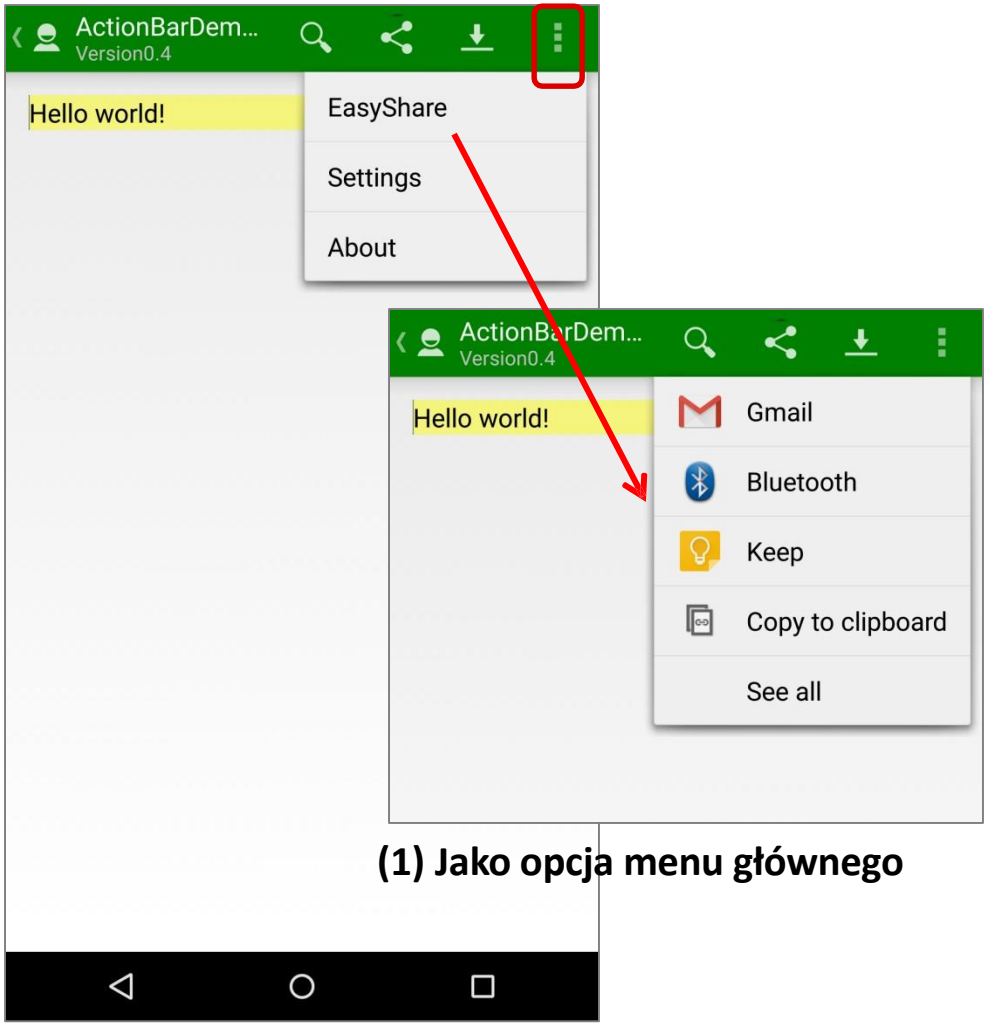
Wybrana opcja dodawana jest jako *domyślna*.



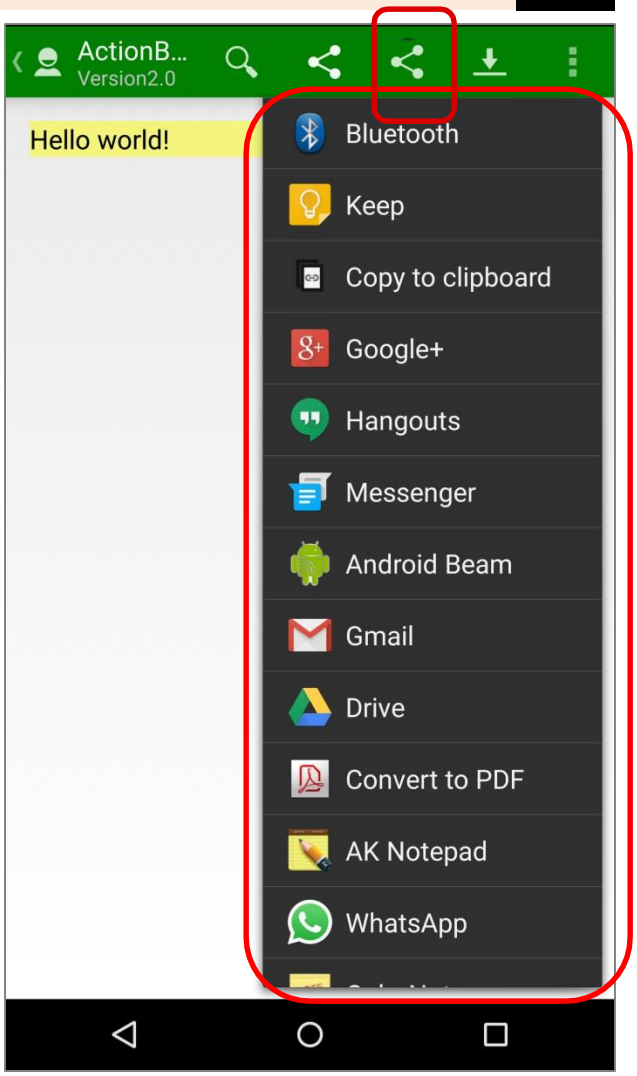
(b) Łatwe udostępnianie

Przykład 4 – Przycisk do udostępniania

Widok **łatwego udostępniania** może być wyświetlany zarówno na komponencie ActionBar jak i w menu głównym.



(1) Jako opcja menu głównego

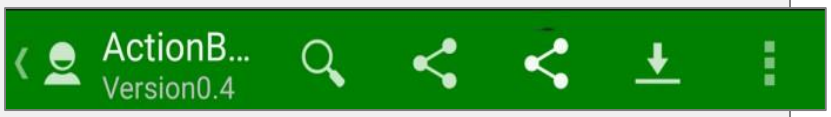


(2) Jako część ActionBar'a

Przykład 4 – Przycisk do udostępniania

Plik zasobów **MENU** uwzględniający widżet **ShareActionProvider:**

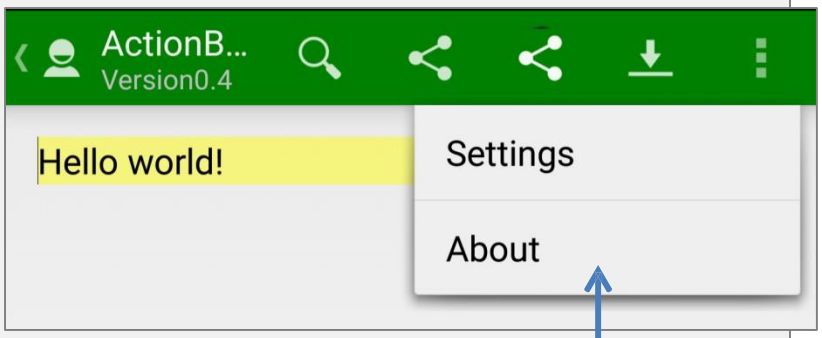
```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="csu.matos.MainActivity" >

    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>

    <item
        android:id="@+id/action_share_chooser"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="ShareChooser" />

     <item
        android:id="@+id/action_share_easy"
        android:orderInCategory="145"
        android:showAsAction="always"
        android:title="EasyShare"
        android:actionProviderClass="android.widget.ShareActionProvider" />
```



Przykład 4 – Przycisk do udostępniania

Plik zasobów **MENU** uwzględniający widżet **ShareActionProvider:**

```
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>

    <item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>

    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>

</menu>
```



Przykład 4 – Przycisk do udostępniania

Kod wzorowany na Przykładzie 2.

```
public class MainActivity extends Activity {

    EditText txtMsg;
    ActionBar actionBar;
    ShareActionProvider shareActionProvider;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    txtMsg = (EditText)findViewById(R.id.txtMsg);

    // setup ActionBar
    actionBar = getActionBar();

    actionBar.setTitle("ActionBarDemo4");
    actionBar.setSubtitle("Version0.4");
    actionBar.setLogo(R.drawable.ic_action_logo);
    actionBar.setBackgroundDrawable(getResources()
             .getDrawable(R.drawable.mybackground0));

    actionBar.setDisplayShowCustomEnabled(true); // allow custom views to be shown
    actionBar.setDisplayHomeAsUpEnabled(true);   // show 'UP' affordance < button
    actionBar.setDisplayShowHomeEnabled(true);   // allow app icon – logo to be shown
    actionBar.setHomeButtonEnabled(true);        // needed for API14 or greater

  }
```

Przykład 4 – Przycisk do udostępniania

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {

   // Inflate the menu, add ShareChooser & EasyShare tiles
   getMenuInflater().inflate(R.menu.main, menu);

   // Locate MenuItem holding ShareActionProvider
    MenuItem easySharedItem = menu.findItem(R.id.action_share_easy);

   // prepare EASY SHARE action tile:
   // Fetch and store a ShareActionProvider for future usage
   // an intent assembles the email(or SMS), you need only to select carrier
   shareActionProvider = (ShareActionProvider) easySharedItem.getActionProvider();

   // prepare an EMAIL
   shareActionProvider.setShareIntent( emailIntent() );

   // prepare an SMS - try this later...
   // shareActionProvider.setShareIntent( smsIntent() );

    return super.onCreateOptionsMenu(menu);
}
```

1

Przykład 4 – Przycisk do udostępniania

```
// return a SHARED intent to deliver an email
 private Intent emailIntent() {

   Intent intent = new Intent(Intent.ACTION_SEND);
   intent.setType("text/plain");
   intent.putExtra(Intent.EXTRA_EMAIL, new String[] { "v.matos@csuohio.edu" });
   intent.putExtra(Intent.EXTRA_SUBJECT, "subject here...");
   intent.putExtra(Intent.EXTRA_TEXT, "this is the email-text to be sent...");

   return intent;

 }


// return a SHARED intent to deliver an SMS text-message
 private Intent smsIntent() {

   Intent intent = new Intent(Intent.ACTION_VIEW);
   String yourNumber="+ 1 216 555-4321";
   intent.setData( Uri.parse("sms:" + yourNumber) );
   intent.putExtra("sms_body", "Here goes my msg");

   return intent;

 }
```

Przykład 4 – Przycisk do udostępniania

```
   @Override
   public boolean onOptionsItemSelected(MenuItem item) {
      // Handle action bar item clicks here. Observe EasyShare is NOT handled here!
      int id = item.getItemId();
      if (id == R.id.action_search) {
         txtMsg.setText("Search...");
         return true;
      }
      else if (id == R.id.action_share_chooser) {
         txtMsg.setText("Classic Share Chooser - Send Email...");

4        startActivity( Intent.createChooser( emailIntent(), "Send EMAIL Using...") );
         //startActivity(Intent.createChooser(smsIntent(), "Send SMS Using..."));
         return true;
      }
      else if (id == R.id.action_download) {
         txtMsg.setText("Download...");
         return true;
      }
      else if (id == R.id.action_about) {
         txtMsg.setText("About...");
         return true;
      }
      else if (id == R.id.action_settings) {
         txtMsg.setText("Settings...");
         return true;
      }
      return false;
   }
}//Activity
```

Przykład 4 – Przycisk do udostępniania

Komentarz

1. Metoda **onCreateOptionsMenu** jest wykorzystywana do stworzenia obiektów na podstawie specyfikacji XML. Element *@id+action_share_easy* zawierający widżet *ShareAccessProvider* jest powiązany z obiektem *easyShareProvider*. Jest to używane do kontrolowania wersji z łatwym udostępnianiem. Później wspomniany obiekt jest wiązany z intencją (ACTION_VIEW, ACTION_SENDTO) która zostanie użyta by dostarczyć wybrane dane.

2. Metoda **`emailIntent()`** zwraca intencję w której zostaje stworzona prosta wiadomość mailowa. Wiadomość zawiera typ, odbiorców, temat i treść.

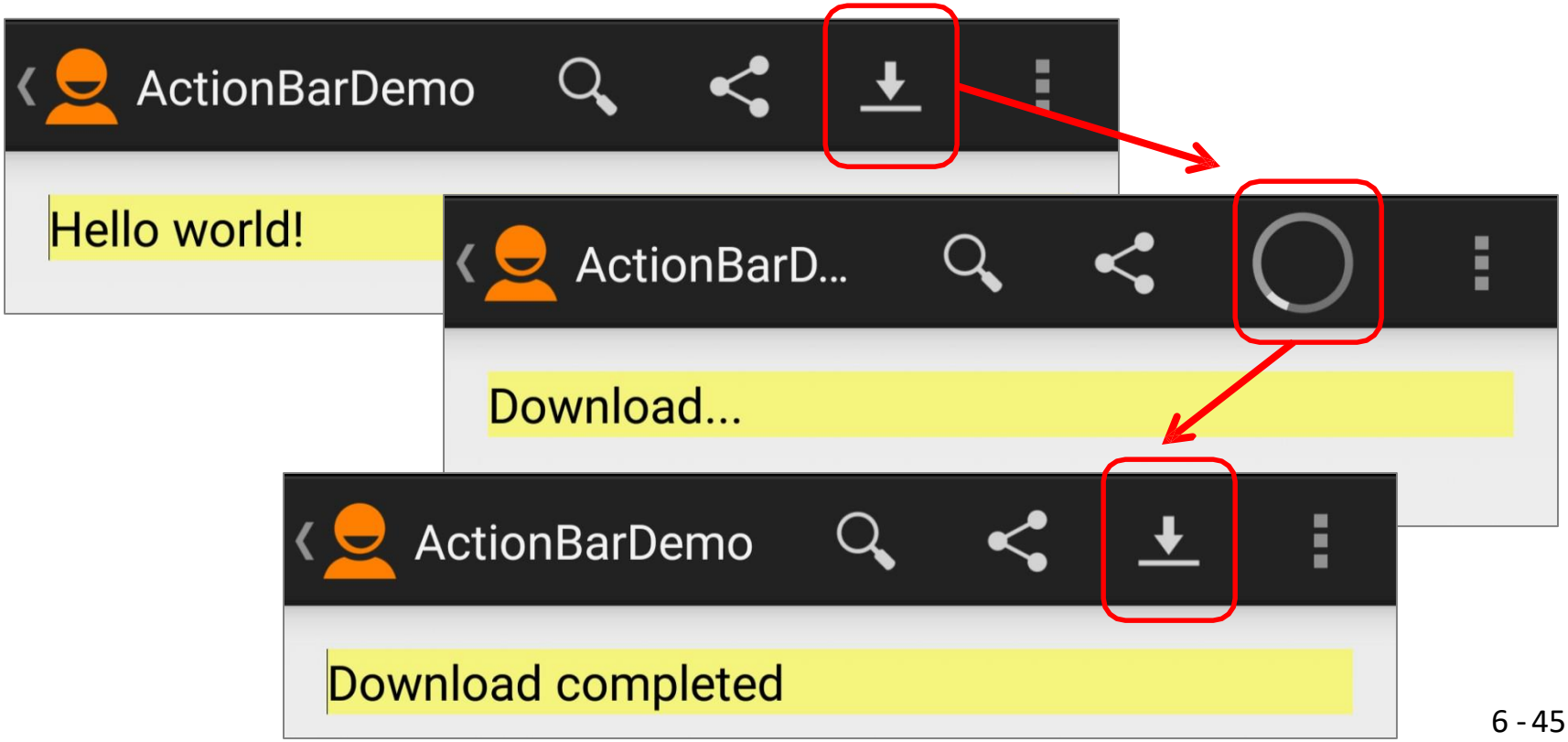
3. Metoda **`smsIntent()`** zwraca intencję w której zostaje stworzona wiadomość SMS. Wiadomość zawiera typ, odbiorcę oraz treść.

4. Metoda **`onOptionsItemSelected()`** jest wykorzystywana by rozpoznać wybór użytkownika (gdy wykorzystywany jest wariant z dialogiem). Implikuje to wywołanie metody *Intent.createChooser*(…) by wyświetlić listę aplikacji umożliwiających wysyłanie danych. Wariant z łatwym udostępnianiem nie jest przetwarzany w tej metodzie.

Przykład 5 – Kafelek do ściągania

W tym przykładzie ActionBar wyświetla ikonę **ściągania**. Po kliknięciu wywołuje ***zdefiniowaną przez użytkownika długotrwałą operację*** (jak przeszukiwanie bazy danych czy operacje internetowe). Tego typu czynności nie powinny być wykonywane w głównym wątku aplikacji.

Zadanie wykonywane jest w tle. Użytkownik jest informowany o stanie wątku w tle poprzez okrągły pasek postępu. Gdy zadanie zostanie ukończone, aktywność jest o tym informowana przez odpowiednie wywołanie zwrotne.



Przykład 5 – Kafelek do ściągania

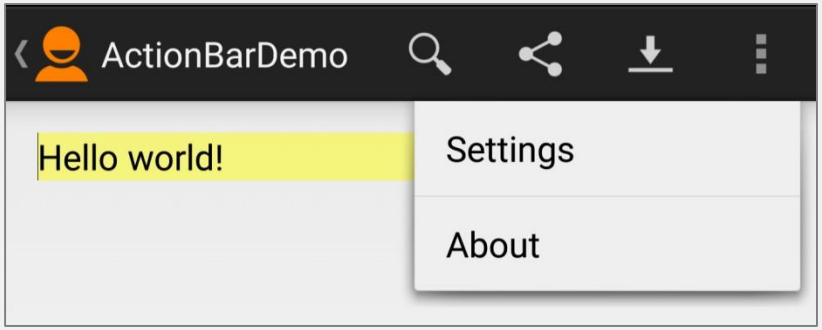
UKŁAD XML:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="csu.matos.MainActivity" >

    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>

    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
```

Przykład 5 – Kafelek do ściągania

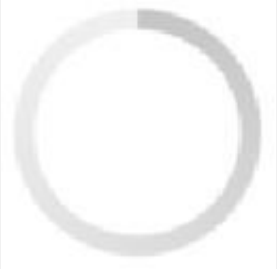
UKŁAD XML:

```
<item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>

</menu>
```



Plik **res/layout/custom_view_download** definiujący okrągły pasek postępu:

```
<?xml version="1.0" encoding="utf-8"?>

<ProgressBar
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/customViewActionProgressBar"
    style="?android:attr/progressBarStyleLarge"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```



Przykład 5 – Kafelek do ściągania

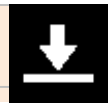
MainActivity

```
public class MainActivity extends Activity {

   EditText txtMsg;
   ActionBar actionBar;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      txtMsg = (EditText) findViewById(R.id.txtMsg);

      // setup the ActionBar
      actionBar = getActionBar();
      actionBar.setDisplayShowCustomEnabled(true);  // allow custom views to be shown
      actionBar.setDisplayHomeAsUpEnabled(true);    // show 'UP' affordance < button
      actionBar.setDisplayShowHomeEnabled(true);    // allow app icon – logo to be shown
      actionBar.setHomeButtonEnabled(true);         // needed for API.14 or greater
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the resource menu file: main.xml
      getMenuInflater().inflate(R.menu.main, menu);
      return true;

   }
```

Przykład 5 – Kafelek do ściągania

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   // Handle clicking of ActionBar items here.
   int id = item.getItemId();

   if (id == R.id.action_search) {
      txtMsg.setText("Search...");
      return true;
   } else if (id == R.id.action_share) {
      txtMsg.setText("Share...");
      return true;

   } else if (id == R.id.action_download) {
      txtMsg.setText("Download...");
      // Temporarily replace the download action-tile with circular progress bar
1 →   performSlowOperation(item);

      return true;

   } else if (id == R.id.action_about) {
      txtMsg.setText("About...");
      return true;
   } else if (id == R.id.action_settings) {
      txtMsg.setText("Settings...");
      return true;
   }

   return false;
}
```

Przykład 5 – Kafelek do ściągania



```
   private void performSlowOperation(MenuItem item) {
      // temporally replace Download action-icon with a progress-bar
      final MenuItem downloadActionItem = item;
2 →   downloadActionItem.setActionView(R.layout.custom_view_download);
      downloadActionItem.expandActionView();
      // define an Android-Handler control to receive messages from a
      // background thread were the slow work is to be done
3 →   final Handler handler = new Handler() {
         @Override
         public void handleMessage(Message msg) {
            super.handleMessage(msg);
            txtMsg.setText("Download completed");      // announce completion of slow job
            downloadActionItem.collapseActionView();   // dismiss progress-bar
            downloadActionItem.setActionView(null);
         }
      };
      // a parallel Thread runs the slow-job and signals its termination
4 →   new Thread() {
         @Override
         public void run() {
            // real 'BUSY_WORK' goes here...(we fake 2 seconds)
            long endTime = SystemClock.uptimeMillis() + 2000; //now + 2 seconds
            handler.sendMessageAtTime(handler.obtainMessage(), endTime);
            super.run();
         }
      }.start();
   }

}//Activity
```

Przykład 5 – Kafelek do ściągania

Komentarz

1. Po kliknięciu przez użytkownika w kafelek ściągania metoda **onOptionsItemSelected()** rozpoznaje zdarzenie oraz wywołuje nowy wątek działający w tle.

2. Wątek rozpoczyna działanie od zamiany ikony do ściągania na okrągły pasek postępu (patrz metody .**setActionView(…)** oraz **expandActionView()**).

3. Tworzony jest obiekt do nasłuchiwania komunikatów wysyłanych przez wątek pracujący w tle. Gdy nadejdzie wiadomość ”zakończono” obiekt usuwa pasek postępu i przywraca oryginalny stan ActionBar’a (patrz metody .**collapseActionView()** oraz **.setActionView(null**)).

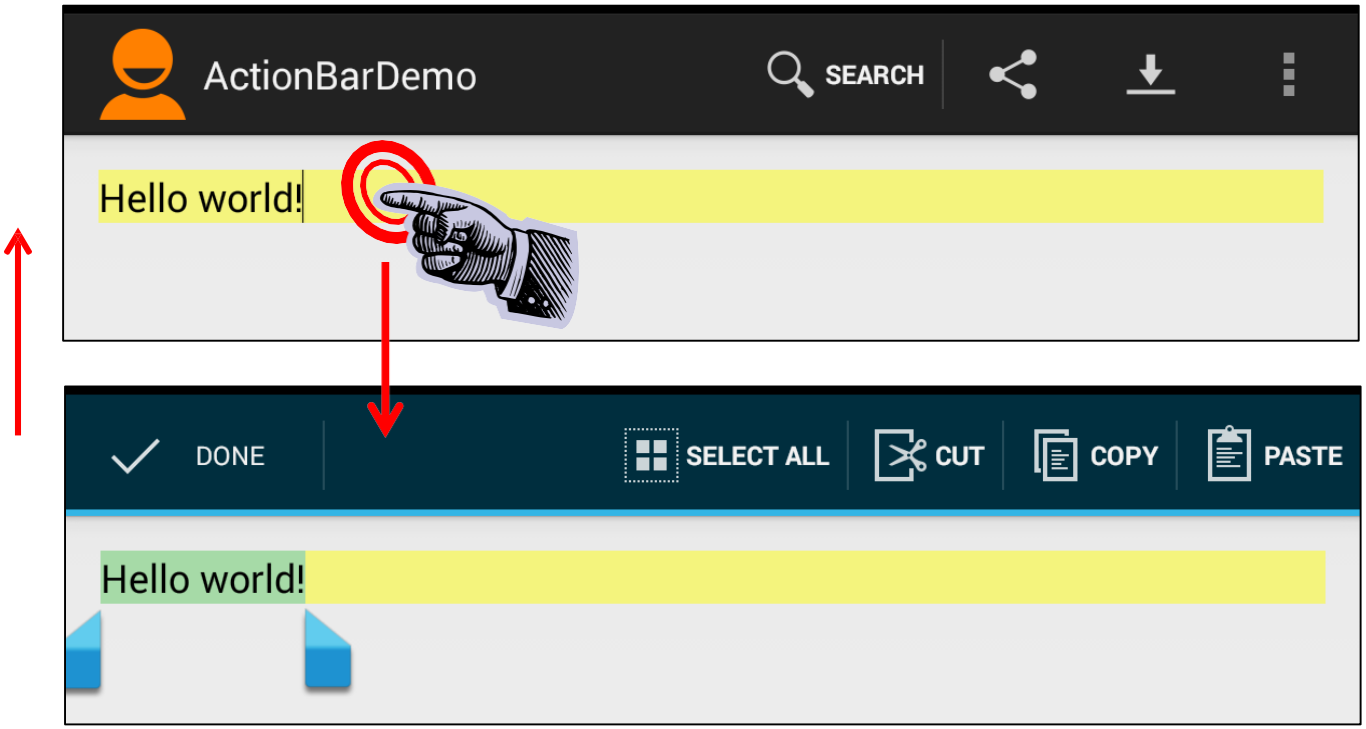
4. Osobny wątek symuluje długotrwałą operację trwającą 5 sekund. Dzięki pracy w tle, główny wątek w dalszym ciągu jest responsywny i użytkownik może (podczas oczekiwania na zakończenie zadania) wykonywać również inne czynności.

   Więcej na temat wielowątkowości znajduje się w osobnej prezentacji.

Przykład 6 – Kontekstowy ActionBar (CAB)

Standardowy ActionBar może zostać rozszerzony o reagowanie na kontekst danego zdarzenia – wywoływanego przez przytrzymanie palca na określonym komponencie. Wówczas ActionBar jest zastępowany przez tzw. **Kontekstowy ActionBar** (**CAB**). Jest to niejako alternatywa dla wykorzystania menu kontekstowego, co było popularnym wzorcem w aplikacjach pisanych dla starszych wersji Androida.

Przykład 1 Prezentował najprostszy sposób wykorzystania CAB.



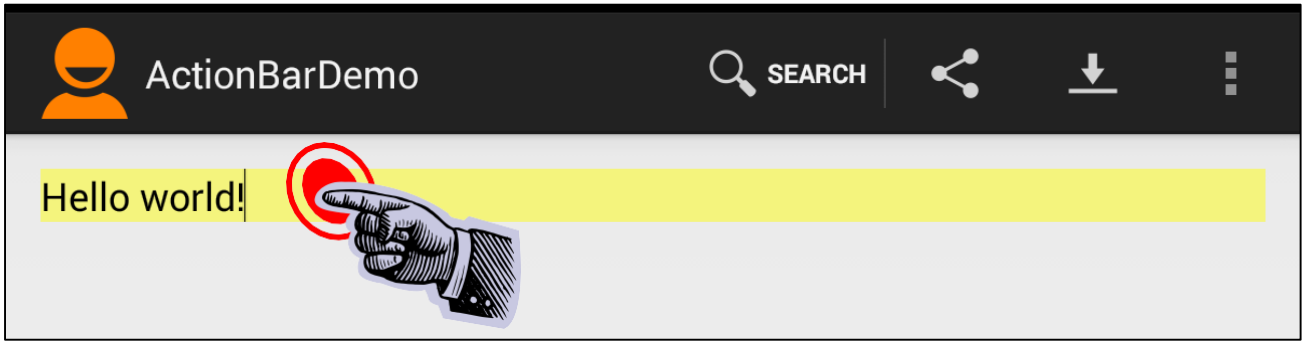
⟵ **ActionBar**

⟵ **Domyślny CAB**

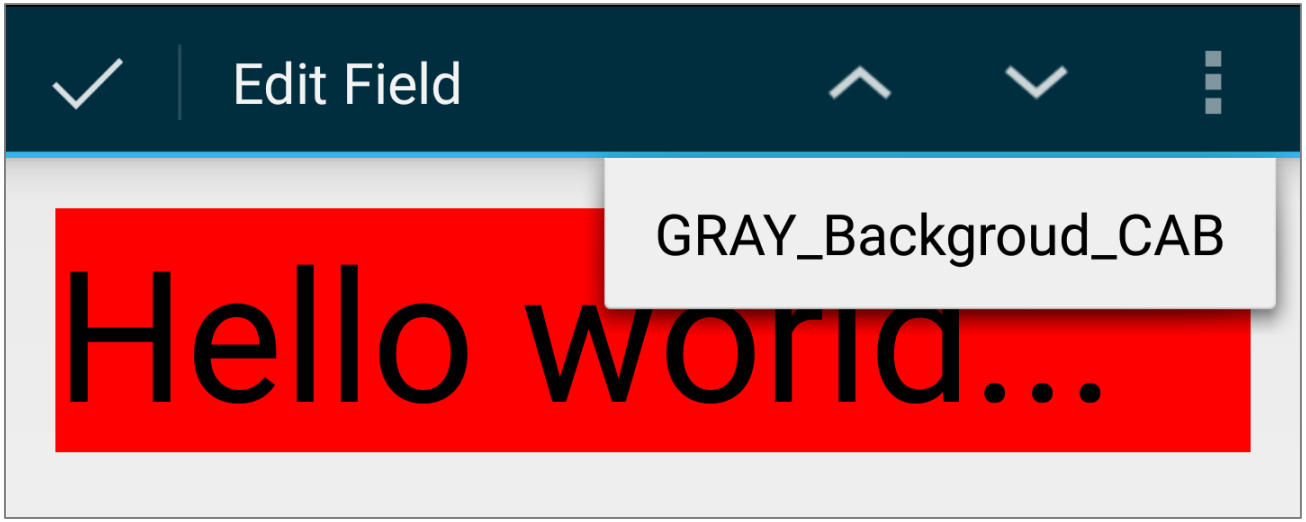
Przykład 6 – Kontekstowy ActionBar (CAB)

Można oczywiście tworzyć własne CABy. W tym celu programista musi stworzyć **tymczasowe menu** które zostanie nałożone na istniejący ActionBar. Należy również zaimplementować interfejs **ActionMode.Callback** by wskazać jak reagować na wybór konkretnej pozycji z CAB.

CAB jest usuwany w momencie w którym użytkownik odznaczy elementy GUI, naciśnie przycisk WSTECZ, lub wybierze opcję *Done* znajdującą się po lewej stronie ActionBara.



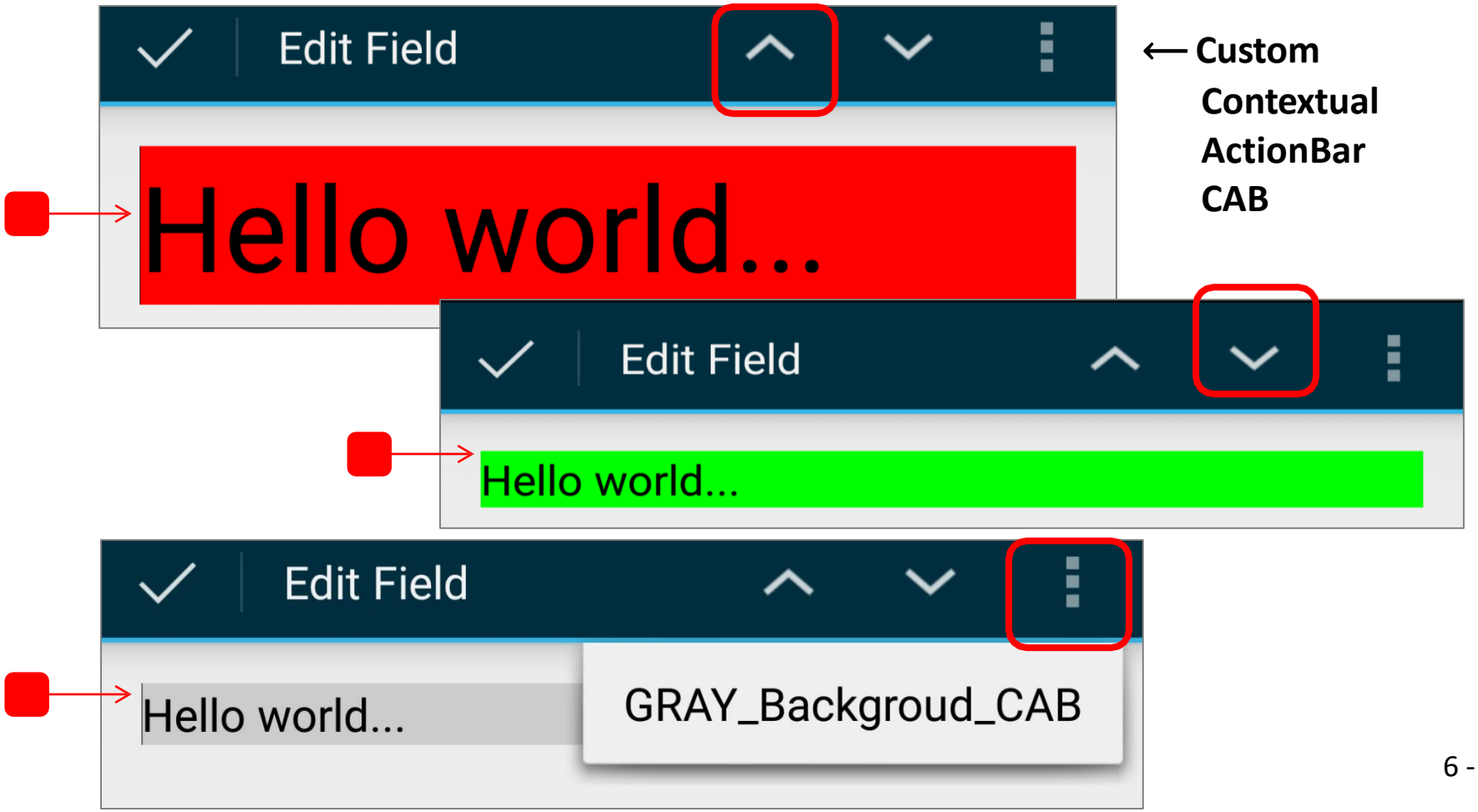
⟵ ActionBar



⟵ Własny CAB

Przykład 6 – Kontekstowy ActionBar (CAB)

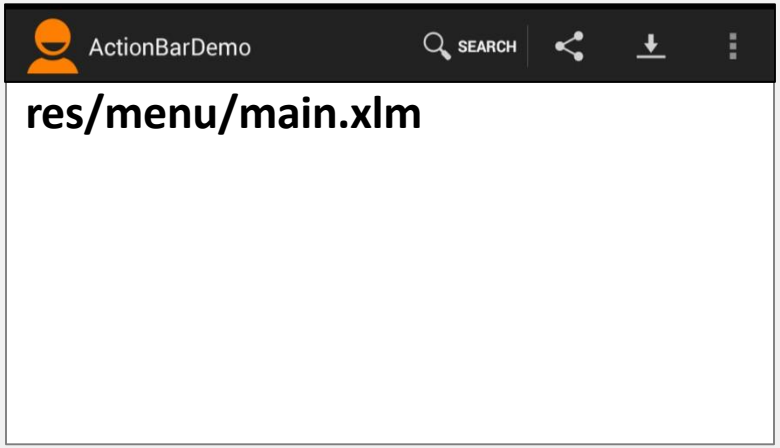
W prezentowanym przykładzie przytrzymanie palca na napisie “Hello world…” aktywuje własny CAB. CAB oferuje dwa kafelki oraz jedną opcję w menu rozwijanym. Pierwszy kafelek (∧) zwiększa wielkość tekstu oraz zmienia tło na czerwone, drugi (∨) zmniejsza rozmiar tekstu i ustawia tło na zielone, natomiast opcja w menu głównym zmienia kolor tła na szary.



← **Custom Contextual ActionBar CAB**

Przykład 6 – Kontekstowy ActionBar (CAB)

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="csu.matos.MainActivity" >
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
    <item
        android:id="@+id/action_settings
        "  android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about
        "
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>
</menu>
```



Przykład 6 – Kontekstowy ActionBar (CAB)


```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.example.x3.MainActivity" >

    <item
        android:id="@+id/action_gray_background_cab"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="GRAY_Backgroud_CAB"/>

    <item
        android:id="@+id/action_increase"
        android:orderInCategory="120"
        android:icon="@drawable/ic_action_increase"
        android:showAsAction="always|withText"
        android:title="Increase"/>

     <item
        android:id="@+id/action_decrease"
        android:orderInCategory="160"
        android:icon="@drawable/ic_action_decrease"
        android:showAsAction="always|withText"
        android:title="Decrease"/>

</menu>
```

res/menu/ txtmsg_cab_menu.xlm

Kontekstowy układ XML

Przykład 6 – Kontekstowy ActionBar (CAB)

MainActivity

```
public class MainActivity extends Activity {

   EditText txtMsg;
   public ActionMode actionmode;
   private TxtMsgCallbacks txtMsgCallbacks;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      txtMsg = (EditText) findViewById(R.id.txtMsg);
      txtMsg.setOnLongClickListener(new OnLongClickListener() {

         @Override
         public boolean onLongClick(View v) {

            if (actionmode == null) {
               txtMsgCallbacks = new TxtMsgCallbacks(MainActivity.this, txtMsg);
            } else {
               Toast.makeText(getApplicationContext(), "REUSING",
                              Toast.LENGTH_LONG).show();
            }
            actionmode = startActionMode(txtMsgCallbacks);
            actionmode.setTitle("Edit Field");
            return true;
         }
      });

   } //onCreate
```

1 → `txtMsg = (EditText) findViewById(R.id.txtMsg);`

2 → `txtMsgCallbacks = new TxtMsgCallbacks(MainActivity.this, txtMsg);`

3 → `actionmode = startActionMode(txtMsgCallbacks);`

Przykład 6 – Kontekstowy ActionBar (CAB)

```
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }

   @Override
   public boolean onOptionsItemSelected(MenuItem item) {
      int id = item.getItemId();

      if (id == R.id.action_search) {
         txtMsg.setText("Search...");
         return true;
      } else if (id == R.id.action_share) {
         txtMsg.setText("Share...");
         return true;
      } else if (id == R.id.action_download) {
         txtMsg.setText("Download...");
         return true;
      } else if (id == R.id.action_about) {
         txtMsg.setText("About...");
         return true;
      } else if (id == R.id.action_settings) {
         txtMsg.setText("Settings...");
         return true;
      }

      return false;
   } //onOptionsItemSelected

} //MainActivity
```

Przykład 6 – Kontekstowy ActionBar (CAB)

Komentarze – MainActivity

1. Nasłuchiwacz **LongClickListener** jest ustawiony dla pola tekstowego (typu EditText). Zadaniem nasłuchiwacza jest uaktywnienie własnego, kontekstowego paska ActionBar.

2. Gdy zdarzenie typu *LongClick* zostaje wykryte, obiekt **ActionMode** jest tworzony by zarządzać CAB. Tego typu obiekty są przydatne w sytuacjach, w których wymagana jest tymczasowa podmiana GUI i zmiana sposobu interakcji z użytkownikiem. W tym przykładzie obiekt ActionMode przykrywa globalny pasek ActionBar. Obiekt ten otrzymuje określony napis (“Edit Field”), referencję do widoku EditText oraz referencję do kontekstu aplikacji.

3. Dodatkowo obiekt **TxtMsgCallback** jest tworzony (bądź wykorzystywany podobnie jeśli nie jest to pierwszy raz) i zostaje powiązany z komponentem ActionMode by obsłużyć zdarzenia związane z interakcją użytkownika.

Przykład 6 – Kontekstowy ActionBar (CAB)

Klasa TxtMsgCallbacks

```
public class TxtMsgCallbacks  implements ActionMode.Callback {
   // This class handles the aActions shown by the custom CAB
   MainActivity mainContext;
   TextView txtMsg;

   public TxtMsgCallbacks(MainActivity mainContext, TextView txtMsg) {
      // this is the EditText view controlled by the CAB
      this.txtMsg = txtMsg;
      this.mainContext = mainContext;
   }

   @Override
   public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
1 →
      //set value of 10px (regardless of screen density)
      float tenPixels = TypedValue.applyDimension( TypedValue.COMPLEX_UNIT_DIP, 10,
                                    mainContext.getResources().getDisplayMetrics());
      //detect current text-size
      float oldSize = txtMsg.getTextSize();
2 →
      //increase by 10px text-size with red background
      if (item.getItemId() == R.id.action_increase) {
         txtMsg.setTextSize(TypedValue.COMPLEX_UNIT_PX, oldSize + tenPixels);
         txtMsg.setBackgroundColor(Color.RED);

3 →   //decrease by 10px text-size with green background
      } else if (item.getItemId() == R.id.action_decrease) {
         txtMsg.setTextSize(TypedValue.COMPLEX_UNIT_PX, oldSize - tenPixels);
         txtMsg.setBackgroundColor(Color.GREEN);
```

Przykład 6 – Kontekstowy ActionBar (CAB)

Klasa TxtMsgCallbacks

4

5

```
        //set background to gray
        } else if (item.getItemId() == R.id.action_gray_background_cab) {
            txtMsg.setBackgroundColor(Color.LTGRAY);
        }

        return false;
    }

    // showing other states from the  ActionMode life-cycle
    @Override
    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        mode.getMenuInflater().inflate(R.menu.txtmsg_cab_menu, menu);
        return true;
    }

    @Override
    public void onDestroyActionMode(ActionMode mode) {
        Toast.makeText(mainContext, "Destroy CAB", Toast.LENGTH_LONG).show();
    }

    @Override
    public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
        Toast.makeText(mainContext, "Prepare CAB", Toast.LENGTH_LONG).show();
        return false;
    }
}
```

Przykład 6 – Kontekstowy ActionBar (CAB)

Komentarze – TxtMsgCallbacks

Klasa **ActionMode** jest odpowiedzialna za wizualizację własnego CAB. Klasa pomocnicza **TxtMsgCallbacks** tworzy obiekty na podstawie specyfikacji (**txtmsg_cab_menu.xml**) które zakrywają globalny komponent ActionBar oraz monitoruje kliknięcia przycisków wyświetlanych na CAB.

1. Metoda **`onActionItemClicked()`** jest wywoływana jeśli użytkownik kliknie wybrany kafelek. Jej pierwszy krok to określenie ile faktycznie zajmuje 10 pikseli na zadanym ekranie.

2. Po wyborze pierwszej akcji wielkość tekstu (txtMsg) jest zwiększana o 10 px, a kolor tła zmieniany jest na czerwony.

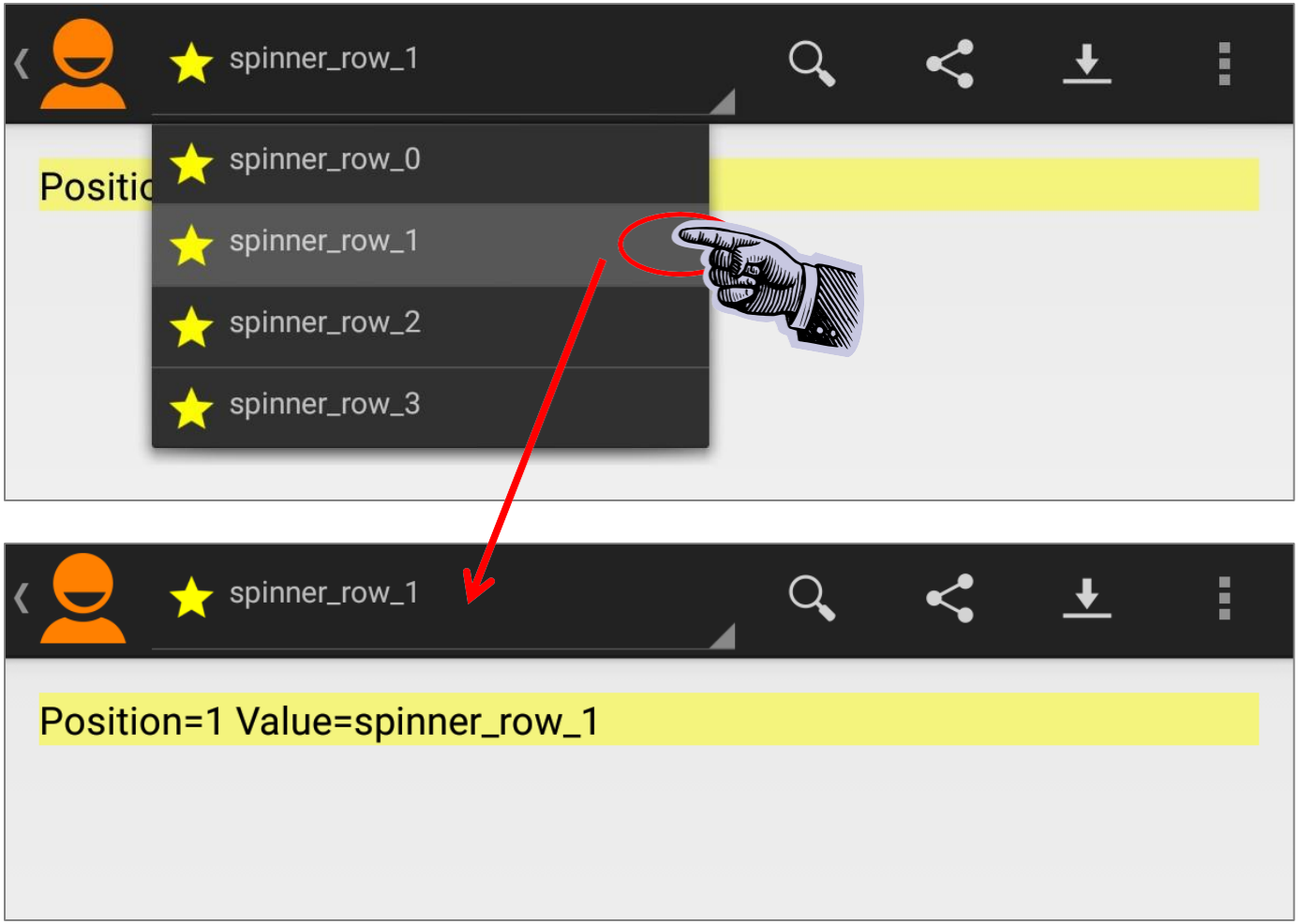
3. Po wyborze drugiej akcji wielkość tekstu (txtMsg) jest zmniejszana o 10 px, a kolor tła zmieniany jest na zielony.

4. Wybór trzeciej opcji skutkuje zmianą koloru tła na szary.

5. Komunikat typu Toast jest wyświetlany przy przejściu do innego stanu.

Przykład 7 – Spinner

W tym przykładzie komponent typu **Spinner** jest dodawany do ActionBar. Plik **res/menu/main.xml** jest taki sam jak w pierwszym przykładzie.



Przykład 7 – Spinner

Pierwszym krokiem jest zdefiniowanie układu XML zawierający komponent typu Spinner, który później zostanie umieszczony na pasku Actionbar:

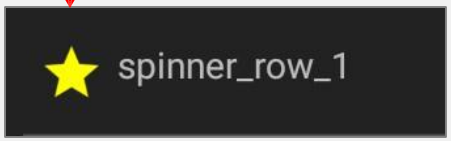
custom_spinner_view_on_actionbar.xml

```
<Spinner
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/spinner_data_row"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Układ określający formatowanie poszczególnych pozycji spinnera.

custom_spinner_row_icon_caption.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="wrap_content"
    android:orientation="horizontal"    android:padding="6dp" >
    <ImageView
        android:id="@+id/imgSpinnerRowIcon"
        android:layout_width="25dp" android:layout_height="25dp"
        android:layout_marginRight="5dp"
        android:src="@drawable/ic_launcher" />
    <TextView
        android:id="@+id/txtSpinnerRowCaption"
        android:layout_width="wrap_content"  android:layout_height="wrap_content" />
</LinearLayout>
```



Przykład 7 – Spinner

W metodzie **onCreate** pole rozwijane (Spinner) jest przypisywane do ActionBara. Ponadto tworzony jest DataAdapter i przypisywany nasłuchiwacz 'ItemSelected'.

```
@Override
protected void onResume() {
     super.onResume();
     actionBar = getActionBar();                    // setup the ActionBar
     actionBar.setDisplayShowCustomEnabled(true);  // allow custom views to be shown
     actionBar.setDisplayHomeAsUpEnabled(true);    // show 'UP' affordance < button
     actionBar.setDisplayShowHomeEnabled(true);    // allow app icon – logo to be shown
     actionBar.setHomeButtonEnabled(true);         // needed for API.14 or greater

     // move the spinner to the actionBar as a CustomView
     actionBar.setCustomView(R.layout.custom_spinner_view_on_actionbar);

     // create the custom adapter to feed the spinner
     customSpinnerAdapter = new SpinnerCustomAdapter(
                              getApplicationContext(),
                              SpinnerDummyContent.customSpinnerList);

     // plumbing - get access to the spinner widget shown on the actionBar
     customSpinner = (Spinner)
     actionBar.getCustomView().findViewById(R.id.spinner_data_row);

     // bind spinner and adapter
     customSpinner.setAdapter(customSpinnerAdapter);

     // put a listener to wait for spinner rows to be selected
     customSpinner.setOnItemSelectedListener(this);
     customSpinner.setSelection(selectedSpinnerRow);

}//onResume
```

Przykład 7 – Spinner

SpinnerCustomAdapter. Jest to definicja własnego adaptera by wypełnić poszczególne wiersze Spinnera tekstem oraz grafiką na podstawie: **custom_spinner_row_icon_caption.xml.**

```
public class SpinnerCustomAdapter extends BaseAdapter {

  private ImageView spinnerRowIcon;
  private TextView spinnerRowCaption;
  private ArrayList<SpinnerRow> spinnerRows;
  private Context context;

  public SpinnerCustomAdapter(Context applicationContext,
                              ArrayList<SpinnerRow> customSpinnerList) {
    this.spinnerRows = customSpinnerList;
    this.context = applicationContext;
  }

  @Override
  public int getCount() {  return
    spinnerRows.size();
  }

  @Override
  public Object getItem(int index) {
    return spinnerRows.get(index);
  }

  @Override
  public long getItemId(int position) {
    return position;
  }
```

Przykład 7 – Spinner

SpinnerCustomAdapter. Jest to definicja własnego adaptera by wypełnić poszczególne wiersze Spinnera tekstem oraz grafiką na podstawie: **custom_spinner_row_icon_caption.xml.**

```
  @Override
  public View getView(final int position, View convertView, ViewGroup parent) {

        if (convertView == null) {
          LayoutInflater mInflater = (LayoutInflater) context.getSystemService(
                                      Activity.LAYOUT_INFLATER_SERVICE);
          convertView=mInflater.inflate(R.layout.custom_spinner_row_icon_caption, null);
        }

        spinnerRowIcon = (ImageView) convertView.findViewById(R.id.imgSpinnerRowIcon);
        spinnerRowCaption = (TextView) convertView.findViewById(
                                       R.id.txtSpinnerRowCaption);

        spinnerRowIcon.setImageResource(spinnerRows.get(position).getIcon());

        spinnerRowCaption.setText(spinnerRows.get(position).getCaption());

        convertView.setId(position);

        return convertView;
  }

}
```

Przykład 7 – Spinner

Sztuczne dane źródłowe (SpinnerDummyContent.java). Kod służy wygenerowaniu danych pokazowych dla komponentu Spinner.

```
public class SpinnerDummyContent {

   public static ArrayList<SpinnerRow> customSpinnerList = new
                                       ArrayList<SpinnerRow>();
   static {

      // preparing spinner data (a set of [caption, icon] objects)
      customSpinnerList.add( new SpinnerRow("spinner_row_0",
                             R.drawable.ic_spinner_row_icon));
      customSpinnerList.add( new SpinnerRow("spinner_row_1",
                             R.drawable.ic_spinner_row_icon));
      customSpinnerList.add( new SpinnerRow("spinner_row_2",
                             R.drawable.ic_spinner_row_icon));
      customSpinnerList.add( new SpinnerRow("spinner_row_3",
                             R.drawable.ic_spinner_row_icon));

   }
```

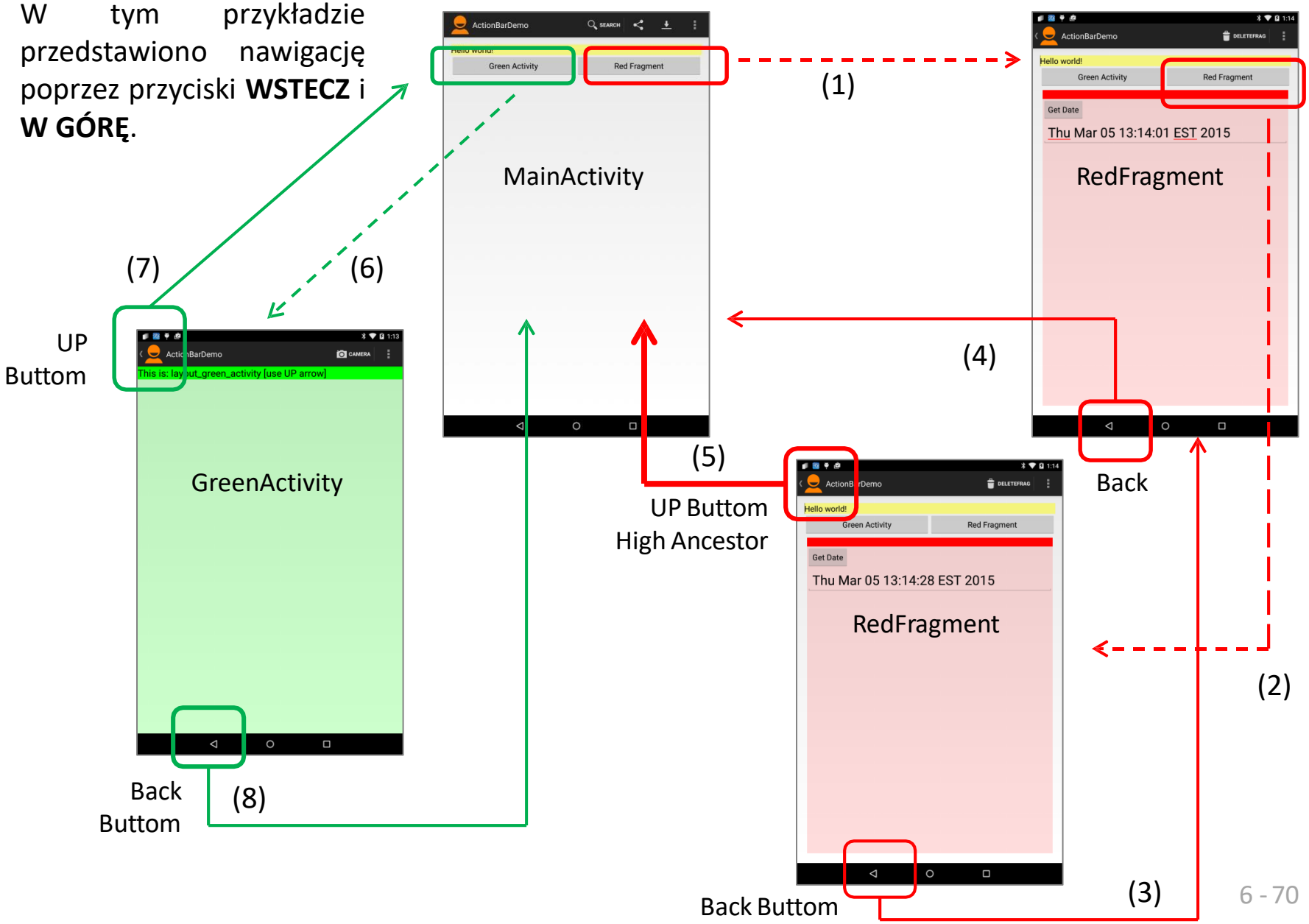
Przykład 7 – Spinner

Sztuczne dane źródłowe. Kod służy wygenerowaniu danych pokazowych dla komponentu Spinner.

```
   public static class SpinnerRow { // each row consists of [caption, icon]
      private String caption;
      private int icon;

      public SpinnerRow(String caption, int icon) {
         this.caption = caption;  this.icon = icon;
      }

      public String getCaption() {
         return this.caption;
      }

      public int getIcon() {
         return this.icon;
      }

      @Override
      public String toString() {
         return caption;
      }
   }//SpinnerRow

}
```

Przykład 8 – Nawigacja

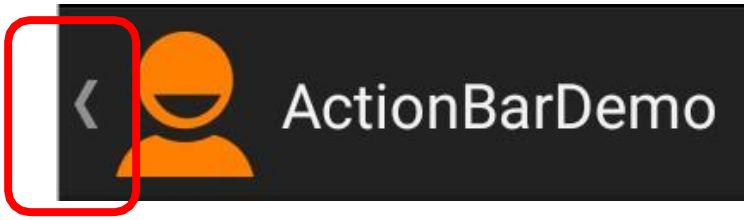
W tym przykładzie przedstawiono nawigację poprzez przyciski **WSTECZ** i **W GÓRĘ**.



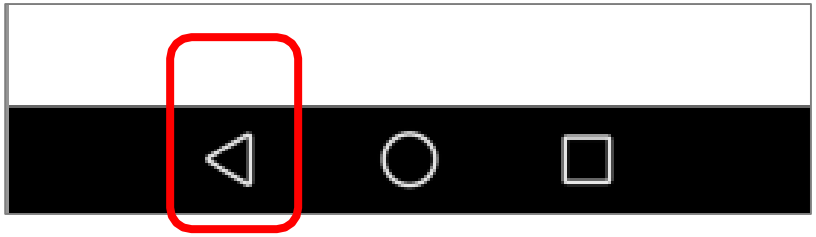
Przykład 8 – Nawigacja

W tym przykładzie zakłada się, że interfejs **MainActivity** może zostać tymczasowo przysłonięty przez inną aktywność (**GreenActivity**) lub może być jedynie częściowo przysłonięty przez fragmenty typu **RedFragments**.

Nawigacja między wymienionymi elementami odbywać się będzie przez wybór klawisza **WSTECZ** lub **W GÓRĘ.**



W GÓRĘ (UP-KEY)



WSTECZ (BACK-KEY)

{Przycisk **W GÓRĘ** używany jest by przekierować widok na dowolny inny element hierarchii. Przycisk **WSTECZ** zapewnia nawigację o jeden element wcześniej zgodnie z ich kolejnością wyświetlania.

Przykład 8 – Nawigacja

IDEA DZIAŁANIA PRZYKŁADU

1. Po kliknięciu na przycisk ‘Red Fragment’ główna aktywność pokazuje fragment RedFragment (Można wówczas kliknąć przycisk ‘Get Date’ by zmienić stan i dane pokazywane przez fragment).

2. Drugi fragment (redFragment) jest tworzony i umieszczany nad pierwszym.

3. Kliknięcie przycisku WSTECZ usuwa aktualny widok (drugi fragment) zmieniając stan aplikacji by pokazywała tylko jeden fragment.

4. Kliknięcie przycisku WSTECZ jeszcze raz spowoduje powrót do głównej aktywności.

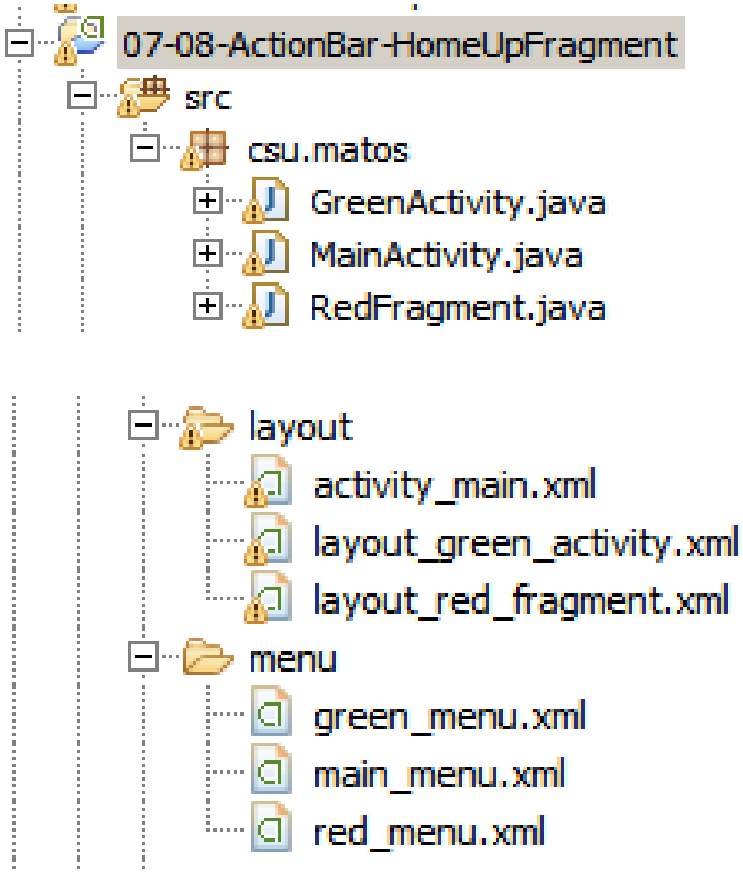
5. Kliknięcie przycisku W GÓRĘ powoduje od razu przejście do poprzedniego elementu w hierarchii widoków (w tym przypadku jest to główna aktywność).

6. Wykorzystywana jest intencja by przywołać GreenActivity, która staje się aktywną i widoczną..

7. Plik AndroidManifest posiada wpis mówiący, że MainActivity jest rodzicem dla GreenActivity. Zatem wykorzystanie przycisku W GÓRĘ powoduje przejście do aktywności rodzica (wyżej w hierarchii).

Przykład 8 – Nawigacja

STRUKTURA APLIKACJI

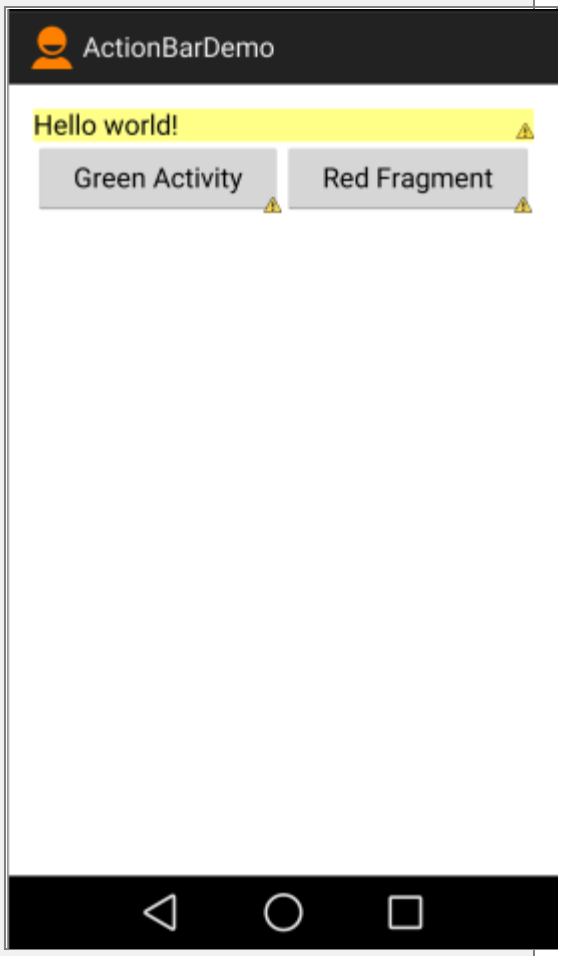


Układy dla MainActivity, GreenActivity i RedFragment

Każdy z tych komponentów będzie miało swoje menu wizualizowane na ActionBar

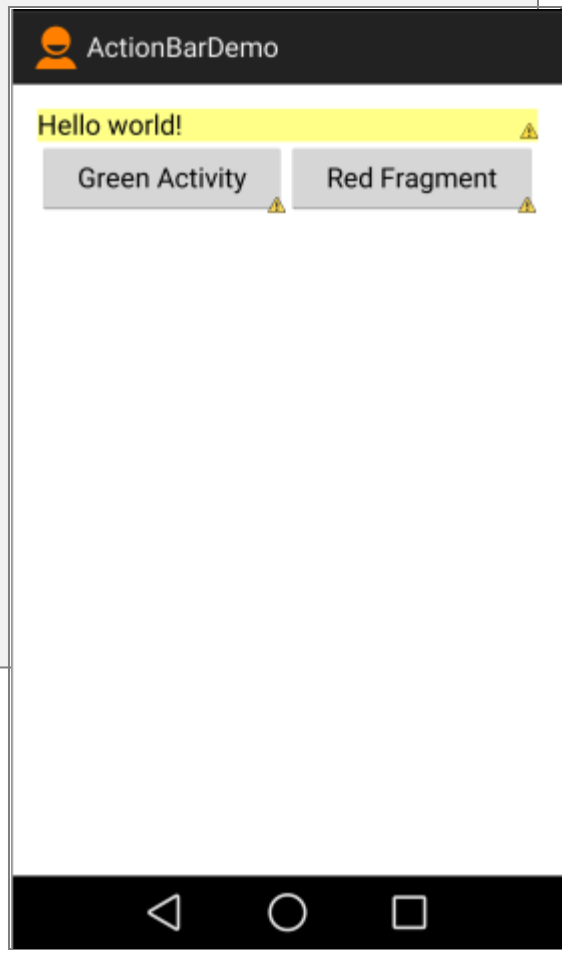
Przykład 8 – UKŁAD: activity_main.xml


```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="csu.matos.MainActivity" >

    <EditText
        android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#77ffff00"
        android:text="@string/hello_world" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/btnGreenActivity"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Green Activity" />
```



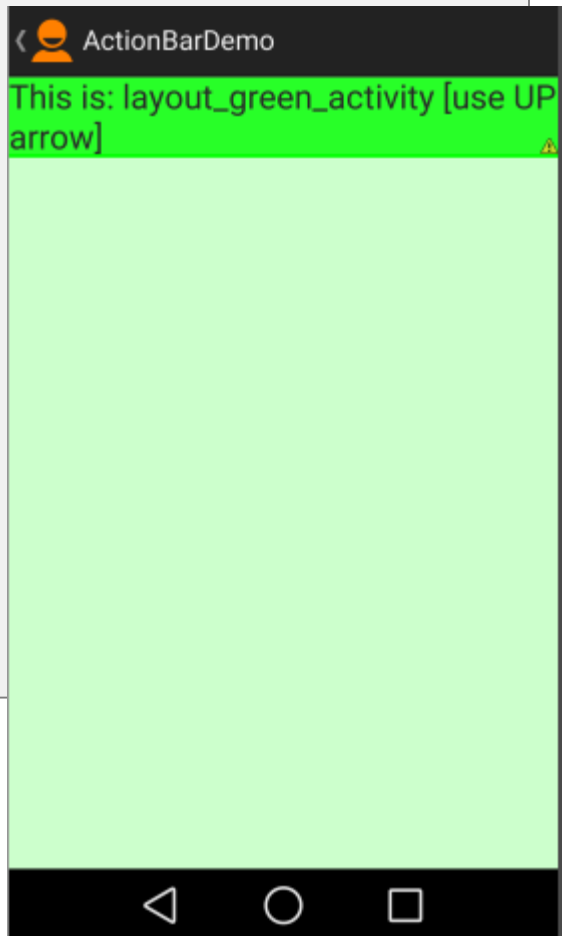
Przykład 8 – UKŁAD: activity_main.xml


```
        <Button
            android:id="@+id/btnRedFragment"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Red Fragment" />
    </LinearLayout>

    <FrameLayout
        android:id="@+id/frag_container_inside_activity_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="6dp"
        android:layout_weight="2" />

</LinearLayout>
```



Przykład 8 – UKŁAD: layout_green_activity.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#3300ff00"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textViewGreen1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff00ff00"
        android:text=
          "This is: layout_green_activity [use UP arrow]"
        android:textAppearance=
          "?android:attr/textAppearanceLarge" />

</LinearLayout>
```

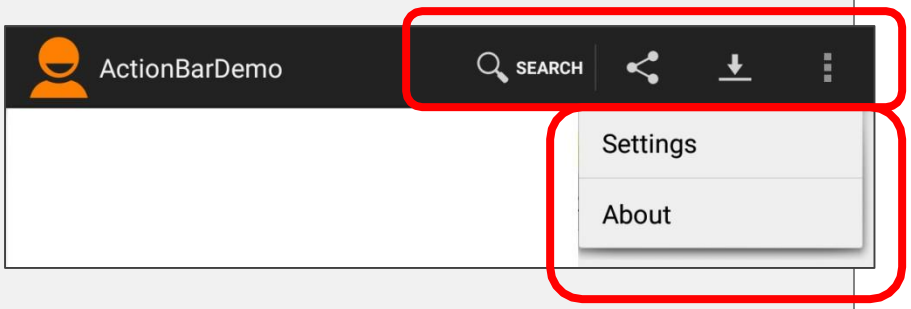


Przykład 8 – UKŁAD: layout_red_fragment.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#22ff0000"
    android:layout_margin="6dp"
    android:orientation="vertical" >

     <View
         android:background="#ffff0000"
         android:layout_width="match_parent"
         android:layout_height="20dp" />
    <Button
         android:id="@+id/btn_date_red_fragment"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Get Date" />

    <EditText
         android:id="@+id/txtMsgFragmentRed"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:textSize="30sp"
         android:ems="10"  />
```

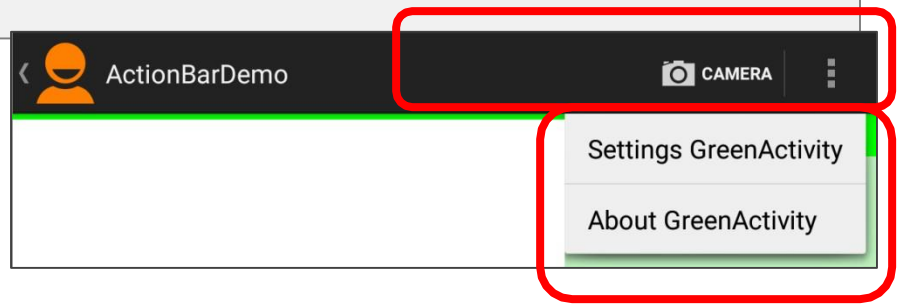
Przykład 8 – MENU: main_menu.xml

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>
</menu>
```



Przykład 8 – MENU: green_menu.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_camera_green_activity"
        android:icon="@drawable/ic_action_camera"
        android:orderInCategory="100"
        android:showAsAction="ifRoom|withText"
        android:title="Camera"/>
    <item
        android:id="@+id/action_settings_green_activity"
        android:icon="@drawable/ic_launcher"
        android:orderInCategory="120"
        android:showAsAction="never"
        android:title="Settings GreenActivity"/>
    <item
        android:id="@+id/action_about_green_activity"
        android:icon="@drawable/ic_launcher"
        android:orderInCategory="140"
        android:showAsAction="never"
        android:title="About GreenActivity"/>
</menu>
```



Przykład 8 – MENU: red_menu.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/action_delete_red_frag"
        android:icon="@drawable/ic_action_delete"
        android:orderInCategory="100"
        android:showAsAction="ifRoom|withText"
        android:title="DeleteFrag"/>

    <item
        android:id="@+id/settings_red_frag"
        android:orderInCategory="120"
        android:showAsAction="never"
        android:title="Settings_Red_Frag"/>

    <item
        android:id="@+id/about_red_frag"
        android:orderInCategory="140"
        android:showAsAction="never"
        android:title="About_Red_Frag"/>

</menu>
```

Przykład 8 – AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="csu.matos" android:versionCode="1“  android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="16"
        android:targetSdkVersion="21" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:windowSoftInputMode="stateHidden" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name=".GreenActivity"
            android:label="@string/app_name"
            android:parentActivityName="MainActivity" >
        </activity>

    </application>

</manifest>
```

Przykład 8 – MainActivity.java


```
public class MainActivity extends Activity implements OnClickListener {

   EditText txtMsg;
   Button btnGreenActivity;
   Button btnRedFragment;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      txtMsg = (EditText) findViewById(R.id.txtMsg);
      btnGreenActivity = (Button) findViewById(R.id.btnGreenActivity);
      btnRedFragment = (Button) findViewById(R.id.btnRedFragment);
      btnGreenActivity.setOnClickListener(this);
      btnRedFragment.setOnClickListener(this);

   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; add items to the action bar
      getMenuInflater().inflate(R.menu.main_menu, menu);
      return true;
   }
```

Przykład 8 – MainActivity.java


```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    // Handle action bar item clicks here.
    int id = item.getItemId();

    if (id == R.id.action_search) {
        txtMsg.setText("Search...");
        return true;
    } else if (id == R.id.action_share) {
        txtMsg.setText("Share...");
        return true;
    } else if (id == R.id.action_download) {
        txtMsg.setText("Download...");
        return true;
    } else if (id == R.id.action_about) {
        txtMsg.setText("About...");
        return true;
    } else if (id == R.id.action_settings) {
        txtMsg.setText("Settings...");
        return true;
    }
    return false;
}
```

1

Przykład 8 – MainActivity.java


```
  // "GreenActivity" button used to invoke a supporting Activity called via
  // Intent while "RedFragment" button replaces the MainActivity's UI with
  // a new instance of a "FragmentRed"
  @Override
  public void onClick(View v) {

    if (v.getId() == R.id.btnGreenActivity) {
      Intent greenActivityIntent = new Intent(MainActivity.this,
                                              GreenActivity.class);
      // if needed put data items inside the bundle
      Bundle datainfo = new Bundle();
      greenActivityIntent.putExtra("data", datainfo);
      startActivityForResult(greenActivityIntent, 0);
    } else

    if( v.getId() == R.id.btnRedFragment ){
      // create a new RED fragment - show it
      FragmentTransaction ft = getFragmentManager().beginTransaction();
      RedFragment fragmentRed = RedFragment.newInstance("new-red-frag-arg1");
      ft.replace(R.id.frag_container_inside_activity_main,fragmentRed,"red_frag");
      ft.addToBackStack("red_tran");  //allows BackButton pop-navigation
      ft.commit();
    }

  }//onClick

}//MainActivity
```

2

3

Przykład 8 – MainActivity.java

Komentarz

1. W tym przykładzie każda aktywność i fragment posiada indywidualne menu. Aktywność główna tworzy ActionBar na podstawie specyfikacji **main_menu.xml**. Menu zawiera przede wszystkim opcje do ściągania, udostępniania i wyszukiwania.

2. Gdy użytkownik kliknie na przycisk ‘Green Activity’ wysyłana jest intencja by wywołać ekran GreenActivity. Wraz z intencją mogą być przesyłane dane dodatkowe. Metoda ‘*startActivityForResults*(…)’ umożliwia monitorowanie wartości zwróconych przez Green Activity.

3. Kliknięcie na przycisk ‘Red Fragment’ powoduje stworzenie nowej instancji klasy RedFragment. Metoda *.replace*(…) usuwa jakikolwiek poprzedni widok znajdujący się w określonym miejscu MainActivity na ten nowostworzony. Informacje o przeprowadzonej transakcji dodawane są do **BackStack** w celu łatwiejszego przywrócenia poprzedniego stanu aplikacji.

Przykład 8 – GreenActivity.java


```
public class GreenActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.layout_green_activity);

    // enable the home button
    getActionBar().setDisplayHomeAsUpEnabled(true);
    getActionBar().setHomeButtonEnabled(true);

  }//onCreate


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.green_menu, menu);
      return true;

    }
```

Przykład 8 – GreenActivity.java


```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    switch (item.getItemId()) {

        case android.R.id.home:
        // the user pressed the UP button - where to go now? look for
        // Manifest's entry: android:parentActivityName="MainActivity"
            if (getParentActivityIntent() == null) {
                Log.i("ActivityGreen",
                      "Fix Manifest to indicate the parentActivityName");
                onBackPressed(); //terminate the app

            } else {
                NavUtils.navigateUpFromSameTask(this); //back to parent activity
            }
            return true;

        default:
            return super.onOptionsItemSelected(item);
    }
 }
}
```

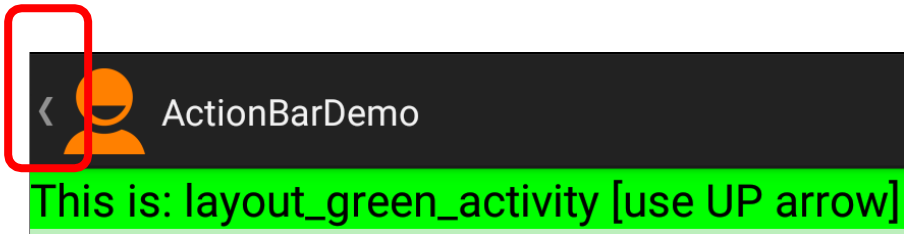
1

Przykład 8 – GreenActivity.java

Komentarz

1. Powiązanie hierarchii aktywności obywa się poprzez odpowiednią zmianę w pliku AndroidManifest.xml:

```
…
<activity
    android:name=".GreenActivity“
    android:label="@string/app_name“
    android:parentActivityName="MainActivity" >
</activity>
…
```



Pozycja menu **`android.R.id.home`** reprezentuje kliknięcie przycisku W GÓRĘ. Pierwszym krokiem jest sprawdzenie, czy plik manifestu zawiera odwołanie do aktywności nadrzędnej. Jeżeli takie odwołanie nie istnieje, wywoływana jest metoda *onBackPressed*() by cofnąć się w hierarchii widoków. W przeciwnym przypadku, wykorzystywana jest klasa **NavUtils** by przejść do wskazanego miejsca w hierarchii.

Przykład 8 – RedFragment.java


```
public class RedFragment extends Fragment {

   TextView txtMsgFragmentRed =  null;

   public static RedFragment newInstance(String strArg) {
      RedFragment fragmentRed = new RedFragment();
      Bundle args = new Bundle();
      args.putString("strArg1", strArg);
      fragmentRed.setArguments(args);
      return fragmentRed;
   }

   @Override
   public void onCreate(Bundle arg0) {
      super.onCreate(arg0);
      setHasOptionsMenu(true);

      // enable the home button
      getActivity().getActionBar().setDisplayHomeAsUpEnabled(true);
      getActivity().getActionBar().setHomeButtonEnabled(true);
   }

   @Override
   public View onCreateView(LayoutInflater inflater,
                            ViewGroup container,
                            Bundle savedInstanceState) {

      // inflate layout_blue.xml holding a TextView and a ListView
      LinearLayout redLayout = (LinearLayout) inflater.inflate(
                                  R.layout.layout_red_fragment, null );
```

1

Przykład 8 – RedFragment.java


```
        // plumbing - get a reference to textview and listview
        txtMsgFragmentRed = (TextView) redLayout.findViewById(R.id.txtMsgFragmentRed);

        final Button button = (Button)redLayout.findViewById(R.id.btn_date_red_fragment);
        button.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                String text = (new Date()).toString();
                txtMsgFragmentRed.setText(text);
            }
        });
        return redLayout;
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        menu.clear();
        inflater.inflate(R.menu.red_menu, menu);

    }
     @Override
     public boolean onOptionsItemSelected(MenuItem item) {
         // Handle action bar item clicks here.
         int id = item.getItemId();

         if (id == android.R.id.home) {
2         clearBackStack();           //TRY: jump to MainActivity (HigAncestor)
          //showPreviousRedScreen(); //TRY: same as pressing Back button
         }else {
```

Przykład 8 – RedFragment.java


```
            //for now -just show the action-id
            txtMsgFragmentRed.setText("ACTION_ID="+id);
           }

           return super.onOptionsItemSelected(item);
        }

       private void clearBackStack() {
          try {
             FragmentTransaction ft = getFragmentManager().beginTransaction();

             android.app.FragmentManager fragmentManager = getFragmentManager();

             fragmentManager.popBackStackImmediate(null,
                                FragmentManager.POP_BACK_STACK_INCLUSIVE);

              ft.commit();

          } catch (Exception e) {
             Log.e("CLEAR-STACK>>> ", e.getMessage() );

          }

       }//clearBackStack()
```

3

Przykład 8 – RedFragment.java


```
    private void showPreviousRedScreen() {
      try {
        FragmentTransaction ft = getFragmentManager().beginTransaction();
        android.app.FragmentManager fragmentManager = getFragmentManager();

        //determine the size n of the BackStack (0,1,..n-1)
        int bsCount = fragmentManager.getBackStackEntryCount();

        //see (without removing) the stack's top entry
        BackStackEntry topEntry = fragmentManager.getBackStackEntryAt(bsCount-1);

        //obtain the numeric ID and name tag of the top entry
        String tag = topEntry.getName();
        int id = topEntry.getId();
        Log.e("RED Top Fragment name: ", "" + tag + " " + id);

        //pop the top entry (until matching id) and reset UI with its state data
        //fragmentManager.popBackStackImmediate(id, 1);
        fragmentManager.popBackStackImmediate();

        ft.commit();

      } catch (Exception e) {
          Log.e("REMOVE>>> ", e.getMessage() );
      }

    }//showPreviousRedScreen

}
```

4

Przykład 8 – Nawigacja

Komentarz

1. Po przysłonięciu aktywności przez RedFragment możliwe jest wykorzystanie przycisku W GÓRĘ.

2. Prezentowany przykład umożliwia dwa typy nawigacji: historyczną oraz względem przodka.

3. **Nawigacja względem przodka**. Metoda ***clearBackStack***() jest wywoływana by usunąć wszystkie wpisy w stosie BackStack dotyczące RedFragment. Możliwy jest powrót do „czystej” aktywności.

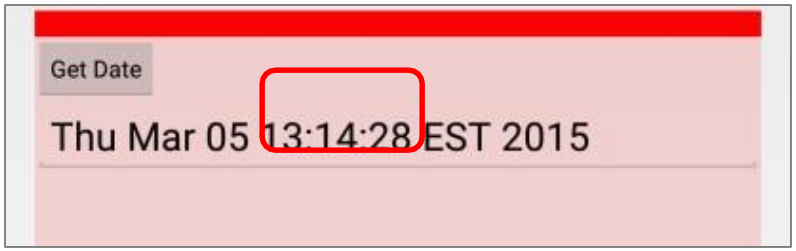
   Wywołanie .*popBackStackImmediate*(*tag, flag*) wydobywa wszystkie wpisy z BackStack aż do napotkania określonego tag’a.

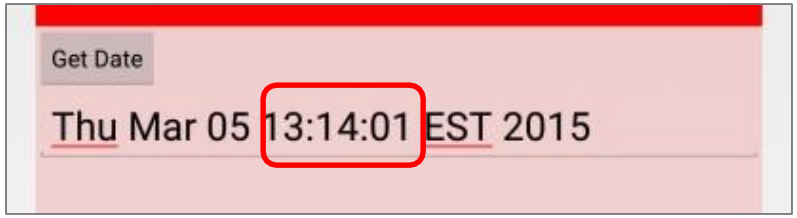
   Jeżeli ustawiono flagę **POP_BACK_STACK_INCLUSIVE** wszystkie elementy zostaną wydobyte ze stosu aż do trafienia na poszukiwany. W przeciwnym przypadku wszystkie wpisy aż do poszukiwanego (bez niego) będą usunięte.

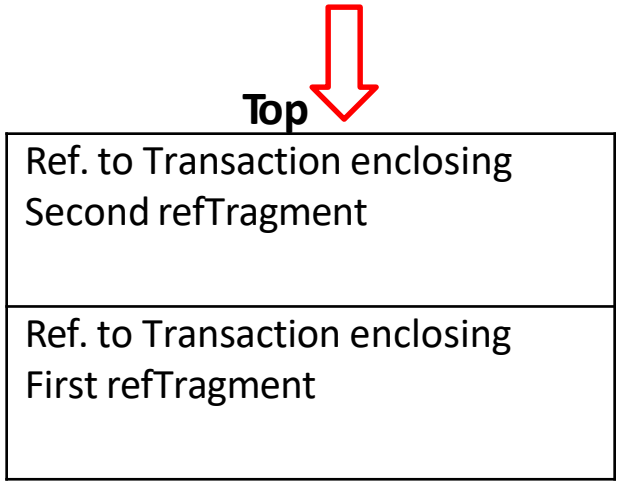
Komentarz

4. **Nawigacja historyczna.** Metoda własna *showPreviousRedScreen*() pokazuje jak można wymusić zachowanie tożsame jak przy wciśnięciu przycisku W GÓRĘ.

   Tranzycja do poprzedzającego obiektu RedFragment (jeżeli jakikolwiek jest dostępny) odbywa się poprzez wywołanie metody **`.popBackStackImmediate()`** która pobiera i wyświetla ostatni stan fragmentu na stosie BackStack.





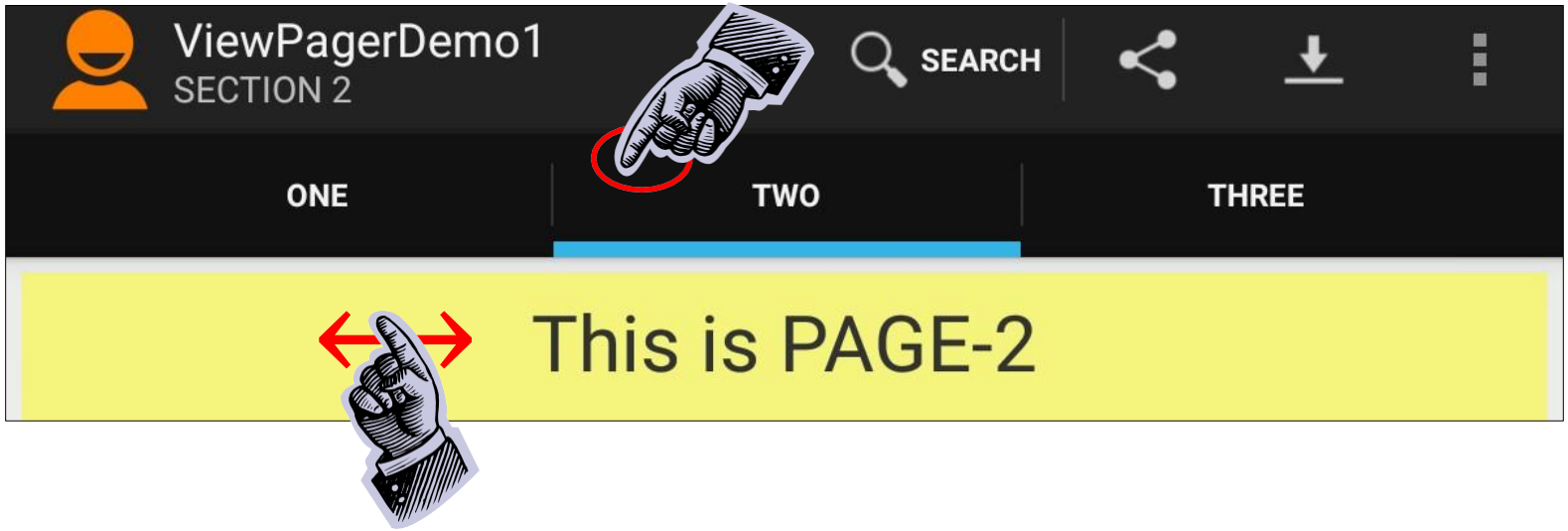


Przykład 9 – Wykorzystanie zakładek: ViewPager

Przesunięcie palcem zakładek jest przykładem tzw. **nawigacji bocznej** w której nowe ekrany pokazywane są na ekranie przez ich przeciągnięcie z końca obszaru roboczego. Widżet **ViewPager** jest jednym z komponentów, który umożliwia praktyczną implementację przesuwania zakładek w poziomie.

Strony zakładek – zwykle reprezentowane przez fragmenty – są generowane przez własny **PagerAdapter**.

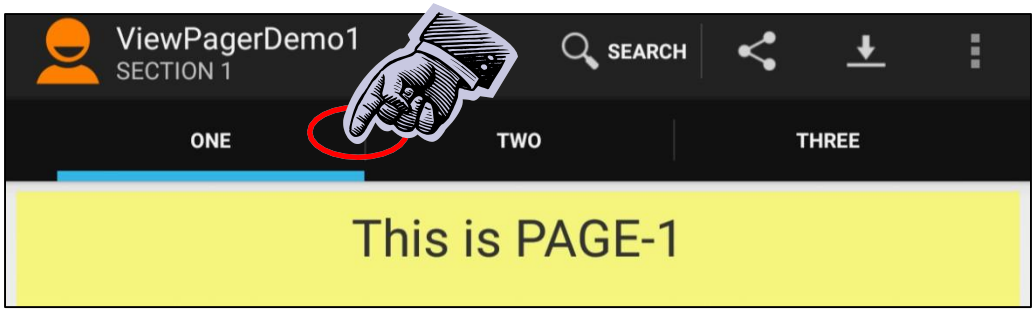
Funkcjonalność ViewPager jest często powiązany z komponentem **ActionBar** lub **TabStrip** by zapewnić nawigację opartą na zakładkach.

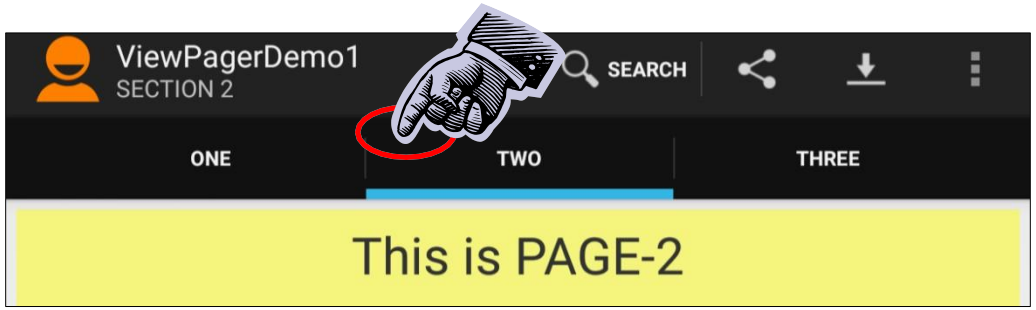


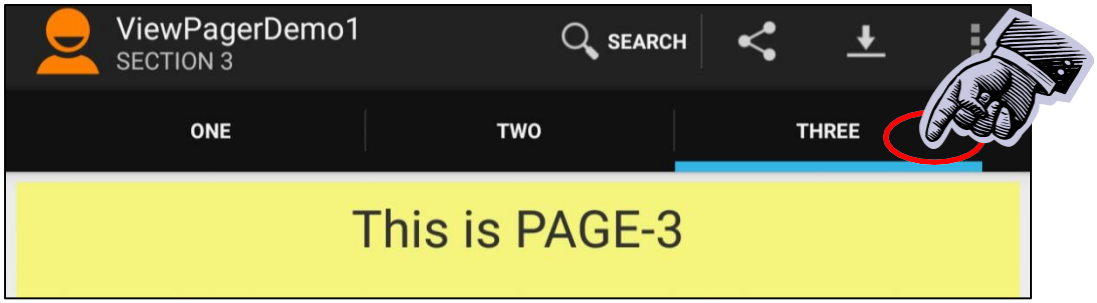
Przykład 9 – Wykorzystanie zakładek: ViewPager

W tym przykładzie komponent **ViewPager** zostanie powiązany z elementem **ActionBar** tworzonej aplikacji.

Jednakże od API-21 takie rozwiązanie ma status deprecated i zostanie wykorzystane jedynie w celu prezentacji ogólnej idei.





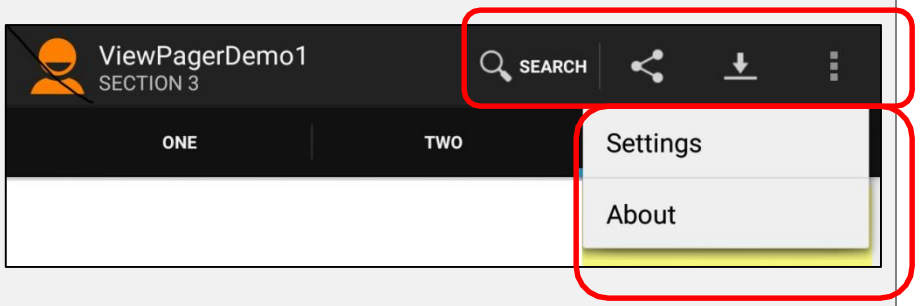


Zakładki są częścią **ActionBar**.

Kliknięcie na daną zakładkę powoduje stworzenie animacji symulującej poziomy ruch przewijania.

Przykład 9 – MENU: main_menu.xml

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>
</menu>
```



Przykład 9 – UKŁADY XML

activity_main.xml

```
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="6dp"
    tools:context="csu.matos.MainActivity" />
```

fragment_page_layout.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="6dp"
    android:background="#77ffff00"
    tools:context="csu.matos.MainActivity$PlaceholderFragment" >

    <TextView
        android:id="@+id/section_label"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Hola"
        android:textSize="30sp" />

</LinearLayout>
```

Przykład 9 – MainActivity.java


```
public class MainActivity extends Activity implements TabListener {
  SectionsPagerAdapter mSectionsPagerAdapter;
  ViewPager mViewPager;
  ActionBar actionBar;
  Context context;
  int duration = Toast.LENGTH_SHORT;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    context = MainActivity.this;

    // adapter returns fragments representing pages added to the ViewPager
    mSectionsPagerAdapter = new SectionsPagerAdapter(getFragmentManager());
1   mViewPager = (ViewPager) findViewById(R.id.pager);
    mViewPager.setAdapter(mSectionsPagerAdapter);

    actionBar = getActionBar();
2   actionBar.addTab(actionBar.newTab().setText("ONE").setTabListener(this));
    actionBar.addTab(actionBar.newTab().setText("TWO").setTabListener(this));
    actionBar.addTab(actionBar.newTab().setText("THREE").setTabListener(this));
    actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

    actionBar.setDisplayShowHomeEnabled(true);
    actionBar.setDisplayShowTitleEnabled(true);
```

Przykład 9 – MainActivity.java

3

```
    mViewPager.setOnPageChangeListener(new OnPageChangeListener() {
    //this listener reacts to the changing of pages
      @Override
      public void onPageSelected(int position) {
        actionBar.setSelectedNavigationItem(position);
        actionBar.setSubtitle(mSectionsPagerAdapter.getPageTitle(position));
      }

      @Override
      public void onPageScrolled(int position, float positionOffset,
                                 int positionOffsetPixels) {
      }

      @Override
      public void onPageScrollStateChanged(int state) {
      }
    });
  }//onCreate

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
  }
```

Przykład 9 – MainActivity.java


```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  // Handle action bar item clicks here.
  int id = item.getItemId();

  if (id == R.id.action_search) {
    Toast.makeText(context, "Search...", duration).show();
    return true;
  }
  else if (id == R.id.action_share) {
    Toast.makeText(context, "Share...", duration).show();
    return true;
  }
  else if (id == R.id.action_download) {
    Toast.makeText(context, "Download...", duration).show();
    return true;
  }
  else if (id == R.id.action_about) {
    Toast.makeText(context, "About...", duration).show();
    return true;
  }
  else if (id == R.id.action_settings) {
    Toast.makeText(context, "Settings...", duration).show();
    return true;
  }
  return false;
}
```

4

Przykład 9 – MainActivity.java


```
public class SectionsPagerAdapter extends FragmentPagerAdapter {

  public SectionsPagerAdapter(FragmentManager fm) {
    super(fm);
  }

  @Override
  public Fragment getItem(int position) {
5   return PlaceholderFragment.newInstance(position + 1); //return a fragment
  }

  @Override
  public int getCount() {
    return 3; // Show 3 total pages.
  }

  @Override
  public CharSequence getPageTitle(int position) {
    Locale locale = Locale.getDefault();
    switch (position) {
    case 0:   return getString(R.string.title_section1).toUpperCase(locale);
    case 1:   return getString(R.string.title_section2).toUpperCase(locale);
    case 2:   return getString(R.string.title_section3).toUpperCase(locale);
    }
    return null;
  }
}//SectionsPagerAdapter
```

Przykład 9 – MainActivity.java


```
public static class PlaceholderFragment extends Fragment {

  private static final String ARG_SECTION_NUMBER = "section_number";

  public static PlaceholderFragment newInstance(int sectionNumber)
  {
    PlaceholderFragment fragment = new PlaceholderFragment();
    Bundle args = new Bundle();
    args.putInt(ARG_SECTION_NUMBER, sectionNumber);
    fragment.setArguments(args);
    return fragment;
  }
  public PlaceholderFragment() { }

  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
    int pagePosition = getArguments().getInt(ARG_SECTION_NUMBER, -1);
    View rootView = inflater.inflate( R.layout.fragment_main, container,
    false);  TextView txtMsg = (TextView)
    rootView.findViewById(R.id.section_label);  String text = "This is PAGE-" +
    pagePosition;
    txtMsg.setText(text);
    return rootView;
  }
}//PlaceHolderFragment
```

6

7

Przykład 9 – MainActivity.java


```
  // Implementing ActionBar TAB listener
  @Override
  public void onTabSelected(Tab tab, FragmentTransaction ft) {
    // move to page selected by clicking a TAB
8   mViewPager.setCurrentItem(tab.getPosition());
  }

  @Override
  public void onTabUnselected(Tab tab, FragmentTransaction ft) {
    // TODO: nothing - needed by the interface

  }

  @Override
  public void onTabReselected(Tab tab, FragmentTransaction ft) {
    // TODO: nothing - needed by the interface


  }

}
```

Przykład 9 – Wykorzystanie zakładek: ViewPager

Komentarz

1. Widżet ViewPager (zdefiniowany w układzie **activity_main.xml** jako **pager**), będzie zajmował całą przestrzeń roboczą. Własny **SectionsPagerAdapter** musi zostać zdefiniowany by pokazać osobne strony w ramach kontenera **pager**.

2. Po uzyskaniu dostępu do **ActionBar'a**, dodawane są trzy zakładki o nazwach “ONE”, “TWO” i “THREE”. Klauzula `setNavigationMode()` jest wykorzystana do aktywacji nawigacji po zakładkach (zakładki nie są pokazywane w przypadku pominięcia tej klauzuli).

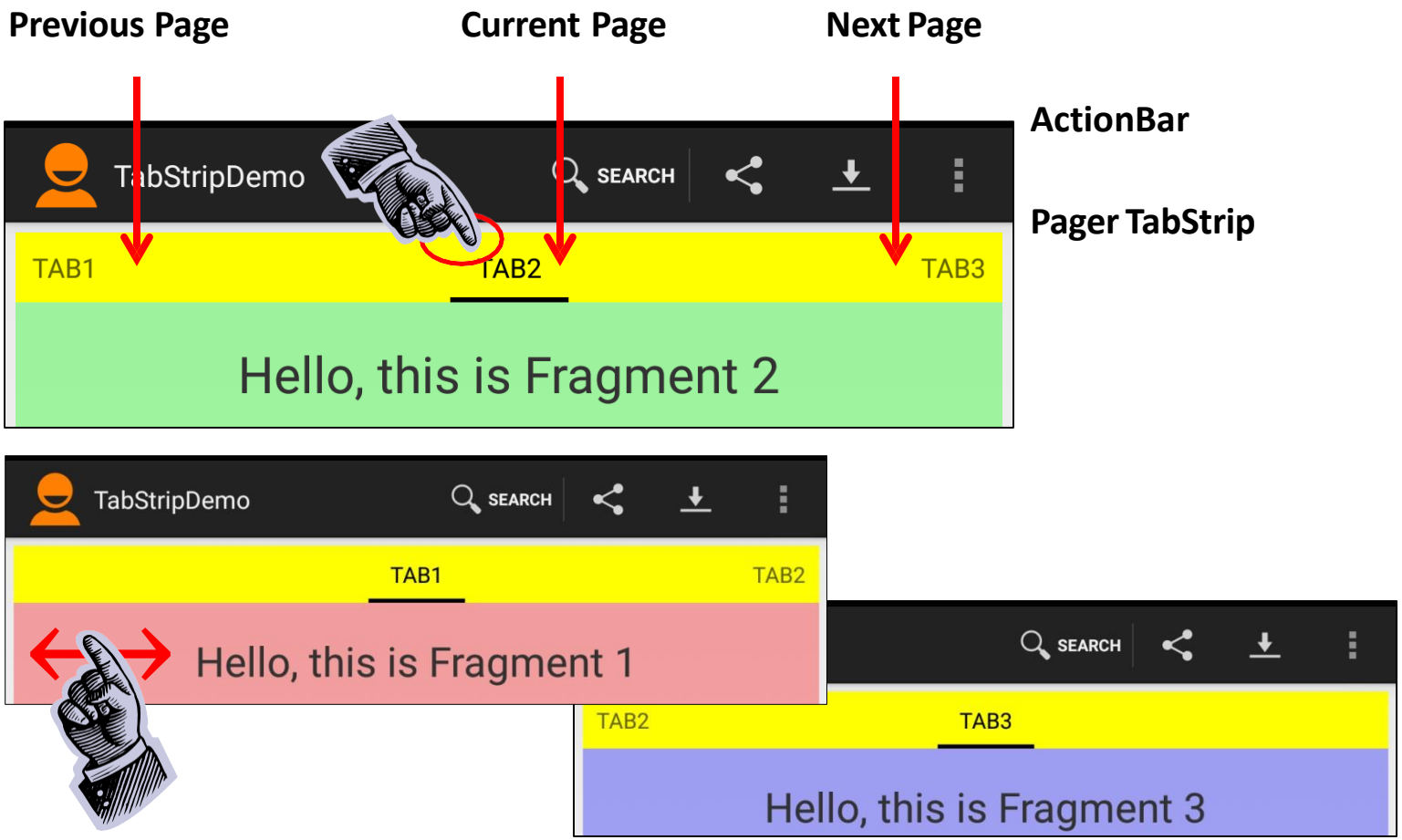
3. Nasłuchiwacz `onPageChangeListener` reaguje na zmianę zakładek. Wówczas zmieniana jest również prezentowana treść zgodnie z wybraną zakładką.

4. Prócz zakładek ActionBar prezentuje również wszystkie komponenty zdefiniowane w pliku **res/menu/main.xml**. Metoda `onOptionsMenuItemSelected` odpowiedzialna jest za wykonanie akcji powiązanej z daną pozycją menu.

5. Przy każdej zmianie zakładki nowy fragment jest tworzony by wypełnić aktualny widok. Metoda `getItem()` zwraca fragment dla odpowiedniej wybranej pozycji.

Przykład 10 – ViewPager oraz TabStrip

Widżet **PagerTabStrip** jest poziomym paskiem mogącym prezentować do 3 zakładek. Zakładki powiązane są z *aktualną, następną* i *poprzednią* stroną komponentu ViewPager. Każda zakładka składa się z *napisu i wskaźnika wyboru (podkreślenia)*. Wskaźnik aktualnie wybranej zakładki znajduje się pod jej nazwą.



Przykład 10 – ViewPager oraz TabStrip

UKŁAD: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="6dp" >

    <android.support.v4.view.PagerTabStrip
        android:id="@+id/pager_tab_strip"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:background="#FFFFFF00"
        android:padding="10dp" />

</android.support.v4.view.ViewPager>
```

1 → xmlns:android="http://schemas.android.com/apk/res/android"

2 → <android.support.v4.view.PagerTabStrip

- Komponenty **ViewPager** oraz **TabStrip** zajmują cały ekran.
- Określone strony są osobnymi fragmentami tworzonymi na podstawie ich specyfikacji XML.

Przykład 10 – ViewPager oraz TabStrip

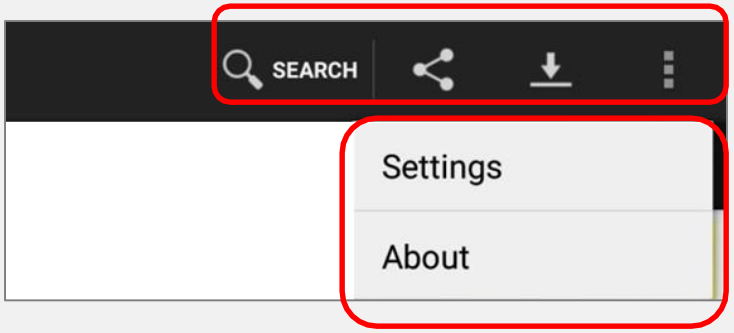
UKŁAD: fragment_page1.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="22dp"
        android:textSize="30sp"
        android:text="Hello, this is Fragment 1" />

</RelativeLayout>
```

Dwa pozostałe układy: **fragment_page2** oraz **fragment_page3** są podobne, różnią się tylko etykietą i kolorem tła.

Przykład 10 – MENU: main_menu.xml

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>
</menu>
```



The app's menu is placed on the ActionBar. Keep in mind that the Tabs shown by the PagerTabStrips widget are NOT connected with the ActionBar.

Przykład 10 – MainActivity.java


```
public class MainActivity extends FragmentActivity {
   Context context;
   int duration = Toast.LENGTH_SHORT;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      // Get the view from activity_main.xml
      setContentView(R.layout.activity_main);
      context = getApplication();

      // Locate the viewpager in activity_main.xml
      ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);

1 →   // Set the ViewPagerAdapter into ViewPager
      viewPager.setAdapter(new MyViewPagerAdapter(getSupportFragmentManager()));

   }//onCreate

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is present.
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }
```

Przykład 10 – MainActivity.java

2

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   int id = item.getItemId();

   if (id == R.id.action_search) {
      Toast.makeText(context, "Search...", duration).show();
      return true;
   }
   else if (id == R.id.action_share) {
      Toast.makeText(context, "Share...", duration).show();
      return true;
   }
   else if (id == R.id.action_download) {
      Toast.makeText(context, "Download...", duration).show();
      return true;
   }
   else if (id == R.id.action_about) {
      Toast.makeText(context, "About...", duration).show();
      return true;
   }
   else if (id == R.id.action_settings) {
      Toast.makeText(context, "Settings...", duration).show();
      return true;
   }
   return false;
}
```

Przykład 10 – FragmentPage1.java

```
public class FragmentPage1 extends Fragment {

   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {

      // Get the view design from file: res/layout/fragment_page1.xml
      View view = inflater.inflate(R.layout.fragment_page1, container, false);

      view.setBackgroundColor(Color.parseColor("#55FF0000"));

      return view;
   }

}
```

3

Dwa pozostałe fragmenty: **FragmentPage2** i **FragmentPage3** są podobne.

Przykład 10 – MiViewPagerAdapter.java

```
public class MyViewPagerAdapter extends FragmentPagerAdapter {
   // Tab Captions
   private String tabCaption[] = new String[] { "TAB1", "TAB2", "TAB3" };

   public MyViewPagerAdapter(FragmentManager fragmentManager) {
      super(fragmentManager);
   }

   @Override
   public int getCount() {
      return tabCaption.length;  // return 3 (numbers of tabs in the example)
   }
3 →
   @Override
   public Fragment getItem(int position) {
      switch (position) {
      case 0:  return new FragmentPage1();
      case 1:  return new FragmentPage2();
      case 2:  return new FragmentPage3();
      }
      return null;
   }

   @Override
   public CharSequence getPageTitle(int position) {
      return tabCaption[position];    // return tab caption
   }
}
```

Przykład 10 – ViewPager oraz TabStrip

Komentarz

1. Aplikacja składa się z trzech niezależnych części: ActionBar na górze, PagerTabStrip oraz ViewPager. Komponent ViewPager zajmuje najwięcej obszaru roboczego. Wymaga stworzenia własnego adaptera by móc wyświetlić szereg stron. Elementem bazowym każdej strony jest określony fragment.

2. ActionBar tworzony jest na podstawie specyfikacji zawartej w **res/menu/main.xml**. Nasłuchiwacz **onOptionsMenuItemSelected** rejestruje zdarzenia kliknięcia oraz wykonuje powiązane z nimi operacje. Należy zaznaczyć, że ActionBar oraz zakładki PagerTabStrips to dwa niezależne komponenty.

3. **MyViewPageAdapter** jest odpowiedzialny za stworzenie etykiety oraz zawartości obiektów ViewPagerviews. Dwie najważniejsze metody w ramach tej klasy to: `getItem()` oraz `getPageTitle()`. Obie jako parametr przyjmują określny indeks (0, 1, 2,…) i zwracają odpowiednio stronę lub etykietę strony.

Przykład 11 – Pasek narzędziowy

Komponent typu **Toolbar** jest nową kontrolką wprowadzoną w SDK 5.0. Funkcjonalnie stanowi odpowiednik ActionBar’a, posiadając jednak kilka unikalnych funkcji niedostępnych w starszym bracie.

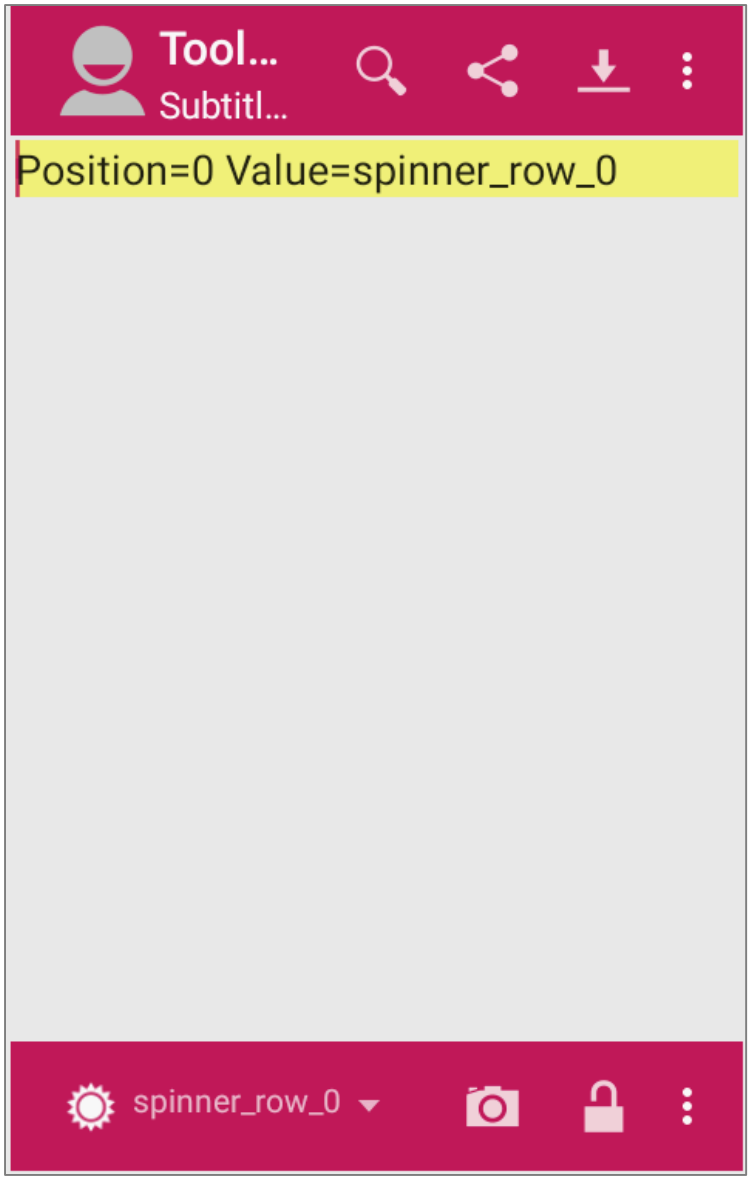
Podobnie jak ActionBar, pasek Toolbar może zawierać przyciski nawigacyjne, logo, tytuł, podtytuł, kafelki oraz dowolny własny widok.

W przeciwieństwie do ActionBar'a – który może być umiejscowiony tylko na górze ekranu – pasek narzędziowy może być umieszczony gdziekolwiek na ekranie. Możliwe jest wykorzystaniu wielu pasków narzędziowych naraz.

Nie sposób wizualnie rozróżnić komponent Toolbar od ActionBar

Przykład 11 – Pasek narzędziowy

Toolbar
Na górze



Toolbar
Na dole

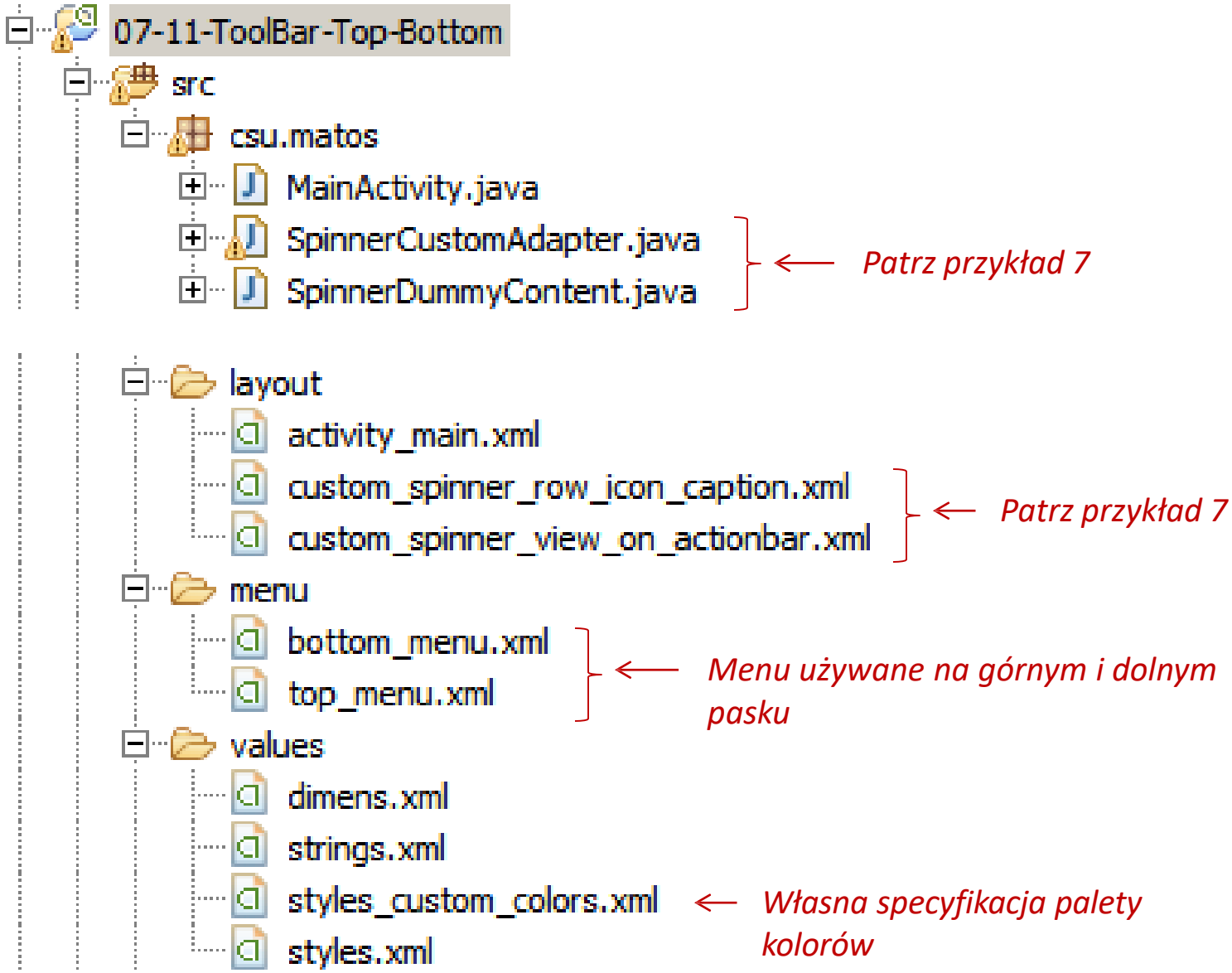
W tym przykładzie aplikacja ma dwa niezależne paski.

Pasek górny zawiera: logo, tytuł, podtytuł, przyciski oraz menu.

Dolny pasek zawiera własny widok (składający się z obiektu Spinner), oraz dodatkowe przyciski.

Każdy pasek narzędziowy może wykorzystywać odrębny styl oraz paletę kolorów.

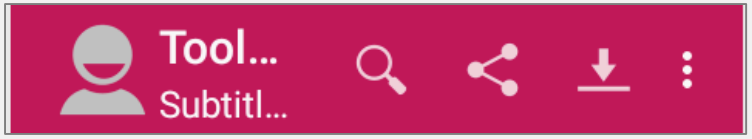
Przykład 11 – Pasek narzędziowy



Przykład 11 – Układ: activity_main.xml


```
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="?android:attr/colorBackground"
    android:orientation="vertical"
    android:padding="2dp" >

    <Toolbar
        android:id="@+id/Toolbar_top"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?android:attr/colorPrimary"
        android:minHeight="?android:attr/actionBarSize"
        android:popupTheme="@android:style/ThemeOverlay.Material.Light"
        android:theme="@android:style/ThemeOverlay.Material.Dark.ActionBar" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:orientation="vertical" >
        <EditText
            android:id="@+id/txtMsg"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="2dp"
            android:background="#77ffff00"
            android:text="@string/hello_world" />
    </LinearLayout>
```

Toolbar górny

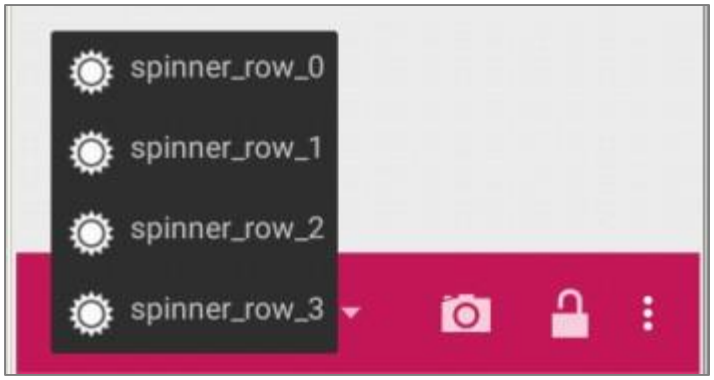


Przykład 11 – Układ: activity_main.xml

Toolbar dolny

```
    <Toolbar
        android:id="@+id/Toolbar_bottom"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?android:attr/colorPrimary"
        android:minHeight="?android:attr/actionBarSize"
        android:popupTheme="@android:style/ThemeOverlay.Material.Light"
        android:theme="@android:style/ThemeOverlay.Material.Dark.ActionBar" />

</LinearLayout>
```

Później obiekt typu Spinner zostanie dodany.



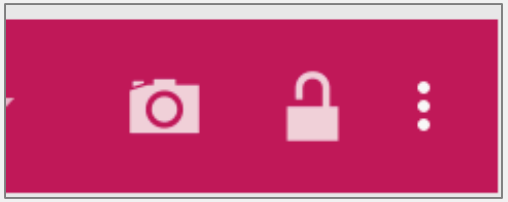
Przykład 11 – MENU: top_menu.xml

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:orderInCategory="120"
        android:showAsAction="always|withText"
        android:title="Search"/>
    <item
        android:id="@+id/action_share"
        android:icon="@drawable/ic_action_share"
        android:orderInCategory="140"
        android:showAsAction="always"
        android:title="Share"/>
    <item
        android:id="@+id/action_download"
        android:icon="@drawable/ic_action_download"
        android:orderInCategory="160"
        android:showAsAction="always"
        android:title="Download"/>
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="180"
        android:showAsAction="never"
        android:title="Settings"/>
    <item
        android:id="@+id/action_about"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="About"/>
</menu>
```

Część z menu jest wyświetlane przez górny pasek

Przykład 11 – MENU: bottom_menu.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/action_camera_bottom"
        android:icon="@drawable/ic_action_camera"
        android:orderInCategory="100"
        android:showAsAction="always"
        android:title="Camera"/>
    <item
        android:id="@+id/action_unlock_bottom"
        android:icon="@drawable/ic_action_unlock"
        android:orderInCategory="110"
        android:showAsAction="always"
        android:title="Unlock"/>
    <item
        android:id="@+id/action_settings_bottom"
        android:icon="@drawable/ic_launcher"
        android:orderInCategory="120"
        android:showAsAction="never"
        android:title="Settings Bottom "/>

</menu>
```



Druga część menu jest wyświetlana na dolnym pasku

Przykład 11 – STYLE: style_custom_colors.xml

Należy zmodyfikować plik **AndroidManifest** by zmienić paletę kolorów. Należy dodać następującą klauzulę do elementu **<application>**: **android:theme="@style/MyPinkTheme**

```
<resources>
<style name="MyPinkTheme" parent="@android:style/Theme.Material.Light.DarkActionBar">
    <!-- Customizing the COLOR PALETTE                                        -->
    <!-- Using PINK theme colors from suggested Android palette. Visit link:  -->
    <!-- http://www.google.com/design/spec/style/color.html#color-color-palette -->
    <!-- Elements of material design described at link:                       -->
    <!-- https://developer.android.com/training/material/theme.html           -->

    <!-- do not show app title       -->
    <item name="android:windowNoTitle">true</item>
    <!-- do not show ActionBar       -->
    <item name="android:windowActionBar">false</item>

    <!-- colorPrimary: PINK700 (Toolbar background) -->
    <item name="android:colorPrimary">#C2185B</item>
    <!-- colorPrimaryDark: PINK900 (status bar)     -->
    <item name="android:colorPrimaryDark">#880E4F</item>
    <!-- colorAccent: PINK700 (UI widgets like: checkboxes, text fields )  -->
    <item name="android:colorAccent">#C2185B</item>

    <!-- colorBackground: PINK100 (window background)  -->
    <item name="android:colorBackground">#F8BBD0</item>

  </style>
</resources>
```

Definicja dodana w **res/values/styles_custom_colors.xml**

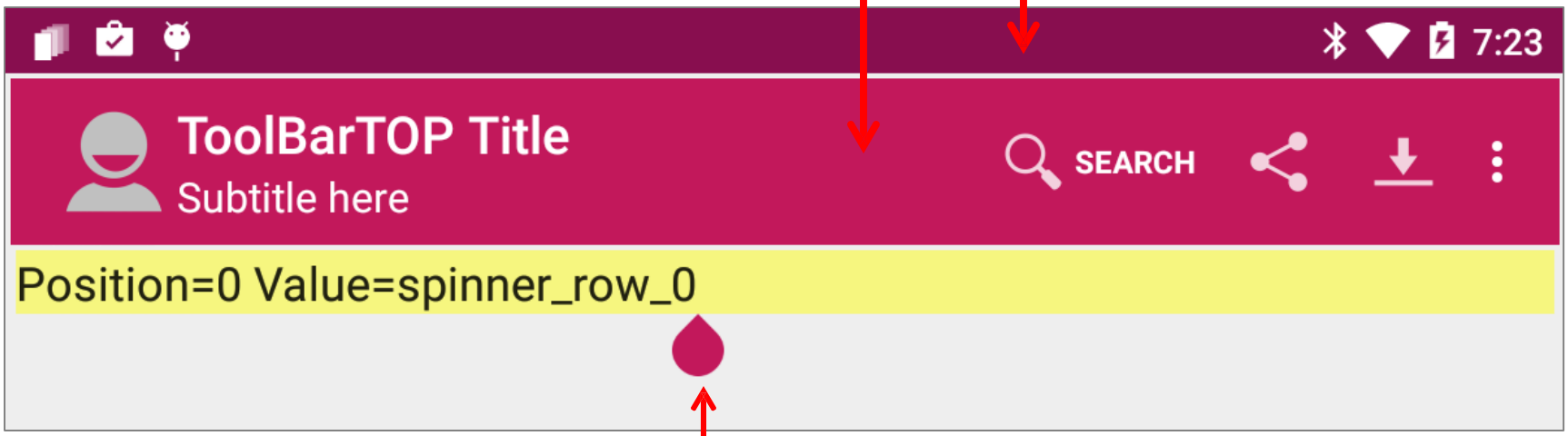
Przykład 11 – Pasek narzędziowy

Paleta kolorów zdefiniowana w:
styles_custom_colors.xml

android:colorPrimaryDark

android:colorPrimary

Toolbar górny



android:colorAccent

Przykład 11 – MainActivity.java


```
public class MainActivity extends    Activity
             implements OnMenuItemClickListener, OnItemSelectedListener {
   EditText txtMsg;
   Toolbar toolbarTop;
   Toolbar toolbarBottom;
   int selectedSpinnerRow = 0;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      txtMsg = (EditText) findViewById(R.id.txtMsg);

1     //TOP toolbar -------------------------------------------------------
      toolbarTop = (Toolbar) findViewById(R.id.toolbar_top);
      toolbarTop.setTitle("ToolBarTOP Title");
      toolbarTop.setSubtitle("Subtitle here");
      toolbarTop.inflateMenu(R.menu.top_menu);
      toolbarTop.setLogo(R.drawable.ic_action_app_logo);
      toolbarTop.setOnMenuItemClickListener(this);
      //BOTTOM toolbar ----------------------------------------------------
      toolbarBottom = (Toolbar) findViewById(R.id.toolbar_bottom);
      //toolbarBottom.setTitle("ToolBarBOTTOM Title");
      //toolbarBottom.setSubtitle("Subtitle here");
      //toolbarBottom.setLogo(R.drawable.ic_action_app_logo);
      toolbarBottom.inflateMenu(R.menu.bottom_menu);
      //adding a custom-view
2     prepareCustomView(toolbarBottom);
      toolbarBottom.setOnMenuItemClickListener(this);
   }
```

Przykład 11 – MainActivity.java

3

```
@Override
public boolean onMenuItemClick(MenuItem item) {
   int id = item.getItemId();

   // Handle TOP action bar item clicks here.
   if (id == R.id.action_search) {
      txtMsg.setText("Search...");      return true;
   } else if (id == R.id.action_share) {
      txtMsg.setText("Share...");       return true;
   } else if (id == R.id.action_download) {
      txtMsg.setText("Download...");    return true;
   } else if (id == R.id.action_about) {
      txtMsg.setText("About...");       return true;
   } else if (id == R.id.action_settings) {
      txtMsg.setText("Settings...");    return true;
   }

   // Handle BOTTOM action bar item
   if (id == R.id.action_camera_bottom) {
      txtMsg.setText("Camera...");      return true;
   } else if (id == R.id.action_unlock_bottom) {
      txtMsg.setText("Unlock...");      return true;
   } else if (id == R.id.action_settings_bottom) {
      txtMsg.setText("Settings-Bottom...");   return true;
   }

   return false;
}
```

Przykład 11 – MainActivity.java

4

```
private void prepareCustomView(Toolbar toolbar) {

  // setup the BOTTOM toolbar
  LayoutInflater inflater = (LayoutInflater) getApplication()
                            .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

  toolbar.addView(inflater.inflate(R.layout.custom_spinner_view_on_actionbar, null));

  // create the custom adapter to feed the spinner
  SpinnerCustomAdapter customSpinnerAdapter = new SpinnerCustomAdapter(
                    getApplicationContext(),
                    SpinnerDummyContent.customSpinnerList);

  // plumbing - get access to the spinner widget shown on the actionBar
  Spinner customSpinner = (Spinner) toolbar.findViewById(R.id.spinner_data_row);

  // bind spinner and adapter
  customSpinner.setAdapter(customSpinnerAdapter);

  // put a listener to wait for spinner rows to be selected
  customSpinner.setOnItemSelectedListener(this);

  customSpinner.setSelection(selectedSpinnerRow);

}//prepareCustomView
```

Przykład 11 – MainActivity.java

5

```
  @Override
  public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {

    selectedSpinnerRow = position;

    TextView caption = (TextView) toolbarBottom.findViewById(
                                   R.id.txtSpinnerRowCaption);

    txtMsg.setText( "Position=" + position + " Value=" + caption.getText());

  }

  @Override
  public void onNothingSelected(AdapterView<?> parent) {
    // TODO nothing to do here...

  }

}
```

Przykład 11 – Pasek narzędziowy

Komentarz

1. Górny pasek narzędziowy zyskuje logo, tytuł, podtytuł oraz pozycje menu głównego.

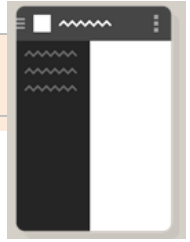
2. Podobnie dolny pasek zyskuje listę rozwijalną i inne pozycje menu.

3. Metoda `onMenuItemClick()` jest wspólna dla górnego i dolnego paska, a jej zadaniem jest obsługa zdarzenia kliknięcia na poszczególne opcje.

4. Metoda `prepareCustomView()` tworzy obiekty zgodnie z podaną specyfikacją xml. Każdy wiersz listy zawiera etykietę tekstową oraz obrazek (zgodnie z `custom_spinner_row_icon_caption.xml`). Własny adapter – jak w przykładzie 7 – jest odpowiedzialny za dodanie elementów do komponentu Spinner.

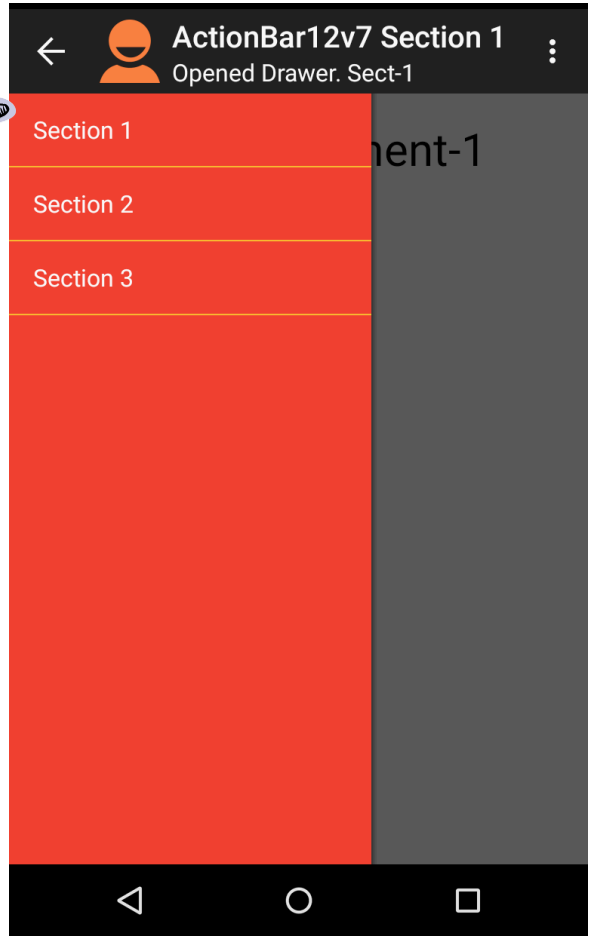
5. Gdy użytkownik wybierze pozycję z listy wywoływana jest metoda `onItemSelected()` by zapamiętać wybraną pozycję oraz na tej podstawie podjąć odpowiednie działania.

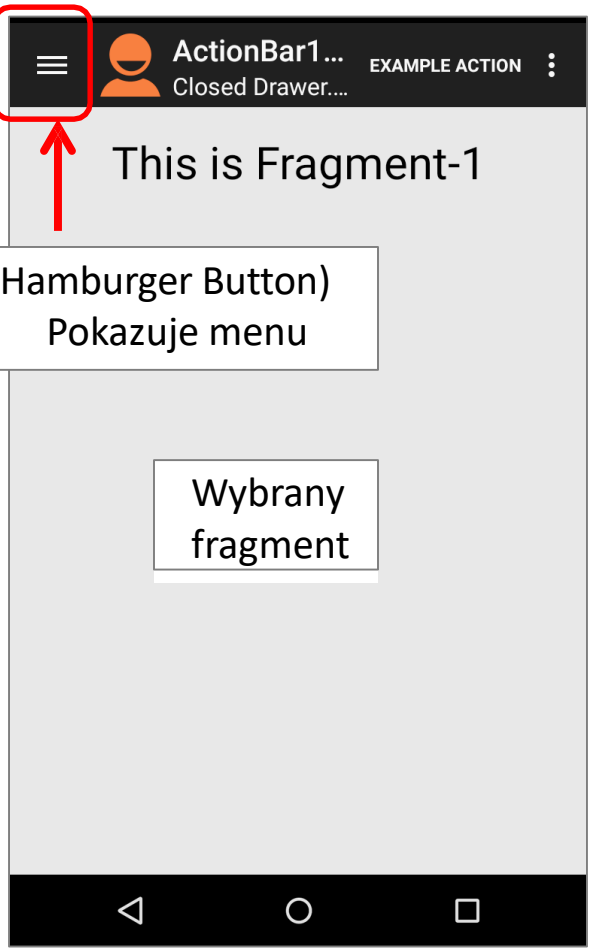
Przykład 12 – Pływające menu



- ActionBar może być zmodyfikowany by zawierał specjalny przycisk (tzw. **HAMBURGER button**).
- Po kliknięciu, tymczasowo przykrywa ekran prezentując tzw. **DrawerLayout**.
- Zwykle ogranicza się do prezentacji listy opcji (zwanych sekcjami lub fragmentami).
- Po wyborze jednej z opcji, menu znika a widok aktywności jest aktualizowany na podstawie wybranej opcji.

Pływające menu



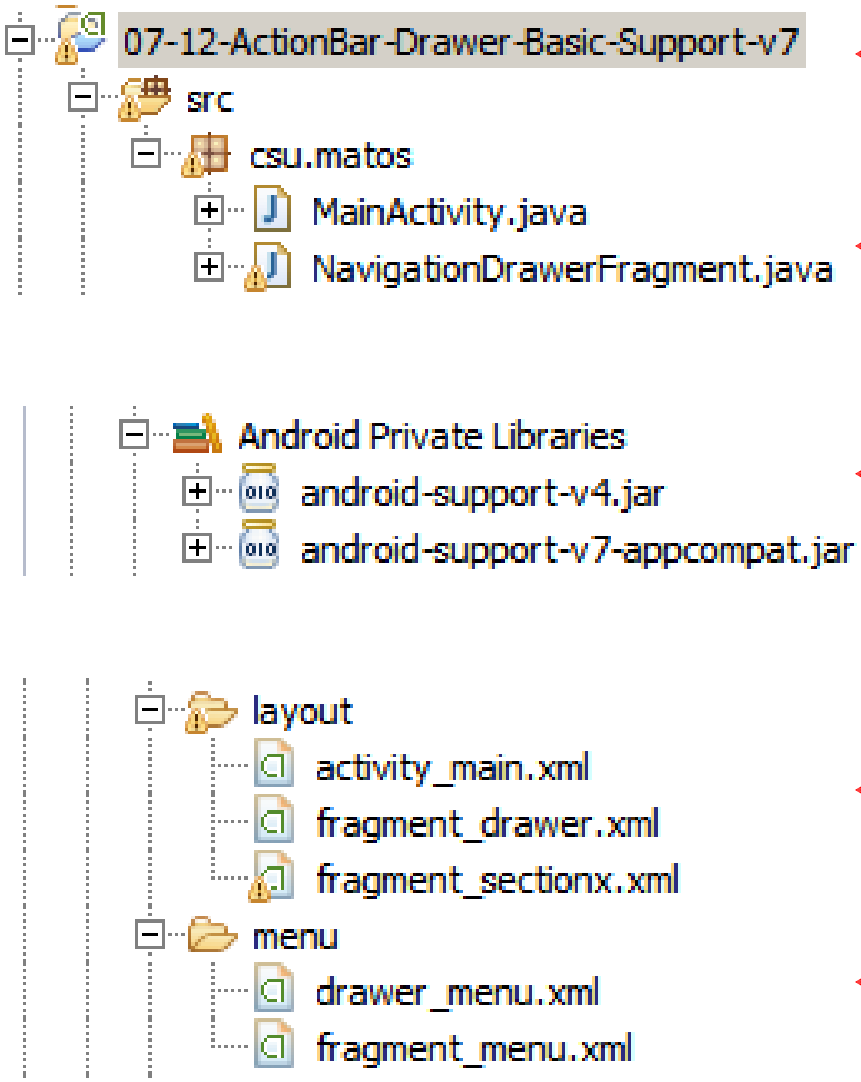


(Hamburger Button)
Pokazuje menu

Wybrany fragment

Przykład 12 – Pływające menu

ARCHITEKTURA APLIKACJI



Przykład stworzony na podstawie odpowiedniego szablonu aplikacji w Android Studio.

Definicja sposobu wyświetlania listy z pływającym menu

Biblioteka **appcompat-v4** dla fragmentów, a **appcompat-v7** by wspierać ActionBar.

Podstawowe układy XML

Wygląd opcji menu

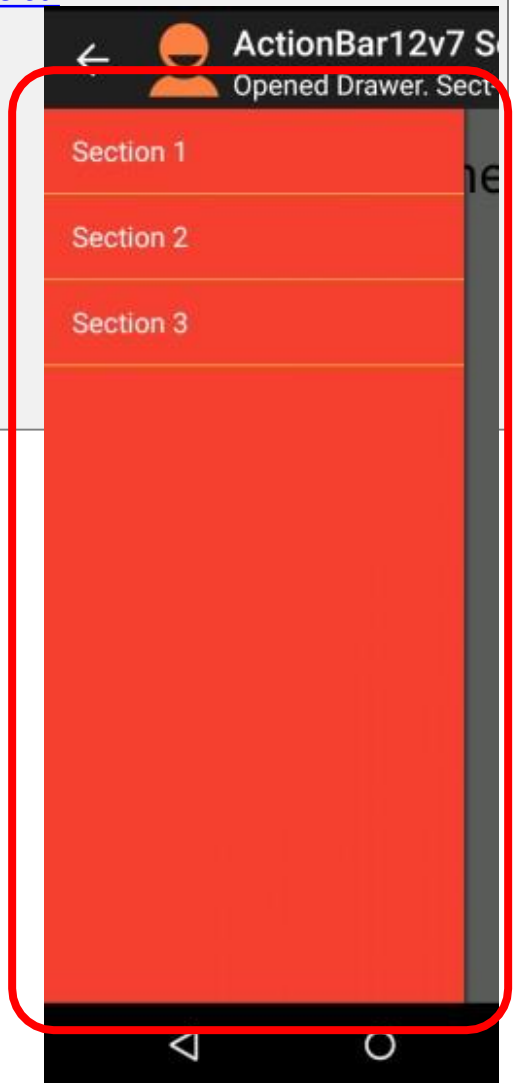
Przykład 12 – UKŁAD: activity_main.xml

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout_activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="csu.matos.MainActivity" >

    <!-- main content view, consumes the entire screen -->
    <FrameLayout
        android:id="@+id/main_fragment_frameLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!-- the navigation drawer will partially cover the screen -->
    <fragment
        android:id="@+id/navigationDrawerFragment_host_container"
        android:name="csu.matos.NavigationDrawerFragment"
        android:layout_width="@dimen/navigation_drawer_width"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        tools:layout="@layout/fragment_navigation_drawer" />

</android.support.v4.widget.DrawerLayout>
```

Miejsce na pływający panel menu

Przykład 12 – UKŁAD: fragment_drawer.xml

```
<ListView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#F44336"
    android:choiceMode="singleChoice"
    android:divider="@android:color/holo_orange_light"
    android:dividerHeight="1dp"
    tools:context="csu.matos.NavigationDrawerFragment" />
```



Prezentowana jest prosta lista. Możliwa jest zmiana wyglądu wierszy przez stworzenie własnego adaptera.

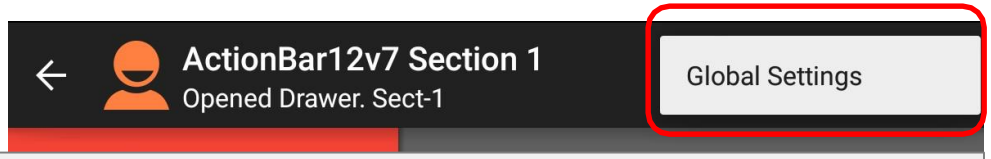
Przykład 12 – UKŁAD: fragment_sectionx.xml

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="csu.matos.MainActivity$PlaceholderFragment" >

    <TextView
        android:id="@+id/txt_section_label"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="This is fragment-n"
        android:textColor="#ff000000"
        android:textSize="30sp" />

</RelativeLayout>
```

Potrzebny jest osobny układ dla każdego elementu menu pływającego. Dla uproszczenia pokazano układ dla jednego elementu.

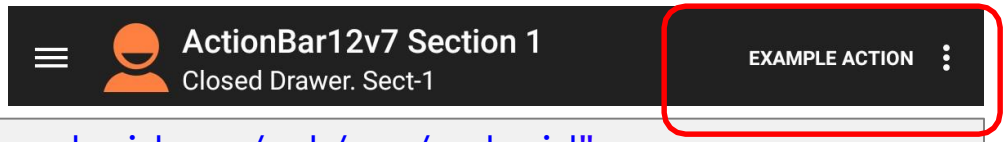
Przykład 12 – MENU: fragment_main.xml

res/menu/drawer_menu.xml




```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto" >
    <item
        android:id="@+id/action_global_settings"
        android:orderInCategory="100"
        android:title="Global Settings"
        app:showAsAction="never"/>
</menu>
```

res/menu/fragment_menu.xml




```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools" >
    <item
        android:id="@+id/action_example"
        android:title="@string/action_example"
        app:showAsAction="withText|ifRoom"/>
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never"/>
</menu>
```

Przykład 12 – MainActivity.java


```
import ...
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
1
public class MainActivity extends ActionBarActivity
             implements NavigationDrawerFragment.NavigationDrawerCallbacks {

   private NavigationDrawerFragment mNavigationDrawerFragment;
   private CharSequence mTitle;
   ActionBar actionBar;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      // this is the navigation fragment
2     mNavigationDrawerFragment = (NavigationDrawerFragment)
                                  getSupportFragmentManager().findFragmentById(
                                  R.id.navigationDrawerFragment_host_container);

      // link navigation drawerFragment to main_activity layout
      mNavigationDrawerFragment.setUp(
                              R.id.navigationDrawerFragment_host_container,
                              R.id.drawer_layout_activity_main );
      prepareActionBar();
   }
```

Przykład 12 – MainActivity.java


```
   @Override
   public void onNavigationDrawerItemSelected(int position) {
3     // update the main GUI content with selected
      fragment(SectionX)  FragmentManager fragmentManager =
      getSupportFragmentManager();  fragmentManager
            .beginTransaction()
            .replace(R.id.main_fragment_frameLayout,
                    PlaceholderFragment.newInstance(position + 1))
            .commit();
   }

   public void onSectionAttached(int number) {
4     switch (number) {
      case 1:
         mTitle = getString(R.string.title_section1);
         break;
      case 2:
         mTitle = getString(R.string.title_section2);
         break;
      case 3:
         mTitle = getString(R.string.title_section3);
         break;
      }

   }
```

Przykład 12 – MainActivity.java


```
public void prepareActionBar() {
   actionBar = getSupportActionBar();
5  actionBar.setDisplayShowTitleEnabled(true);
   actionBar.setDisplayShowHomeEnabled(true);
   actionBar.setLogo(R.drawable.ic_launcher);
   actionBar.setDisplayUseLogoEnabled(true);
   actionBar.setTitle(getTitle() + " " + mTitle);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
   if (!mNavigationDrawerFragment.isDrawerOpen()) {
      getMenuInflater().inflate(R.menu.fragment_menu, menu);
6     prepareActionBar();
      return true;
   }
   return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
7  // Handle action bar item clicks here.
   int id = item.getItemId();
   if (id == R.id.action_settings) {
      return true;
   }
   return super.onOptionsItemSelected(item);
}
```

Przykład 12 – MainActivity.java


```
public static class PlaceholderFragment extends Fragment {

    private static final String ARG_SECTION_NUMBER = "section_number";

8   public static PlaceholderFragment newInstance(int sectionNumber) {
        PlaceholderFragment fragment = new PlaceholderFragment();
        Bundle args = new Bundle();
        args.putInt(ARG_SECTION_NUMBER, sectionNumber);
        fragment.setArguments(args);
        return fragment;
    }

    public PlaceholderFragment() { }

    @Override
9   public View onCreateView(LayoutInflater inflater, ViewGroup
    container,
                                Bundle savedInstanceState) {
        View fragmentView = inflater.inflate(R.layout.fragment_sectionx,
                                            container, false);
        TextView txtCaptionFragment = (TextView)fragmentView.findViewById(
                                                R.id.txt_section_label);
        txtCaptionFragment.setText("This is Fragment-"
                        + getArguments().getInt(ARG_SECTION_NUMBER, 1));
        return fragmentView;

    }
```

Przykład 12 – MainActivity.java

A

```
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        ((MainActivity) activity).onSectionAttached(getArguments()
                                    .getInt(ARG_SECTION_NUMBER));
    }

  }//PlaceHolderFragment

}
```

Przykład 12 – MainActivity.java

Komentarze

Aplikacja została stworzona na podstawie wbudowanego szablonu '**Drawer Navigation Activity**'.

Aplikacja składa się z dwóch rodzajów fragmentów. Jeden jest dedykowany do stworzenia opcji menu pływającego. Nazwijmy ten fragment ***drawer-fragment*** *. Pozostałe fragmenty dedykowane są obsłudze ekranów wyświetlanych po wyborze odpowiedniej opcji z listy przez użytkownika. Nazwijmy je* ***section-fragments****.*

1. Główna aktywność dziedziczy po klasie **ActionBarActivity,** która jest klasą bazową dla aktywności wykorzystujących bibliotekę zapewniającą kompatybilność wsteczną i implementację ActionBar'a (oraz metody takie jak `getSupportFragmentManager()`, `getSupportActionBar()`, itp.). Możliwe jest ich wykorzystanie dla API poziomu 7 bądź wyższego, pod warunkiem ustawienia skórki jako *Theme.AppCompat* (lub podobnej). Za obsługę pływającego menu (otwarcie, zamknięcie itp.) odpowiedzialna jest biblioteka appcompat-v4.

Przykład 12 – MainActivity.java

2. Plik zasobów **activity_main.xml** definiuje element `<fragment>` w ramach którego umieszczone jest menu pływające. Punkt 2 odnosi się do momentu w którym obiekt *drawer-fragment* jest tworzony i osadzony w miejsce komponentu oznaczonego znacznikiem <fragment>. Pływające menu zostanie wyświetlone na ekranie gdy użytkownik kliknie specjalny przycisk (**Hamburger** buton) lub przesunie palcem przez długość ekranu.

3. Metoda **`onNavigationDrawerItemSelected`** stanowi implementację interfejsu **Callback** którego celem jest połączenie obiektu *drawer-fragment* oraz głównej aktywności. Po jej wywołaniu tworzony jest obiekt *section-fragment* zgodnie z wyborem użytkownika z listy.

4. Gdy fragment section-fragment jest dołączony do okna aplikacji (zwykle zastępując poprzedni widok) metoda **`onSectionAttached`** jest wywoływana. Następuje wówczas aktualizacja zmiennej globalnej **`mTitle`** na nazwę wybranej pozycji.

5. Pobierana jest referencja do komponentu ActionBar i dokonywana jest jego inicjalizacja podając tytuł, logo i przycisk główny.

Przykład 12 – MainActivity.java

6. Każdy fragment section-fragment może mieć swoje indywidualne menu. Metoda **`onCreateOptionsMenu`** odpowiedzialna jest za tworzenie właściwego menu i aktualizację komponentu ActionBar.

7. Gdy pozycja menu umieszczona na komponencie ActionBar zostaje wybrana, metoda **`onOptionsItemSelected`** jest wywoływana by obsłużyć żądanie użytkownika.

8. Dla uproszczenia, prezentowany przykład tworzy tylko jeden typ fragmentu *section-fragment*. Klasa **`PlaceholderFragment`** reprezentuje odpowiedź aplikacji na wybór użytkownika.

9. Każdy z fragmentów *section-fragments* będzie posiadał unikalny alias (‘This is fragment-1’, ‘This is fragment-2, …). Metoda **`onCreateView`** aktualizuje widok za każdym razem, gdy użytkownik wybierze pozycję z listy.

Przykład 12 – NavigationDrawerFragment.java


```
import ...
import android.support.v4.app.Fragment;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBar;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.app.ActionBarDrawerToggle;

1 →
public class NavigationDrawerFragment extends Fragment {

   private static final String STATE_SELECTED_POSITION
   ="chosen_drawer_position";
   private static final String PREF_USER_LEARNED_DRAWER = "drawer_learned";

   private NavigationDrawerCallbacks mCallbacks;
   private ActionBarDrawerToggle mDrawerToggle;
   private DrawerLayout mainActivityLayout;
   private ListView mDrawerListView;
   private View mDrawerFragmentHost;
   private ActionBar actionBar;

   private int mCurrentSelectedPosition = 0;
   private boolean mFromSavedInstanceState;
   private boolean mUserLearnedDrawer;

   public NavigationDrawerFragment() {
   }
```

Przykład 12 – NavigationDrawerFragment.java 2 z 9

```
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);

   // Read in the flag indicating whether or not the user has demonstrated
   // awareness of the drawer. See PREF_USER_LEARNED_DRAWER for details.
   SharedPreferences sp = PreferenceManager
                          .getDefaultSharedPreferences(getActivity());
   mUserLearnedDrawer = sp.getBoolean(PREF_USER_LEARNED_DRAWER, false);

   if (savedInstanceState != null) {
      mCurrentSelectedPosition = savedInstanceState
                                 .getInt(STATE_SELECTED_POSITION);
      mFromSavedInstanceState = true;
   }

   // Select either the default item (0) or the last selected item.
   selectItem(mCurrentSelectedPosition);

}

@Override
public void onActivityCreated(Bundle savedInstanceState) {
   super.onActivityCreated(savedInstanceState);
   // this fragment may alter the action bar.
   setHasOptionsMenu(true);
}
```

2

3

Przykład 12 – NavigationDrawerFragment.java


```
@Override
public View onCreateView( LayoutInflater inflater, ViewGroup container,
                          Bundle savedInstanceState) {

  mDrawerListView = (ListView)  inflater.inflate(
                                R.layout.fragment_drawer, container, false);

  mDrawerListView.setOnItemClickListener(new
  AdapterView.OnItemClickListener(){
        @Override
        public void onItemClick(AdapterView<?> parent, View view,
                                int position, long id) {
          selectItem(position);
        }
      });
  mDrawerListView.setAdapter(new ArrayAdapter<String>(
          getActionBar().getThemedContext(),
          android.R.layout.simple_list_item_1,
          android.R.id.text1,
          new String[] { getString(R.string.title_section1),
                         getString(R.string.title_section2),
                         getString(R.string.title_section3), } ) );

  mDrawerListView.setItemChecked(mCurrentSelectedPosition, true);
  return mDrawerListView;

 }
```

4

5

Przykład 12 – NavigationDrawerFragment.java


```
public boolean isDrawerOpen() {
   return mainActivityLayout != null
         && mainActivityLayout.isDrawerOpen(mDrawerFragmentHost);
}

public void setUp(int drawerFragmentId, int activity_main_Id) {
   //find the place in the main layout where drawer fragment is to be shown
   mDrawerFragmentHost = getActivity().findViewById(drawerFragmentId);
   //access the full activity_main layout (DrawerView)
   mainActivityLayout =  (DrawerLayout) getActivity().findViewById(
                                                        activity_main_Id);
   // set custom shadow to overlay main content when the drawer opens
   mainActivityLayout.setDrawerShadow(R.drawable.drawer_shadow,
                                      GravityCompat.START);
   // set up the drawer's list view with items and click listener
   actionBar = getActionBar();
   actionBar.setDisplayHomeAsUpEnabled(true);
   actionBar.setHomeButtonEnabled(true);

   try {
      mDrawerToggle = new CustomActionBarDrawerToggle(mainActivityLayout);
      mainActivityLayout.setDrawerListener(mDrawerToggle);

   } catch (RuntimeException e) {
      Toast.makeText(getActivity(), "Error:"+e.getMessage(),
                     Toast.LENGTH_SHORT).show();
   }
```

6

Przykład 12 – NavigationDrawerFragment.java


```
        //the unseen drawer should be shown to the user (once at launch time)
        if (!mUserLearnedDrawer && !mFromSavedInstanceState) {
            mainActivityLayout.openDrawer(mDrawerFragmentHost);
        }

        // Defer code dependent on restoration of previous instance state.
        mainActivityLayout.post(new Runnable() {
            @Override
            public void run() {
                mDrawerToggle.syncState();
            }
        });

    }

    private void selectItem(int position) {
        mCurrentSelectedPosition = position;
7   if (mDrawerListView != null) { //user is learning - section checked!
            mDrawerListView.setItemChecked(position, true);
        }
        if (mainActivityLayout != null) {
            mainActivityLayout.closeDrawer(mDrawerFragmentHost);
        }
        if (mCallbacks != null) {
            mCallbacks.onNavigationDrawerItemSelected(position);
        }
    }
```

Przykład 12 – NavigationDrawerFragment.java


```
@Override
public void onAttach(Activity activity) {
   super.onAttach(activity);
   try {
8     mCallbacks = (NavigationDrawerCallbacks) activity;
   } catch (ClassCastException e) {
      throw new ClassCastException("Main must implement Nav.DrawerCallbacks.");
   }
}

@Override
public void onDetach() {
   super.onDetach();
   mCallbacks = null;
}

@Override
public void onSaveInstanceState(Bundle outState) {
9  super.onSaveInstanceState(outState);
   outState.putInt(STATE_SELECTED_POSITION, mCurrentSelectedPosition);
}

@Override
public void onConfigurationChanged(Configuration newConfig) {
   super.onConfigurationChanged(newConfig);
   // Forward the new configuration the drawer toggle component.
   mDrawerToggle.onConfigurationChanged(newConfig);
}
```

Przykład 12 – NavigationDrawerFragment.java


```
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
   // (Opened Drawer) show the global app actions in the action bar.
   if (mainActivityLayout != null && isDrawerOpen()) {
      inflater.inflate(R.menu.drawer_menu, menu);

   }
   super.onCreateOptionsMenu(menu, inflater);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
   if (mDrawerToggle.onOptionsItemSelected(item)) {
      return true;
   }

   if (item.getItemId() == R.id.action_example) {
      Toast.makeText(getActivity(),"Ex. action.",Toast.LENGTH_SHORT).show();
      return true;
   }

   return super.onOptionsItemSelected(item);
}

private ActionBar getActionBar() {
   return ((ActionBarActivity) getActivity()).getSupportActionBar();
}
```

A

Przykład 12 – NavigationDrawerFragment.java


```
public static interface NavigationDrawerCallbacks {

  void onNavigationDrawerItemSelected(int position);
}



private class CustomActionBarDrawerToggle extends ActionBarDrawerToggle {

  public CustomActionBarDrawerToggle(DrawerLayout mainActivityLayout) {
    super(getActivity(), mainActivityLayout,
          R.string.navigation_drawer_open,R.string.navigation_drawer_close );
  }

  @Override
  public void onDrawerClosed(View view) {
    actionBar.setSubtitle("Closed Drawer. Sect"+(mCurrentSelectedPosition + 1));
    getActivity().invalidateOptionsMenu(); //next is onPrepareOptionsMenu()

  }
```

B

C

Przykład 12 – NavigationDrawerFragment.java 9 z 9

D

```
    @Override
    public void onDrawerOpened(View drawerView) {

      if (!isAdded()) {
        return;
      }

      if (!mUserLearnedDrawer) {
        // The user manually opened the drawer; store this flag to
        // prevent auto-showing the navig.drawer in the future.
        mUserLearnedDrawer = true;

        SharedPreferences sp = PreferenceManager
                                .getDefaultSharedPreferences(getActivity());

        sp.edit().putBoolean(PREF_USER_LEARNED_DRAWER, true).apply();
      }

      actionBar.setSubtitle("Open Drawer. Sect-"+(mCurrentSelectedPosition + 1));
      getActivity().invalidateOptionsMenu(); // next is onPrepareOptionsMenu()
    }

  }//CustomActionBarDrawerToggle

}// Fragment
```

Komentarze

1. Klasa **`NavigationDrawerFragment`** jest odpowiedzialna za obsługę zdarzeń dla obiektu *drawer-fragment*. Gdy użytkownik wybierze daną opcję, fragment reprezentujący wybór prezentowany jest na ekranie.

2. Gdy aplikacja jest uruchamiana po raz pierwszy, nie ma żadnego uprzednio zapisanego stanu aplikacji. Dlatego całe menu pływające zostanie wyświetlone na ekranie. Gdy użytkownik wypróbuje każdą opcję w menu, flaga boolowska `mUserLearnedDrawer` zostanie ustawiona by zapamiętać ten fakt. Przy każdym następnym uruchomieniu aplikacji, na podstawie zapamiętanej wartości zostanie podjęta decyzja czy pokazać panel z menu pływającym czy też nie.

3. Metoda `setHasOptionsMenu()` jest wywoływana by powiadomić system, że dany fragment chce wyświetlić własne menu.

Przykład 12 – NavigationDrawerFragment.java 2 z 4

4. Wywołanie `onCreateView()` skutkuje stworzeniem obiektów na podstawie układu *drawer-fragment*. Nasłuchiwacz kliknięcia zapamiętuje pozycję wybranego wiersza listy. W naszym przykładzie tworzona jest zwykła lista łańcuchów znaków, jednak może prezentować dane dowolnego typu.

5. Przykład wykorzystuje prosty `ArrayAdapter<String>` by wypełnić wszystkie wiersze listy.

6. Menu pływające jest przygotowywane przez wywołanie metody `setUp()`. Na początku lokalizuje ona miejsce na ekranie gdzie należy umieścić fragment o nazwie *drawer-fragment*. Następnie dodawany jest cień do prawej krawędzi menu. Następnie do ActionBar'a dodawany jest przycisk typu **Hamburger** button, informujący użytkownika o istnieniu menu pływającego. Kliknięcie na ten przycisk powoduje wyświetlenie menu.

Przykład 12 – NavigationDrawerFragment.java

7. Gdy użytkownik wybierze pozycję z listy jej indeks jest zapamiętywany, jest ona oznaczona jako „odwiedzona”, a menu zostaje zamknięte. Główna aktywność jest informowana poprzez wywołanie zwrotne o dokonanym wyborze.

8. Fragment *drawer-fragment* musi mieć pewność, że główna aktywność nasłuchuje na jego komunikaty. Metoda `onAttach()` weryfikuje zatem implementację interfejsu Callback.

9. Zanim aplikacja zostanie zamknięta, pozycja ostatnio wybranego elementu jest zapamiętywana w kolekcji typu Bundle. Następna sesja z aplikacją rozpocznie się od wyświetlenia tej pozycji.

A. Fragment *drawer-fragment* tworzy obiekty odpowiedzialne za wyświetlenie menu.

B. Interfejs wywołania zwrotnego jest definiowany. W tym przykładzie składa się z pojedynczej metody `onNavigationDrawerItemSelected()` akceptującej liczbę całkowitą reprezentującą wybraną pozycję listy.

Przykład 12 – NavigationDrawerFragment.java 4 z 4

C. Klasa `CustomActionBarDrawerToggle` opisuje co się dzieje w momencie otwarcia i zamknięcia menu pływającego. Metoda `onDrawerClosed()` aktualizuje tytuł ActionBar'a zgodnie z wybraną pozycją. Następuje również zmiana pozycji w menu.

D. Za pierwszym wywołaniem metody `onDrawerOpenned()` fragment *drawer-fragment* nie został jeszcze dodany do aktywności bazowej, co implikuje wyświetlenie menu bocznego. Informacja o wyborze tej pozycji z menu jest przechowywana w kolekcji typu klucz-wartość. Metoda aktualizuje później tytuł ActionBar'a (który początkowo jest pusty) i zmienia zawarte tam menu.

Wykorzystanie menu i ActionBar

PYTANIA?