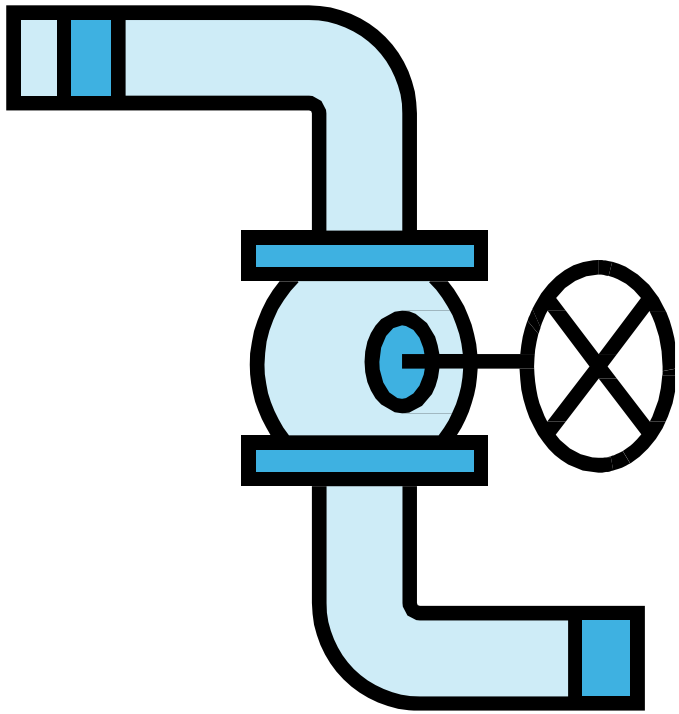
Pracownia aplikacji mobilnych

Powiązanie XML z kodem JAVA - elementy

Powiązanie układu z kodem Java

Należy „połączyć” elementy XML i odpowiadające im obiekty w ramach instancji klasy aktywności napisanej w Javie.

XLM Layout
<xml….
. . .
. . .
</xml>



JAVA code
public class ….
{
. . .
. . .
}

Powiązanie układu z kodem Java

Załóżmy, że GUI zostało stworzone w *res/layout/main.xml*. Ten układ może zostać powiązany z daną aktywnością poprzez instrukcję:

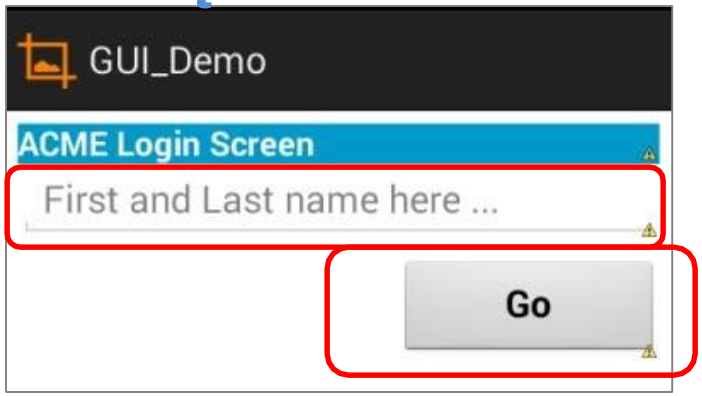
```
setContentView(R.layout.main);
```

Poszczególne widżety, jak np. *myButton* mogą zostać zaadresowane z pomocą polecenia `findViewByID(...)` tzn:

```
Button btn= (Button) findViewById(R.id.myButton);
```

gdzie **R** jest klasą automatycznie wygenerowaną by możliwe było śledzenie zasobów dostępnych dla danej aplikacji. W szczególności **R.id...** stanowi kolekcję widżetów zdefiniowanych w ramach interfejsu/ów XML.

Powiązanie układu z kodem Java




```
<!- XML LAYOUT -->
<LinearLayout
     android:id="@+id/myLinearLayout"
     ... >

     <TextView
          android:text="ACME Login Screen"
          ...  />

     <EditText
          android:id="@+id/edtUserName"
          ... />

     <Button
          android:id="@+id/btnGo"
          ...  />

</LinearLayout>
```

Java code

```
package csu.matos.gui_demo;
import android…;

public class MainActivity extends Activity {

    EditText edtUserName;

    Button   btnGo;

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

     setContentView(R.layout.activity_main);

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

     btnGo = (Button) findViewById(R.id.btnGo);
     ...

   }

  ...
}
```

Powiązanie układu z kodem Java

Dodawanie nasłuchiwaczy do widżetów

W tym momencie przycisk btn może zostać wykorzystany np. poprzez zdefiniowanie dla niego reakcji na kliknięcie:

```
btn.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        updateTime();
    }
});

private void updateTime() {
    btn.setText(new Date().toString());
}
```

Czym jest kontekst?

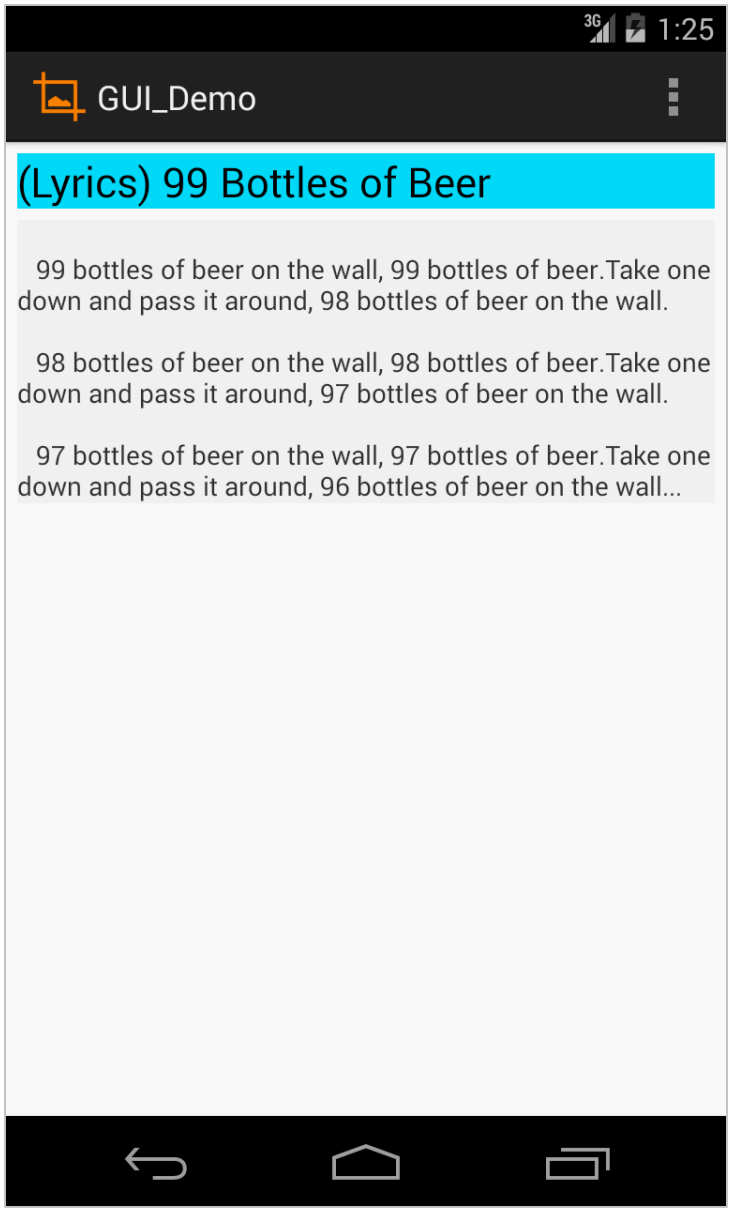
Dla Androida **Context** definiuje tzw. **środowisko** na którym aplikacja może tworzyć i używać zasobów.

- Kiedy widżet jest tworzony, jest mu przyporządkowany określony kontekst. Poprzez afiliację do tego środowiska, ma on wówczas dostęp do innych składowych hierarchii do której został przyporządkowany.

- Dla prostej aplikacji składającej się wyłącznie z jednej aktywności np. MainActivity metoda **getApplicationContext()** oraz referencja **MainActivity.this** zwracają ten sam rezultat.

- Aplikacja ma jednak zwykle **wiele aktywności**. W takiej sytuacji dostępny jest jeden globalny kontekst aplikacji oraz po jednym kontekście dla każdej z aktywności, każdy umożliwiający dostęp do zasobów zdefiniowanych w ramach tego kontekstu.

Etykiety

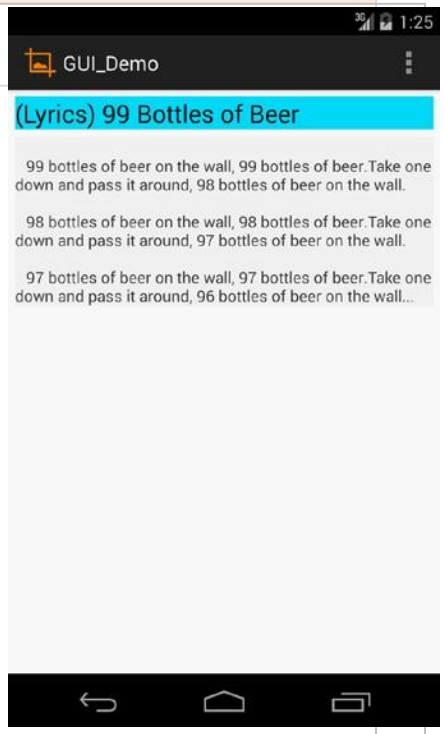


- Etykieta w kontekście Androida nazywana jest **TextView**.
- Zwykle wykorzystywana jest do prezentacji danych tekstowych bądź opisywania komponentów aplikacji.
- Etykiety nie umożliwiają edycji tekstu przez użytkownika.
- Tekst może zawierać pewne znaki specjalne jak np. **\n** (nowa linia)
- Można również wykorzystać formatowanie HTML:

```
Html.fromHtml("<b>bold</b> string")
```

Etykiety - przykład

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="6dp" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/holo_blue_bright"
        android:text="(Lyrics) 99 Bottles of Beer"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="6dp"
        android:background="@color/gray_light"
        android:text="\n\t99 bottles of beer on the wall, 99 bottles of beer.Take one down and
pass it around, 98 bottles of beer on the wall.\n\n\t98 bottles of beer on the wall, 98 bottles
of beer.Take one down and pass it around, 97 bottles of beer on the wall. \n\n\t97 bottles of
beer on the wall, 97 bottles of beer.Take one down and pass it around, 96 bottles of beer on
the wall... "
        android:textSize="14sp" />

</LinearLayout>
```



Przyciski

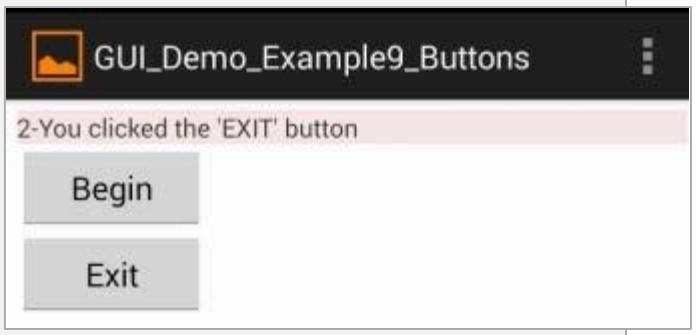
- Widżet **Button** jest graficzną reprezentacją standardowego przycisku.
- **Button** jest podklasą **TextView**. Dlatego formatowanie przycisku przebiega podobnie jak formatowanie **TextView**.
- Można zmienić domyślny wygląd przycisku poprzez realizację własnej specyfikacji *drawable.xml (podanej jako „tło")*. Można wówczas wskazać kształt, kolor, obramowanie, wierzchołki, gradient oraz wygląd poszczególnych stanów (naciśnięty, posiada focus).

```
<Button
        android:id="@+id/btnClickMeNow"
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="5dp“
        android:gravity="center"
        android:padding="5dp"
        android:text="Click Me Now!"
        android:textColor="#ffff0000"
        android:textSize="20sp"
        android:textStyle="bold" />
```

Podłączanie wielu przycisków

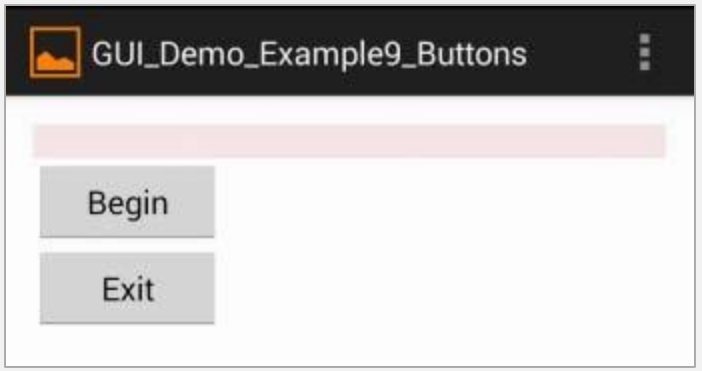
```
public class MainActivity extends Activity implements OnClickListener {
    TextView txtMsg;
    Button btnBegin;
    Button btnExit;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main );

        txtMsg = (TextView) findViewById(R.id.txtMsg);
        btnBegin = (Button) findViewById(R.id.btnBegin);
        btnExit = (Button) findViewById(R.id.btnExit);

        btnBegin.setOnClickListener(this);
        btnExit.setOnClickListener(this);
    }//onCreate


    @Override
    public void onClick(View v) {
        if (v.getId() == btnBegin.getId()) {
            txtMsg.setText("1-You clicked the 'BEGIN' button");
        }
        if (v.getId() == btnExit.getId()) {
            txtMsg.setText("2-You clicked the 'EXIT' button");
        }

    }//onClick
}
```

Proszę zwrócić uwagę na implementację interfejsu **OnClickListener**, jako alternatywną wersje obsługi wielu przycisków. Metoda **onClick** sprawdza, który z przycisków jest źródłem zdarzenia i wykonuje zadaną akcję.



Podłączanie wielu przycisków

```
<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:orientation="vertical"
    android:padding="6dp" >

    <TextView
        android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#88eed0d0" />

    <Button
        android:id="@+id/btnBegin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="5"
        android:text="Begin" />

    <Button
        android:id="@+id/btnExit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="5"
        android:text="Exit" />

</LinearLayout>
```



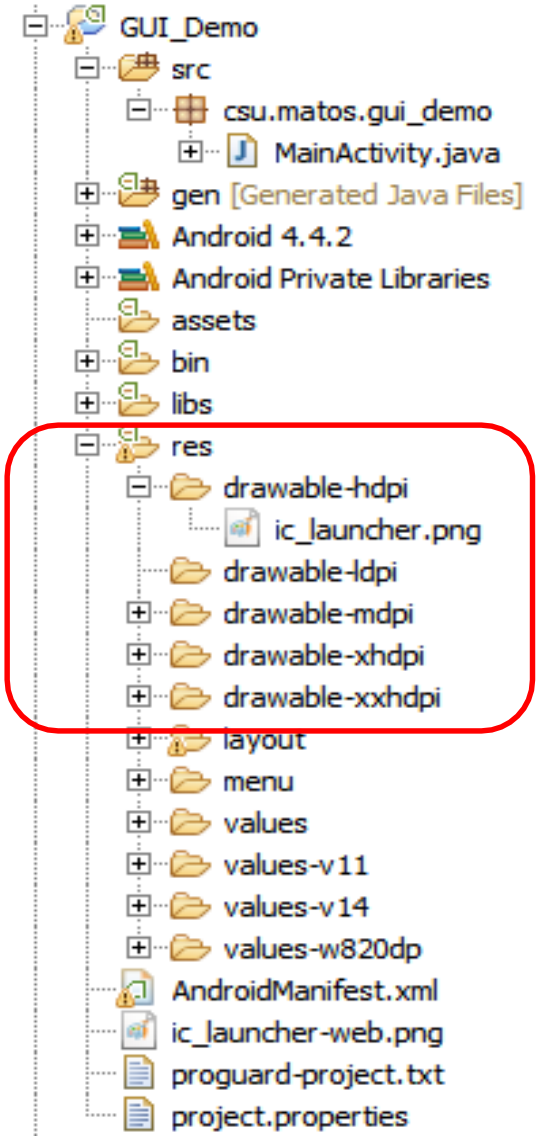
ImageView i ImageButton

- **ImageView** oraz **ImageButton** pozwalają na umieszczenie grafik (gif, jpg, png, etc).

- Stanowią analogię dla komponentów odpowiednio *TextView* oraz *Button*

- Każdy taki widżet posiada atrybut
  `android:src` lub
  `android:background`
  (w pliku XML) pozwalający określić używany obrazek.

- Obrazki przechowywane są w folderze **res/drawable** (opcjonalnie można dodać różne wersje tego samego obrazka w katalogach z przedrostkami *medium, high, x-high, xx-high i xxx-high*). Szczegóły dostępne są na:
  http://developer.android.com/design/style/iconography.html



ImageView i ImageButton

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="6dp"
    android:orientation="vertical" >

    <ImageButton
        android:id="@+id/imgButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" >
    </ImageButton>

    <ImageView
        android:id="@+id/imgView1"
        android:layout_width="200dp"
        android:layout_height="150dp"
        android:scaleType="fitXY"
        android:src="@drawable/flowers1" >
    </ImageView>

</LinearLayout>
```

ImageView i ImageButton

Zwykły przycisk typu **Button** też może wyświetlać grafikę, ale **ImageButton** ma więcej opcji

```
<LinearLayout
    . . .

 <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawableLeft="@drawable/ic_launcher"
        android:gravity="left|center_vertical"
        android:padding="15dp"
        android:text="Click me" />

</LinearLayout>
```

Jak Android wykorzystuje ikony?

Ikony – małe obrazki reprezentujące aplikację bądź jej część. Mogą pojawiać się w wielu miejscach aplikacji takich jak:

- Ekran domowy
- Launcher
- Menu główne
- Action Bar
- Pasek stanu
- Zakładkach
- Dialogach
- Listach

mdpi (761 bytes)
1x = 48 x 48 pixels
BaseLine

hdpi (1.15KB)
1.5x = 72 x 72 px

x-hdpi (1.52KB)
2x = 96 x 96 px

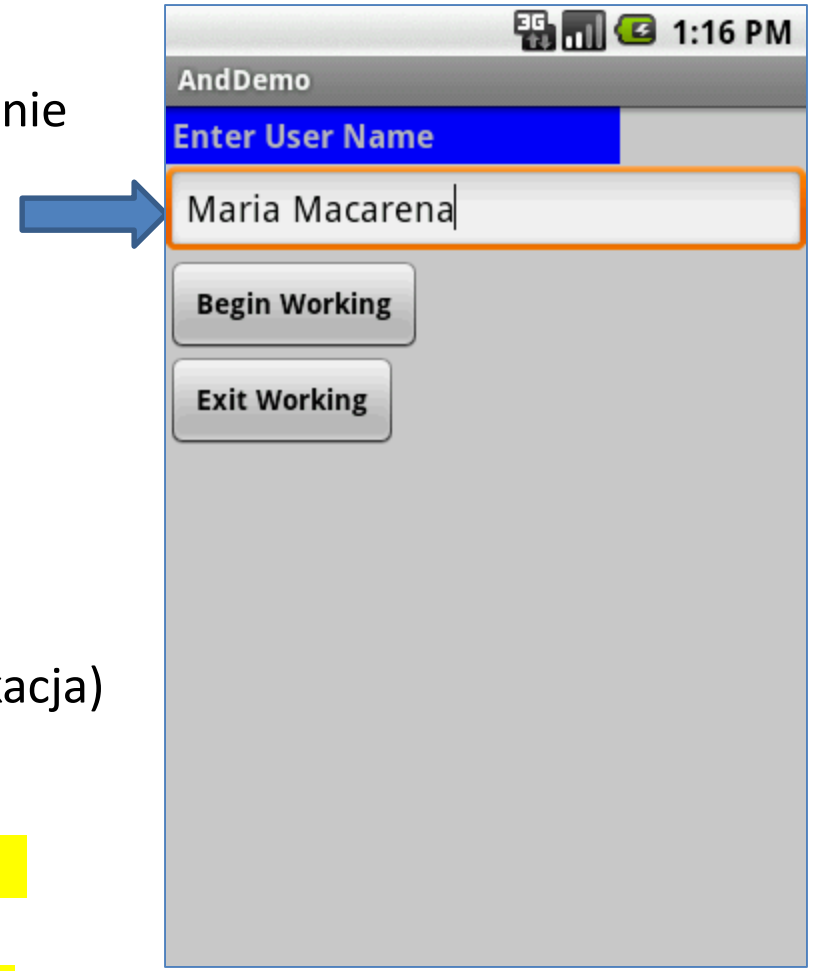
xx-hdpi (2.47KB)
3x = 144 x 144 px

Więcej informacji dostępnych jest pod:
http://developer.android.com/design/style/iconography.html

WSKAZÓWKA: Wiele witryn oferuje darmową konwersję między różnymi formatami graficznymi:
http://www.prodraw.net/favicon/index.php
http://converticon.com/

Pola tekstowe

- Widżet EditText jest rozszerzeniem TextView umożliwiającym wprowadzenie tekstu przez użytkownika.



- Prócz zwykłego tekstu można używać podstawowych znaczników HTML by uzyskać pogrubienie, podkreślenie, kursywę. Umożliwia to metoda: **Html.fromHtml(html_text)**

- Dostęp do tekstu (pobieranie, modyfikacja) możliwy jest dzięki metodom:

```
txtBox.setText("tekst")
```

```
txtBox.getText().toString()
```

Pola tekstowe

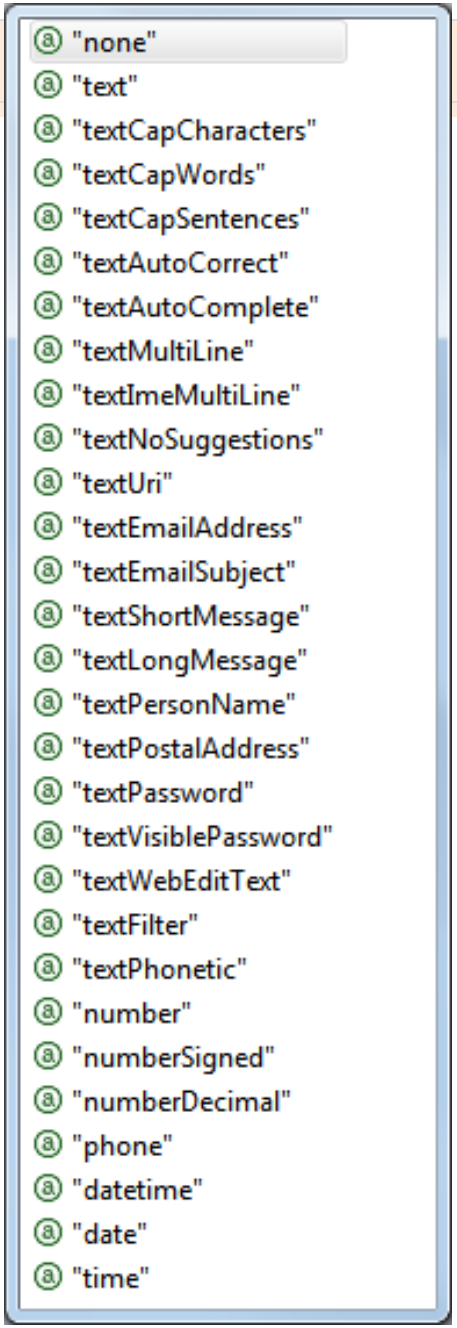
Format wprowadzania danych

Komponent EditText może być ustawiony by przyjmował tylko określone typu danych: numery (ze znakiem i częścią ułamkową bądź bez), numery telefonów, daty, czas, odnośniki itp.

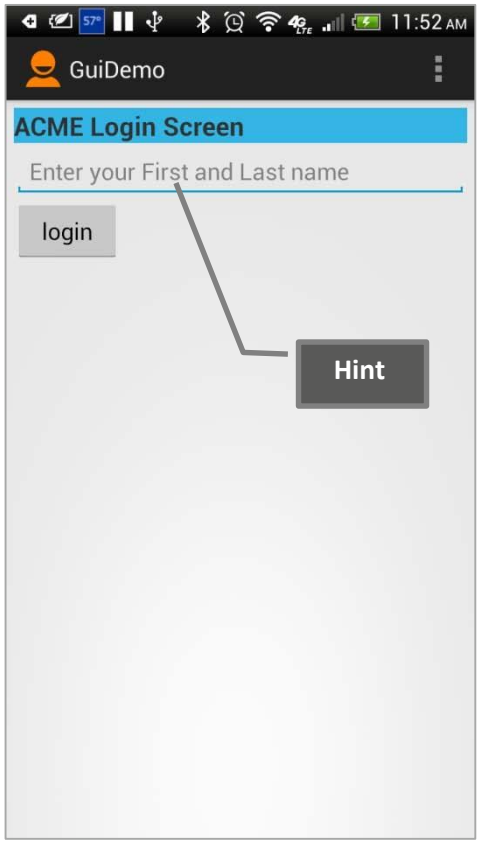
Ustawienie EditText by akceptował tylko wybrany typ danych umożliwia klauzula XML:

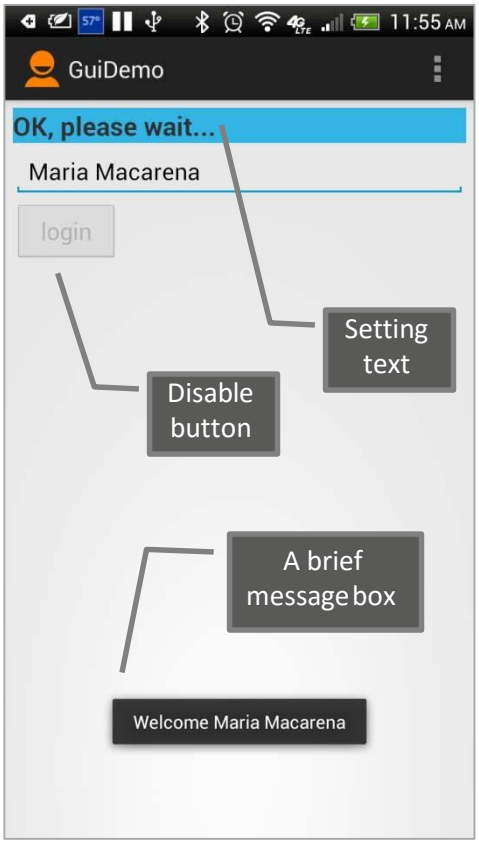
```
android:inputType="choices"
```

gdzie **choices** zawiera dowolne wartości pokazane na slajdzie. Wartości można łączyć, przykładowo zapis `textCapWords|textAutoCorrect` akceptuje słowa zaczynające się z wielkiej litery i umożliwia automatyczną kontrolę pisowni.



Ekran logowania - przykład





Zdjęcia z HTC-One

Ekran logowania - przykład

Układ 1 z 2

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="6dp" >

    <TextView
        android:id="@+id/txtLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_blue_light"
        android:text="@string/ACME_Login_Screen"
        android:textSize="20sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/edtUserName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="2dp"
        android:hint="@string/Enter_your_First_and_Last_name"
        android:inputType="textCapWords|textAutoCorrect"
        android:textSize="18sp" >
        <requestFocus />
    </EditText>
```

Ekran logowania - przykład

Układ 2 z 2

```
    <Button
        android:id="@+id/btnLogin"
        android:layout_width="82dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="2dp"
        android:text="@string/login" />
</LinearLayout>
```

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- this is the res/values/strings.xml file -->
<resources>

    <string name="app_name">GuiDemo</string>
    <string name="action_settings">Settings</string>
    <string name="login">Login</string>
    <string name="ACME_Login_Screen">ACME Login Screen</string>
    <string name="Enter_your_First_and_Last_name">Enter your First and Last name</string>

</resources>
```

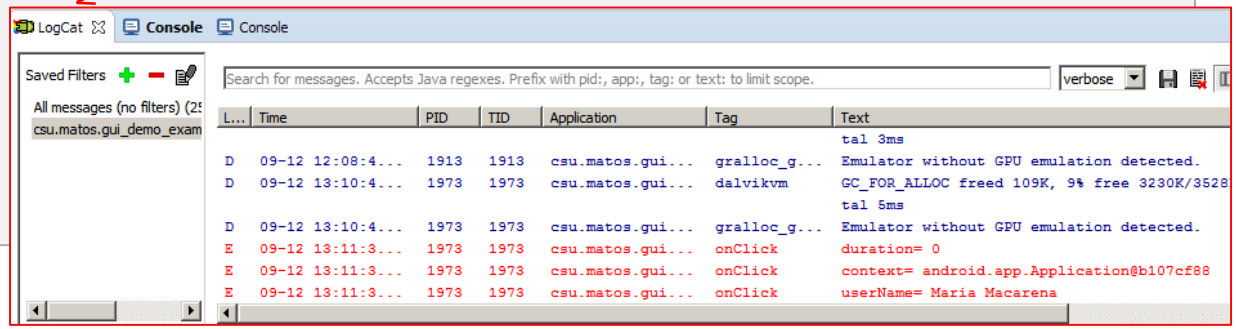
Ekran logowania - przykład

```
public class MainActivity extends ActionBarActivity {

 // class variables representing UI controls to be controlled from the Java program
 TextView txtLogin;
 EditText edtUserName;
 Button btnLogin;

 // variables used with the Toast message class
 private Context context;
 private int duration = Toast.LENGTH_SHORT;

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

  // show the login screen
  setContentView(R.layout.activity_main);
  context = getApplicationContext();

  // binding the UI's controls defined in "main.xml" to Java code
  txtLogin = (TextView) findViewById(R.id.txtLogin);
  edtUserName = (EditText) findViewById(R.id.edtUserName);
  btnLogin = (Button) findViewById(R.id.btnLogin);
```

Ekran logowania - przykład

```
// LISTENER: allowing the button widget to react to user interaction
btnLogin.setOnClickListener(new OnClickListener() {
 @Override
 public void onClick(View v) {

  String userName = edtUserName.getText().toString();

  Log.e("onClick ", "duration= " + duration);
  Log.e("onClick ", "context= " + context.toString());
  Log.e("onClick ", "userName= " + userName);

  if (userName.equals("Maria Macarena")) {
   txtLogin.setText("OK, please wait...");
   Toast.makeText(getApplicationContext(),
     "Welcome " + userName, duration).show();
   btnLogin.setEnabled(false);
  } else {
   Toast.makeText(context, userName + "  is not a valid USER",
     duration).show();
  }
 }
});// onClick

}// onCreate
```

Log.e jako debug – *usuń później*!!!



Ekran logowania - przykład

```
 @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;
 }

 @Override
 public boolean onOptionsItemSelected(MenuItem item) {
  // Handle action bar item clicks here. The action bar will
  // automatically handle clicks on the Home/Up button, so long
  // as you specify a parent activity in AndroidManifest.xml.
  int id = item.getItemId();
  if (id == R.id.action_settings) {
   return true;
  }
  return super.onOptionsItemSelected(item);
 }
}
```

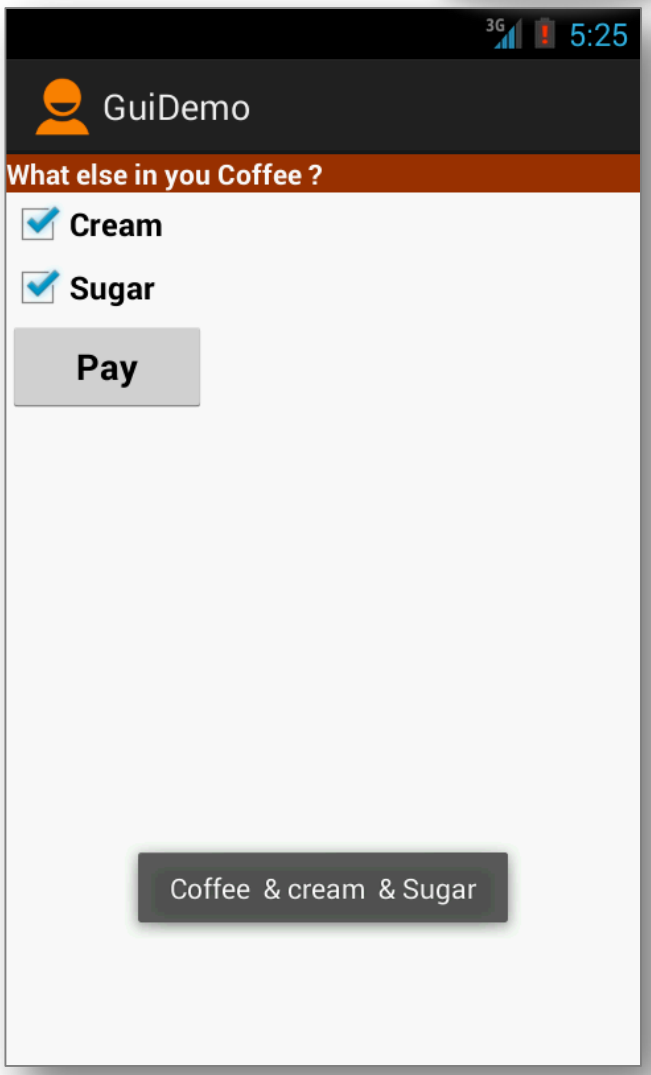
Pole typu CheckBox

Pole typu checkbox można interpretować jako **dwustanowy** przycisk który może być *zaznaczony* bądź *nie*.

Dany ekran może zawierać wiele niezależnie działających pól wyboru. W danym czasie więcej niż jeden checkbox może zostać zaznaczony.

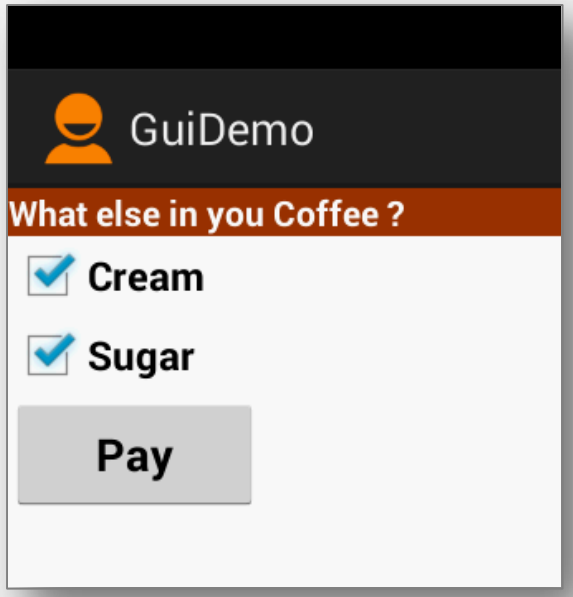
W przykładzie wykorzystano komponenty CheckBox do możliwości wyboru dodatków do kawy (cukier, śmietanka).

Po kliknięciu przycisku Pay użytkownikowi prezentowany jest komunikat zawierający wybraną kombinację zamówienia.



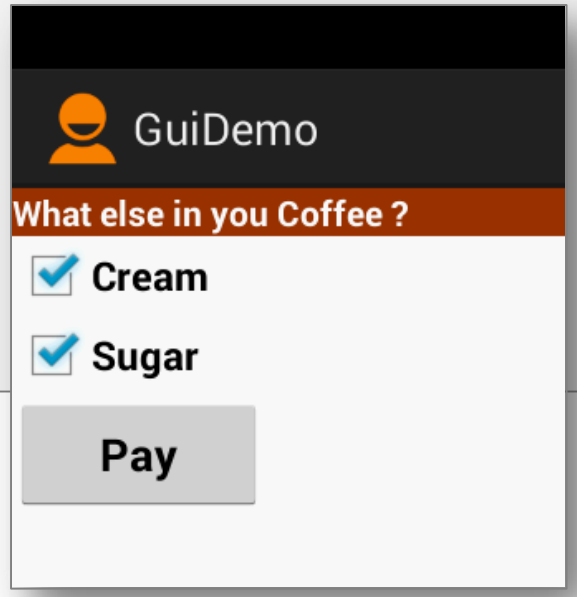
Checkbox - przykład

```
<?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:padding="6dp"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/labelCoffee"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff993300"
        android:text="@string/coffee_addons"
        android:textColor="@android:color/white"
        android:textStyle="bold" />

    <CheckBox
        android:id="@+id/chkCream"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cream"
        android:textStyle="bold" />
```



Checkbox - przykład

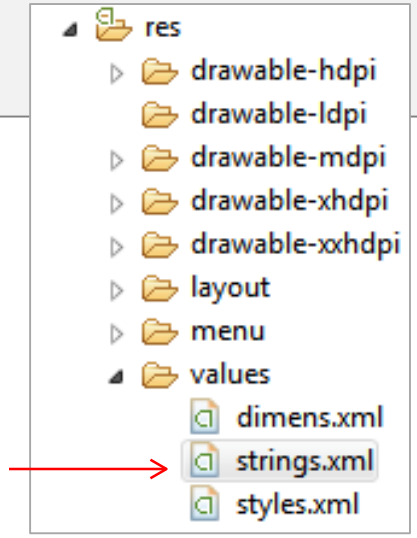
```
    <CheckBox
        android:id="@+id/chkSugar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/sugar"
        android:textStyle="bold" />

    <Button
        android:id="@+id/btnPay"
        android:layout_width="153dp"
        android:layout_height="wrap_content"
        android:text="@string/pay"
        android:textStyle="bold" />

</LinearLayout>
```



Checkbox – przykład [@string/…]

Zasoby: res/values/strings

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

    <string name="app_name">GuiDemo</string>
    <string name="action_settings">Settings</string>

    <string name="click_me">Click Me</string>
    <string name="sugar">Sugar</string>
    <string name="cream">Cream</string>
    <string name="coffee_addons">What else in your coffee?</string>
    <string name="pay">Pay</string>
</resources>
```



Checkbox - przykład

```
public class MainActivity extends Activity {

    CheckBox chkCream;
    CheckBox chkSugar;
    Button btnPay;

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //binding XMl controls with Java code
        chkCream = (CheckBox)findViewById(R.id.chkCream);
        chkSugar = (CheckBox)findViewById(R.id.chkSugar);
        btnPay = (Button) findViewById(R.id.btnPay);
```

Checkbox - przykład

```
 //LISTENER: wiring button-events-&-code
        btnPay.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
   String msg = "Coffee ";
   if (chkCream.isChecked()) {
      msg += " & cream ";
   }
   if (chkSugar.isChecked()){
      msg += " & Sugar";
   }
   Toast.makeText(getApplicationContext(),
                  msg, Toast.LENGTH_SHORT).show();
   //go now and compute cost...

   }//onClick
   });

  }//onCreate

}//class
```

Pole typu RadioButton

- Pole typu **RadioButton** (jak CheckBox) jest dwu-stanowym przyciskiem, który może być *zaznaczony* bądź *nie*.

- Zwykle pola tego typu są agregowane w kontener o nazwie **RadioGroup**. Zastosowanie tego kontenera powoduje, że pola wyboru zachowują się jako **wzajemnie wykluczające się selektory.** Oznacza to, że zmiana jednego z przycisków powoduje odznaczenie pozostałych.

- Właściwości dotyczących kroju, stylu czy koloru czcionki zarządzane są podobnie jak w przypadku etykiety TextView.

- Można wywołać metodę ***isChecked()*** by sprawdzić czy poszczególny przycisk jest zaznaczony, natomiast metoda **toggle()** odpowiada za zmianę jego stanu.

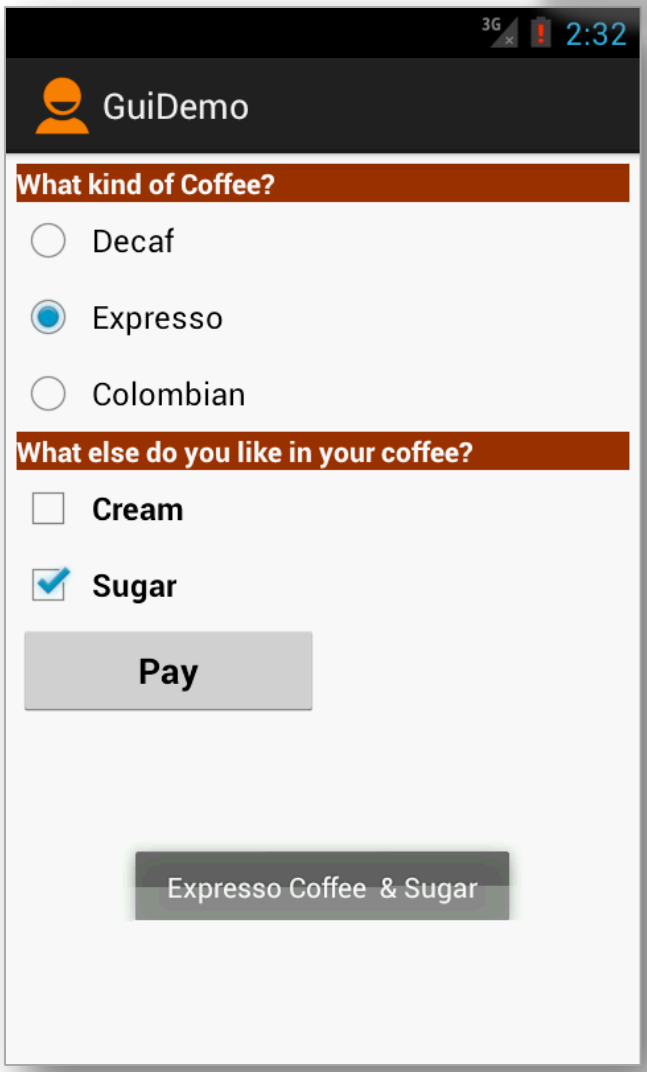
RadioButton - przykład

Przykład

*Rozszerzono poprzednią aplikację dodając komponent **RadioGroup** by umożliwić wybór konkretnego typu kawy.*

RadioGroup

Podsumowanie



RadioButton - przykład

Na podstawie poprzedniego przykładu – tylko nowe rzeczy są prezentowane

```
<TextView
      android:id="@+id/textView1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:background="#ff993300"
      android:text="@string/kind_of_coffee"
      android:textColor="#ffffff"
      android:textStyle="bold" />
```


```
<RadioGroup
      android:id="@+id/radioGroupCoffeeType"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" >

      <RadioButton
          android:id="@+id/radDecaf"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content
          "  android:text="@string/decaf" />

      <RadioButton
          android:id="@+id/radExpresso"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content
          "  android:text="@string/expresso"
          />

      <RadioButton
          android:id="@+id/radColombian"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content
          "  android:checked="true"
          android:text="@string/colombian" />
</RadioGroup>
```

RadioButton - przykład

```
public class MainActivity extends Activity {

   CheckBox chkCream;
   CheckBox chkSugar;
   Button btnPay;

   RadioGroup radCoffeeType;
     RadioButton radDecaf;
     RadioButton radExpresso;
     RadioButton radColombian;

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

      chkCream = (CheckBox) findViewById(R.id.chkCream);
      chkSugar = (CheckBox) findViewById(R.id.chkSugar);
      btnPay = (Button) findViewById(R.id.btnPay);

      radCoffeeType = (RadioGroup) findViewById(R.id.radioGroupCoffeeType);

      radDecaf = (RadioButton) findViewById(R.id.radDecaf);
      radExpresso = (RadioButton) findViewById(R.id.radExpresso);
      radColombian = (RadioButton)
      findViewById(R.id.radColombian);
```

RadioButton - przykład

```
        // LISTENER: wiring button-events-&-code
        btnPay.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                String msg = "Coffee ";
                if (chkCream.isChecked())
                    msg += " & cream ";
                if (chkSugar.isChecked())
                    msg += " & Sugar";

                // get selected radio button ID number
                int radioId = radCoffeeType.getCheckedRadioButtonId();

                // compare selected's Id with individual RadioButtons ID
                if (radColombian.getId() == radioId)
                    msg = "Colombian " + msg;
                // similarly you may use .isChecked() on each RadioButton
                if (radExpresso.isChecked())
                    msg = "Expresso " + msg;
                // similarly you may use .isChecked() on each RadioButton
                if (radDecaf.isChecked())
                    msg = "Decaf " + msg;

                Toast.makeText(getApplicationContext(), msg, 1).show();
                // go now and compute cost...
            }// onClick
        });
    }// onCreate

}// class
```

RadioButton - przykład

Wskazówka

```
radGroupradioId = (RadioGroup)findViewById(R.id.radioGroup1);

int radioId  = radGroupradioId.getCheckedRadioButtonId();

switch (radioId) {
  case R.id.radColombian: msg += " Colombian "; break;
  case R.id.radExpresso: msg += " Expresso "; break;
  case R.id.radDecaf: msg += " Decaf "; break;
}
```

Alternatywny przykład zarządzania **RadioGroup** – nie wymaga badania stanu każdego z przycisków z osobna.

Przydatne atrybuty XML i metody Java

By kontrolować focus (XML):

```
android:visibility      true/false – widoczność
android:background      color, image, drawable
<requestFocus />        ustaw domyślny focus
```

Metody Java

```
myButton.requestFocus()
myTextBox.isFocused()
myWidget.setEnabled()
myWidget.isEnabled()
```

Dodatek A. Wykorzystanie dyrektywy @string

Dobry programista Android **NIE** wprowadza bezpośrednio literałów znakowych w kodzie aplikacji.

Przykładowo, definiując komponent **TextView** by pokazać np. lokalizację przedsiębiorstwa, nie powinno się stosować zapisu `android:text="Cleveland"` (powoduje to też wyświetlenie ostrzeżenia **Warning** *`[I18N] Hardcoded string "Cleveland", should use @string resource`*)

Zamiast tego powinno stosować się 2-etapową procedurę:

1. Dodać literał np.. *headquarter* do **res/values/string.xml.**
   ```
   <string name="headquarter">Cleveland</string>
   ```

2. Aby odwołać się do stworzonego literału należy wykorzystać zapis *@string/headquarter*. W przypadku etykiety wystarczy zapis
   ```
   android:text="@string/headquarter"
   ```

Dlaczego?

Dodatek B.

Android Asset Studio

LINK: http://romannurik.github.io/AndroidAssetStudio/

Wspomniane narzędzie oferuje możliwość tworzenia różnych rodzajów ikon oraz innych elementów graficznych

Generator ikon	Inne generatory	Pozostałe narzędzia
Launcher icons Action bar and tab icons Notification icons Navigation drawer indicator Generic icons	Device frame generator Simple nine-patch gen.	Android Action Bar Style Generator Android Holo Colors Generator

Dodatek C. Wielkość elementów graficznych

P. Co to **dpi** (znane jako **dp, ppi**) ?

Skrót od *dots per inch*. Można go wyliczyć korzystając z następującego wzoru:

$$dpi = \sqrt{szerokośćPiksele^2 + wysokośćPiksele^2} / przekątnaCale$$

G1 (320x480)	155.92 dpi	(3.7 in)
Nexus (480x800)	252.15 dpi	
HTC One (1080x1920)	468 dpi	(4.7 in)
Samsung S4 (1080x1920)	441 dpi	(5.5 in)

Q. Jaka jest różnica między **dp, dip** oraz **sp**?

dp *Density-independent Pixels* – Abstrakcyjna metryka oparta na faktycznej gęstości upakowania pikseli ekranu. Należy wykorzystywać do każdego zasobu prócz czcionek.

sp *Scale-independent Pixels* – podobnie jak w przypadku dp, ale wykorzystywany do ustawienia wielkości **czcionki**.

Dodatek D. Rozdzielczość ekranu

Rozdzielczość ekranu – jak Android sobie z tym radzi?

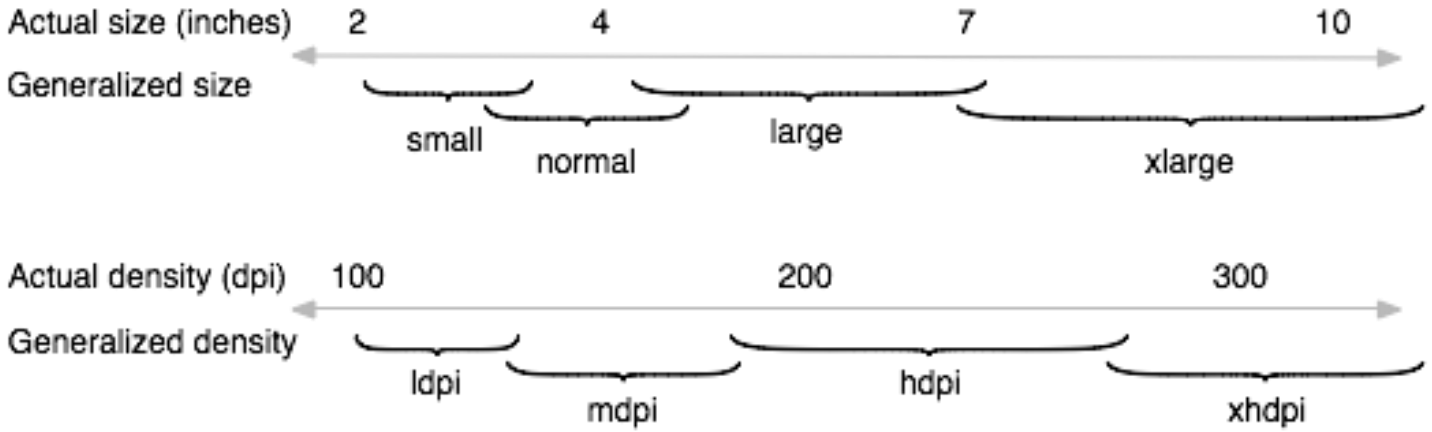
Poniżej znajdują się zapisy jak Android radzi sobie z klasyfikowaniem różnych rozdzielczości ekranu.

Zbiór czterech klas wielkości ekranu

xlarge ekran ma minimum 960dp x 720dp
large ekran ma minimum 640dp x 480dp
normal ekran ma minimum 470dp x 320dp
small ekran ma minimum 426dp x 320dp

Ogólne klasy gęstości:

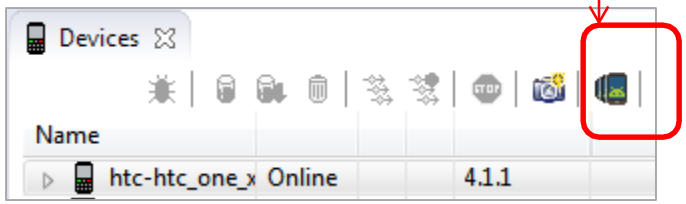
ldpi ~120dpi (low)
mdpi ~160dpi (medium)
hdpi ~240dpi (high)
xhdpi ~320dpi (extra-high)
xxhdpi ~480dpi (extra-extra-high)
Xxxhdpi ~640dpi (extra-extra-extra-high)

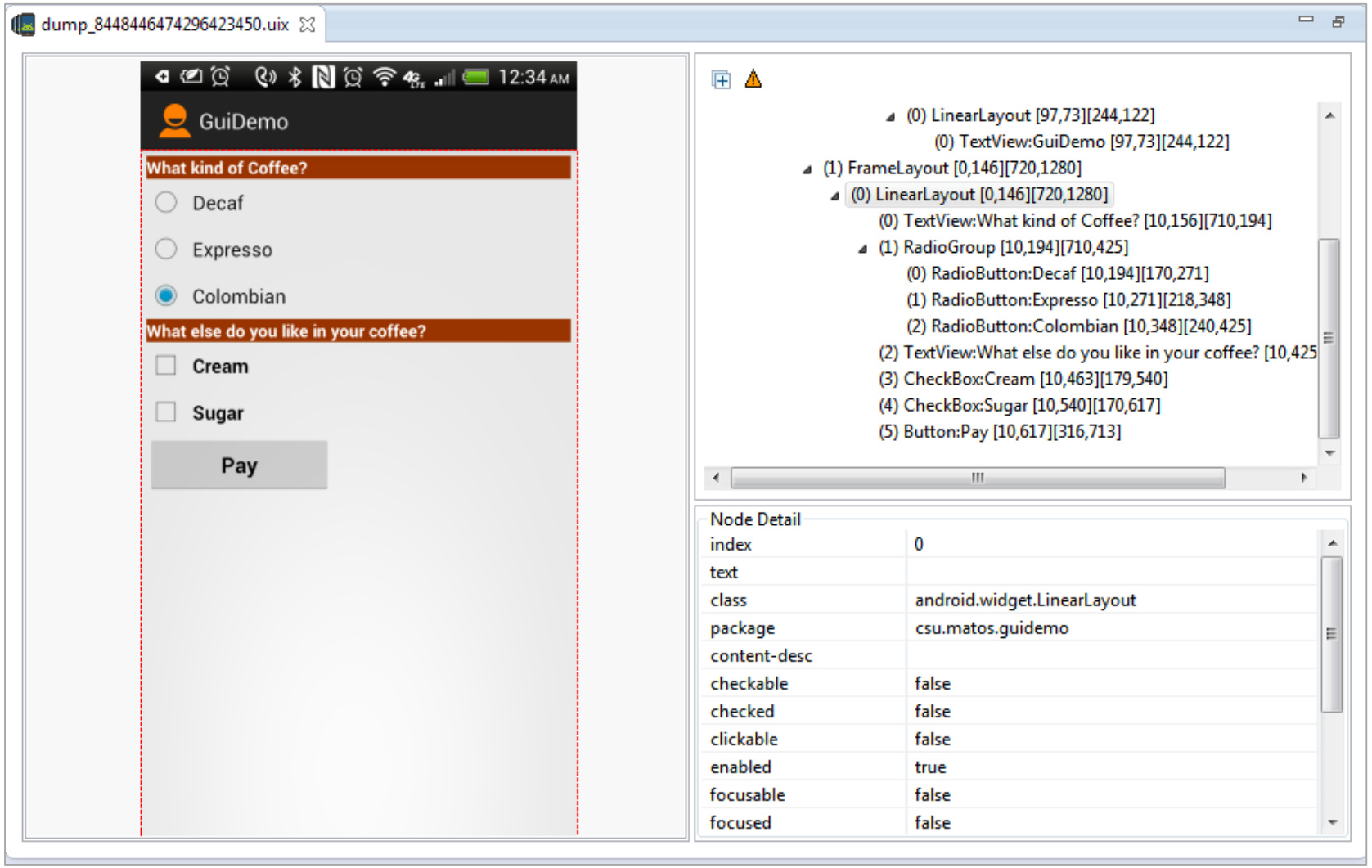


Źródło: http://developer.android.com/guide/practices/screens_support.html

Dodatek E. Hierarchy Viewer

Narzędzie HierarchyViewer umożliwia przejrzenie faktycznej hierarchii komponentów na interfejsie graficznym.





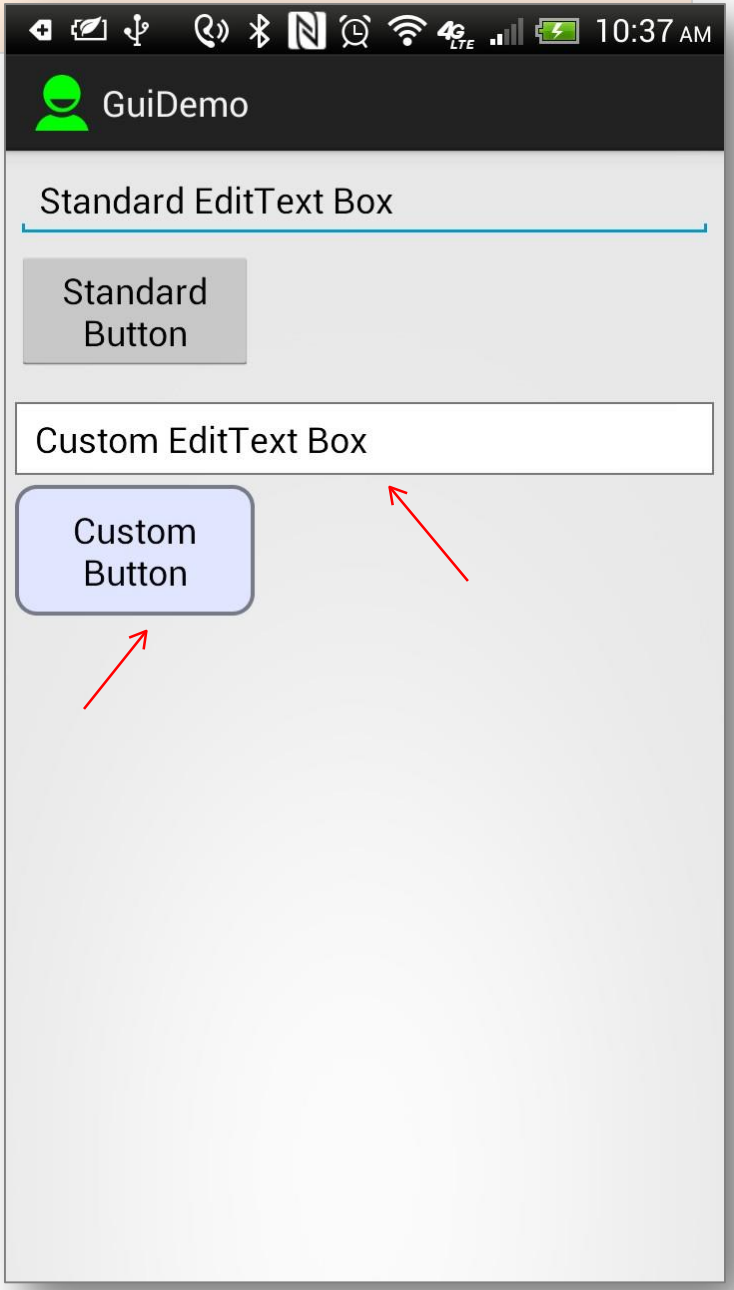
Dodatek F. Zmiana wyglądu widżetu

1. Zmiana wyglądu widżetu jest możliwa w bardzo prosty sposób. Można zmienić kształt, obramowanie, kolor, marginesy i inne właściwości.

2. Podstawowe figury to: prostokąt, elipsa, linia, pierścień.

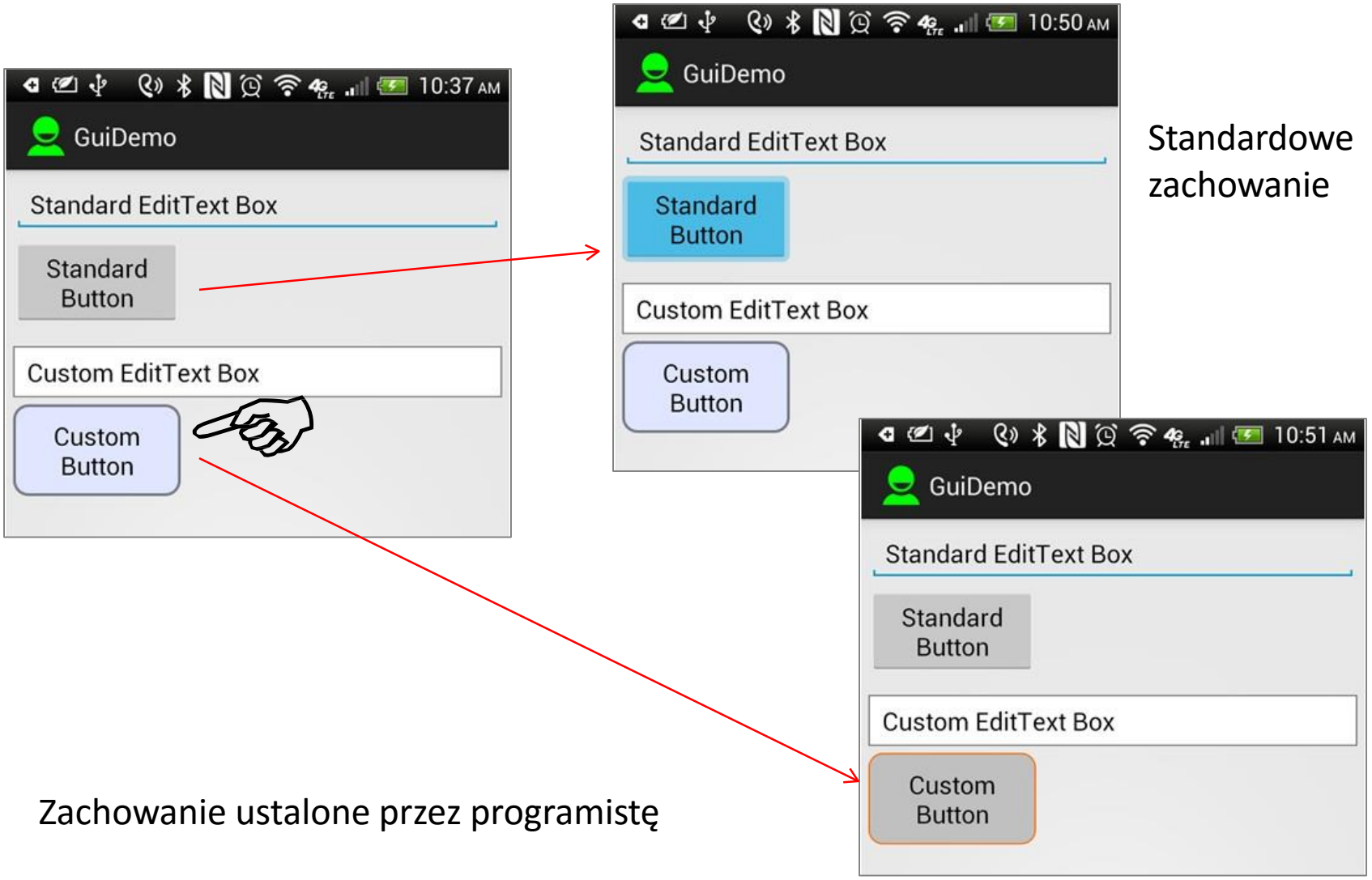
3. Prócz zmian typowo wizualnych w domyślnym stanie, można wprowadzić zmiany do stanów jak: kliknięty, posiada focus itp.

4. Rysunek przedstawia EditText i Button prezentowane w sposób standardowy na urządzeniu z androidem 4.4. Dolna część przedstawia te komponenty po odpowiednich zmianach.



Dodatek F. Zmiana wyglądu widżetu

Zmiana wyglądu z podziałem na stany przycisku

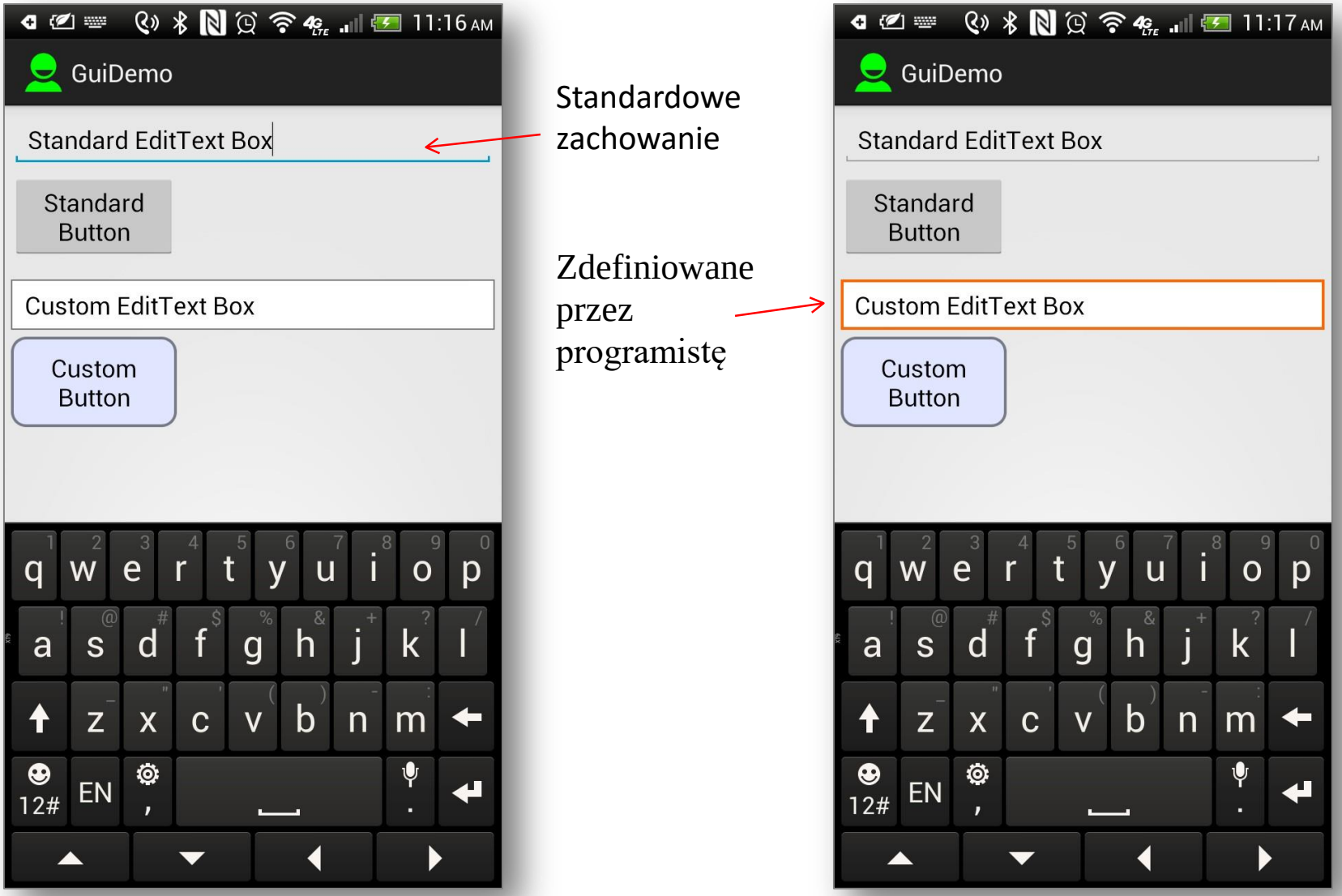


Standardowe zachowanie

Zachowanie ustalone przez programistę

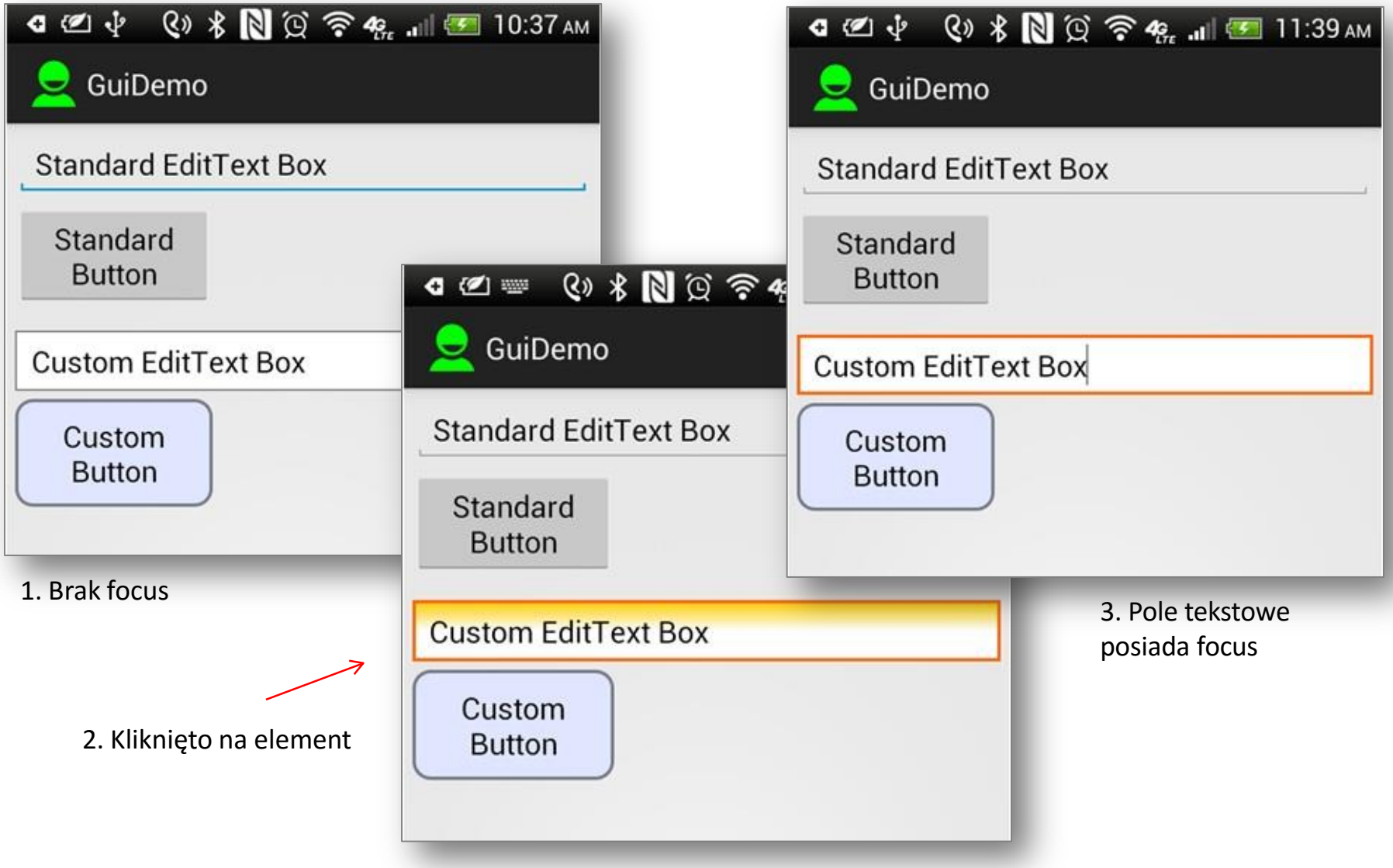
Dodatek F. Zmiana wyglądu widżetu

Zmiana wyglądu, gdy pole tekstowe posiada focus



Dodatek F. Zmiana wyglądu widżetu

Zmiana wyglądu, gdy użytkownik kliknął w pole tekstowe – użycie gradientu.

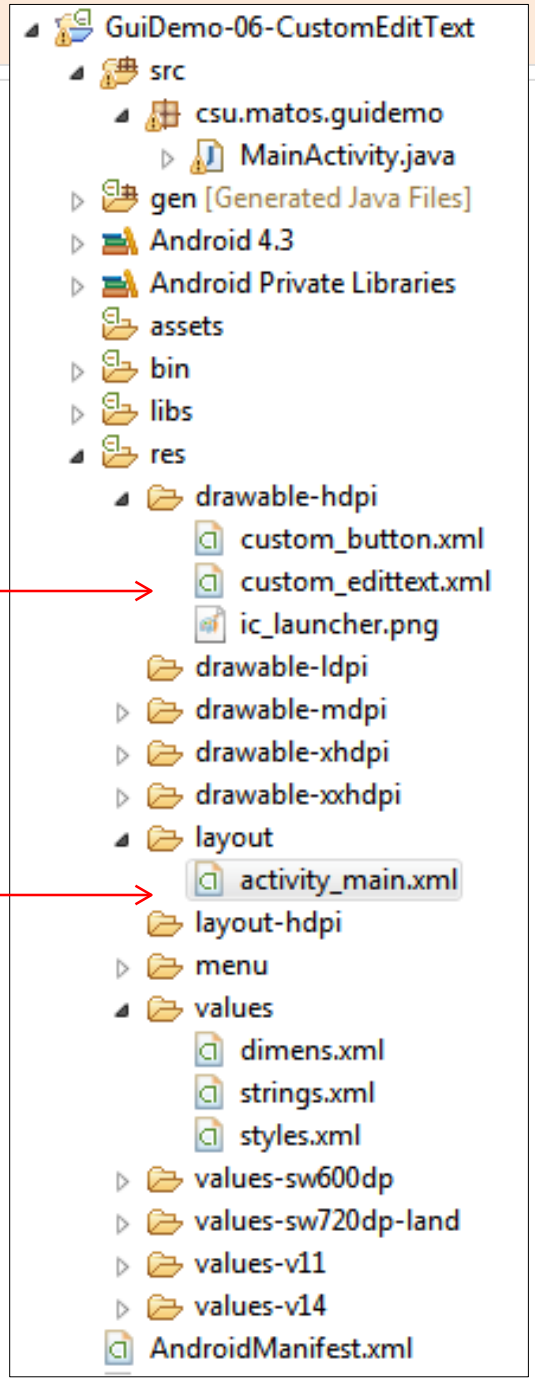


Dodatek F. Zmiana wyglądu widżetu

Organizing the application

Własne szablony wyglądu komponentów

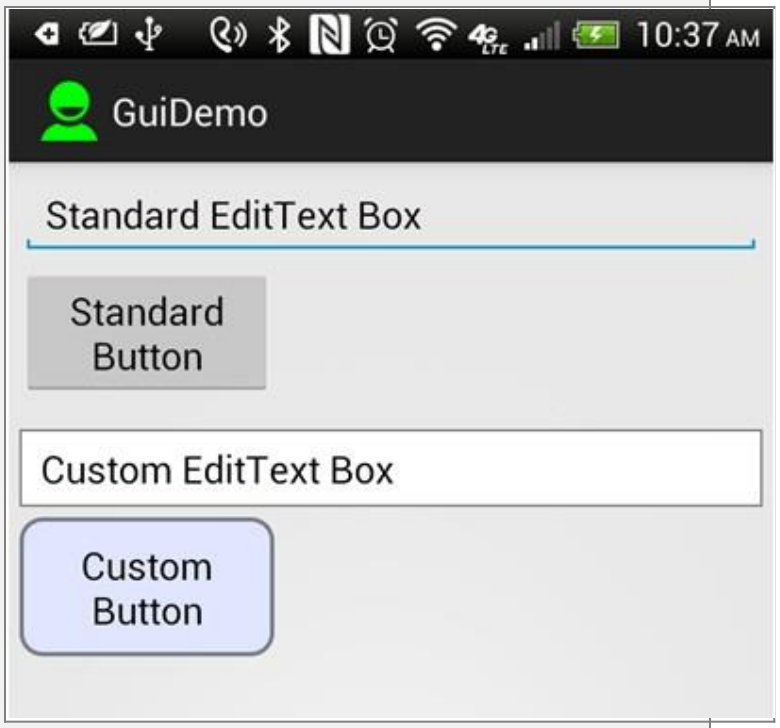
Główny wygląd aktywności



Dodatek F. Zmiana wyglądu widżetu

Układ

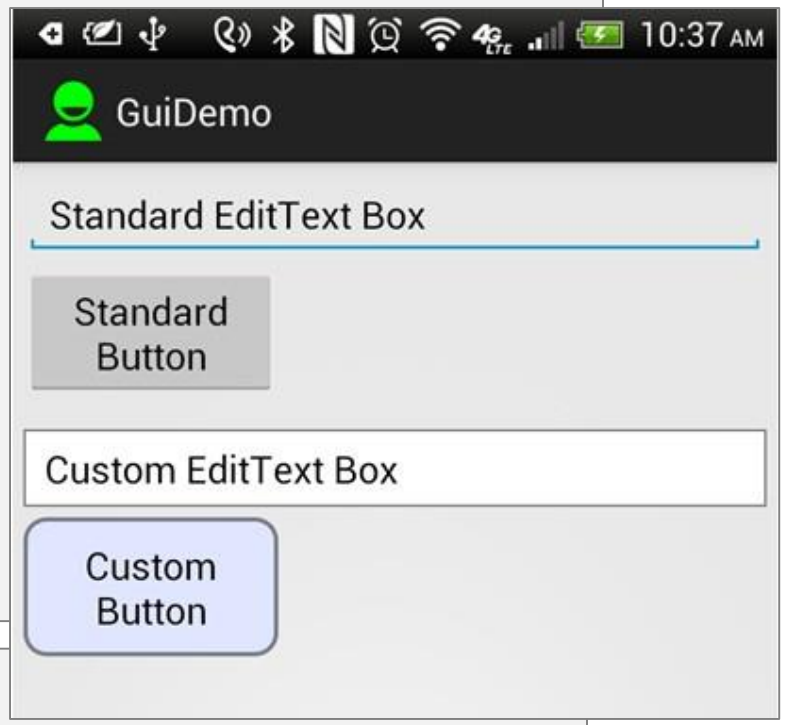
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="5dp" >

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="5dp"
        android:ems="10"
        android:inputType="text"
        android:text="@string/standard_edittext" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/button1"
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="15dp"
        android:text="@string/standard_button" />
```



Dodatek F. Zmiana wyglądu widżetu

Układ oraz Resource: res/values/strings

```
    <EditText
        android:id="@+id/editText2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="5dp"
        android:background="@drawable/custom_edittext"
        android:ems="10"
        android:inputType="text"
        android:text="@string/custom_edittext" />

    <Button
        android:id="@+id/button2"
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:background="@drawable/custom_button"
        android:text="@string/custom_button" />

</LinearLayout>
```




```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">GuiDemo</string>
    <string name="action_settings">Settings</string>
    <string name="standard_button">Standard Button</string>
    <string name="standard_edittext">Standard EditText Box</string>
    <string name="custom_button">Custom Button</string>
    <string name="custom_edittext">Custom EditText Box</string>
</resources>
```

Dodatek F. Zmiana wyglądu widżetu

Zasób: res/drawable/custom_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:state_pressed="true">
        <shape android:shape="rectangle">
            <corners  android:radius="10dp"/>
            <solid    android:color="#ffc0c0c0" />
            <padding  android:left="10dp"
                      android:top="10dp"
                      android:right="10dp"
                      android:bottom="10dp"/>
            <stroke   android:width="1dp" android:color="#ffFF6600"/>
        </shape>
    </item>
    <item android:state_pressed="false">
        <shape android:shape="rectangle">
            <corners  android:radius="10dp"/>
            <solid    android:color="#ffE0E6FF"/>
            <padding  android:left="10dp"
                      android:top="10dp"
                      android:right="10dp"
                      android:bottom="10dp"/>
            <stroke   android:width="2dp" android:color="#ff777B88"/>
        </shape>
    </item>
</selector>
```

Dodatek F. Zmiana wyglądu widżetu

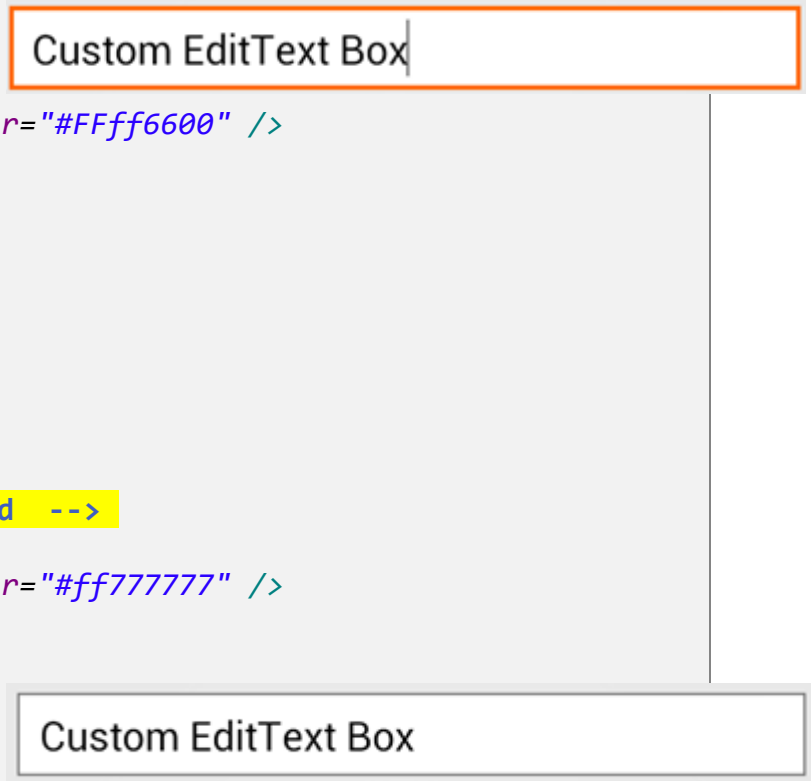
Zasób: res/drawable/custom_button.xml

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

<item android:state_pressed="true">
    <shape android:shape="rectangle">
          <gradient
               android:angle="90"
               android:centerColor="#FFffffff
               "  android:endColor="#FFffcc00"
               android:startColor="#FFffffff"
               android:type="linear" />

        <stroke   android:width="2dp"
                  android:color="#FFff6600" />
        <corners  android:radius="0dp" />
        <padding  android:left="10dp"
                  android:top="6dp"
                  android:right="10dp"
                  android:bottom="6dp" />
    </shape>
</item>
```

Dodatek F. Zmiana wyglądu widżetu

Zasób: res/drawable/custom_button.xml

```
<item android:state_focused="true">
    <shape>
        <solid    android:color="#FFffffff" />
        <stroke   android:width="2dp" android:color="#FFff6600" />
        <corners  android:radius="0dp" />
        <padding  android:left="10dp"
                  android:top="6dp"
                  android:right="10dp"
                  android:bottom="6dp" />
    </shape>
</item>

<item>
    <!-- state: "normal" not-pressed & not-focused  -->
    <shape>
        <stroke   android:width="1dp" android:color="#ff777777" />
        <solid    android:color="#ffffffff" />
        <corners  android:radius="0dp" />
        <padding  android:left="10dp"
                  android:top="6dp"
                  android:right="10dp"
                  android:bottom="6dp" />
    </shape>
</item>
</selector>
```

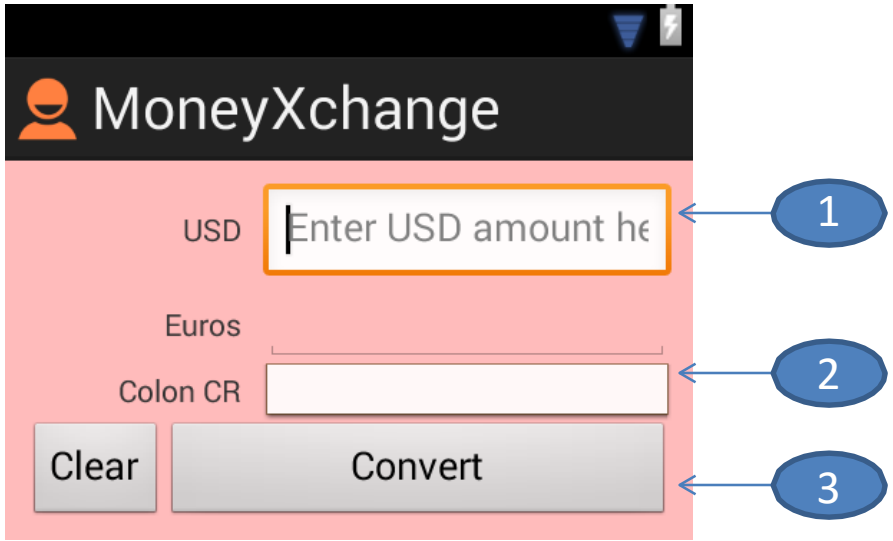


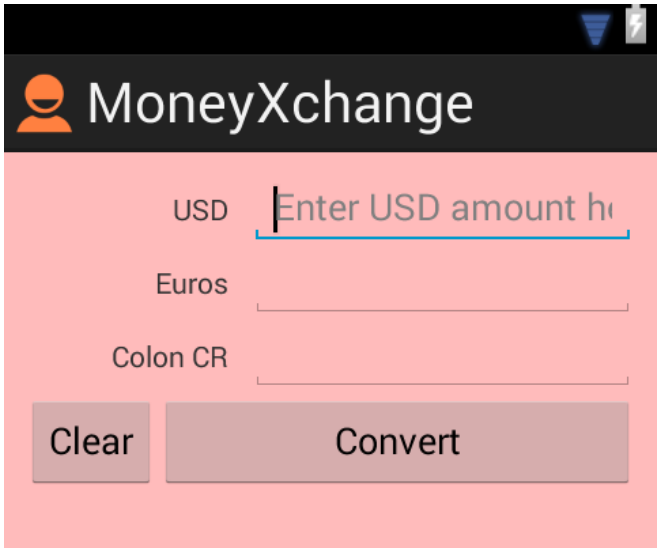
Dodatek G. Problemy z tłem

Zmiana koloru tła może ograniczyć się do zmiany odpowiedniej właściwości w pliku XML `android:background="#44ff0000"` (pół-przeźroczysty czerwony).

W zależności od skórki (theme) urządzenia może zdarzyć się problem z wypełnieniem komponentów – niektóre z nich przejmują globalny kolor tła.

Rozwiązanie jest bardzo proste – nadanie tym komponentom innego koloru tła poprzez wykorzystanie atrybutu `android:background`.






```
1. android:background="@android:drawable/edit_text"
2. android:background="@android:drawable/editbox_dropdown_light_frame"
3. android:background="@android:drawable/btn_default"
```