Programowanie urządzeń mobilnych

Tworzenie dialogów i wykorzystanie klasy Toast

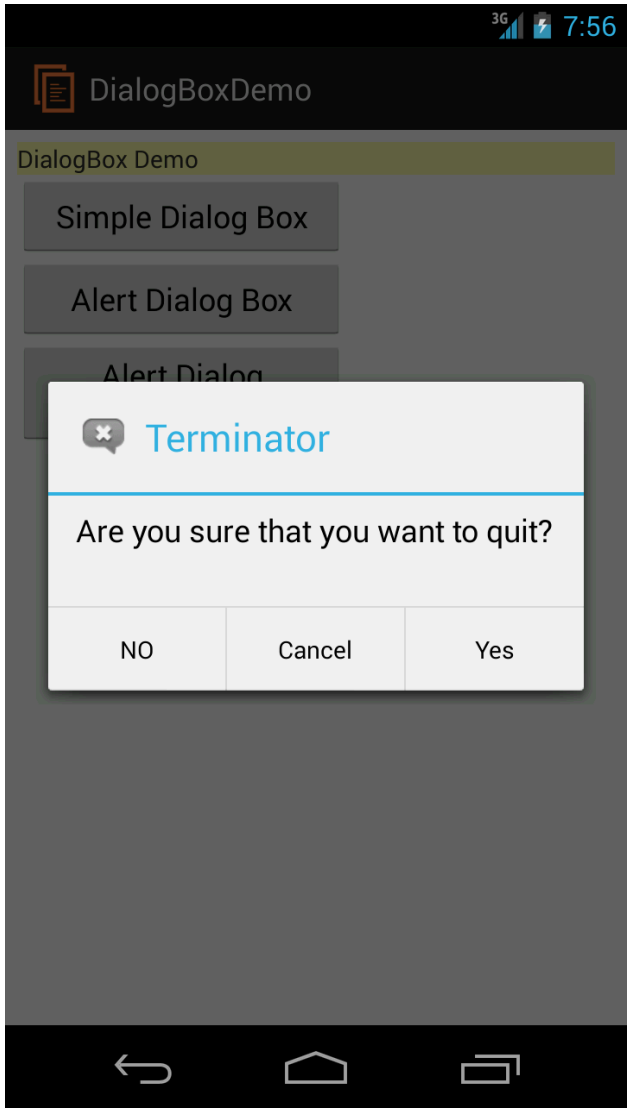
Tłumaczenie i adaptacja materiałów: dr Tomasz Xięski.
Na podstawie prezentacji udostępnionych przez Victor Matos, Cleveland State University.

Wykorzystanie Dialogów

Android umożliwia tworzenie dwóch typów dialogów:

1. **Wykorzystując klasę AlertDialog.**

2. **Wykorzystując klasę Toast**



Toast to transparentny dialog wyświetlający – na kilka sekund – zadaną informację. Znika bez udziału użytkownika.

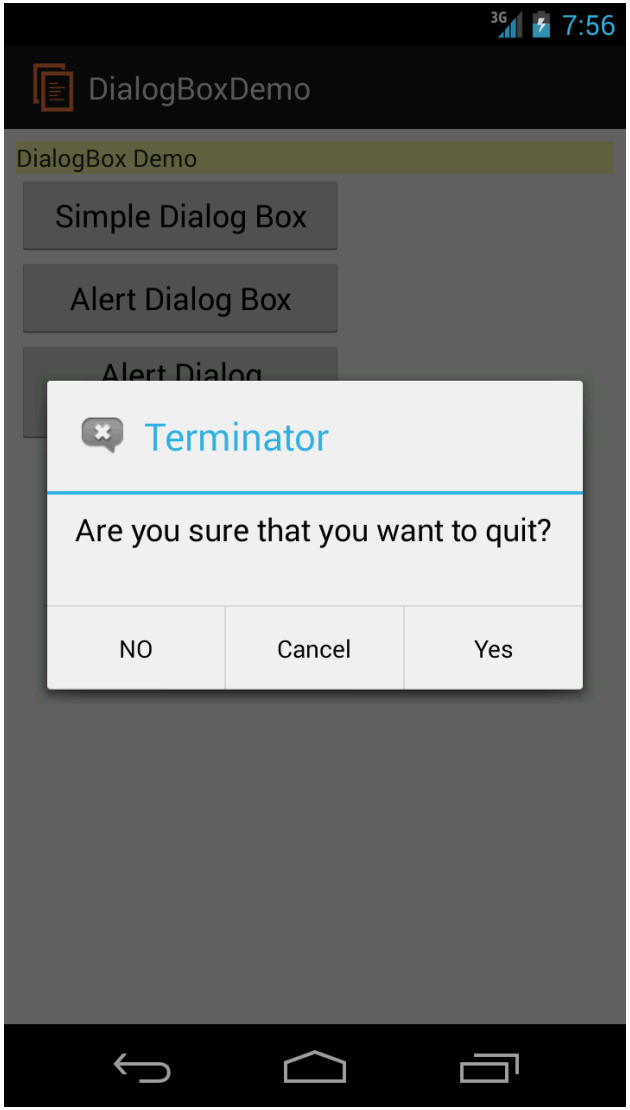
AlertDialog

AlertDialog posiada następujące właściwości:

(1) Wyświetla małe okno pływające, znajdujące się nad aktualnym widokiem.

(2) Prezentuje użytkownikowi komunikat wraz z trzema opcjonalnymi przyciskami.

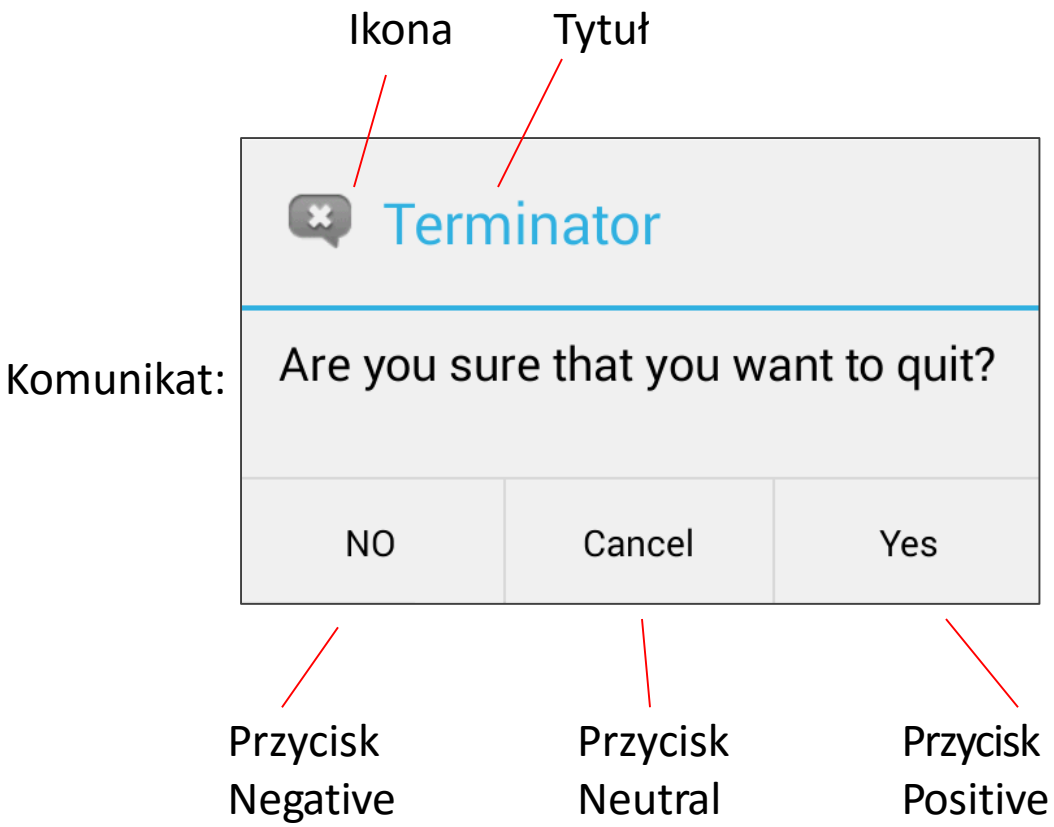
(3) Dialog jest zamykany poprzez kliknięcie danego przycisku bądź dotknięcie obszaru poza dialogiem.



Uwaga:
Dialog tego typu nie jest dialogiem modalnym!

AlertDialog

Struktura AlertDialog:



Rysunek wykorzystuje:
Theme_Holo_Light_Dialog oraz
STYLE_NORMAL

AlertDialog

Przykład 1. AlertDialog

Widok aplikacji składa się z trzecich przycisków. Po kliknięciu na dany przycisk, określony typ AlertDialog jest wyświetlany.

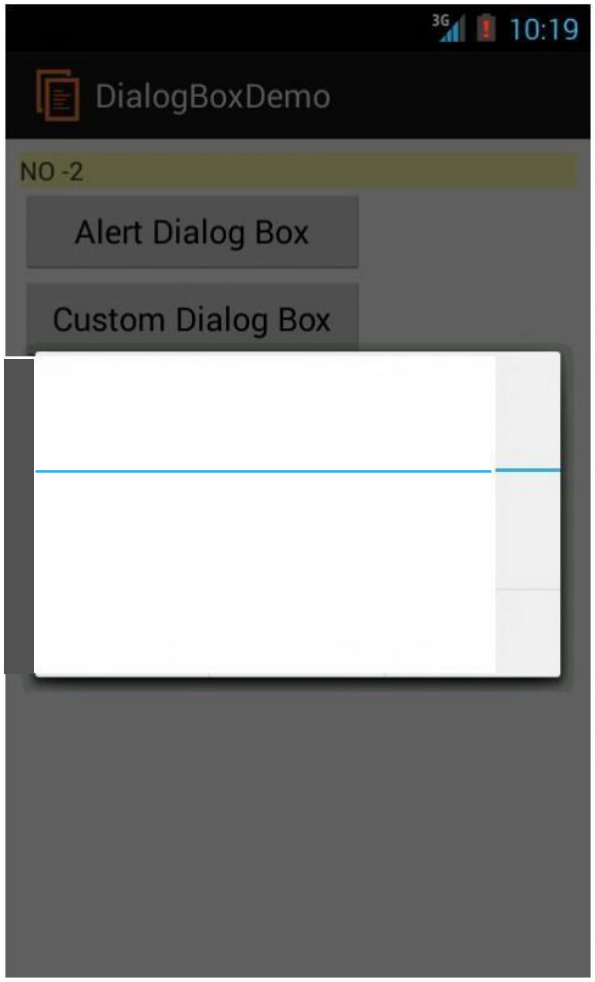
1. Po kliknięciu pierwszego przycisku wyświetlany jest standardowy **AlertDialog** z 3 przyciskami.

2. Drugi wybór to **własna implementacja dialogu**, w którym użytkownik może wprowadzić dane.

3. Ostatnia opcja wykorzystuje koncepcję fragmentów.

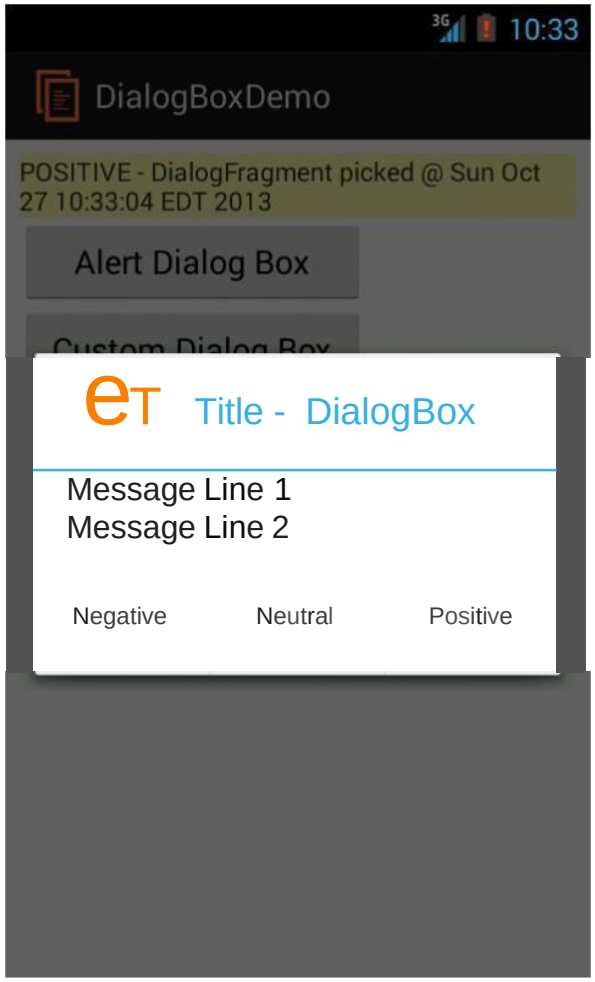
AlertDialog

Przykład 1. AlertDialog



Prosty **AlertDialog** z trzema opcjami.

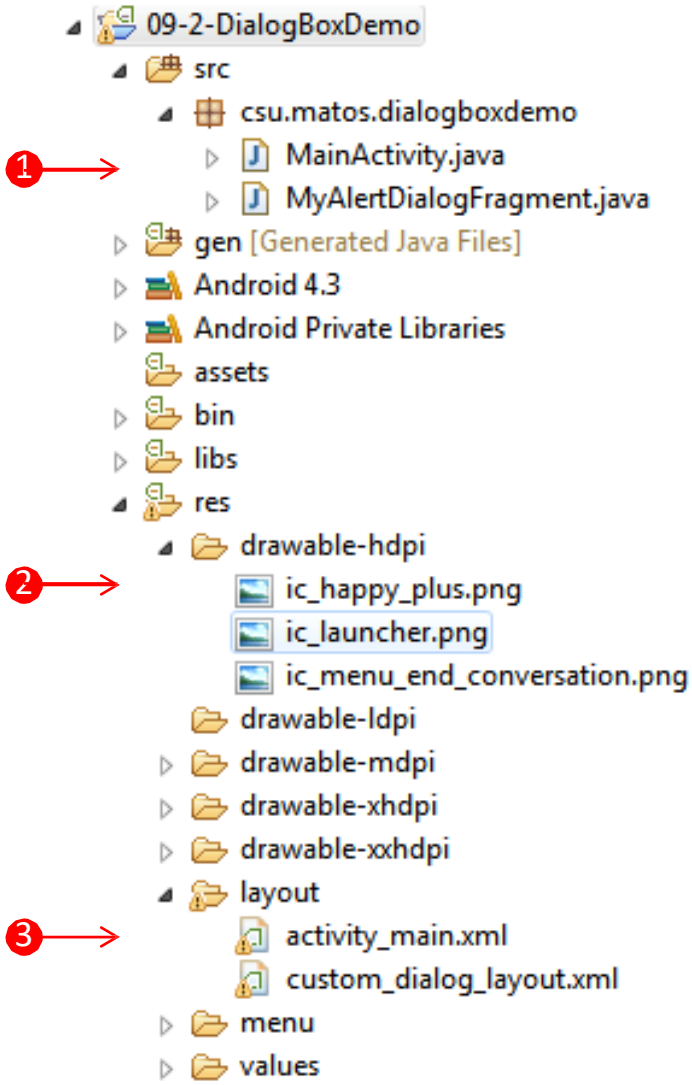
Własny AlertDialog do wprowadzania danych.



Fragment z trzema opcjami.

AlertDialog

Przykład 1. Struktura aplikacji



1. Klasa **MainActivity** prezentuje listę opcji oraz umożliwia osadzenie obiektów klasy **DialogFragment**.

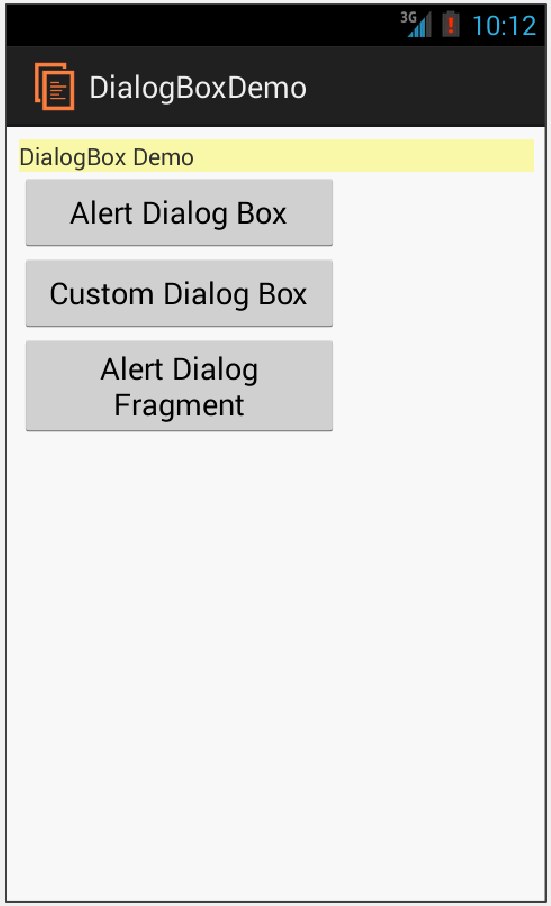
2. Zasoby aplikacji zawierają również zestaw ikon wyświetlanych na oknach dialogowych.

3. Plik XML custom_dialog_layout.xml prezentuje widok dla własnej implementacji dialogu.

AlertDialog

Przykład 1. Układ XML – activity_main.xml

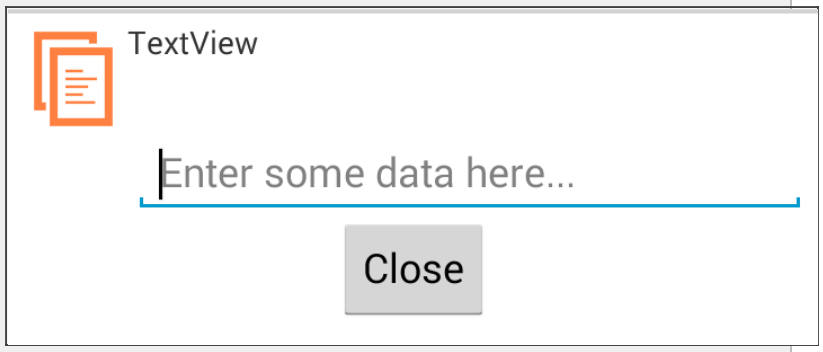
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"     android:layout_height="match_parent"
    android:orientation="vertical"          android:padding="7dp"   >

    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#55ffff00"
        android:text="DialogBox Demo" />

    <Button
        android:id="@+id/btn_alert_dialog1"
        android:layout_width="190dp"
        android:layout_height="wrap_content"
        android:text="Alert Dialog Box" />

    <Button
        android:id="@+id/btn_custom_dialog"
        android:layout_width="190dp"
        android:layout_height="wrap_content"
        android:text="Custom Dialog Box" />

    <Button android:id="@+id/btn_alert_dialog2"
        android:layout_width="190dp"
        android:layout_height="wrap_content"
        android:text="Alert Dialog Fragment" />
</LinearLayout>
```



AlertDialog

Przykład 1. Układ XML – custom_dialog_layout.xml

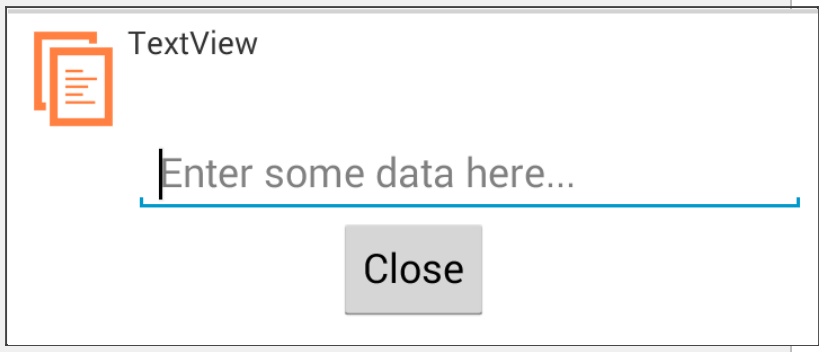
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="5dp" >

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

        <ImageView
            android:id="@+id/imageView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_launcher" />

        <TextView android:id="@+id/sd_textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="TextView" />
    </LinearLayout>
```



AlertDialog

Przykład 1. Układ XML – custom_dialog_layout.xml

```
    <EditText
        android:id="@+id/sd_editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="50dp"
        android:ems="15"
        android:hint="Enter some data here..." >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/sd_btnClose"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Close" />

</LinearLayout>
```



AlertDialog

Przykład 1. MainActivity.java

```
// example adapted from:
// http://developer.android.com/reference/android/app/DialogFragment.html

public class MainActivity extends Activity implements OnClickListener {
   TextView txtMsg;
   Button btnCustomDialog;
   Button btnAlertDialog;
   Button btnDialogFragment;
   Context activityContext;
   String msg = "";

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

      txtMsg = (TextView) findViewById(R.id.txtMsg);
1 →   btnAlertDialog = (Button) findViewById(R.id.btn_alert_dialog1);
      btnCustomDialog = (Button) findViewById(R.id.btn_custom_dialog);
      btnDialogFragment = (Button) findViewById(R.id.btn_alert_dialog2);

      btnCustomDialog.setOnClickListener(this);
      btnAlertDialog.setOnClickListener(this);
      btnDialogFragment.setOnClickListener(this);

   }
```

AlertDialog

Przykład 1. MainActivity.java

```
@Override
public void onClick(View v) {
    if (v.getId() == btnAlertDialog.getId()) {
        showMyAlertDialog(this);
    }
    if (v.getId() == btnCustomDialog.getId()) {
        showCustomDialogBox();
    }
    if (v.getId() == btnDialogFragment.getId()) {
        showMyAlertDialogFragment(this);
    }
}// onClick

private void showMyAlertDialog(MainActivity mainActivity) {
    new AlertDialog.Builder(mainActivity)
            .setTitle("Terminator")
            .setMessage("Are you sure that you want to quit?")
            .setIcon(R.drawable.ic_menu_end_conversation)

            // set three option buttons
            .setPositiveButton("Yes",
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int whichButton) {
                            // actions serving "YES" button go here
                            msg = "YES " + Integer.toString(whichButton);
                            txtMsg.setText(msg);
                        }
                    })// setPositiveButton
```

AlertDialog

Przykład 1. MainActivity.java

```
            .setNeutralButton("Cancel",
                  new DialogInterface.OnClickListener() {
                     public void onClick(DialogInterface dialog,
                           int whichButton) {
                        // actions serving "CANCEL" button go here
                        msg = "CANCEL " + Integer.toString(whichButton);
                        txtMsg.setText(msg);
                     }// OnClick
                  })// setNeutralButton

            .setNegativeButton("NO", new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int whichButton) {
                  // actions serving "NO" button go here
                  msg = "NO " + Integer.toString(whichButton);
                  txtMsg.setText(msg);
               }
            })// setNegativeButton

            .create()
            .show();

      }// showMyAlertDialog
```

AlertDialog

Przykład 1. MainActivity.java

```
private void showCustomDialogBox() {

    final Dialog customDialog = new Dialog(activityContext);
    customDialog.setTitle("Custom Dialog Title");

    // match customDialog with custom dialog layout
    customDialog.setContentView(R.layout.custom_dialog_layout);

    ((TextView) customDialog.findViewById(R.id.sd_textView1))
                .setText("\nMessage line1\nMessage line2\n"
                +"Dismiss: Back btn, Close, or touch outside");

    final EditText sd_txtInputData = (EditText) customDialog
                                                .findViewById(R.id.sd_editText1);

    ((Button) customDialog.findViewById(R.id.sd_btnClose))
          .setOnClickListener(new OnClickListener() {
             @Override
             public void onClick(View v) {
                txtMsg.setText(sd_txtInputData.getText().toString());
                customDialog.dismiss();
             }
          });

    customDialog.show();

}
```

4

AlertDialog

Przykład 1. MainActivity.java

```
    private void showMyAlertDialogFragment(MainActivity mainActivity) {

5 →     DialogFragment dialogFragment = MyAlertDialogFragment
                                        .newInstance(R.string.title);

        dialogFragment.show(getFragmentManager(), "TAG_MYDIALOGFRAGMENT1");

    }

6 →  public void doPositiveClick(Date time) {
        txtMsg.setText("POSITIVE - DialogFragment picked @ " + time);

    }

    public void doNegativeClick(Date time) {
        txtMsg.setText("NEGATIVE - DialogFragment picked @ " + time);

    }

    public void doNeutralClick(Date time) {
        txtMsg.setText("NEUTRAL - DialogFragment picked @ " + time);

    }

}
```

AlertDialog

Przykład 1. MainActivity.java

Komentarz

1, 2. Główny interfejs prezentuje 3 przyciski do wyświetlenia odpowiedniego typu dialogu.

3. Metoda **showMyAlertDialog** używa tzw. mechanizm *builder* by stworzyć nowy *AlertDialog* dodając do niego tytuł, ikonę, komunikat oraz trzy przyciski. Każdy przycisk ma przypisaną indywidualną metodę onClick() odpowiedzialną za określoną reakcję.

4. Własny dialog wykorzystuje metodę `.setContentView(R.layout.custom_dialog_layout)` by ustawić odpowiedni widok. Jego przycisk do zamknięcia ma ustawiony nasłuchiwacz by przesłać wprowadzony tekst do głównej aktywności i zamknąć dialog.

5. W tym kroku obiekt **DialogFragment** jest tworzony. W konstruktorze podawany jest m. in. jego tytuł. Dialog prezentowany jest ponad aktualnie prezentowany widok.

6. **Odwołania zwrotne pod postacią metod** doPositive(), doNegative() itp umożliwiają fragmentowi DialogFragment przekazywanie danych do głównej aktywności.

AlertDialog

Przykład 1. MyAlertDialogFragment.java

```
public class MyAlertDialogFragment extends DialogFragment {

  public static MyAlertDialogFragment newInstance(int title) {

    MyAlertDialogFragment frag = new MyAlertDialogFragment();

    Bundle args = new Bundle();
    args.putInt("title", title);
    args.putString("message", "Message Line 1\nMessage Line 2");
    args.putInt("icon", R.drawable.ic_happy_plus);

    frag.setArguments(args);
    return frag;
  }

  @Override
  public Dialog onCreateDialog(Bundle savedInstanceState) {

    int title = getArguments().getInt("title");
    int icon = getArguments().getInt("icon");
    String message = getArguments().getString("message");

    return new AlertDialog.Builder(getActivity())
        .setIcon(icon)
        .setTitle(title)
        .setMessage(message)
```

17

AlertDialog

Przykład 1. MyAlertDialogFragment.java

```
        .setPositiveButton("Positive",
            new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog,
                  int whichButton) {
                ((MainActivity) getActivity())
                    .doPositiveClick(new Date());
              }
            })
        .setNegativeButton("Negative",
            new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog,
                  int whichButton) {
                ((MainActivity) getActivity())
                    .doNegativeClick(new Date());
              }
            })
        .setNeutralButton("Neutral",
            new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog,
                  int whichButton) {
                ((MainActivity) getActivity())
                    .doNeutralClick(new Date());
              }
            }).create();
  }
}
```

3

18

AlertDialog

Przykład 1. MyAlertDialogFragment.java

Komentarz

1. Klasa rozszerza **DialogFragment**. Metoda odpowiedzialna za jego tworzenie wymaga podania tytułu, komunikatu oraz ikony. Podane argumenty są zapamiętywane w strukturze typu Bundle.

2. Metoda **onCreateDialog** pobiera przekazane dane (tytuł, ikona, komunikat) ze struktury typu Bundle. Mechanizm builder jest wykorzystywany by stworzyć okno dialogowe.

3. Trzy przyciski dodawane są do obiektu DialogFragment. Każdy posiada nasłuchiwacza, który aktywowany wywołuje odpowiednią metodę zwrotną z aktywności MainActivity. Przekazywana jest aktualna data i czas.

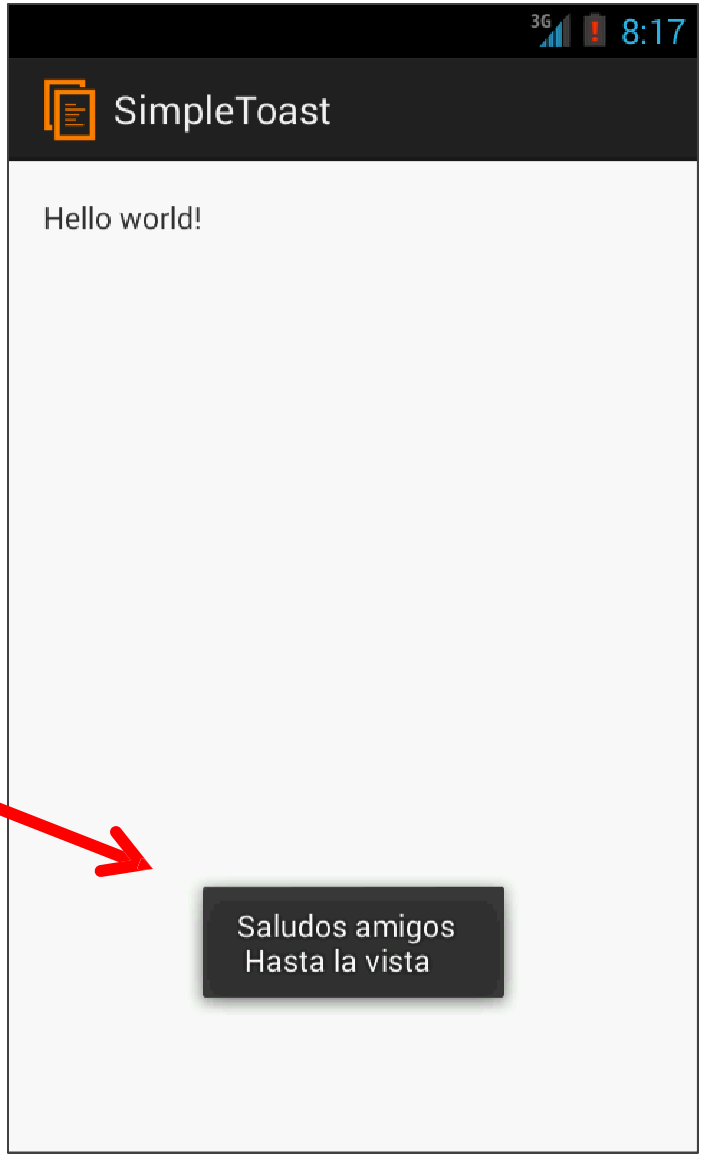
Toast

Obiekty typu Toast stanowią najprostszy mechanizm generowania komunikatów.

Zwykle używane są w sytuacji, gdy **krótka wiadomość** powinna zostać wyświetlona użytkownikowi.

Komunikat wizualizowany jest jako pół-przeźroczysty, pływający dialog, wyświetlany nad aktualnym widokiem. Jego czas życia to 2-4 sekundy.

Toast nigdy nie uzyskuje focus'a!



Toast

Przykład 2. Składnia polecenia makeText

```
Toast.makeText ( context,  message,  duration ).show();
```

Context: Referencja do kontekstu aktualnego widoku bądź aplikacji.

Message: Komunikat do wyświetlenia.

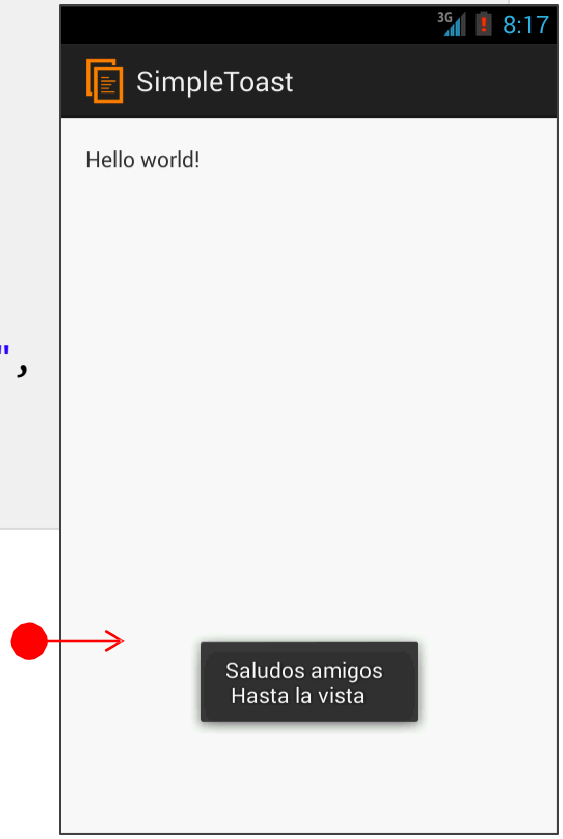
Duration: Toast.*LENGTH_SHORT* (0) to ok. 2 sec
Toast.*LENGTH_* LONG (1) to ok. 3.5 sec

Klasa Toast posiada tylko kilka metod wpływające na położenie komunikatu: *makeText, show, setGravity oraz setMargin.*

Toast

Przykład 2. Prosty Toast

```
public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState)
        { super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Toast.makeText( getApplicationContext(),
                        "Saludos amigos \n Hasta la vista",
                        Toast.LENGTH_LONG).show();
    }
}
```



Przekazanie kontekstu można zrealizować na kilka sposobów: `getApplicationContext()`, `MainActivity.this`, lub **`this`**.

Przykład 3. Zmiana pozycji komunikatu

- **Domyślnie** komunikaty typu Toast są prezentowane jako wycentrowane na **dole** ekranu.
- Jednakże można wpłynąć na pozycję komunikatu wykorzystując następujące metody:

```
void setGravity (int gravity, int xOffset, int yOffset)

void setMargin (float horizontalMargin, float verticalMargin)
```

Toast

Przykład 3. Zmiana pozycji komunikatu

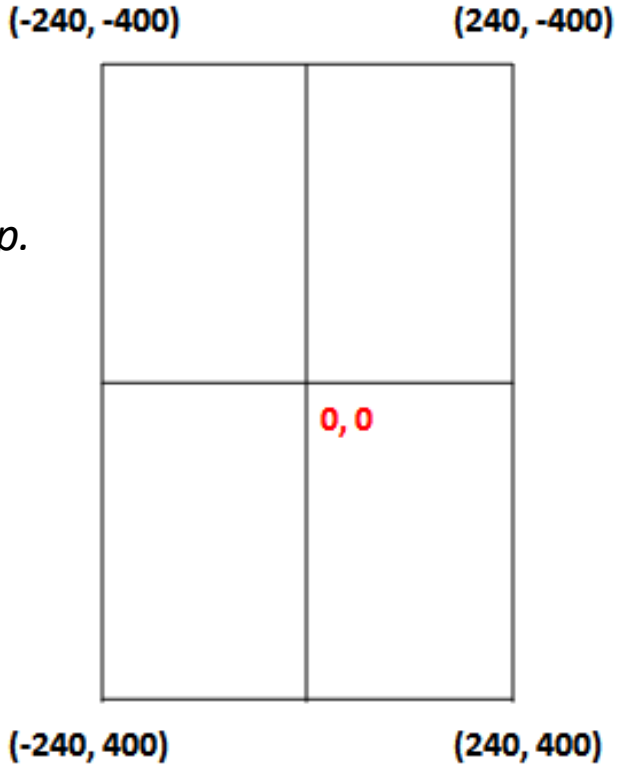
```
void setGravity (int gravity, int xOffset, int yOffset)
```

(Przy założeniu rozdzielczości ekranu **480x800**)

gravity: Ogólne położenie. Przykładowe wartości:
Gravity.CENTER, Gravity.TOP, Gravity.BOTTOM, itp.

xOffset: Przesunięcie poziome -240,…,0,…240
left, center, right

yOffset: Przesunięcie pionowe -400,…,0,…400
top, center, bottom

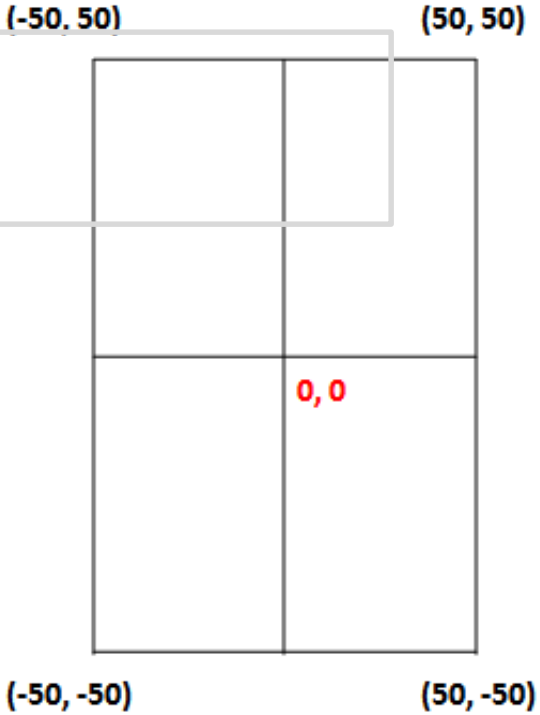


Toast

Przykład 3. Zmiana pozycji komunikatu

- Punkt (0,0) – *środek ekranu* – to miejsce gdzie pionowe i poziome linie na rysunku się krzyżują.
- Ekran jest dzielony po 50% względem środka
- Marginesy są wyrażane jako wartości procentowe: -50,…, 0, …, 50.

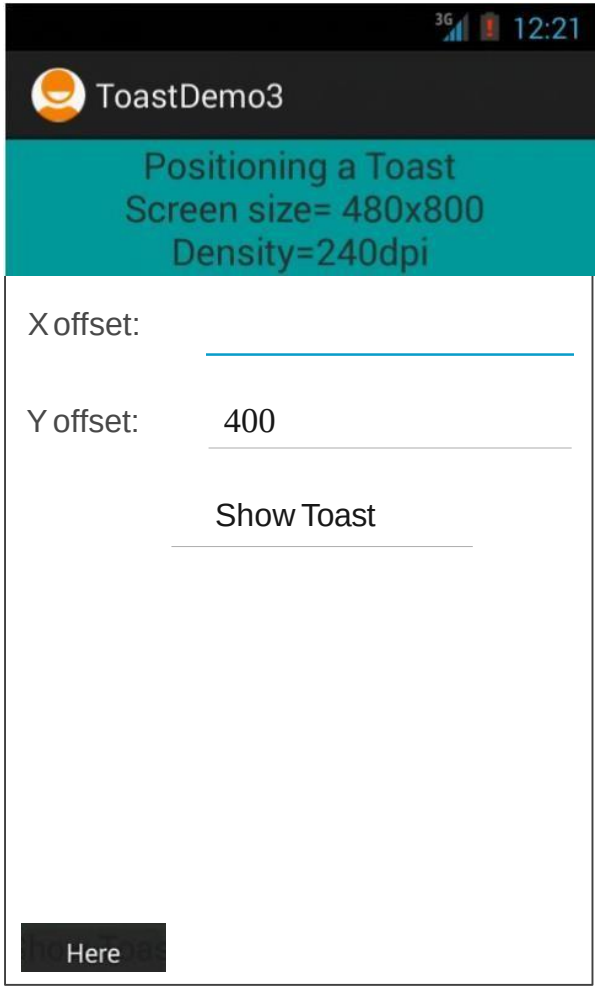
```
void setMargin (float horizontalMargin,
                float verticalMargin)
```

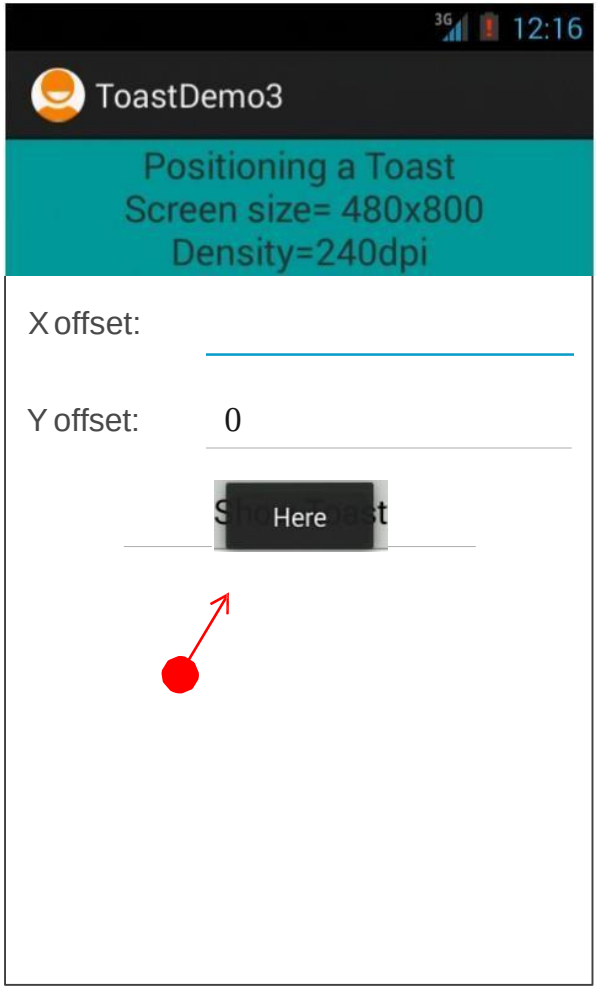


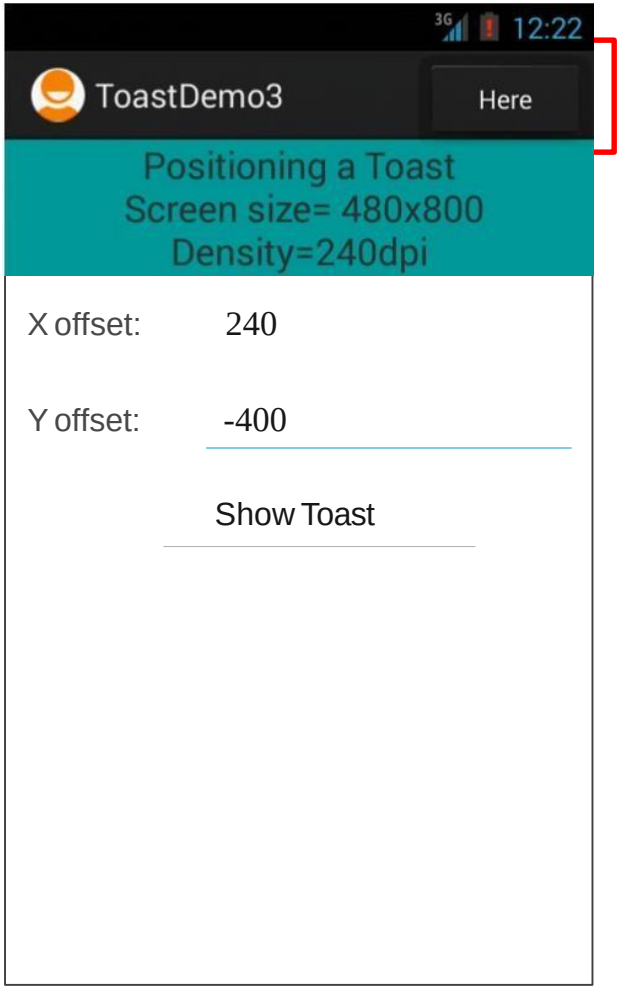
Uwaga! Para marginesów:
(-50, -50) reprezentuje lewy-dolny róg ekranu,
(0, 0) to jego środek, natomiast
(50, 50) to prawy górny róg.

Toast

Example 3. Re-positioning a Toast View



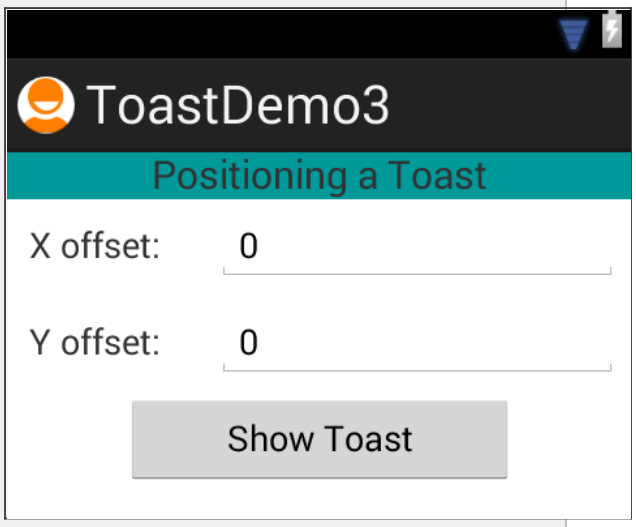




Toast

Przykład 3. Układ XML: activity_main.xml

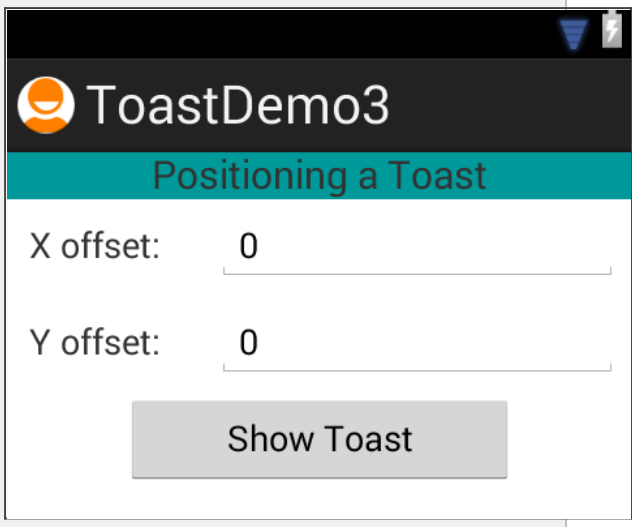
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView android:id="@+id/txtCaption"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff009999"
        android:gravity="center"
        android:text="Positioning a Toast"
        android:textSize="20sp" />

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


        android:padding="10dp" >

        <TextView android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:text=" X offset: "
            android:textSize="18sp" />

 <EditText
            android:id="@+id/txtXCoordinate“
```



Toast

Przykład 3. Układ XML: activity_main.xml

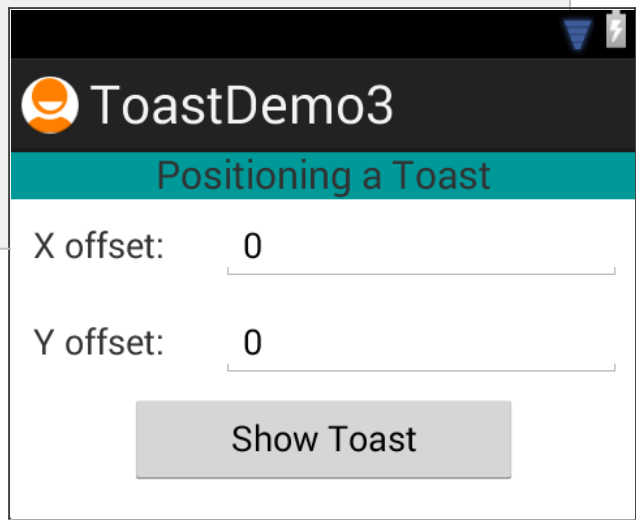
```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:inputType="numberSigned"
            android:text="0"
            android:textSize="18sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp" >

        <TextView android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:text=" Y offset: "
            android:textSize="18sp" />


        <EditText
            android:id="@+id/txtYCoordinate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:inputType="numberSigned"
            android:text="0"
            android:textSize="18sp" />
    </LinearLayout>
```



Toast

Przykład 3. Układ XML: activity_main.xml

```
    <Button android:id="@+id/btnShowToast"
        android:layout_width="200dp"
        android:layout_height="wrap_content"

        android:layout_gravity="center"
        android:text=" Show Toast " >
    </Button>

</LinearLayout>
```



Toast

Przykład 3. MainActivity: ToastDemo3.java

```
public class ToastDemo3 extends Activity {
   EditText  txtXCoordinate;
   EditText  txtYCoordinate;
   TextView txtCaption;

   Button btnShowToast;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activty_main);

1 →   // bind GUI and Java controls
      txtCaption = (TextView) findViewById(R.id.txtCaption);
      txtXCoordinate = (EditText) findViewById(R.id.txtXCoordinate);
      txtYCoordinate = (EditText) findViewById(R.id.txtYCoordinate);
      btnShowToast = (Button) findViewById(R.id.btnShowToast);

2 →   // find screen-size and density(dpi)
      int dpi  = Resources.getSystem().getDisplayMetrics().densityDpi; int
      width=  Resources.getSystem().getDisplayMetrics().widthPixels; int
      height = Resources.getSystem().getDisplayMetrics().heightPixels;
      txtCaption.append("\n Screen size= " + width + "x" + height
                      +" Density=" + dpi + "dpi");
```

Toast

Przykład 3. MainActivity: ToastDemo3.java

```
      // show toast centered around selected X,Y coordinates
      btnShowToast.setOnClickListener(new OnClickListener() {
         @Override
         public void onClick(View v) {
            try {
               Toast myToast = Toast.makeText(getApplicationContext(),
                     "Here", Toast.LENGTH_LONG);

               myToast.setGravity( Gravity.CENTER,
                     Integer.valueOf(txtXCoordinate.getText().toString()),
                     Integer.valueOf(txtYCoordinate.getText().toString()));

               myToast.show();

            } catch (Exception e) {
               Toast.makeText(getApplicationContext(), e.getMessage(),
                     Toast.LENGTH_LONG).show();
            }
         }
      });


   }// onCreate

}// class
```

3

Toast

Przykład 3. MainActivity: ToastDemo3.java

Komentarz

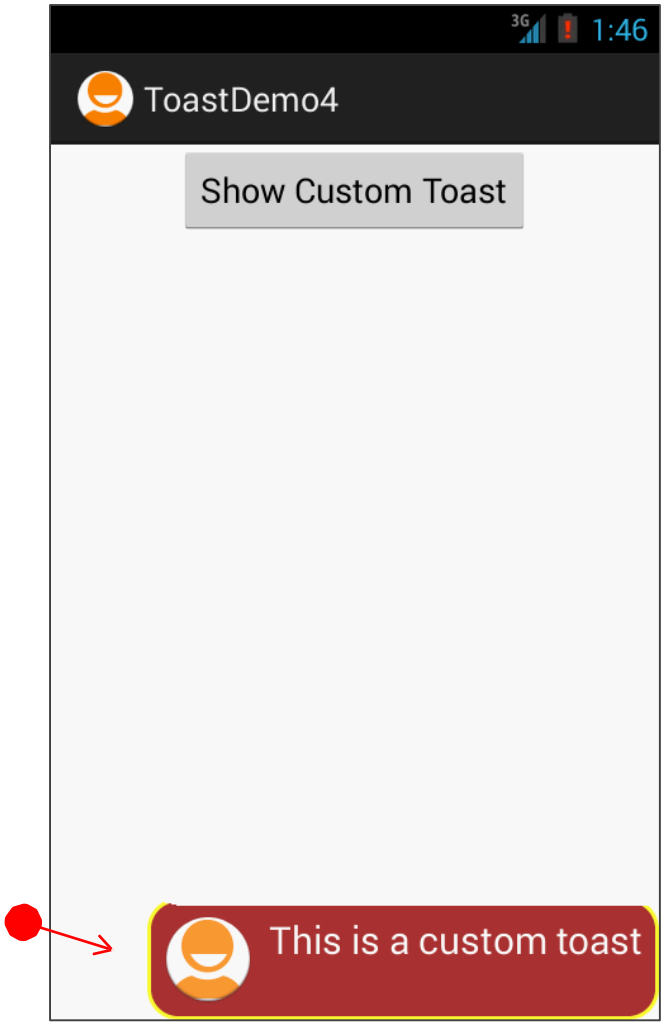
1. Obiekty GUI są wiązane z określonymi komponentami Java.
   Po kliknięciu przycisku komunikat typu Toast jest prezentowany.

2. Wywołanie `Resources.`*`getSystem()`*`.`*`getDisplayMetrics()`*
   wykorzystywane jest by określić wielkość ekranu (wysokość, szerokość) w pikselach, jak również gęstość upakowania pikseli.

3. Instancja klasy Toast jest tworzona przy użyciu metody *makeText*.
   Wywołanie metody `setGravity` jest wykorzystywane by wskazać współrzędne (X,Y) gdzie komunikat typu Toast ma zostać wyświetlony.
   Współrzędne X i Y odwołują się do położenia rzeczywistych pikseli na ekranie urządzenia.

Przykład 4. A Custom-Made Toast View

Komunikaty typu Toast mogą zostać zmodyfikowane by wyświetlić dowolną kombinację kolorów, kształtów, tekstu, rysunków oraz tła.



By stworzyć własny komunikat typu Toast:

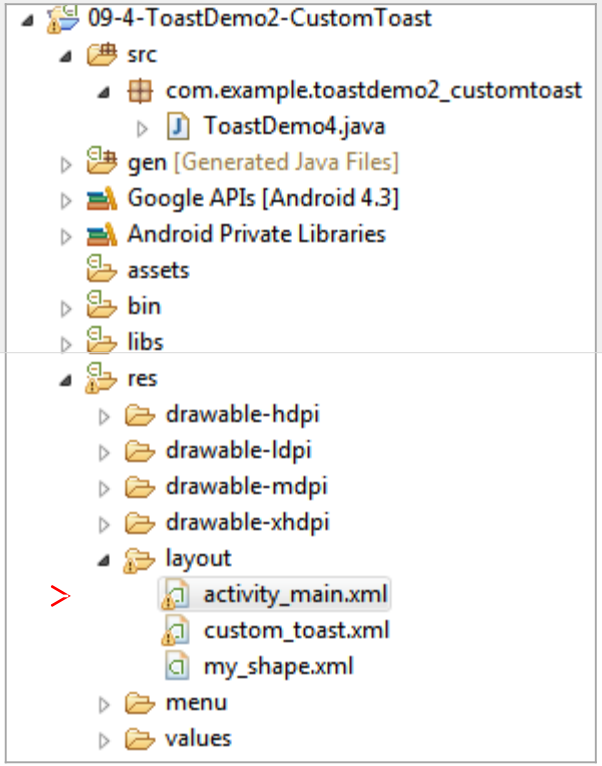
1. Zdefiniuj układ XML jaki ma zostać zaaplikowany do własnego obiektu Toast.
2. Prócz etykiety TextView reprezentującej dany komunikat, inne elementy GUI mogą zostać zastosowane.
3. Dokonaj inflacji widoku XML. Dołącz nowy widok do obiektu typu Toast za pomocą metody `setView()`.

Przykład bazuje na:
http://hustleplay.wordpress.com/2009/07/23/replicating-default-android-toast/
http://developer.android.com/guide/topics/ui/notifiers/toasts.html

Toast

Przykład 4. Układ XML - activity_main.xml

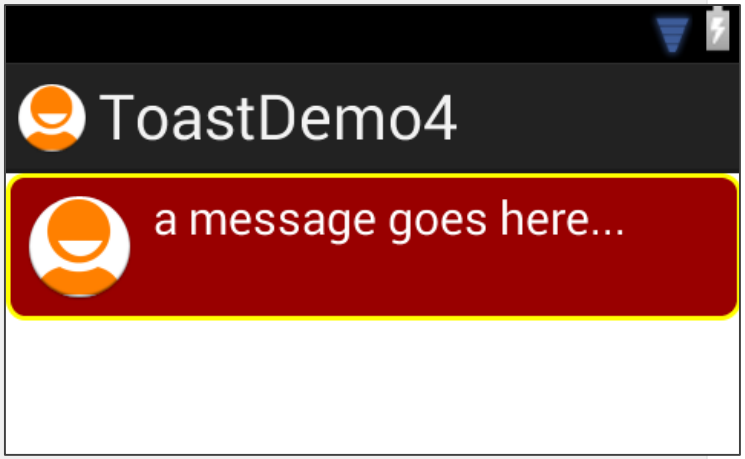
```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Butatnodnroid:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="showCustomToast"
        android:text="Show Custom Toast"
        android:layout_gravity="center"
        tools:context=".ToastDemo4" />

</LinearLayout>
```



Toast

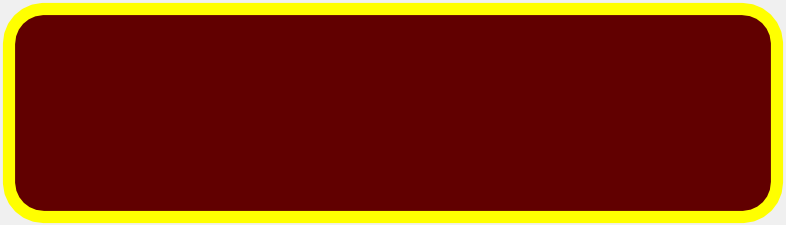
Przykład 4. Układ XML - custom_toast.xml

```
<?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:background="@layout/my_shape"

    android:orientation="horizontal"
    android:padding="8dp" >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="8dp"
        android:src="@drawable/ic_launcher" />

    <TextView android:id="@+id/toast_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="a message goes here..."
        android:textColor="#ffffffff"
        android:textSize="20sp" />

</LinearLayout>
```



Przykład 4. Układ XML - my_shape.xml

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

    <stroke android:width="2dp"
        android:color="#ffffff00" />

    <solid android:color="#ff990000" />

    <padding
        android:bottom="4dp"
        android:left="10dp"
        android:right="10dp"
        android:top="4dp" />

    <corners android:radius="15dp" />

</shape>
```



Uwaga: Shape to element typu drawable jak prostokąt czy owal. Wykorzystuje atrybuty jak stroke (obramowanie) , solid (wewnętrzna część kształtu), narożniki, marginesy itp. Plik znajduje się w katalogu **res/layout**.

Toast

Przykład 4. MainActivity - ToastDemo4.java

```
public class ToastDemo4 extends Activity {

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

    }//onCreate
    public void showCustomToast(View v){
      // ////////////////////////////////////////////////
      // this fragment creates a custom Toast showing
      // image + text + shaped_background
      // triggered by XML button's
      android:onClick=...

      Toast customToast = makeCustomToast(this);

      customToast.show();

    }
```

Toast

Przykład 4. MainActivity - ToastDemo4.java

```
   protected Toast makeCustomToast(Context context) {
      // Reference:
      //
      http://developer.android.com/guide/topics/ui/notifiers/toasts.html
      LayoutInflater inflater = getLayoutInflater();
1 →
      View layout = inflater.inflate( R.layout.custom_toast, null);

      TextView text = (TextView) layout.findViewById(R.id.toast_text);
      text.setText("This is a custom toast");

      Toast toast = new Toast(context);
2 →   toast.setMargin(50,-50); //lower-right
      corner toast.setDuration(Toast.LENGTH_LONG);

3 →   toast.setView(layout);

      return toast;

   }//makeCustomToast

}//ToastDemo2
```

Toast

Przykład 4. MainActivity - ToastDemo4.java

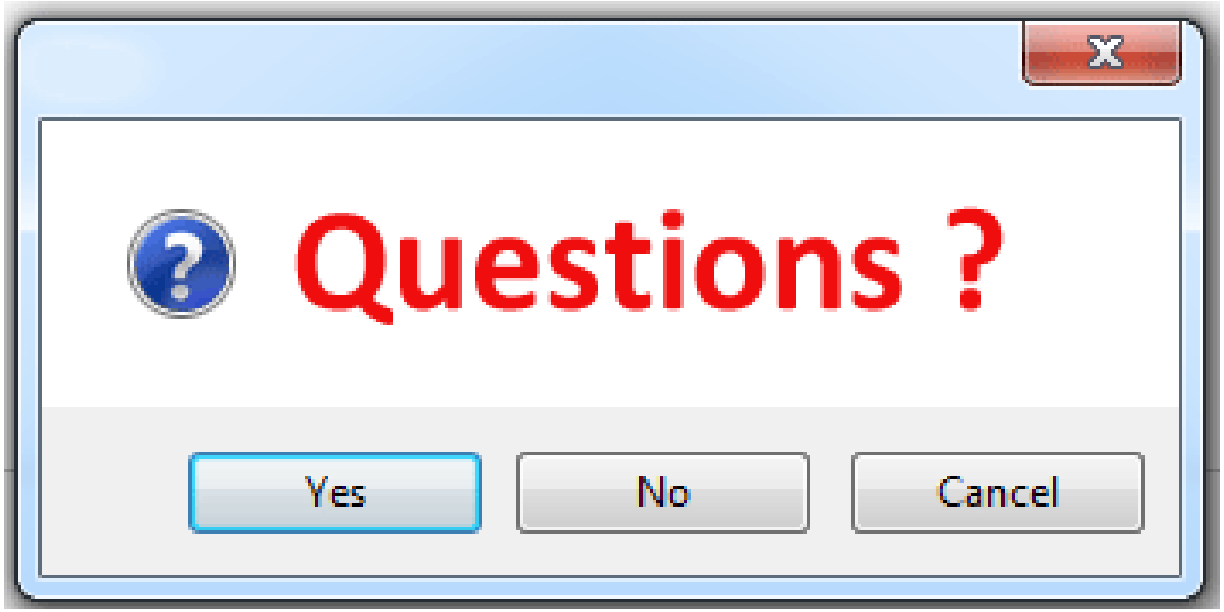
Komentarz

1. Po procesie inflacji układu XML, pobierana jest referencja do etykiety TextView która przechowuje komunikat do wyświetlenia.

2. Obiekt typu toast jest pozycjonowany przy użyciu metod `setMargin()` do dolnego-lewego rogu ekranu (50, -50).

3. Stworzony widok podpinany jest do obiektu typu Toast poprzez metodę `.setView()`.

Tworzenie dialogów i wykorzystanie klasy Toast



Dialogi

DODATEK A.

Shape

W pliku XML definiuje figurę geometryczną .

Dokumentacja:
http://developer.android.com/reference/android/graphics/drawable/shapes/Shape.html

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

http://developer.android.com/reference/android/graphics/drawable/ShapeDrawable.html

```
<?xml version="1.0" encoding="utf-8"?>
<Shape
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"  android:shape=["rectangle"| "oval" | "line" |
    "ring"] >
    <corners android:radius="integer"
        android:topLeftRadius="integer"
        android:topRightRadius="integer"
        android:bottomLeftRadius="integer"
        android:bottomRightRadius="integer"
        />
    <gradient android:angle="integer"
        android:centerX="integer"
        android:centerY="integer"
        android:centerColor="integer"
        android:endColor="color"
        android:gradientRadius="integer"
        android:startColor="color"
        android:type=["linear" | "radial" |
        "sweep"]  android:useLevel=["true" |
        "false"] />
    <padding
        android:left="integer"
        android:top="integer"
        android:right="integer"
        android:bottom="integer"
        />
    <size android:width="integer"
        android:height="integer"
        />
    <solid
        android:color="color" />
    <stroke android:width="integer"
        android:color="color"
        android:dashWidth="integer"
        android:dashGap="integer"
        />
</shape>
```