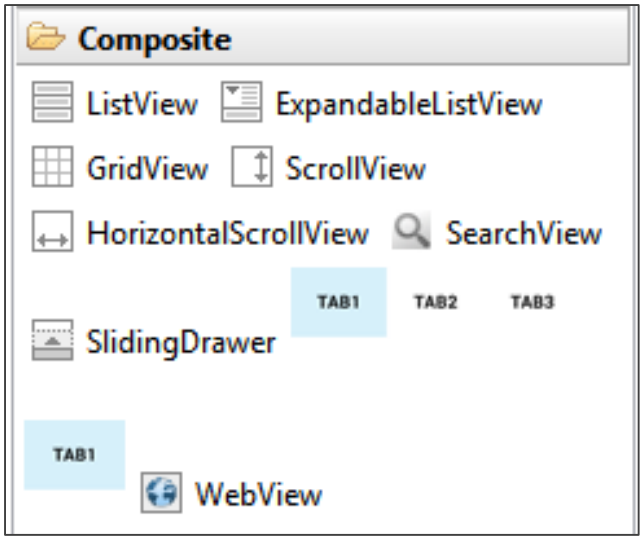
Pracownia aplikacji mobilnych

Wyświetlanie danych na listach

Widżety listowe

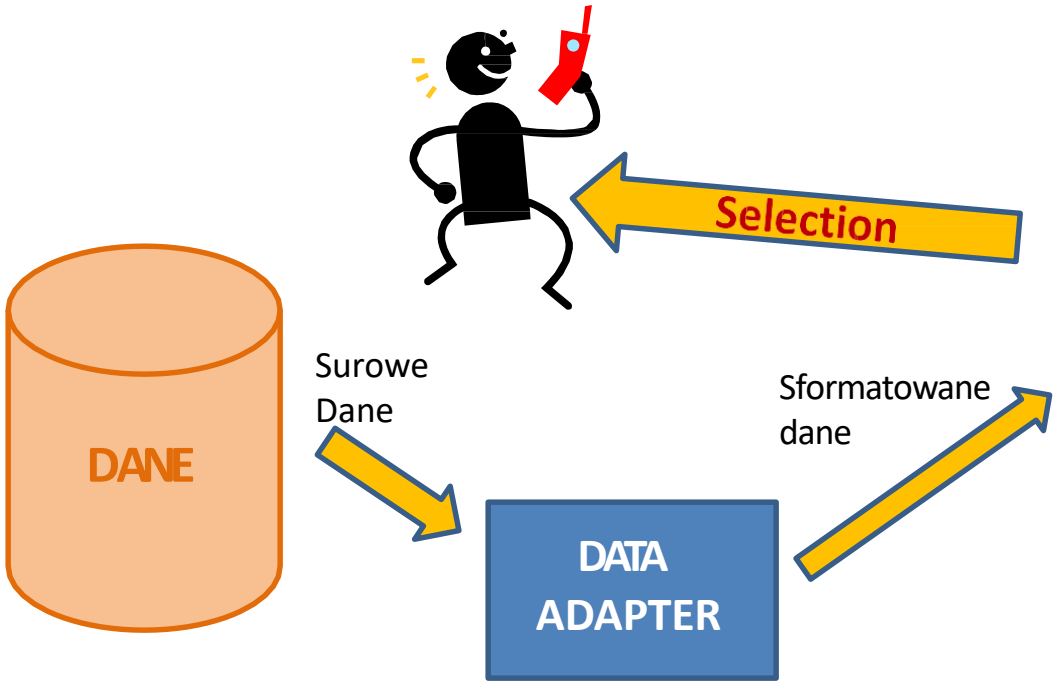
- Widżety RadioButton oraz CheckBox są dobre przy wyborze małej liczby możliwych opcji.

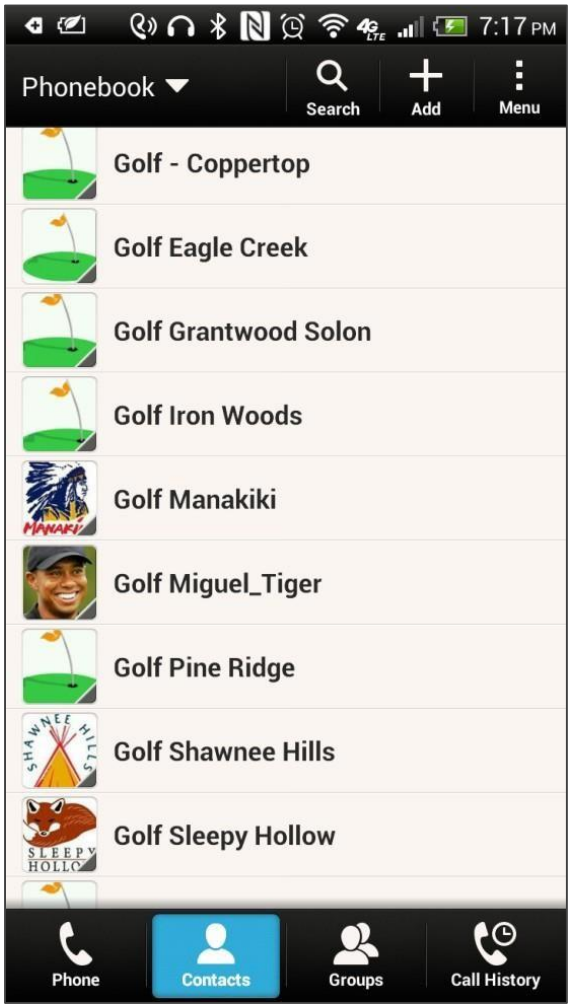
- Dla większej liczby opcji inne widżety oparte o list są bardziej adekwatne.
- Przykłady widżetów opartych o listę to:
  - *ListViews,*
  - *Spinner,*
  - *GridView*
  - *Image Gallery*
  - *ScrollViews*, etc.



Widżety listowe

Przekazywanie danych





Destination layout
Holding a **ListView**

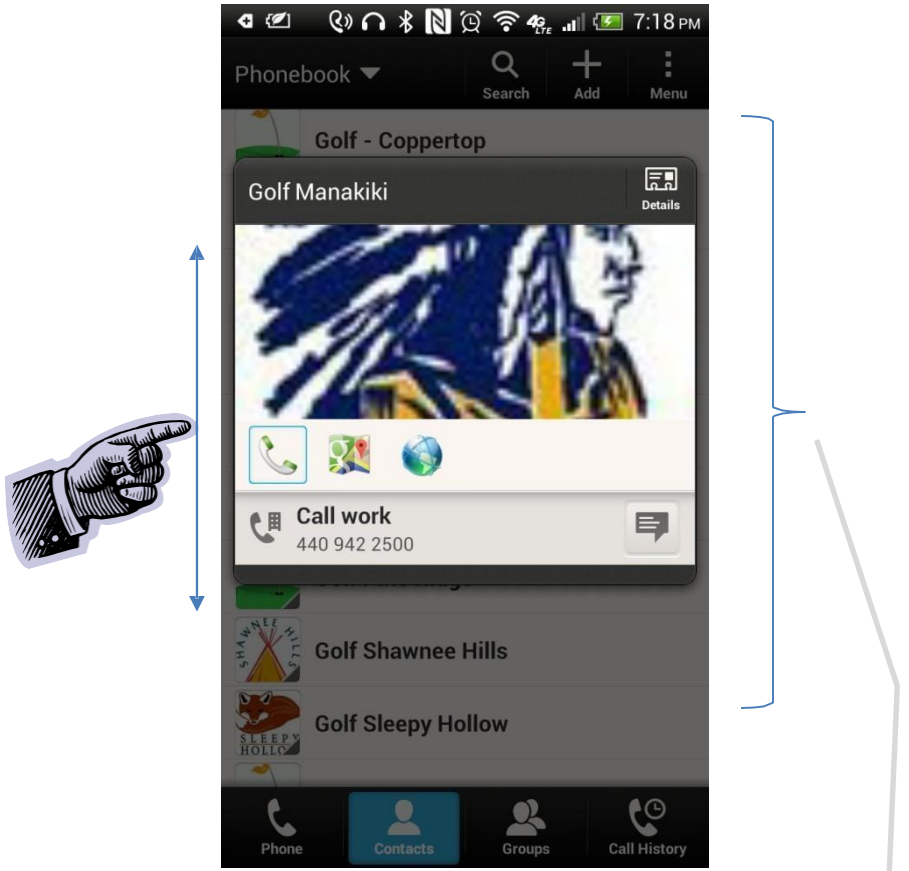
- Klasa ***data adapter*** jest wykorzystywana do populowania danymi listy.
- Surowe dane dla adaptera mogą pochodzić z wielu różnych źródeł, jak małe tablice czy też duże bazy danych.

Widżety listowe

Widżet **ListView** jest najpopularniejszym widżetem, który umożliwia wyświetlanie danych gromadzonych przez **data adapter**.

ListView mają wbudowne paski przewijania, więc nie wszystkie wiersze muszą być widoczne.

Użytkownik dotyka wiersza powodując wybór.



Destination layout
Holding a **ListView**

Rekomendowany zamiennik RecyclerView
https://developer.android.com/guide/topics/ui/layout/recyclerview

ArrayAdapter

- **`ArrayAdapter<T>`** jest rzeczywistą implementacją abstrakcyjnego typu BaseAdapter class.

- Jego zadaniem jest pobranie tablicy elementów (dowolnego typu T), przetworzenie poszczególnych elementów poprzez wywołanie metody **`toString()`** oraz przekazanie sformatowanego wyjścia do **TextView**. Formatowanie wykonywane jest na podstawie dołączonej specyfikacji w formie pliku XML.

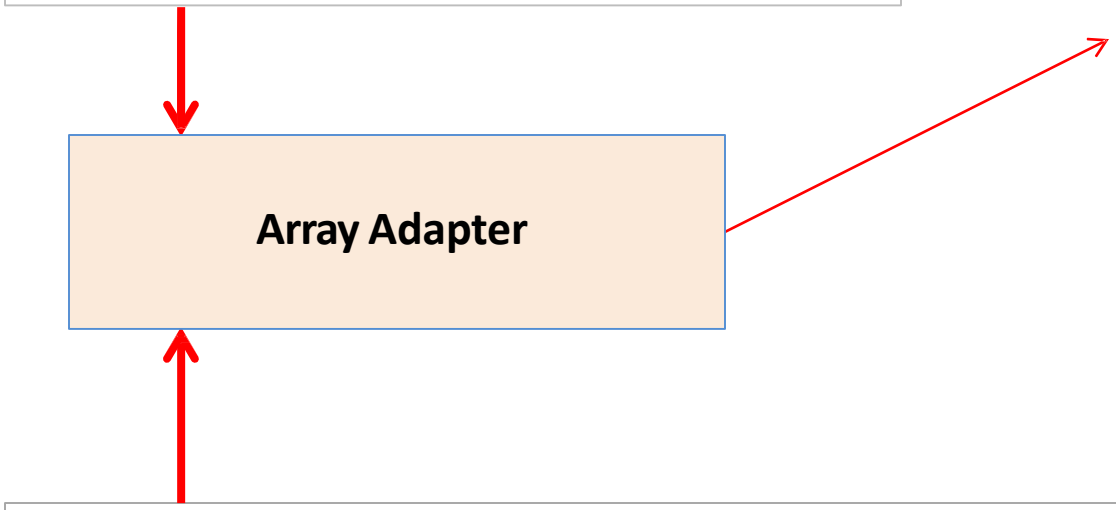
- `ArrayAdapter<T>` jest głównie wykorzystywany dla obiektów typu `<String>`. Dla innych typów danych należy przeciążyć metodę `toString()` by wyznaczyć jaki tekst będzie prezentowany dla konkretnego elementu listy.

- By ListView pokazywał bardziej skomplikowaną aranżację wizualną – tekst i obrazki – należy stworzyć własny adapter, którego metoda `getView(…)` określa jak należy stworzyć i wypozycjonować każdy element interfejsu.

ArrayAdapter

Dane źródłowe - array or java.util.List

{ object$_1$, object$_2$, …, object$_n$ }



Wyjście: ‘Ładne’ GUI

object $_1$.toString()

textviews…

object $_n$.toString()

Specyfikacja XML

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

ArrayAdapter

Typowe użycie ArrayAdapter<String>

```
String[] items = { "Data-0", "Data-1", "Data-2", "Data-3",
                   "Data-4", "Data-5", "Data-6", "Data-7"  };

ArrayAdapter<String> adapter = new ArrayAdapter<String>(
                        this,
                        android.R.layout.simple_list_item_1,
                        items );
```

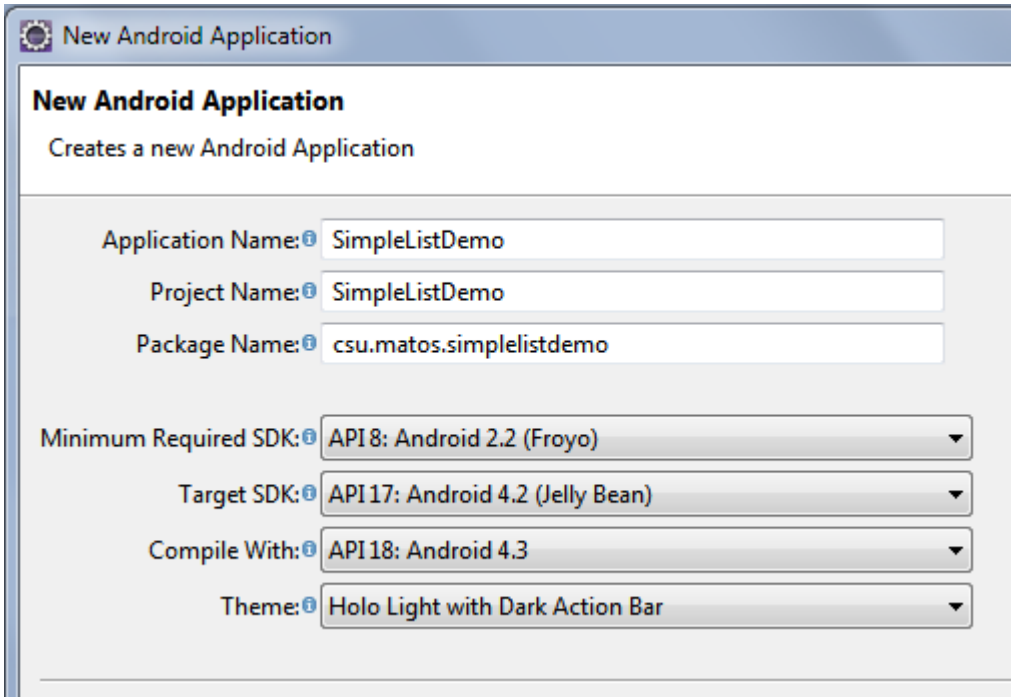
Parameters:

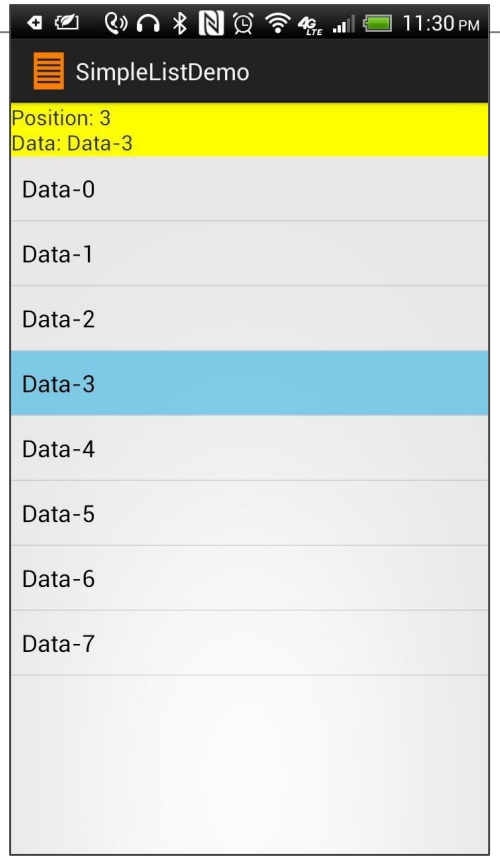
1. Kontekst danej aktywności **(this)**
2. Układ dla **TextView** określający jak każdy wiersz listy jest formatowany (android.R.id.*simple_list_item_1*).
3. Właściwe **źródło danych** (tablica lub Java.List zawierająca elementy, które należy zaprezentować).

Przykład 1: Wykorzystanie ArrayAdapter

Prosta lista

Celem przykładu jest wyświetlenie listy elementów tekstowych. Dane przechowywane są w prostej tabeli typu String[]. Każdy wiersz listy pokazuje poszczególny element tablicy źródłowej. Jeżeli użytkownik zaznaczy dany element, na ekranie pokazywana jest jego wartość oraz indeks z tablicy źródłowej.





Przykład 1: Wykorzystanie ArrayAdapter

Układ 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:orientation="vertical">

    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffffff00"
        android:text="Using ListViews..."

        android:textSize="16sp" />

    <ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"  >
    </ListView>

    <TextView
        android:id="@android:id/empty"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffff0000"
        android:text="empty list"  />

</LinearLayout>
```

Istotne jest nazewnictwo komponentów:

@android:id/list
@android:id/empty

Standardowy układ

Dla pustych list

Przykład 1: Wykorzystanie ArrayAdapter

Układ XML

UWAGA:
ListActivity to nie to samo co Activity. Przypisana jest do ListView

```
package csu.matos;

import ...

public class ListViewDemo extends ListActivity {

   TextView txtMsg;

   String[] items = {  "Data-0", "Data-1", "Data-2", "Data-3",
                       "Data-4", "Data-5", "Data-6", "Data-7"  };

   // next time try an empty list such as:
   // String[] items = {};
```

Źródło danych

Uwaga: Klasa **ListActivity** jest niejawnie powiązana z obiektem o nazwie **@android:id/list**

Przykład 1: Wykorzystanie ArrayAdapter

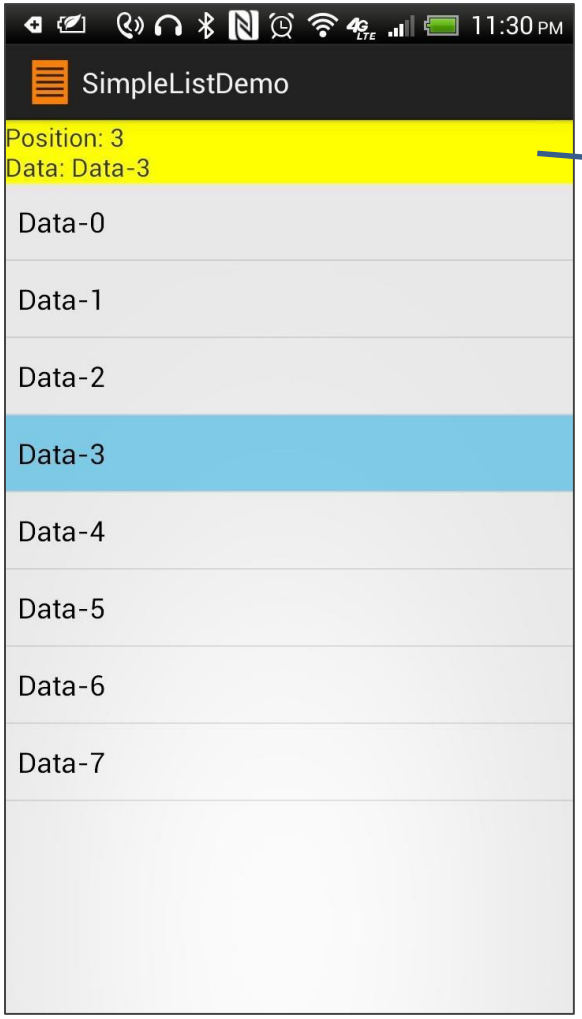
Klasa Main Activity

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

        setListAdapter(new ArrayAdapter<String>(this,
                                        android.R.layout.simple_list_item_1,
                                        items));

        //getListView().setBackgroundColor(Color.GRAY);  //sprawdź

        txtMsg = (TextView) findViewById(R.id.txtMsg);
    }

    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        super.onListItemClick(l, v, position, id);
        String text = " Position: " + position + "    " + items[position];
        txtMsg.setText(text);
    }

}
```

List adapter

List Click Listener

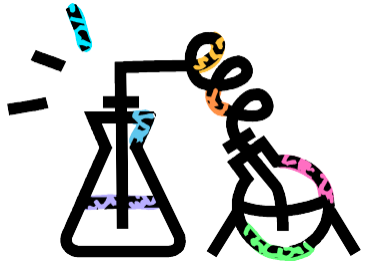
Przykład 1: Wykorzystanie ArrayAdapter



Wybrany element

Tło zmienia się na niebieskie by potwierdzić wybór użytkownika

Przykład 1: Wykorzystanie ArrayAdapter

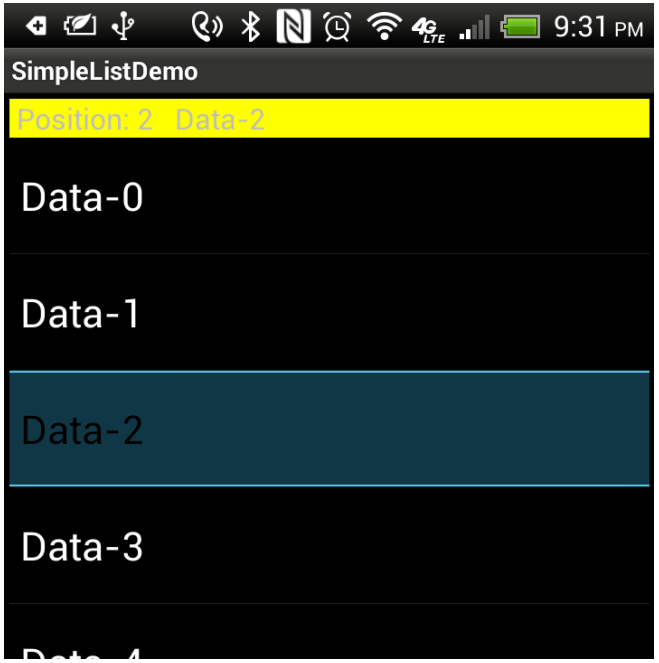


Szybka zmiana grafiki

1. Otwórz plik AndroidManifest.xml. Pod tagiem <Application> znajdź klauzulę `android:theme="@style/AppTheme"`

2. Zmień znalezioną linię na:
   `android:theme="@android:style/Theme.Black"`

3. Spróbuj innych wariantów:
   - `Theme.DeviceDefault`
   - `Theme.Dialog`
   - `Theme.Holo`
   - `Theme.Light`
   - `Theme.Panel`
   - `Theme.Translucent`
   - `Theme.Wallpaper`
   - itp.



Przykład 1B: Wykorzystanie ArrayAdapter

Alternatywna wersja przykładu 1

- Można używać zwykłej klasy **Activity** zamiast **ListActivity**.
- Układ poniżej wykorzystuje listę o identyfikatorze `@+id/my_list` (zamiast `@android:id/list` jak w poprzednim przypadku).

Przykład 1B – Układ 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:orientation="vertical" >

    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffffff00"
        android:text="Using ListViews..."
        android:textSize="16sp" />

    <ListView android:id="@+id/my_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </ListView>

</LinearLayout>

</LinearLayout>
```

Przykład 1B: Wykorzystanie ArrayAdapter

Alternatywna wersja przykładu 1

Przykład 1B – Klasa MainActivity

```
public class ListViewDemo2 extends Activity {
      String[] items = { "Data-0", "Data-1", "Data-2", "Data-3",
                         "Data-4", "Data-5", "Data-6", "Data-7" };
      ListView myListView;
      TextView txtMsg;

      @Override
      public void onCreate(Bundle savedInstanceState)
         { super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
         myListView = (ListView) findViewById(R.id.my_list);

         ArrayAdapter<String> aa = new ArrayAdapter<String>(this,
                                   android.R.layout.simple_list_item_1,
                                   // R.layout.my_text,   //try this
                                   later... items);
         myListView.setAdapter(aa);

         txtMsg = (TextView) findViewById(R.id.txtMsg);
    }//onCreate
}
```

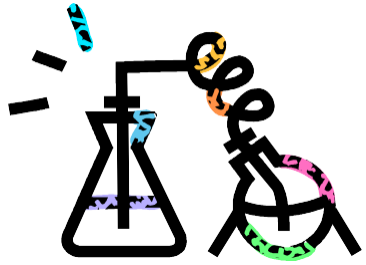
Alternatywna wersja przykładu 1

Example 1B – MainActivity Class

```
myListView.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> av, View v,
                                int position, long id) {

           String text = "Position: " + position
                       + "\nData: " + items[position];

           txtMsg.setText(text);
        }
        });
```

Należy pamiętać o dodaniu nasłuchiwacza dla zdarzenia kliknięcia elementu listy w metodzie **onCreate**.

Przykład 1C: Wykorzystanie ArrayAdapter

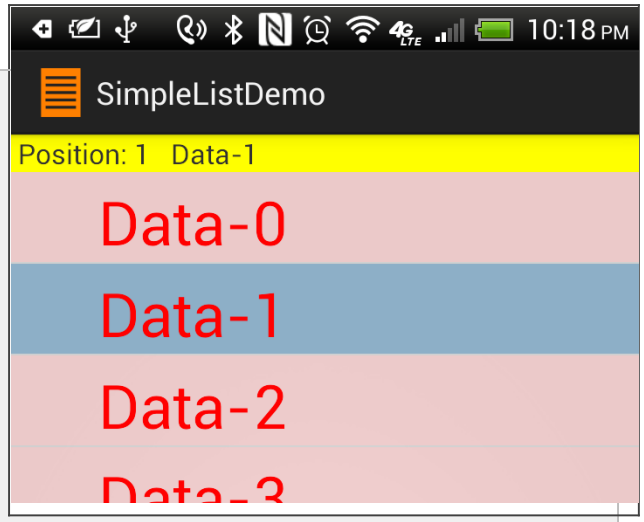


Alternatywna wersja przykładu 1

Wygląd ListView można dowolnie zmieniać zgodnie z **własnymi preferencjami**.

Przykładowo, należy zmienić `android.R.layout.simple_list_item_1` na `R.layout.my_text` gdzie `my_text` to układ zgodny z zaprezentowanym poniżej (przechowywany w **res/layout**).

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="2dp"
    android:paddingTop="5dp"
    android:paddingLeft="50dp"
    android:textColor="#ffff0000"
    android:background="#22ff0000"
    android:textSize="35sp" />
```

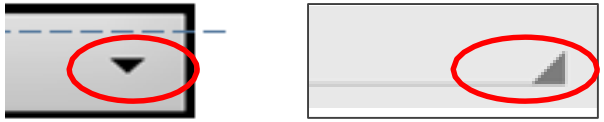


Wskazówka : Od SDK4.0 TextView może również zawierać obrazek (wykorzystując metodę `.setDrawableLeft(some_image)`)

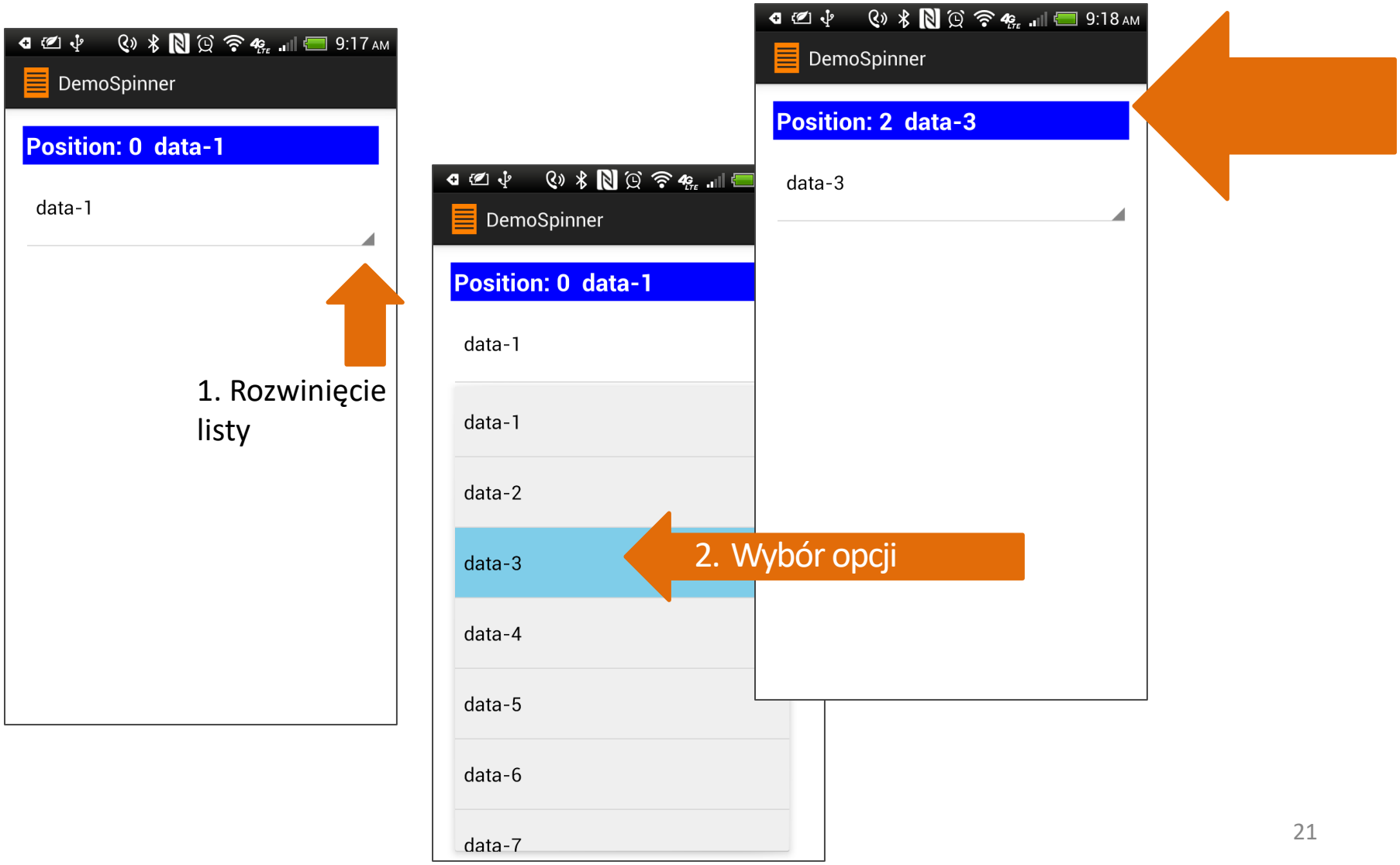
Spinner

Spinner

- **Spinner** jest ekwiwalentem *listy rozwijanej*.
- Spinner ma takie same możliwości jak ListView, ale zajmuje mniej miejsca.
- Obsługą źródła danych zajmuje się ArrayAdapter - ***`setAdapter(…)`***.
- Odpowiedni nasłuchiwacz rejestruje zdarzenia zaznaczenia listy ***`setOnItemSelectedListener(…)`***.
- Metoda ***`setDropDownViewResource(…)`*** pokazuje strzałkę rozwijania listy.



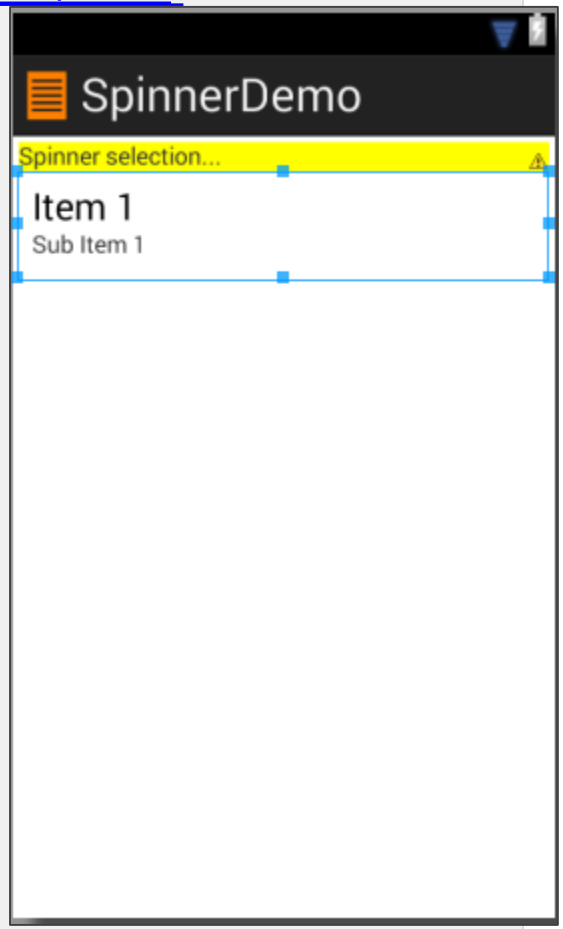
Przykład2: Wykorzystanie Spinner'a



Przykład2: Wykorzystanie Spinner'a

Układ 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:orientation="vertical"
    android:padding="3dp"
    tools:context=".MainActivity" >

    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffffff00"
        android:text="@string/hello_world" />

    <Spinner android:id="@+id/spinner1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```



Przykład2: Wykorzystanie Spinner'a

Klasa MainActivity

```
public class MainActivity extends Activity
                        implements AdapterView.OnItemSelectedListener{

   // GUI objects
   TextView txtMsg;
   Spinner spinner;

   // options to be offered by the spinner
   String[] items = { "Data-0", "Data-1", "Data-2", "Data-3", "Data-4",
                      "Data-5", "Data-6", "Data-7" };

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

      spinner = (Spinner) findViewById(R.id.spinner1);


      // use adapter to bind items array to GUI layout
      ArrayAdapter<String> adapter = new ArrayAdapter<String>(
            this,
            android.R.layout.simple_spinner_dropdown_item,
            items);
```

Przykład2: Wykorzystanie Spinner'a

Klasa MainActivity

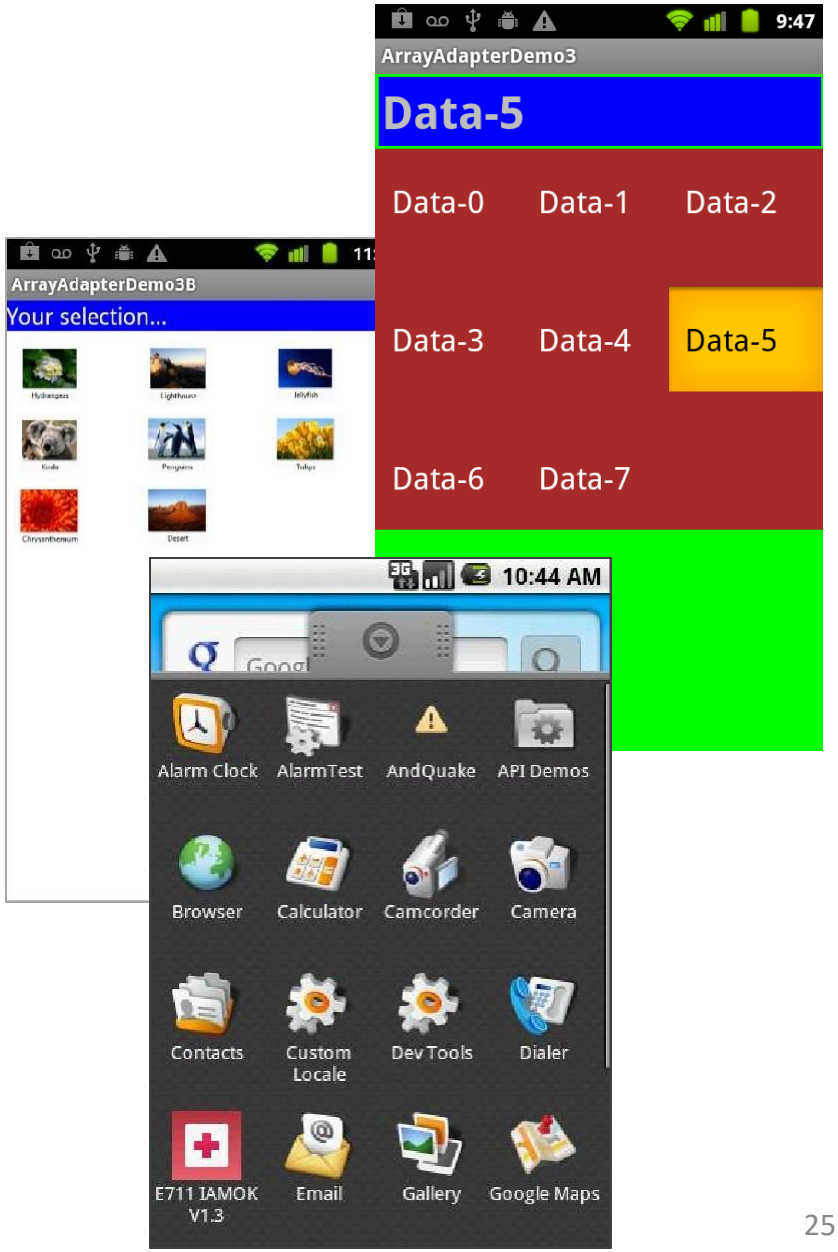
```
        // bind everything together
        spinner.setAdapter(adapter);

        // add spinner a listener so user can meake selections by tapping an item
        spinner.setOnItemSelectedListener(this);

    }
    // next two methods implement the spinner's listener
    @Override
    public void onItemSelected(AdapterView<?> parent, View v, int position,
            long id) {
        // echo on the textbox the user's selection
        txtMsg.setText(items[position]);
    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
        // TODO do nothing - needed by the interface

    }

}
```

GridView

GridView jest implementacją ViewGroup, która wyświetla elementy w formie 2-wymiarowej siatki z możliwością przewijania.

Poszczególne dane wyświetlane w siatce dostarczane są przez warstwę data adaptera.

Komórki siatki mogą wyświetlać zarówno dane tekstowe jak i grafikę.



GridView

Przydatne właściwości

Poniższe właściwości są wykorzystywane by ustalić liczbę i rozmiar kolumn:

- **android:numColumns**
  Określa ile kolumn należy pokazać. Jeżeli wykorzystano stałą “auto_fit”, Android sam ustala liczbę kolumną na podstawie dostępnego wolnego miejsca oraz pozostałych właściwości wymienionych poniżej.

- **android:verticalSpacing** oraz **android:horizontalSpacing**
  określa jak dużo wolnego miejsca powinno zostać między elementami w siatce.

- **android:columnWidth**
  Szerokość kolumn (w **dip**).

- **android:stretchMode**
  Określa jak zmienić rozmiar grafiki, gdy wolne miejsce nie zostało zagospodarowane przez kolumny bądź wymuszone przez spacing.

GridView

Rozciąganie widżetu na cały ekran

Załóżmy, że ekran ma szerokość **320** (dip) pikseli oraz ustawiono właściwości:

android:columnWidth na **100dip** oraz
android:horizontalSpacing na **5dip**.

Trzy kolumny zajmują **310** pikseli (każda kolumna po 100 pikseli oraz dwa pasy pustej przestrzeni po 5 pikseli).

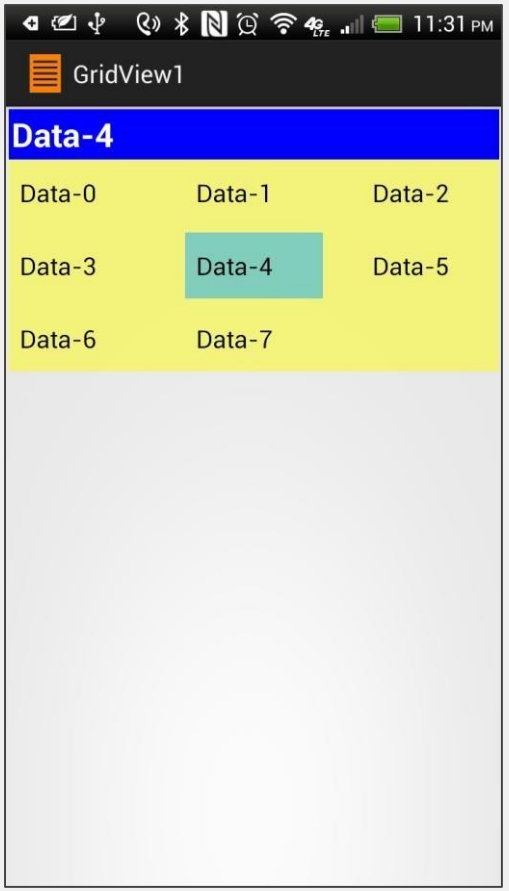
Jeżeli *android:stretchMode* ustawiono na *columnWidth*, każda z 3 kolumn rozciągnięta zostanie o 3-4 piksele by wykorzystać wolną przestrzeń 10 pikseli.

Jeżeli *android:stretchMode* ustawiono na *spacingWidth*, pusta przestrzeń zostanie rozciągnięta o 5 pikseli by zagospodarować pozostałe 10 pikseli.

Przykład 3A: Wykorzystanie GridView

Układ 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="2dp"
    tools:context=".MainActivity" >
    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:textSize="24sp"
        android:textStyle="bold"
        android:textColor="#ffffffff"
        android:padding="2dip"    />

     <GridView android:id="@+id/grid"
        android:background="#77ffff00"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:verticalSpacing="5dip"
        android:horizontalSpacing="5dip"
        android:numColumns="auto_fit"
        android:columnWidth="100dip"
        android:stretchMode="spacingWidth"  />
</LinearLayout>
```



Przykład 3A: Wykorzystanie GridView

Klasa Main Activity

```
public class ArrayAdapterDemo3 extends Activity  {

          TextView txtMsg;

          String[] items = { "Data-0", "Data-1", "Data-2", "Data-3",
                             "Data-4", "Data-5", "Data-6", "Data-7" };
          @Override
          public void onCreate(Bundle icicle) {
                    super.onCreate(icicle);
                    setContentView(R.layout.activity_main);

                    txtMsg = (TextView) findViewById(R.id.txtMsg);
```

Przykład 3A: Wykorzystanie GridView

Klasa Main Activity

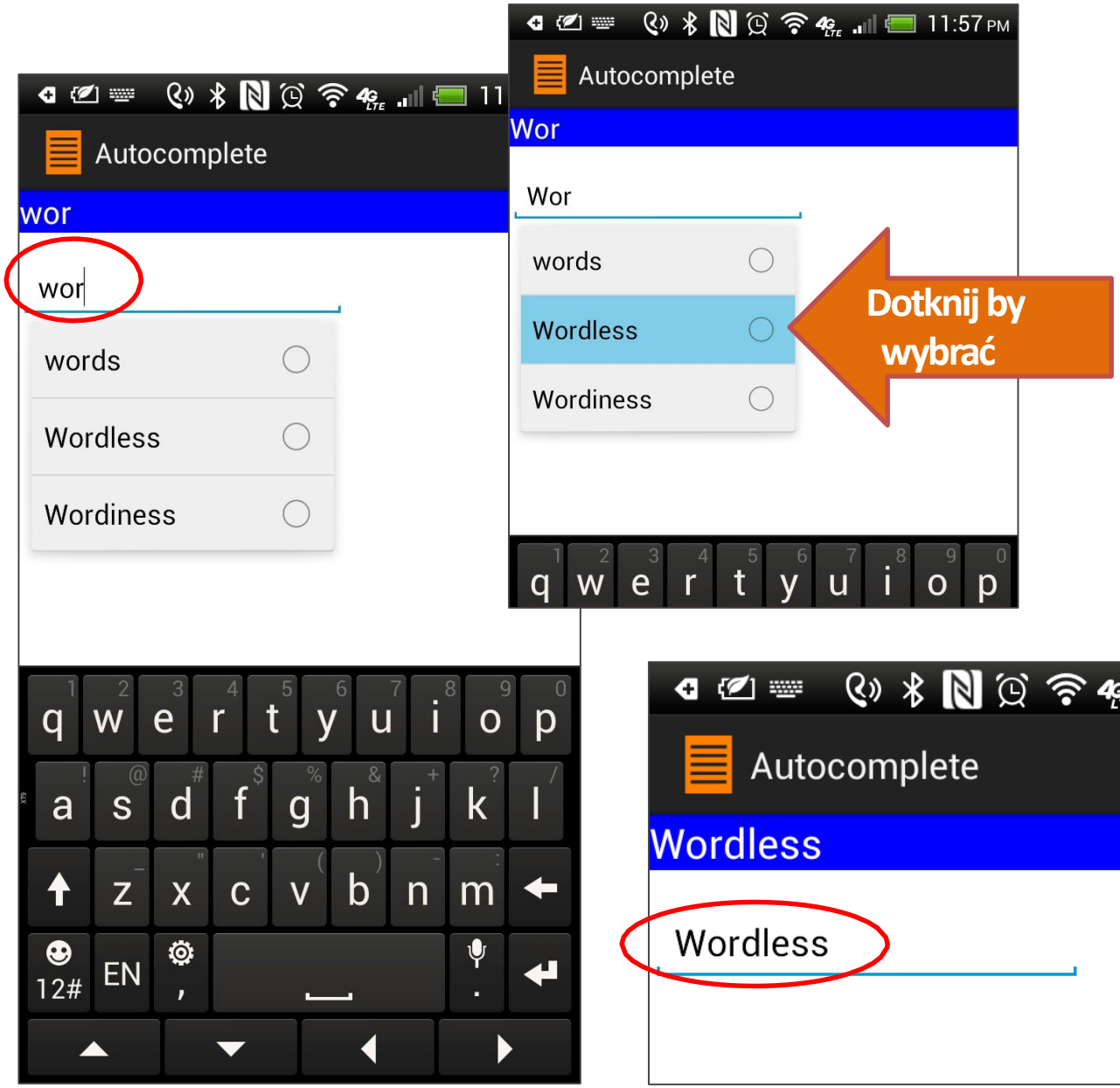
```
      GridView grid = (GridView) findViewById(R.id.grid);

      ArrayAdapter<String> adapter = new ArrayAdapter<String>(
            this,
            android.R.layout.simple_list_item_1,
            items );

      grid.setAdapter(adapter);

      grid.setOnItemClickListener(new OnItemClickListener() {

         @Override
         public void onItemClick(AdapterView<?> container, View
                                 v, int position, long id) {
            txtMsg.setText(items[position]);
         }
      });

   }//onCreate

}// class
```

AutoCompleteTextView

- Jest to bardziej wyspecjalizowany komponent typu EditText.
- Dotychczas wprowadzone znaki są porównywane z początkiem słów przechowywanych w liście dostępnych sugestii.
- Sugestie dopasowane do określonego prefiksu są wyświetlane na *liście wyboru.*
- Użytkownik może wybrać z listy sugestii lub kontynuować wprowadzanie znaków.
- Właściwość `android:completionThreshold` jest wykorzystywana by uaktywnić wyświetlanie listy sugestii. Określa liczbę znaków, którą należy wprowadzić by próbować dopasowania do listy sugestii.

Inne właściwości omawianego widżetu dostępne są w Dodatku B

AutocompleteTextView



Przykład 4.
Słowa zaczynające się od prefiksu "**wor**" lub "**set**" są podstawą do aktywacji listy.

Jeżeli dowolne (3 literowe) prefiksy zostaną wprowadzone mechanizm TextWatcher wyświetli listę sugestii.

AutocompleteTextView

Układ 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:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:textColor="#ffffffff"
        android:background="#ff0000ff" >
       </TextView>

    <AutoCompleteTextView
        android:id="@+id/autoCompleteTextView1"
        android:hint="type here..."
        android:completionThreshold="3"
         android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/txtMsg"
        android:layout_marginTop="15dp"
        android:ems="10" />
</RelativeLayout>
```

Wait 3 chars to work

AutocompleteTextView

Klasa Main Activity

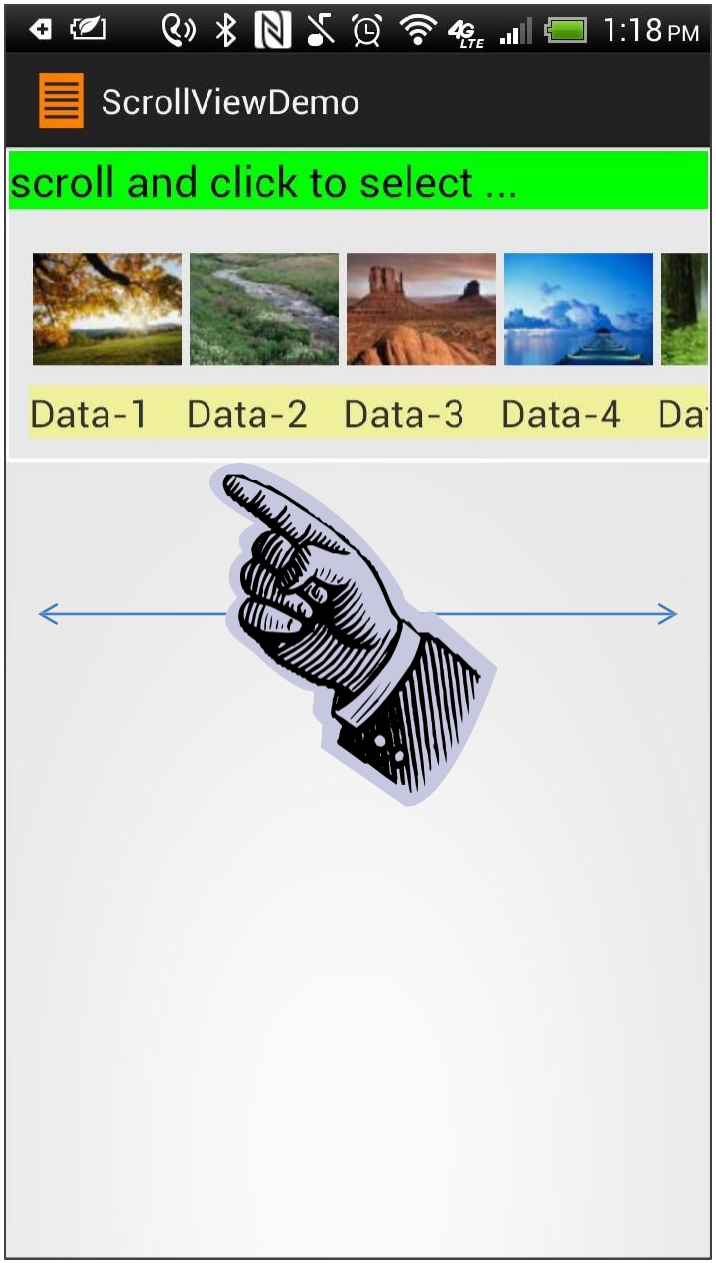
```
public class ArrayAdapterDemo4 extends Activity implements TextWatcher
{

  TextView txtMsg;

  AutoCompleteTextView txtAutoComplete;

  String[] items = { "words", "starting", "with", "set",
      "Setback", "Setline", "Setoffs", "Setouts", "Setters",
      "Setting", "Settled", "Settler", "Wordless", "Wordiness",
      "Adios" };

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

        setContentView(R.layout.activity_main);

       txtMsg = (TextView) findViewById(R.id.txtMsg);
```

AutocompleteTextView

Klasa Main Activity

```
  txtAutoComplete = (AutoCompleteTextView) findViewById(
                                                R.id.autoCompleteTextView1);
  txtAutoComplete.addTextChangedListener(this);

  txtAutoComplete.setAdapter(new ArrayAdapter<String>( this,
                            android.R.layout.simple_list_item_single_choice
                            , items));
  }//onCreate

  public void onTextChanged(CharSequence s, int start, int before, int count) {
    txtMsg.setText(txtAutoComplete.getText());
  }

  public void beforeTextChanged(CharSequence s, int start, int count,int after) {
    // needed for interface, but not used
  }

  public void afterTextChanged(Editable s) {
    // needed for interface, but not used
  }

}//class
```

HorizontalScrollView



Komponent HorizontalScrollView umożliwia w sposób graficzny dokonać wyboru spośród listy elementów (np. miniatur obrazków).

Użytkownik wchodzi w interakcję z komponentem poprzez:

1. Przewijanie (lewo $\leftrightarrow$ prawo)
2. Wybrać miniaturę by uaktywnić daną opcję.

W poniższym przykładzie jeżeli użytkownik kliknie w miniaturę aplikacja reaguję wyświetlając wersję obrazka w wyższej rozdzielczości.

+. Zwykle miniatura ma rozmiar 100x100 pikseli (lub mniej).

Przykład 5: Wykorzystanie HorizontalScrollView

- W tym przykładzie zostaje umieszczony **HorizontalScrollView** na górze ekranu, którego zadaniem jest zaprezentowanie zestawu miniatur.
- Użytkownik może przewijać miniatury i poprzez dotknięcie dokonać wyboru określonej.
- Wersja wybranego obrazka w wyżej rozdzielczości zostanie zaprezentowana na komponencie ImageView poniżej komponentu zawierającego listę miniatur.

Przykład 5: Wykorzystanie HorizontalScrollView

Jak stworzyć miniaturę?

- **Opcja-1.** Wykorzystać konwerter online: http://makeathumbnail.com
- **Opcja-2**. Wykorzystać narzędzie Android Asset Studio http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html

Przykład 5: Wykorzystanie HorizontalScrollView

Wypełnienie widżetu HorizontalScrollView

1. Widżet HorizontalScrollView wyświetli listę ramek, każda składająca się z grafiki oraz tekstu będącego jej podpisem.

2. Układ *frame_icon_caption.xml* określa formatowanie grafiki oraz jej podpisu. Układ zostanie poddanych procesowi **inflacji** by stworzyć odpowiednie elementy GUI.

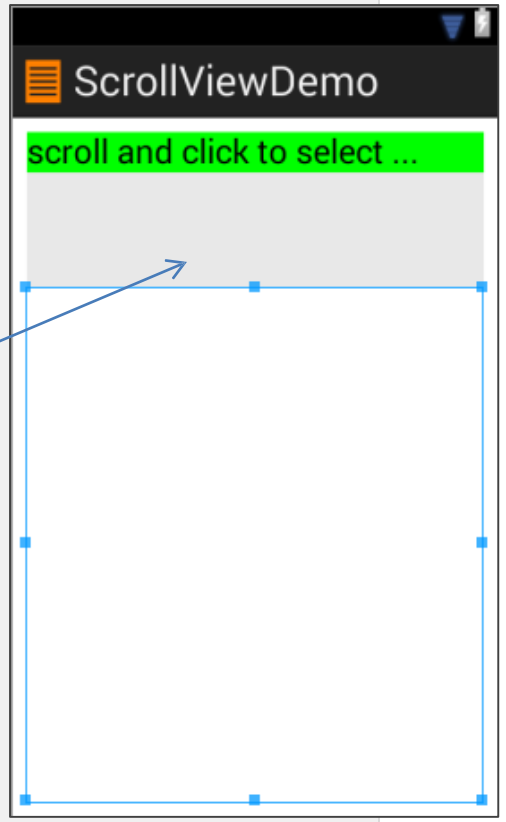
3. Gdy dana *ramka* zostanie wypełniona danymi, zostanie dodana do zwiększającej się liczby widoków wewnątrz kontenera typu *scrollViewgroup* (scrollViewgroup jest osadzony wewnątrz komponentu odpowiedzialnego za poziomy pasek przewijania).

4. Każda *ramka* dostaje unikalne **ID** (określające pozycje wewnątrz scrollViewgroup) jak również unikalny nasłuchiwacz zdarzenia **onClick**.

Przykład 5: Wykorzystanie HorizontalScrollView

Układ XML : *activity_main.xml*

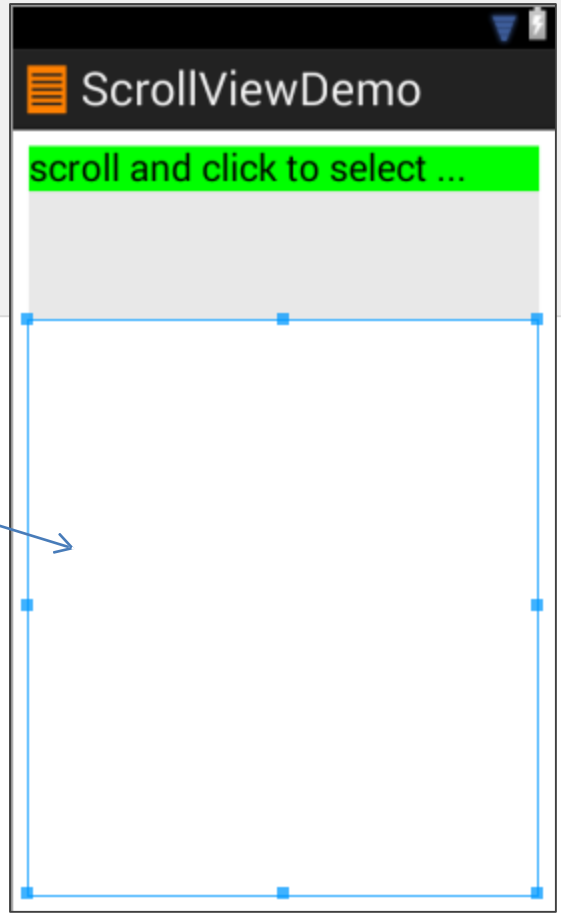
```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#ffffffff"
    android:orientation="vertical"
    android:padding="2dp" >


    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff00ff00"
        android:text="scroll and click to select ..."
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <HorizontalScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#44aaaaaa" >

        <LinearLayout
            android:id="@+id/viewgroup"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:padding="10dip" >
        </LinearLayout>

    </HorizontalScrollView>
```



Przykład 5: Wykorzystanie HorizontalScrollView

Układ XML : *activity_main.xml*

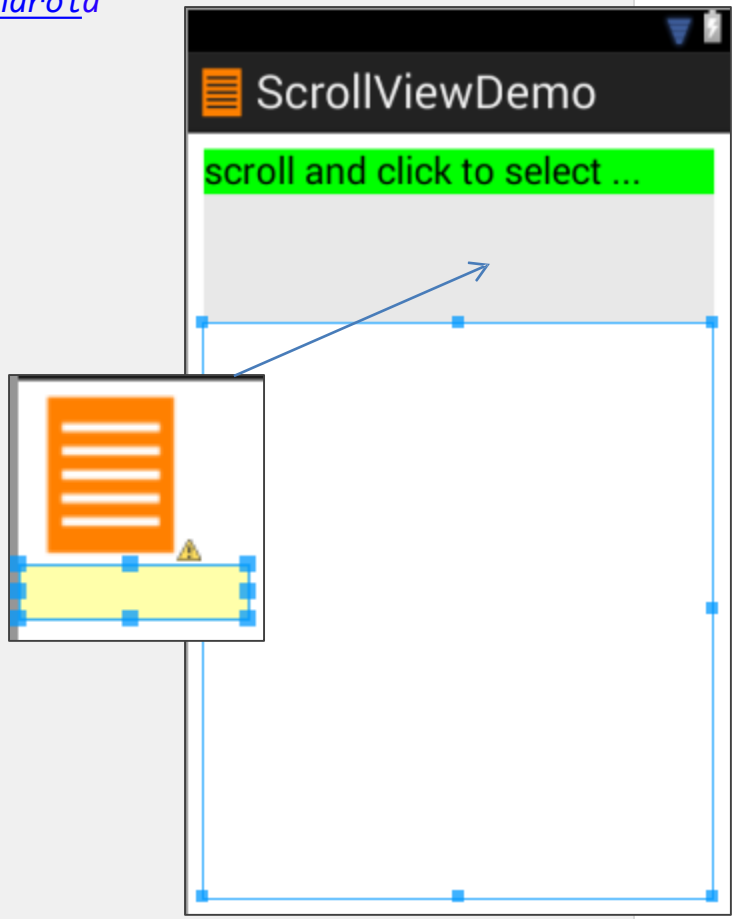
```
    <ImageView
        android:id="@+id/imageSelected"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="2" />

</LinearLayout>
```



Przykład 5: Wykorzystanie HorizontalScrollView

Układ XML : *frame_icon_caption.xml*

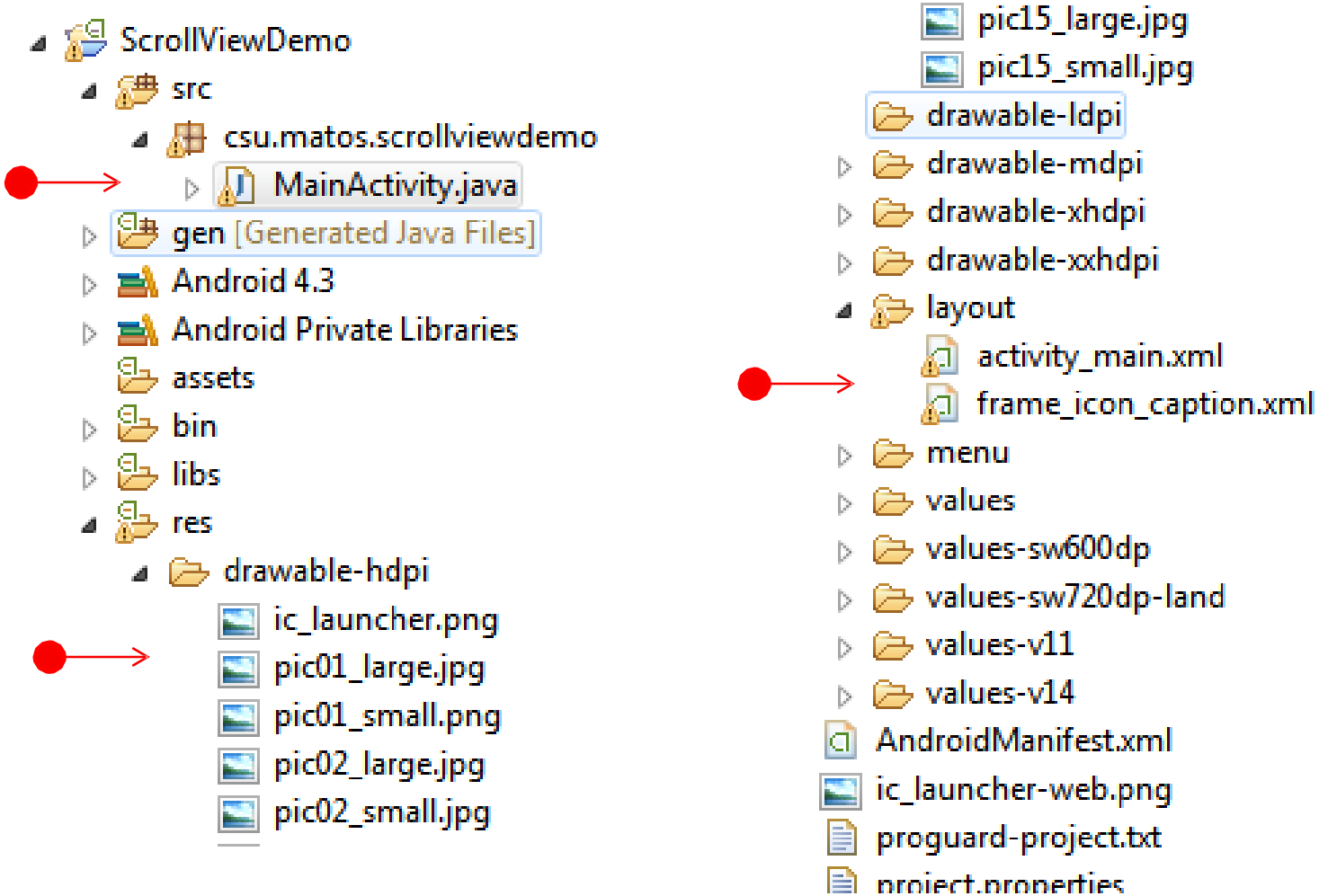
```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/icon"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:paddingLeft="2dp"
        android:paddingRight="2dp"
        android:paddingTop="2dp"
        android:src="@drawable/ic_launcher" />

    <TextView
        android:id="@+id/caption"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#55ffff00"
        android:textSize="20sp" />

</LinearLayout>
```



This layout will be used by an **inflater** to dynamically create new views. These views will be added to the linear layout contained inside the HorizontalScrollerView.

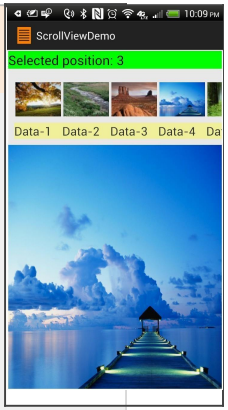
Przykład 5: Wykorzystanie HorizontalScrollView

Struktura aplikacji



Przykład 5: Wykorzystanie HorizontalScrollView

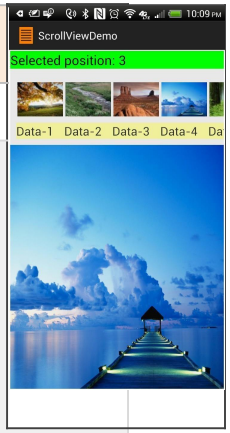
Klasa MainActivity




```
public class MainActivity extends Activity {
    //GUI controls
   TextView txtMsg;
   ViewGroup scrollViewgroup;
   //each frame in the HorizontalScrollView has [icon, caption]
   ImageView icon;
   TextView caption;

   String[] items = { "Data-1", "Data-2", "Data-3", "Data-4", "Data-5",
                      "Data-3", "Data-7", "Data-8", "Data-9", "Data-10", "Data-11",
                      "Data-12", "Data-13", "Data-14", "Data-15" };
   //frame-icons ( 100X100 thumbnails )
   Integer[] thumbnails = { R.drawable.pic01_small, R.drawable.pic02_small,
         R.drawable.pic03_small, R.drawable.pic04_small,
         R.drawable.pic05_small, R.drawable.pic06_small,
         R.drawable.pic07_small, R.drawable.pic08_small,
         R.drawable.pic09_small, R.drawable.pic10_small,
         R.drawable.pic11_small, R.drawable.pic12_small,
         R.drawable.pic13_small, R.drawable.pic14_small,
         R.drawable.pic15_small };
   //large images to be shown (individually) in an ImageView
   Integer[] largeImages = { R.drawable.pic01_large,
         R.drawable.pic02_large, R.drawable.pic03_large,
         R.drawable.pic04_large, R.drawable.pic05_large,
         R.drawable.pic06_large, R.drawable.pic07_large,
         R.drawable.pic08_large, R.drawable.pic09_large,
         R.drawable.pic10_large, R.drawable.pic11_large,
         R.drawable.pic12_large, R.drawable.pic13_large,
         R.drawable.pic14_large, R.drawable.pic15_large };
```

Przykład 5: Wykorzystanie HorizontalScrollView

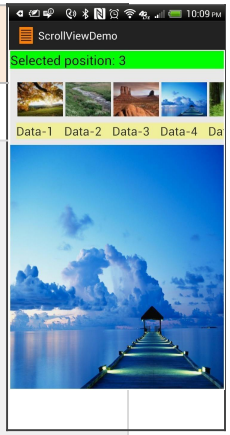
Klasa MainActivity




```
//large image frame for displaying high-quality selected image
ImageView imageSelected;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //bind GUI controls to Java classes
    txtMsg = (TextView) findViewById(R.id.txtMsg);
    imageSelected = (ImageView) findViewById(R.id.imageSelected);

    // this layout goes inside the HorizontalScrollView
    scrollViewgroup = (ViewGroup) findViewById(R.id.viewgroup);

    // populate the ScrollView
    for (int i = 0; i < items.length; i++) {
        //create single frames [icon & caption] using XML inflater
        final View singleFrame = getLayoutInflater().inflate(
                                                R.layout.frame_icon_caption, null );
        //frame-0, frame-1, frame-2, ... and so on
        singleFrame.setId(i);

        TextView caption = (TextView) singleFrame.findViewById(R.id.caption);
        ImageView icon = (ImageView) singleFrame.findViewById(R.id.icon);
        //put data [icon, caption] in each frame
        icon.setImageResource( thumbnails[i] );
        caption.setText( items[i] );
        //add frame to the scrollView
        scrollViewgroup.addView( singleFrame );
```

Przykład 5: Wykorzystanie HorizontalScrollView

MainActivity Class




```
            //each single frame gets its own click listener
            singleFrame.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    String text = "Selected position: " + singleFrame.getId();
                    txtMsg.setText(text);
                    showLargeImage(singleFrame.getId());
                }
            });// listener

        }// for – populating ScrollView

    }

    //display a high-quality version of the image selected using thumbnails
    protected void showLargeImage(int frameId) {
        Drawable selectedLargeImage = getResources()
                                      .getDrawable(largeImages[frameId]);
        imageSelected.setBackground(selectedLargeImage);

    }

}
```

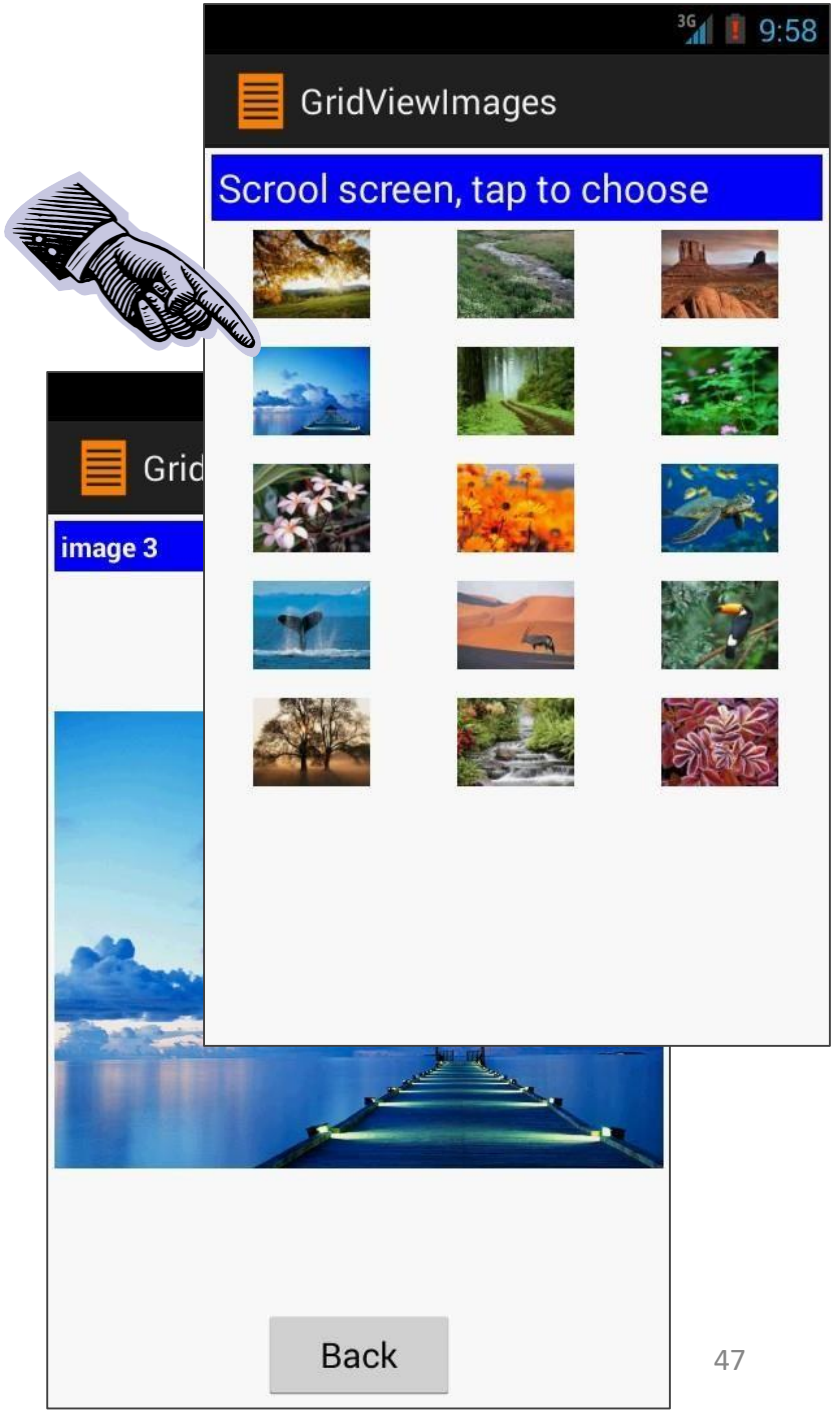
Siatka obrazków

Nieco ciekawszym wizualnie wykorzystaniem komponentu **`GridView`** jest wyświetlanie obrazków zamiast tekstu.

Poniższy przykład pokazuje jako wykorzystać w tym celu ten komponent:

1. Ekran pokazuje tablicę miniatur.
2. Użytkownik dokonuje wyboru poprzez dotknięcie jednej z nich.
3. Aplikacja wyświetla na innym ekranie wersję obrazka o większej rozdzielczości.

Uwaga:
Programista musi stworzyć własny data adapter by określić sposób wyświetlania siatki obrazków.

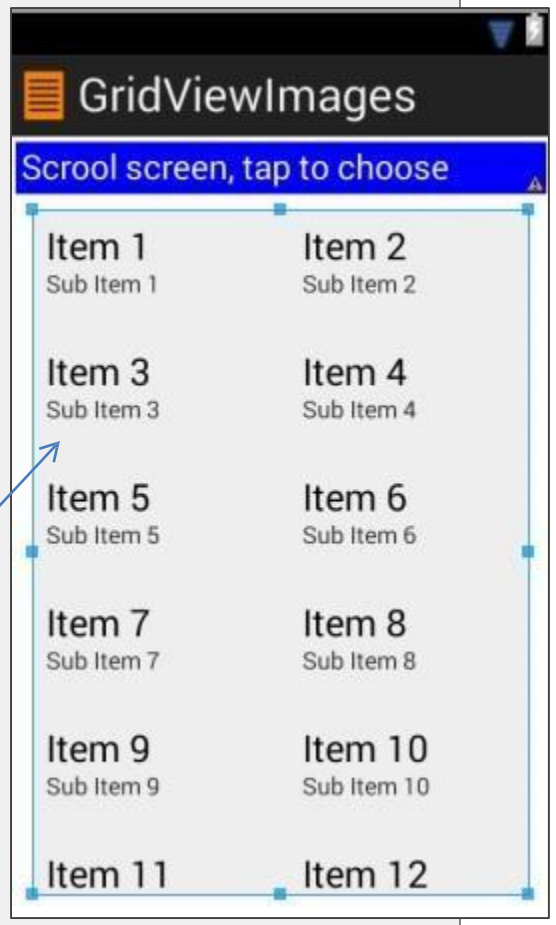


Na podstawie:
http://developer.android.com/guide/topics/ui/layout/gridview.html

Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Układ XML: *activity_main.xml*

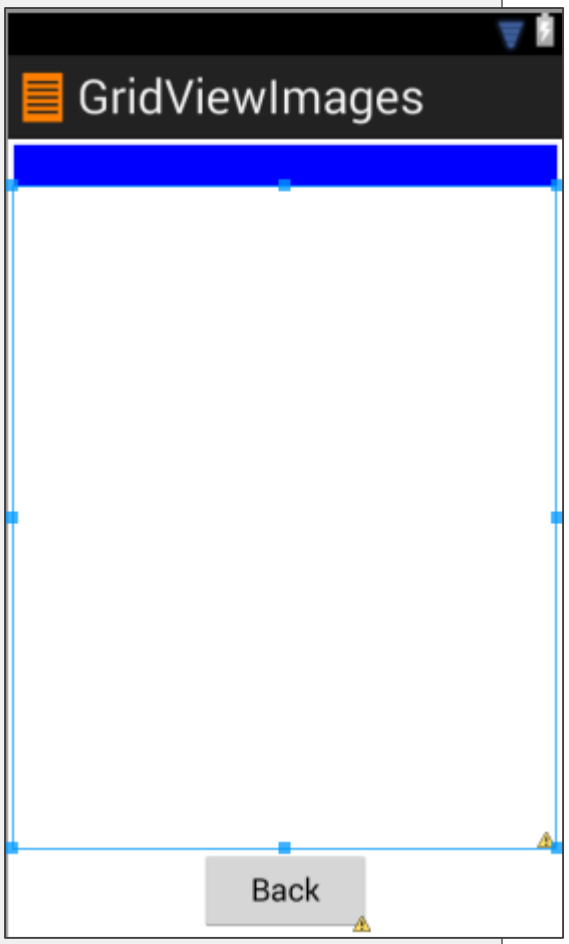
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="3dp" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:padding="3dp"
        android:text="Scrool screen, tap to choose"
        android:textColor="#ffffffff"
        android:textSize="20sp" />

    <GridView android:id="@+id/gridview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="1dp"
        android:columnWidth="100dp"
        android:gravity="center"
        android:horizontalSpacing="5dp"
        android:numColumns="auto_size"
        android:stretchMode="columnWidth"

        android:verticalSpacing="10dp" />
</LinearLayout>
```



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

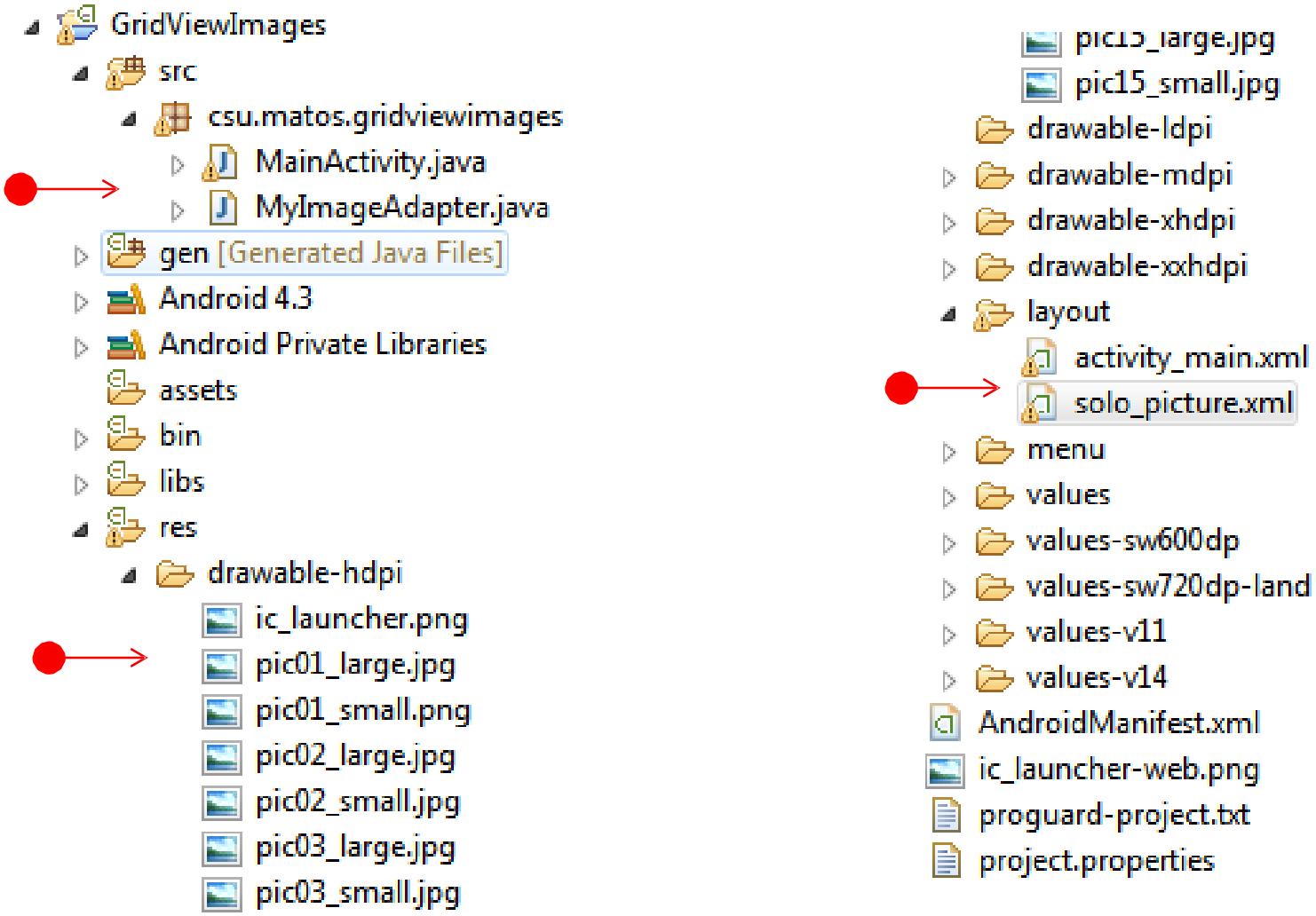
Układ XML: *solo_picture.xml*

```
<?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:padding="3dp"
    android:orientation="vertical" >

    <TextView android:id="@+id/txtSoloMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff" />

    <ImageView
        android:id="@+id/imgSoloPhoto"
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_gravity="center|fill"
        android:layout_weight="2" />

    <Button android:id="@+id/btnBack"
        android:layout_width="100dip"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Back" />


</LinearLayout>
```



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Struktura aplikacji



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Klasa MainActivity


```
public class MainActivity extends Activity implements OnItemClickListener {
  //GUI control  bound to screen1 (holding GidView)
  GridView gridview;
  //GUI controls bound to screen2 (holding single ImageView)
  TextView txtSoloMsg;
  ImageView imgSoloPhoto;
  Button btnBack;
  // initialize array of smallImages (100x75 thumbnails)
  Integer[] smallImages = { R.drawable.pic01_small,
      R.drawable.pic02_small, R.drawable.pic03_small,
      R.drawable.pic04_small, R.drawable.pic05_small,
      R.drawable.pic06_small, R.drawable.pic07_small,
      R.drawable.pic08_small, R.drawable.pic09_small,
      R.drawable.pic10_small, R.drawable.pic11_small,
      R.drawable.pic12_small, R.drawable.pic13_small,
      R.drawable.pic14_small, R.drawable.pic15_small };
  //initialize array of high-resolution images (1024x768)
  Integer[] largeImages = { R.drawable.pic01_large,
      R.drawable.pic02_large, R.drawable.pic03_large,
      R.drawable.pic04_large, R.drawable.pic05_large,
      R.drawable.pic06_large, R.drawable.pic07_large,
      R.drawable.pic08_large, R.drawable.pic09_large,
      R.drawable.pic10_large, R.drawable.pic11_large,
      R.drawable.pic12_large, R.drawable.pic13_large,
      R.drawable.pic14_large, R.drawable.pic15_large };

  //in case you want to use-save state values
  Bundle myOriginalMemoryBundle;
```

Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

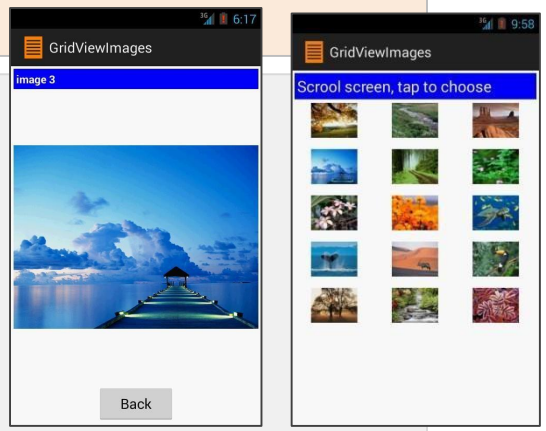
Klasa MainActivity


```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  myOriginalMemoryBundle = savedInstanceState;
  // setup GridView with its custom adapter and listener
  gridview = (GridView) findViewById(R.id.gridview);
  gridview.setAdapter(new MyImageAdapter(this, smallImages));
  gridview.setOnItemClickListener(this);
}

// on response to tapping, display a high-resolution version of the chosen thumbnail
@Override
public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
  showBigScreen(position);
}//onItemClick

private void showBigScreen(int position) {
  // show the selected picture as a single frame
  setContentView(R.layout.solo_picture);
  txtSoloMsg = (TextView) findViewById(R.id.txtSoloMsg);
  imgSoloPhoto = (ImageView) findViewById(R.id.imgSoloPhoto);
  txtSoloMsg.setText("image " + position );

  imgSoloPhoto.setImageResource( largeImages[position] );
```

Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Klasa MainActivity

```
    btnBack = (Button) findViewById(R.id.btnBack);

    btnBack.setOnClickListener(new OnClickListener() {

      @Override
      public void onClick(View v) {
        // redraw the main screen showing the GridView
        onCreate(myOriginalMemoryBundle);
      }

    });

  }// showBigScreen

}
```



Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Własny adapter: Klasa MyImageAdapter

```
// This custom adapter populates the GridView with a visual
// representation of each thumbnail in the input data set.
// It also implements a method -getView()- to access
// individual cells in the GridView.

public class MyImageAdapter extends BaseAdapter{

   private Context context;// calling activity context
    Integer[] smallImages;      // thumbnail data set
   public MyImageAdapter(Context callingActivityContext,
                         Integer[] thumbnails) {
      context = callingActivityContext;
      smallImages = thumbnails;
   }

   // how many entries are there in the data set
   public int getCount() {
      return smallImages.length;
   }

   // what is in a given 'position' in the data set
   public Object getItem(int position) {
      return smallImages[position];
   }

   // what is the ID of data item in given 'position'
   public long getItemId(int position) {
      return position;
   }
```

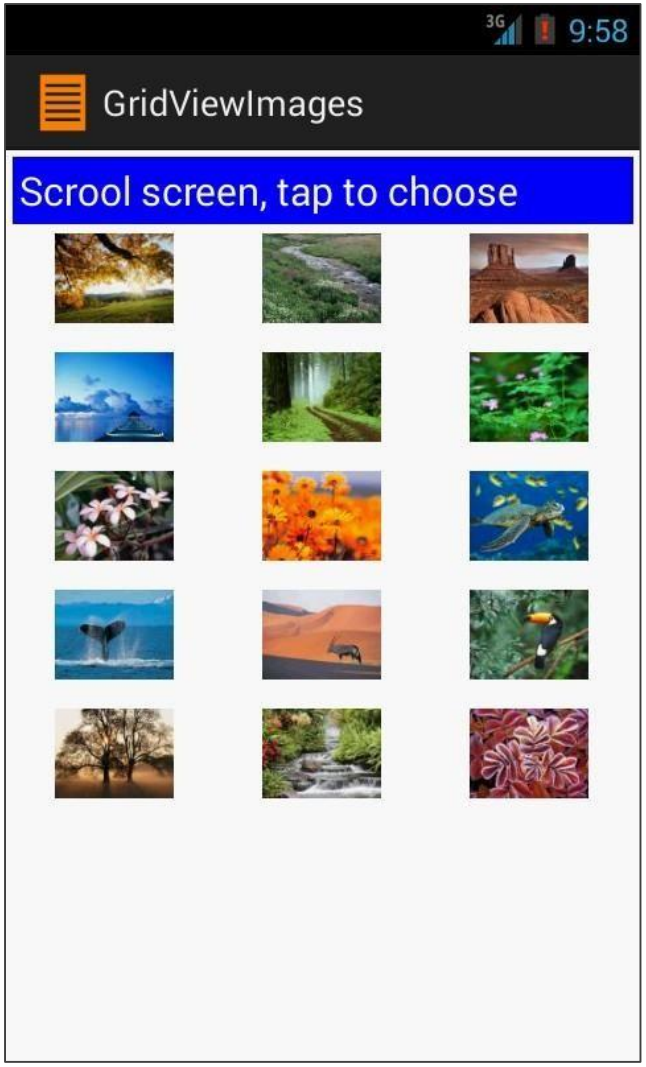
Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Własny adapter: Klasa MyImageAdapter

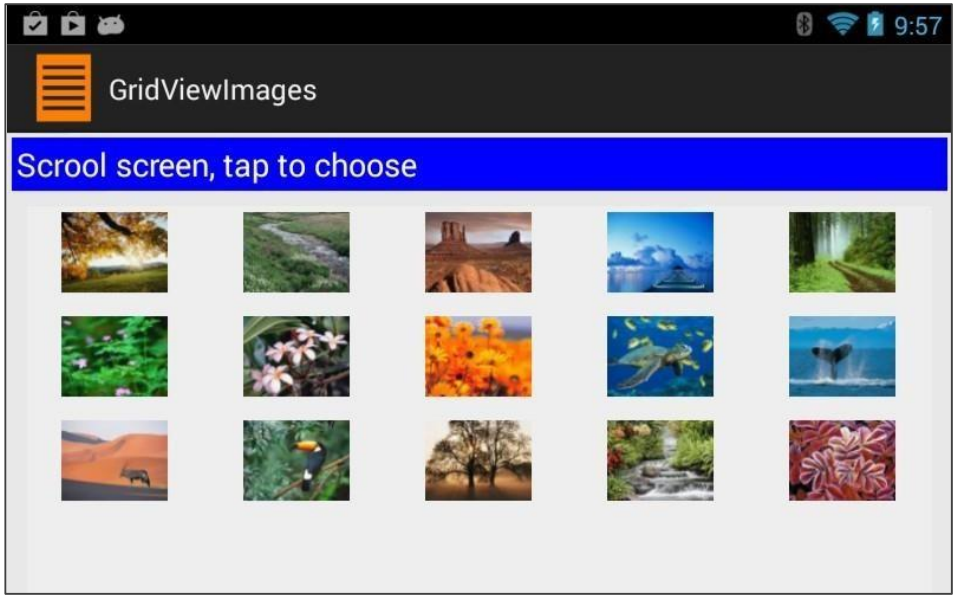

```
    // create a view for each thumbnail in the data set
    public View getView(int position, View convertView, ViewGroup parent) {

        ImageView imageView;

        // if possible, reuse (convertView) image already held in cache
        if (convertView == null) {
            // new image in GridView formatted to:
            // 100x75 pixels (its actual size)
            // center-cropped, and 5dp padding all around

            imageView = new ImageView(context);
            imageView.setLayoutParams( new GridView.LayoutParams(100, 75) );
            imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
            imageView.setPadding(5, 5, 5, 5);

        } else {
            imageView = (ImageView) convertView;
        }

        imageView.setImageResource(smallImages[position]);

        return imageView;
    }

}//MyImageAdapter
```

Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Wynik działania aplikacji na różnych urządzeniach



Widok na telefonie komórkowym



Widok z tabletu o rozdzielczości 1028x728 tablet. Klauzula:

```
android:numColumns="auto_size"
```

określa automatyczny rozkład i liczbę kolumn.

Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Eksperyment - Zmiana GridView na ListView

Celem eksperymentu jest zmiana poprzedniego przykładu by wyświetlał grafikę w **ListView** zamiast komponencie **GridView**. Tak jak poprzednio, po wyborze danej miniatury z listy, prezentowany jest obrazek w wyższej rozdzielczości.

KROKI

1. Zmień układ **activity_main.xml**. Zmień znacznik **<GridView** … na **<ListView.** Resztę pozostaw bez zmian.
2. W głównej klasie java zamień wszystkie referencje dotyczące GridView na ListView. Przykładowy kod po zmianach:

```
ListView gridview;
...
gridview = (ListView) findViewById(R.id.gridview);
```

3. We własnym adapterze dokonaj następującej zmiany by nowy obrazek (imageView) powinien zostać dodany do ListView (zamiast)

```
imageView.setLayoutParams(new ListView
                                .LayoutParams(100, 75) );
```

4. Pozostały kod adaptera pozostaw bez zmian.

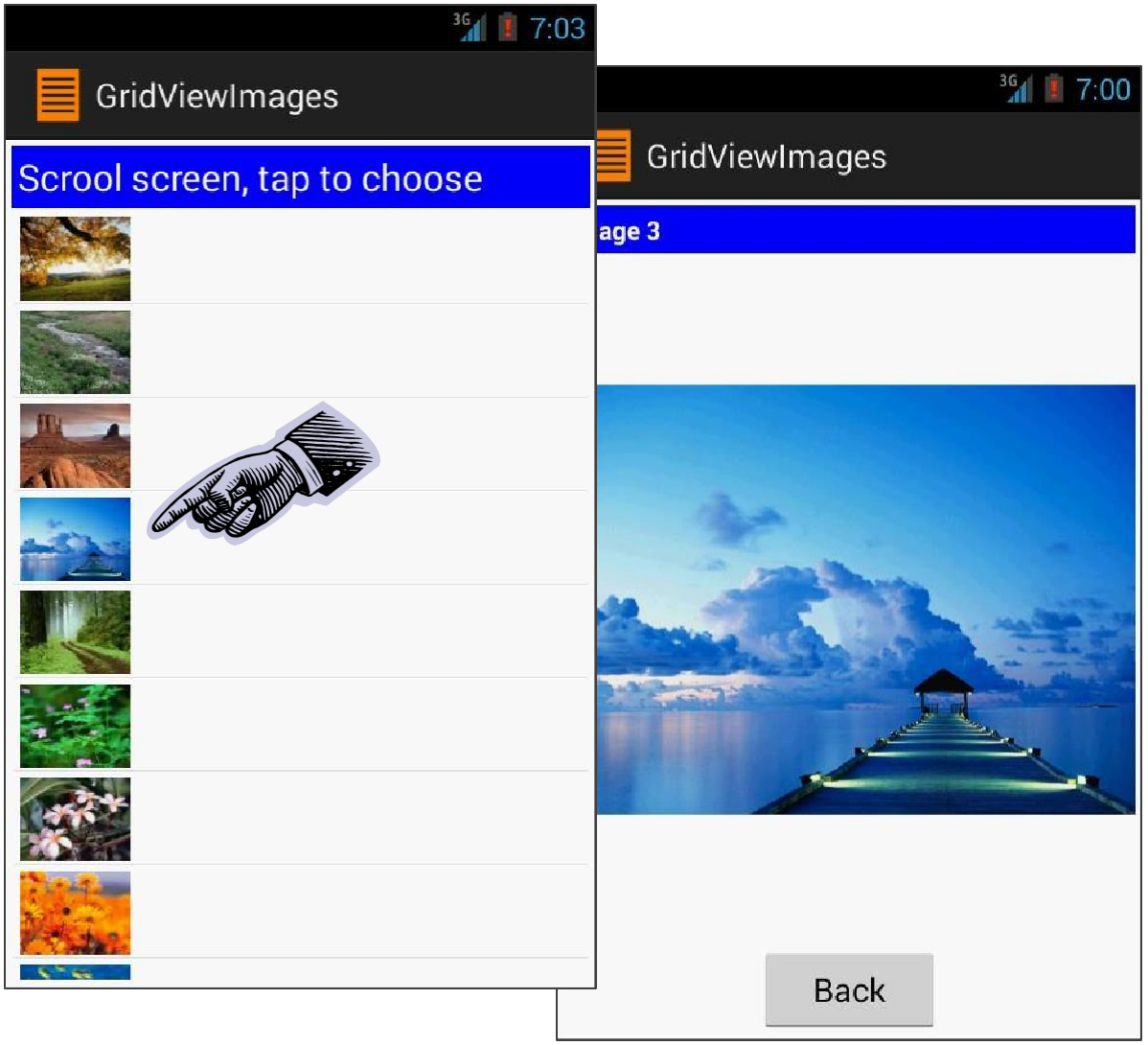
Przykład 6: Wykorzystanie GridView by wyświetlić obrazki

Eksperyment - Zmiana GridView na ListView

Nowa aplikacja powinna wyglądać następująco.

Proszę zauważyć, że główna aktywność prezentuje dane w formie listy.

Więcej w przykładzie 8.

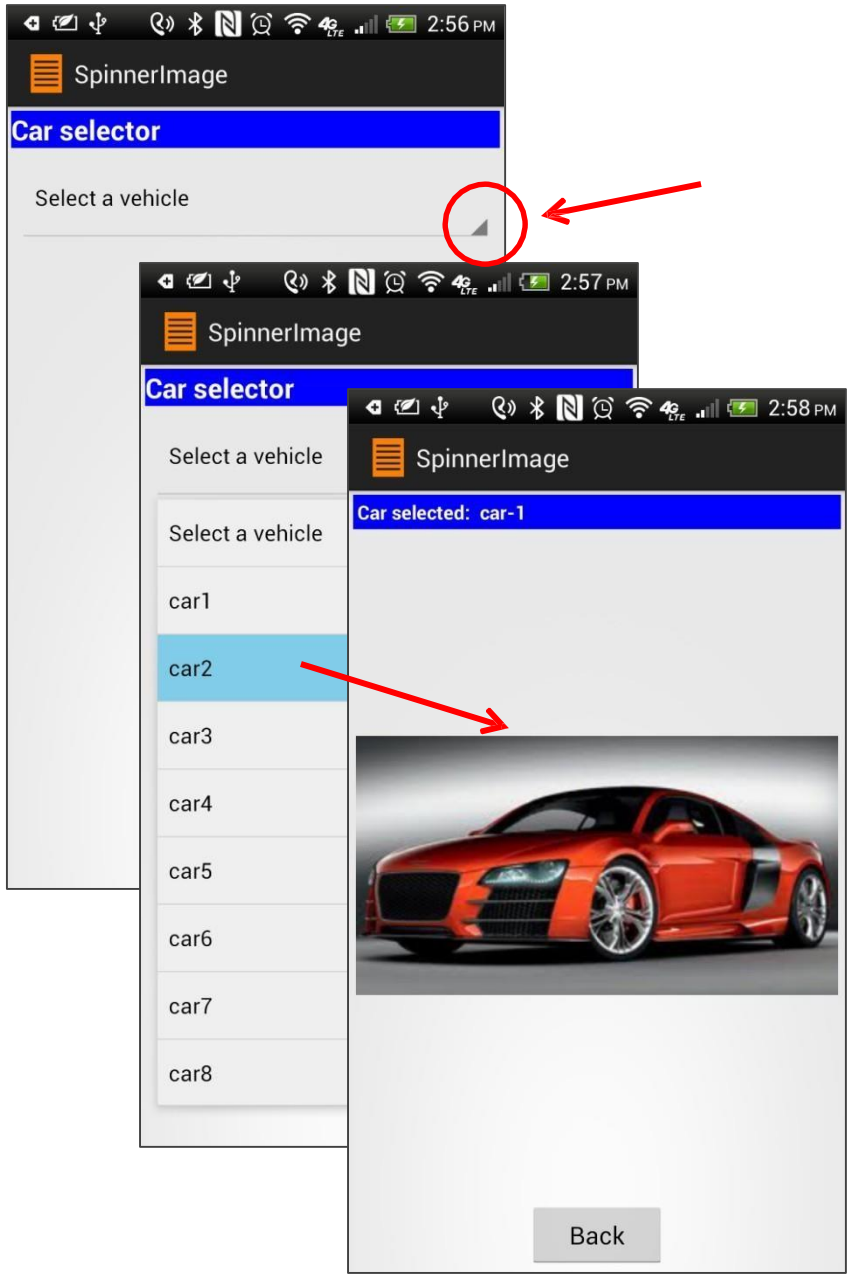


Przykład 7: Wykorzystanie Spinner'a

Jest to inna wersja poprzednio prezentowanego przykładu.

Tym razem lista możliwych wyborów prezentowana jest za pomocą komponentu typu Spinner.

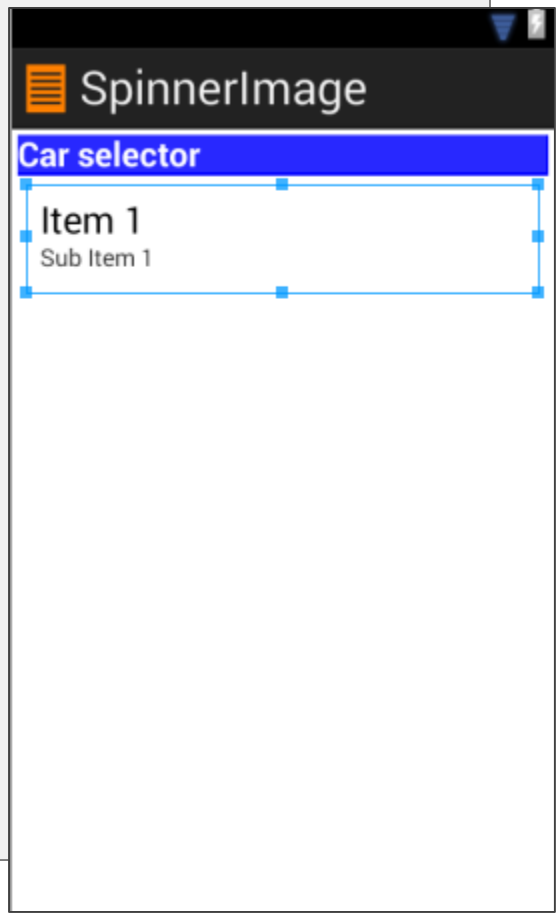
Po kliknięciu przez użytkownika na określonym wierszu, obrazek dotyczący wybranej opcji prezentowany jest na ekranie.



Przykład 7: Wykorzystanie Spinner'a

1. Układ XML: *activity_main.xml*

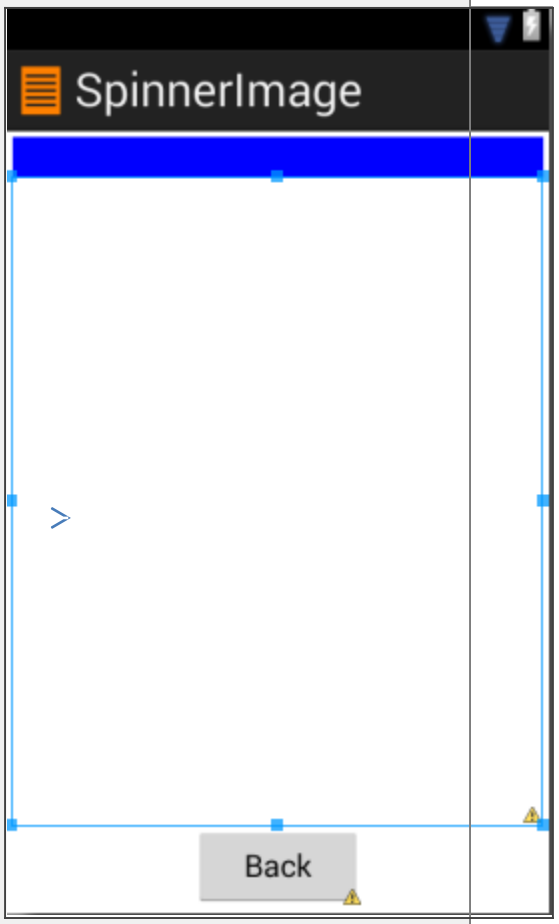
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="3dp" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:text="Car selector"
        android:textColor="#ffffffff"
        android:textSize="20sp"
        android:textStyle="bold" />

    <Spinner android:id="@+id/spinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dip" />

</LinearLayout>
```



Przykład 7: Wykorzystanie Spinner'a

2. Układ XML: *solo_picture.xml*

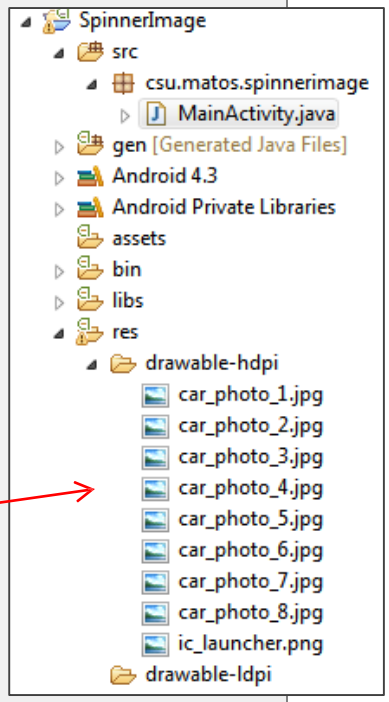
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="3dp"
    android:orientation="vertical" >

    <TextView android:id="@+id/txtSoloMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:textColor="#FFFFFFFF"
        android:padding="3dp"
        android:background="#ff0000ff" />

    <ImageView
        android:id="@+id/imgSoloPhoto"
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_gravity="center|fill"
        android:layout_weight="2" />

    <Button android:id="@+id/btnBack"
        android:layout_width="100dip"
        android:layout_height="wrap_content"


        android:layout_gravity="center_horizontal"
        android:text="Back" />
</LinearLayout>
```



Przykład 7: Wykorzystanie Spinner'a

Klasa MainActivity

```
public class MainActivity extends Activity implements
AdapterView.OnItemSelectedListener {

   // GUI controls in the main screen
   Spinner spinner;

   // GUI controls in the solo_picture screen
   TextView txtSoloMsg;
   ImageView imageSelectedCar;
   Button btnBack;

   // captions to be listed by the spinner
   String[] items = { "Select a vehicle", "car1", "car2", "car3", "car4",
                      "car5", "car6", "car7", "car8" };
   // object IDs of car pictures
   Integer[] carImageArray = new Integer[] { R.drawable.car_photo_1,
         R.drawable.car_photo_2, R.drawable.car_photo_3,
         R.drawable.car_photo_4, R.drawable.car_photo_5,
         R.drawable.car_photo_6, R.drawable.car_photo_7,
         R.drawable.car_photo_8, };
   Bundle myStateInfo;

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

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



Przykład 7: Wykorzystanie Spinner'a

Klasa MainActivity

```
        spinner = (Spinner) findViewById(R.id.spinner);
        spinner.setAdapter(new ArrayAdapter<String>(this,
                                android.R.layout.simple_spinner_dropdown_item,
                                items));
        spinner.setOnItemSelectedListener(this);

    }// onCreate

    // display screen showing image of the selected car
    private void showBigImage(int position) {
        // show the selected picture as a single frame
        setContentView(R.layout.solo_picture);
        txtSoloMsg = (TextView) findViewById(R.id.txtSoloMsg);
        imageSelectedCar = (ImageView) findViewById(R.id.imgSoloPhoto);
        txtSoloMsg.setText("Car selected:  car-" + position);

        imageSelectedCar.setImageResource(carImageArray[position]);

        btnBack = (Button) findViewById(R.id.btnBack);
        btnBack.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                // redraw the main screen showing the spinner
                onCreate(myStateInfo);
            }
        });

    }// showBigScreen
```

Przykład 7: Wykorzystanie Spinner'a

Klasa MainActivity

```
    // next two methods implement the spinner listener
    @Override
    public void onItemSelected(AdapterView<?> parent, View v, int position,
          long id) {
       //ignore position 0. It holds just a label ("SELECT A VEHICLE...")
       if (position > 0) {
          showBigImage(position - 1);
       }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
       // DO NOTHING - needed by the interface
    }

}
```

Własne widżety wykorzystujące listy

Niestandardowe listy

- Android dostarcza wiele układów do prezentowania danych na listach (takie jak: `android.R.layout.simple_list_item_1`, `android.R.layout.simple_list_item_2`, itp).

- Jednakże, w niektórych przypadkach istnieje potrzeba zdecydowanie większej kontroli nad formatowanie i wyglądem poszczególnych elementów.

- W takich przypadkach należy ***stworzyć własną podklasę rozszerzającą Data Adapter.***

- Zostanie to pokazane na następnym przykładzie.

Przykład 8: Własne listy

Tworzenie własnego Data Adapter

By stworzyć własny Data Adapter należy:

1. Stworzyć klasę dziedziczącą po klasie `ArrayAdapter`
2. Przeciążyć metodę **`getView()`**
3. Stworzyć (przez inflację) poszczególne wiersze samemu.

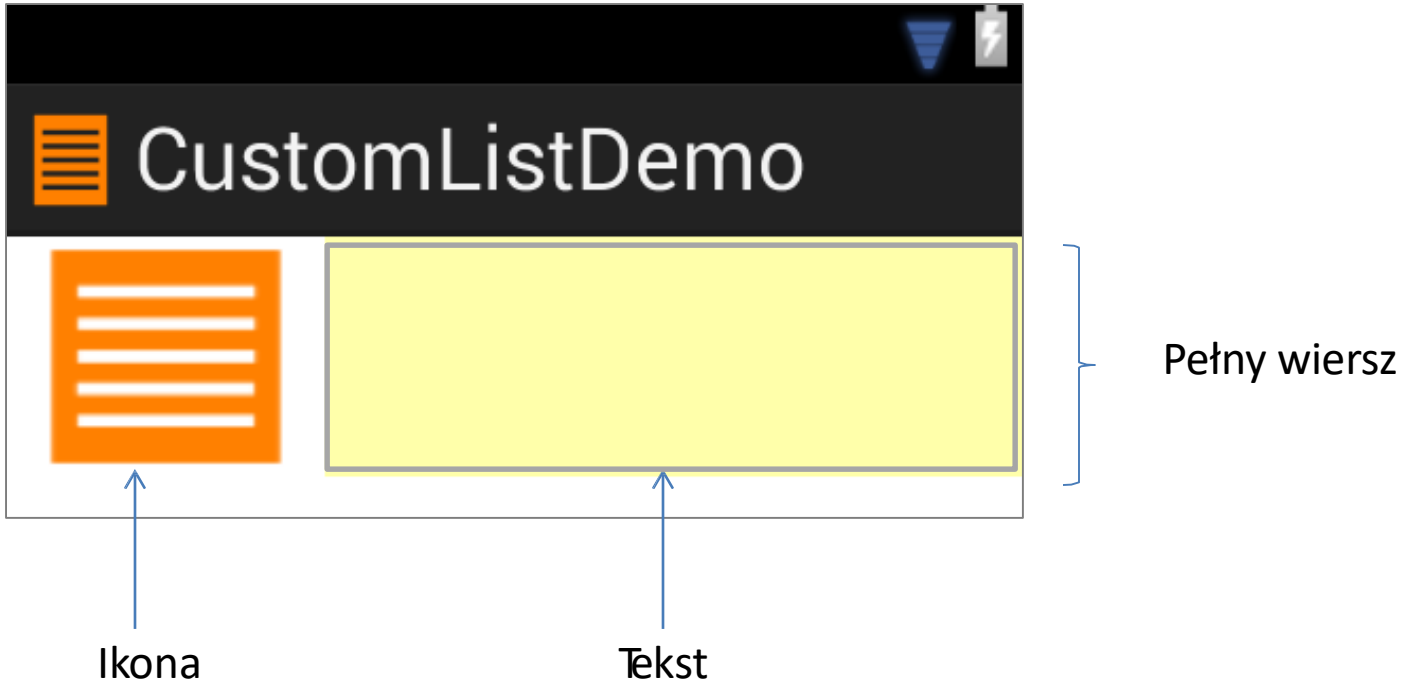
```
public class MyCustomAdapter extends ArrayAdapter{

    // class variables go here ...

    public MyCustomAdapter(...) {   }

    public View getView(...) {    }

}//MyCustomAdapter
```

Dla każdego elementu dostarczanego przez adapter metoda `getView()` **zwraca jego wizualną reprezentację.**

Przykład 8: Własne listy

Problem: Stworzenie własnego wyglądu dla wierszy

Każdy wiersz będzie pokazywał ikonę (po lewej) oraz tekst (po prawej).



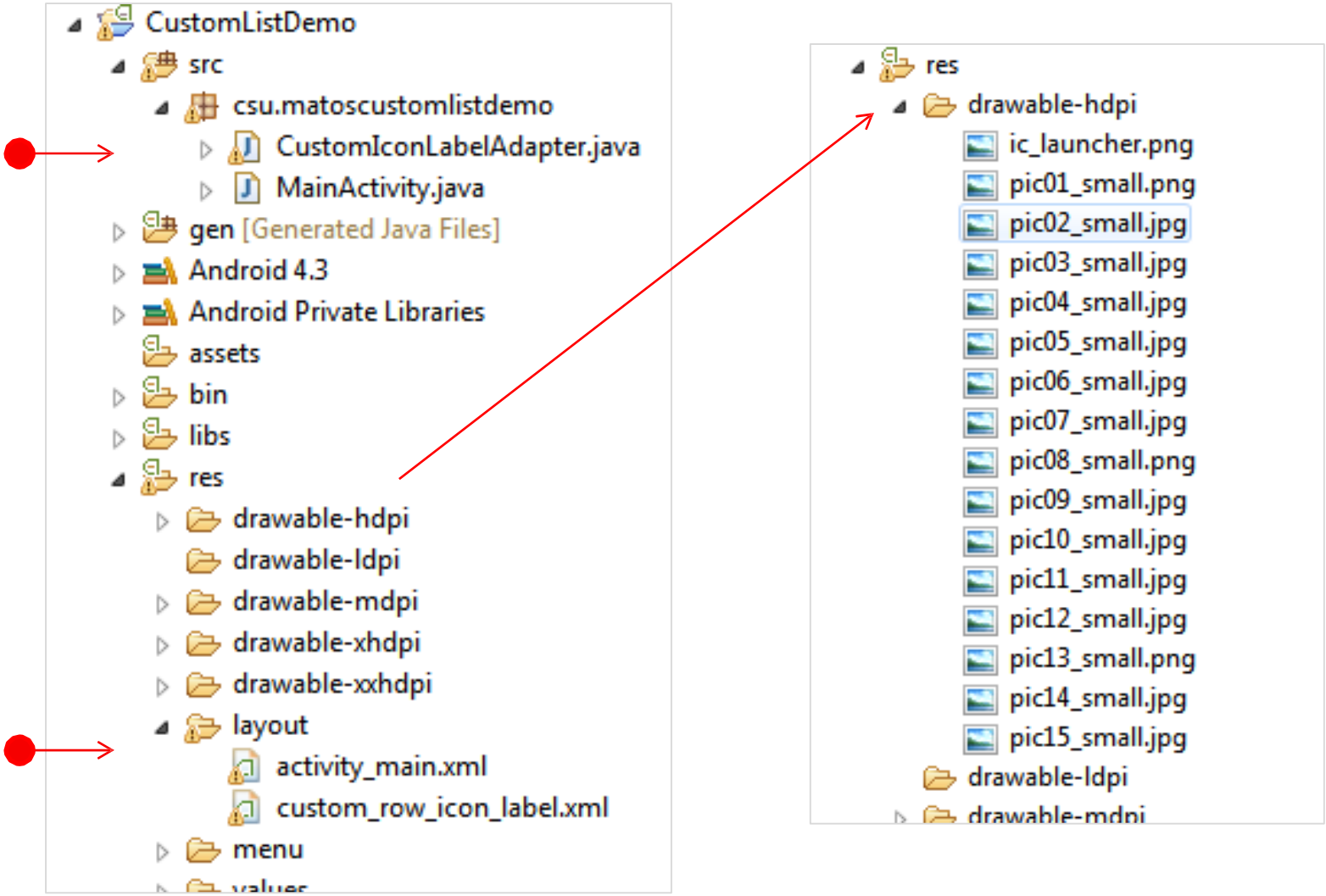
Przykład 8: Własne listy

Wygląd aplikacji

Każdy wiersz składa się z ikony i tekstu

Przykład 8: Własne listy

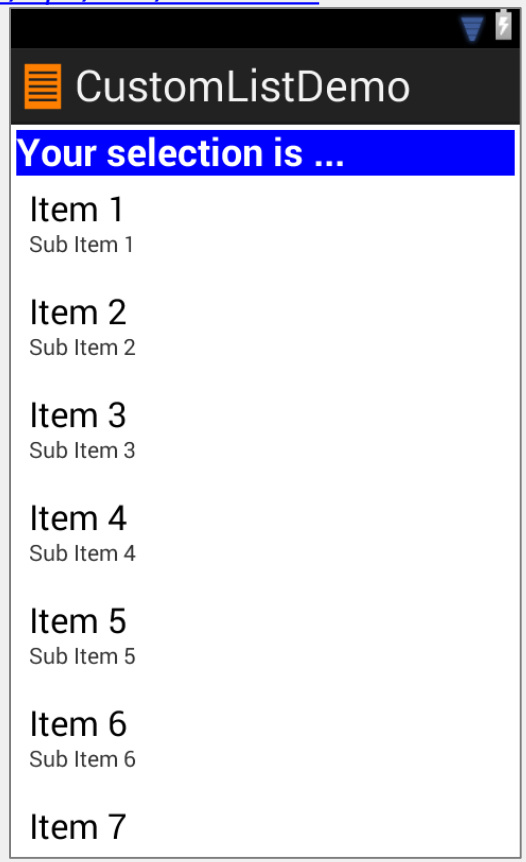
1. Struktura aplikacji



Przykład 8: Własne listy

1. Układ XML: *activity_main.xml*

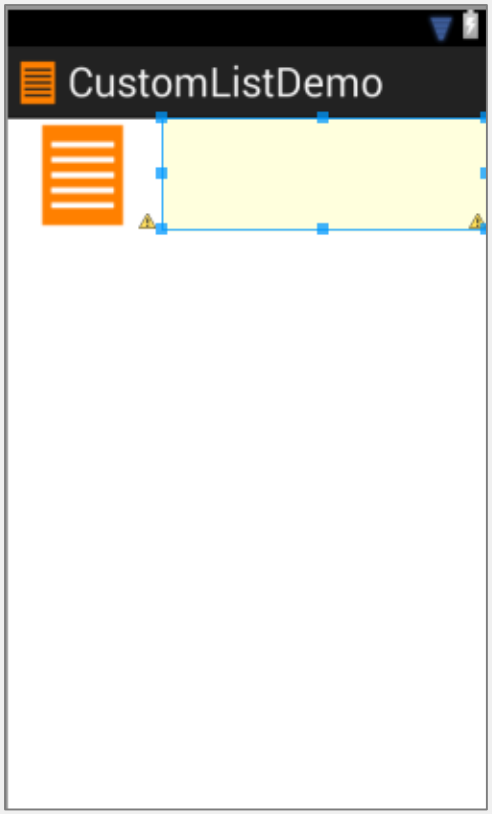
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="3dp"
    android:orientation="vertical" >

    <TextView android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:text="Your selection is ..."
        android:textColor="#ffffffff"
        android:textSize="24sp"
        android:textStyle="bold" />

    <ListView android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </ListView>

</LinearLayout>
```



Przykład 8: Własne listy

2. Układ XML: *custom_row_icon_label.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <ImageView android:id="@+id/icon"
        android:layout_width="100dp"
        android:layout_height="75dp"
        android:layout_marginRight="3dp"
        android:src="@drawable/ic_launcher" />

    <TextView android:id="@+id/label"
        android:layout_width="match_parent"
        android:layout_height="75dp"
        android:background="#22ffff00"
        android:textSize="20sp" />

</LinearLayout>
```



Przykład 8: Własne listy

2. Klasa MainActivity

```
public class MainActivity extends ListActivity {
   TextView txtMsg;
   // The n-th row in the list will consist of [icon, label]
   // where icon = thumbnail[n] and label=items[n]
   String[] items = { "Data-1", "Data-2", "Data-3", "Data-4", "Data-
        5", "Data-6", "Data-7", "Data-8", "Data-9", "Data-10", "Data-
        11", "Data-12", "Data-13", "Data-14", "Data-15" };

   Integer[] thumbnails = { R.drawable.pic01_small,
        R.drawable.pic02_small, R.drawable.pic03_small,
        R.drawable.pic04_small, R.drawable.pic05_small,
        R.drawable.pic06_small, R.drawable.pic07_small,
        R.drawable.pic08_small, R.drawable.pic09_small,
        R.drawable.pic10_small, R.drawable.pic11_small,
        R.drawable.pic12_small, R.drawable.pic13_small,
        R.drawable.pic14_small, R.drawable.pic15_small };

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

Przykład 8: Własne listy

2. Klasa MainActivity

```
      // the arguments of the custom adapter are:
      // activityContex, layout-to-be-inflated, labels, icons
      CustomIconLabelAdapter adapter = new CustomIconLabelAdapter(
                        this,
                        R.layout.custom_row_icon_label,
                        items,
                        thumbnails);
      // bind intrinsic ListView to custom
      adapter setListAdapter(adapter);

   }//onCreate

   // react to user's selection of a row
   @Override
   protected void onListItemClick(ListView l, View v, int position, long id) {
      super.onListItemClick(l, v, position, id);
      String text = " Position: " + position + "    " + items[position];
      txtMsg.setText(text);
   }//listener

}//class
```

Przykład 8: Własne listy

3. Klasa CustomIconLabelAdapter

```
class CustomIconLabelAdapter extends ArrayAdapter <String>
   { Context context;
   Integer[] thumbnails;
   String[] items;

   public CustomIconLabelAdapter( Context context, int
                                  layoutToBeInflated, String[] items,
                                  Integer[] thumbnails) {
      super(context, R.layout.custom_row_icon_label, items);
      this.context = context;
      this.thumbnails = thumbnails;
      this.items = items;
   }
   @Override
   public View getView(int position, View convertView, ViewGroup parent)
   {

      LayoutInflater inflater = ((Activity)
      context).getLayoutInflater(); View row =
      inflater.inflate(R.layout.custom_row_icon_label, null);

      TextView label = (TextView) row.findViewById(R.id.label);
      ImageView icon = (ImageView)
      row.findViewById(R.id.icon);

      label.setText(items[position]);
      icon.setImageResource(thumbnails[position]);
      return row;

}
```

Przykład 8: Własne listy

`LayoutInflater()`

- Klasa **LayoutInflater** konwertuje układ XML w rzeczywista hierarchię obiektów klasy View. Obiekty poddane inflacji dołączane są do aktualnego widoku. Omawiana klasa działa zwykle w porozumieniu z ArrayAdapter.

- Prosty **ArrayAdapter** wymaga podania 3 argumentów: aktualnego **kontekstu**, **układu** wedle którego wiersze są formatowane, danych **źródłowych.**
  - Przeciążona metoda `getView()` tworzy niestandardowy układ wierszy układając grafikę i test z danych źródłowych na podstawie podanej specyfikacji układu XML.
  - Po stworzeniu, widok (wiersz) jest prezentowany (dodawany do listy).
  - Ten proces jest powtarzany dla każdego elementu dostarczonego przez ArrayAdapter.

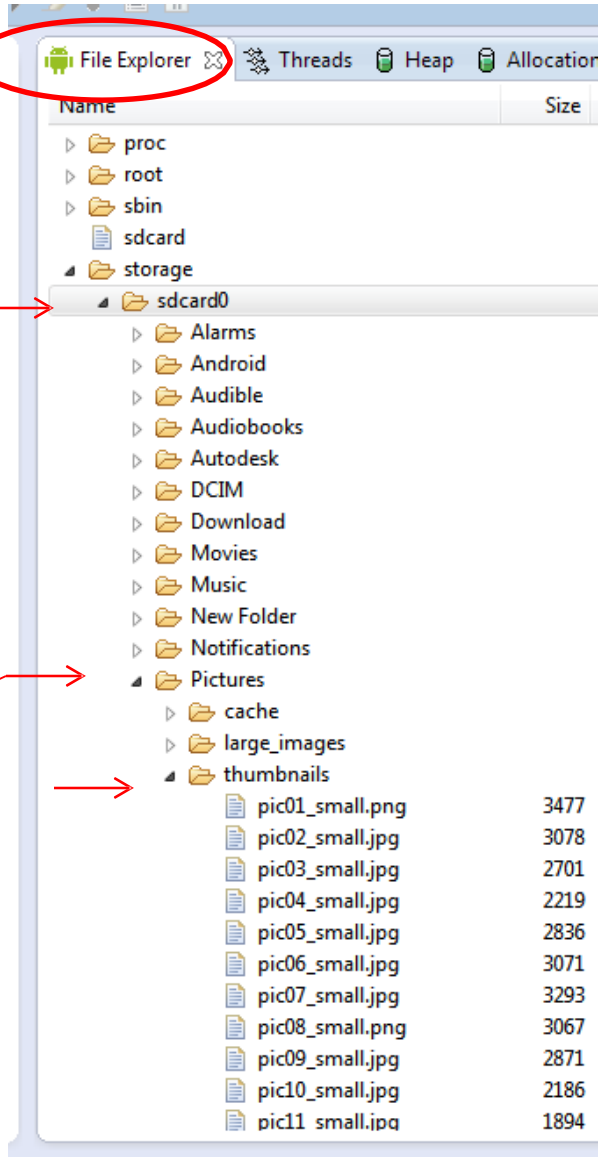
Przykład 9: Przechowywanie grafik na karcie SD

Wariant przykładu 5.

W poprzednich przykładach, obrazki były przechowywane w pamięci aplikacji (przestrzeni jej zasobów).

Tym razem wykorzystywany będzie zewnętrzny nośnik (**karta SD**).

Zakłada się, że zdjęcia znajdują się w katalogu karty pamięci **/Pictures/thumbnails/**



Przykład 9: Przechowywanie grafik na karcie SD

Klasa MainActivity

```
public class MainActivity extends Activity {

  ViewGroup scrollViewgroup;
  ImageView icon;
  TextView caption;
  TextView txtMsg;
  int index;

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

    // this layout is contained inside the horizontalScrollerView
    scrollViewgroup = (ViewGroup) findViewById(R.id.scrollViewgroup);

    try {
      // Environment class allows access to your 'MEDIA' variables
      String absolutePath2SdCard = Environment.getExternalStorageDirectory()
                                              .getAbsolutePath();

      String path2PicturesOnSdCard = absolutePath2SdCard + "/Pictures/thumbnails/";
      // .listFiles() returns an array of files contained in the directory
      // represented by this file-path (or NULL if not a directory).
      File sdPictureFiles = new File(path2PicturesOnSdCard);
      File[] files = sdPictureFiles.listFiles();
```

Wykorzystywane są dokładnie te same układy co wcześniej.

Przykład 9: Przechowywanie grafik na karcie SD

Klasa MainActivity


```
txtMsg.append("\nNum files: " + files.length);

File file;

for (index = 0; index < files.length; index++) {
  file = files[index];

  final View frame = getLayoutInflater().inflate(
                                        R.layout.frame_icon_label, null);

  TextView caption = (TextView) frame.findViewById(R.id.caption);
  ImageView icon = (ImageView) frame.findViewById(R.id.icon);

  // convert (jpg, png,...) file into an acceptable bitmap
  BitmapFactory.Options bmOptions = new BitmapFactory.Options();

  bmOptions.inSampleSize = 1; // one-to-one scale

  Bitmap bitmap = BitmapFactory.decodeFile( file.getAbsolutePath(),
                                            bmOptions);

  icon.setImageBitmap(bitmap);

  caption.setText("File-" + index);
```

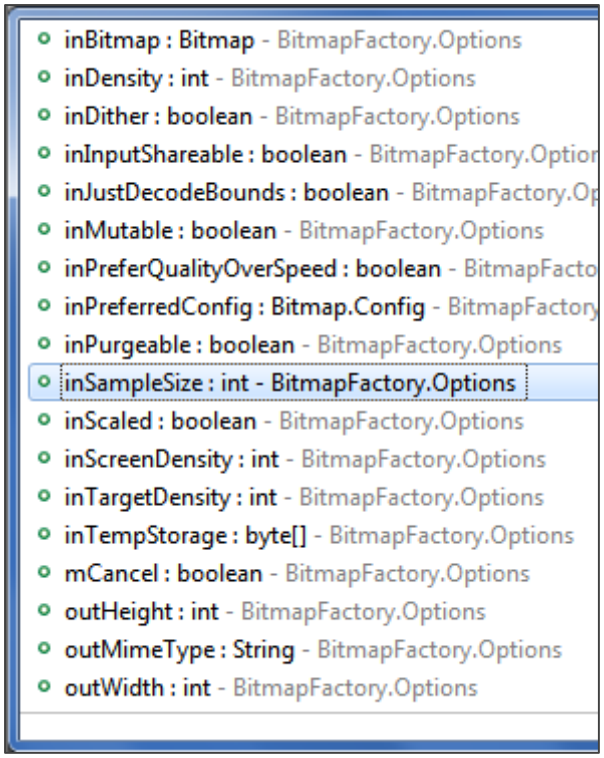
Przykład 9: Przechowywanie grafik na karcie SD

Klasa MainActivity


```
        scrollViewgroup.addView(frame);

        frame.setId(index);

        frame.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
            String text = "Selected position: " + frame.getId();
            txtMsg.setText(text);
          }
        }); // listener

      }// for

    } catch (Exception e) {

      txtMsg.append("\nError: " + e.getMessage());
    }

  } // onCreate
} // class
```

Przykład 9: Przechowywanie grafik na karcie SD

Wykorzystanie klasy BitmapFactory



BitmapFactory
Tworzy obiekty bitmapy z różnych źródeł,
wliczając pliki, strumieni, czy tablic bajtów.

```
Bitmap bitmap= BitmapFactory.decodeFile(
                    file.getAbsolutePath(),
                    Options);
```

Skalowanie wykorzystując właściwość **inSampleSize**:

- Dla wartości **1**, zwraca oryginalny obrazek.
- Jeżeli ustawiono wartość **> 1**, dekoder zwraca mniejszy obrazek.
- Przykładowo, dla inSampleSize **== 4** zostanie zwrócony obrazek o wielkości **1/4** oryginału (co implikuje 1/16 liczbę oryginalnych pikseli).
- Wartość **<** 1 jest traktowana jak w przypadku pierwszym.

Źródło:
http://developer.android.com/reference/android/graphics/BitmapFactory.Options.html

Wyświetlanie danych na listach

Grafiki zrealizowano za pomocą narzędzia: http://android-ui-utils.googlecode.com/hg/asset-studio/dist/device-frames.html

Dodatek A: Gotowe zasoby

SDK Androida zawiera wiele uprzednio zdefiniowanych szablonów/stylów. Niektóre z nich mogą zostać znalezione w:

C:\ Your-Path \Android\android-sdk\platforms\android-xx\data\res\layout
C:\ Your-Path \Android\android-sdk\platforms\android-xx\data\res\values\styles.xml

Przykład:

Poniżej znajduje się definicja szablonu o nazwie: **android.R.layout.simple_list_item_1**. Składa się z pojedynczego pola tekstowego TextView nazwanego “text1”, jego zawartość jest wycentrowana, użyto dużej czcionki i dodano margines.

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:minHeight="?android:attr/listPreferredItemHeight"
    android:paddingLeft="6dip"
    android:textAppearance="?android:attr/textAppearanceLarge"
    />
```

Dodatek A: Gotowe zasoby

Globalne, zdefiniowane szablony

Poniżej znajduje się definicja szablonu: ***simple_spinner_dropdown_item*** w którym każdy wiersz zawiera pole wyboru i tekst.

Źródło: http://code.google.com/p/pdn-slatedroid/source/browse/trunk/eclair/frameworks/base/core/res/res/layout/simple_spinner_dropdown_item.xml?r=44

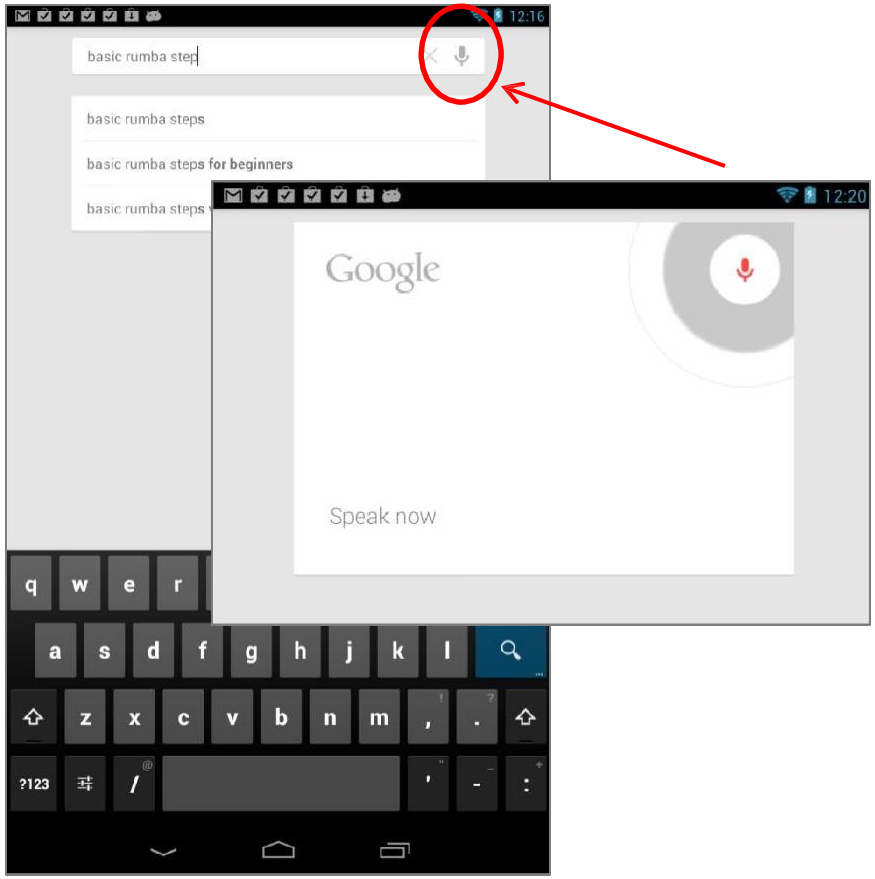
```
<?xml version="1.0" encoding="utf-8"?>
<!--
** Copyright 2008, The Android Open Source Project
** etc...
-->
<CheckedTextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    style="?android:attr/spinnerDropDownItemStyle"
    android:singleLine="true"
    android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:ellipsize="marquee" />
```

Dodatek B: Pola tekstowe i klawiatury

Sposób wprowadzania danych różni się w zależności od fizycznych możliwości danego urządzenia.

Wbudowana klawiatura fizyczna, dostępna po otwarciu pokrywy

Urządzenie na fizyczną klawiaturę dostępną stale.



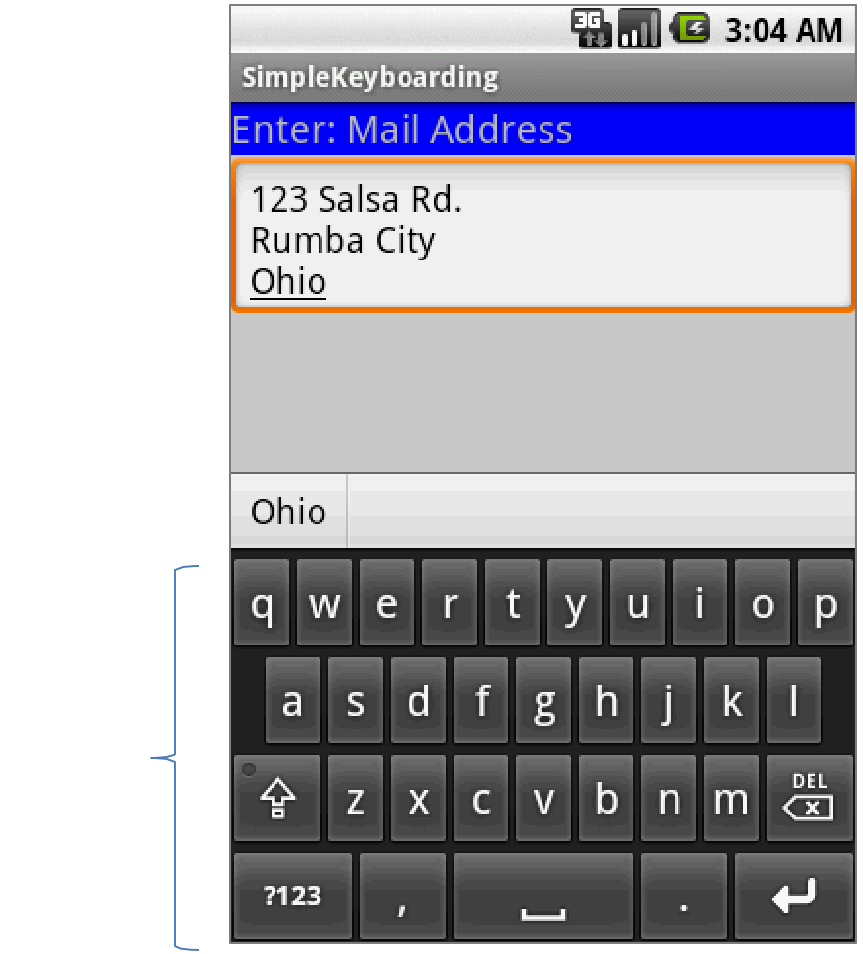
Za dane wejściowe odpowiada wirtualna klawiatura lub rozpoznawanie mowy.

Dodatek B: Pola tekstowe i klawiatury

Gdy użytkownik kliknie w pole **EditText**, Input Media Framework (**IMF**) zapewnia dostęp do:

1. Fizycznej klawiatury (jeśli dostępna) lub
2. Wirtualnej klawiatury (znanej jako IME) najczęściej spotykanej.

Zamknięcie klawiatury wymaga wciśnięcia klawisza **Wstecz**.



Dodatek B: Pola tekstowe i klawiatury

Informacja jakich danych należy się spodziewać

Komponenty TextView mogą używać elementów **XML** lub kodu **Java** by określić jakiego typu dane pole tekstowe powinno akceptować. Przykładowo:

XML

```
android:inputType="phone"
```

Java

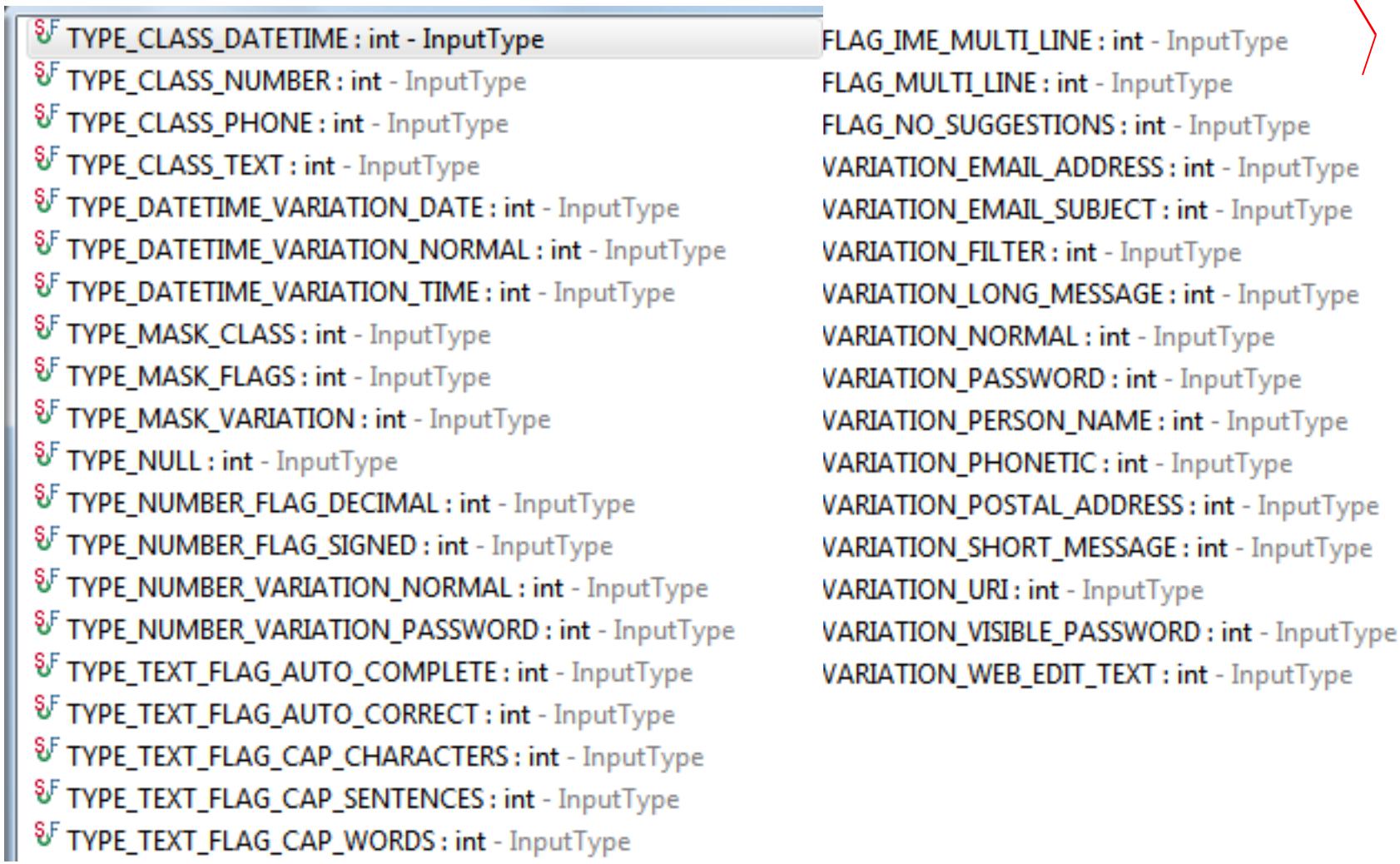
```
editTextBox.setInputType(
        android.text.InputType.TYPE_CLASS_PHONE);
```

Pole inputType ma wpływ na klawiatury wirtualne (oprogramowanie może wówczas wyświetlić najlepszą wersję klawiatury do wprowadzenia danego ciągu znaków)

Dodatek B: Pola tekstowe i klawiatury

Dostępne warianty inputType

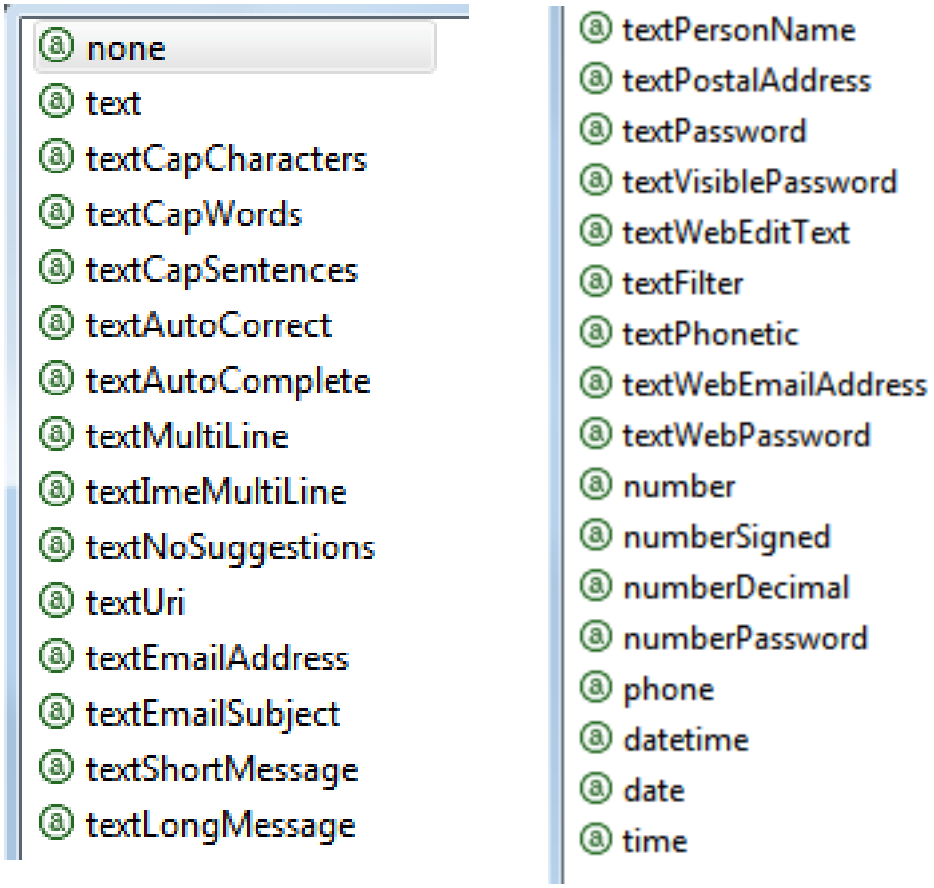
```
editTextBox.setInputType( android.text.InputType.XXX );
```



Dodatek B: Pola tekstowe i klawiatury

Określenie inputType z poziomu XML:

```
<EditText
        …
        android:inputType="numberSigned|numberDecimal"
        … />
```



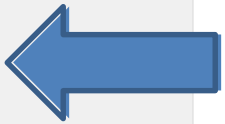
Źródło:
http://developerandroid.com/reference/android/R.styleable.html#TextView_inputType

Dodatek B: Pola tekstowe i klawiatury

Przykład 10: Wykorzystanie wielu wartości naraz

android:inputType="text|textCapWords"

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

<TextView android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:background="#ff0000ff"
     android:text="inputType: text|textCapWords"
     android:textStyle="bold"
     android:textSize="22sp" />

<EditText android:id="@+id/editTextBox"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:padding="5dip"
     android:textSize="18sp"
     android:inputType="text|textCapWords"    />
</LinearLayout>
```



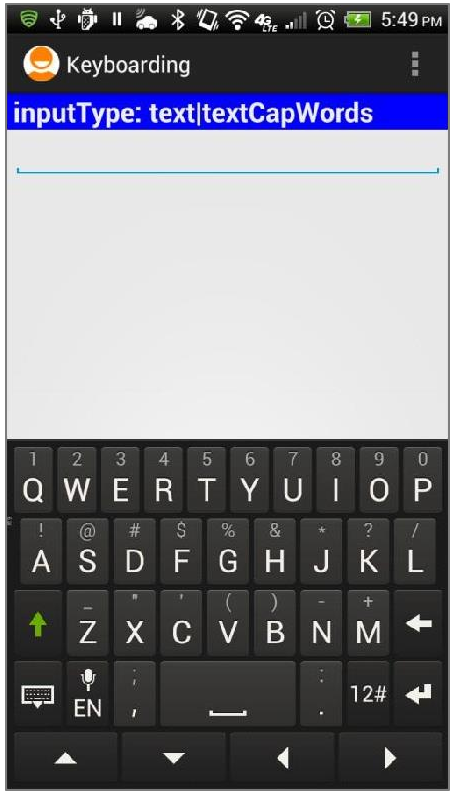
Używaj "potoku" (ozn. |) by separować opcje.

W tym przykładzie zostanie użyta wirtualna klawiatura.

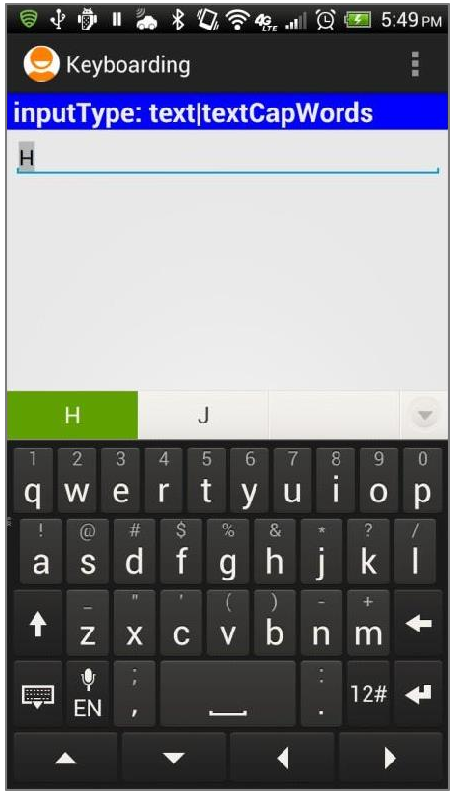
Każde słowo będzie traktowane jak nazwa własna.

Dodatek B: Pola tekstowe i klawiatury

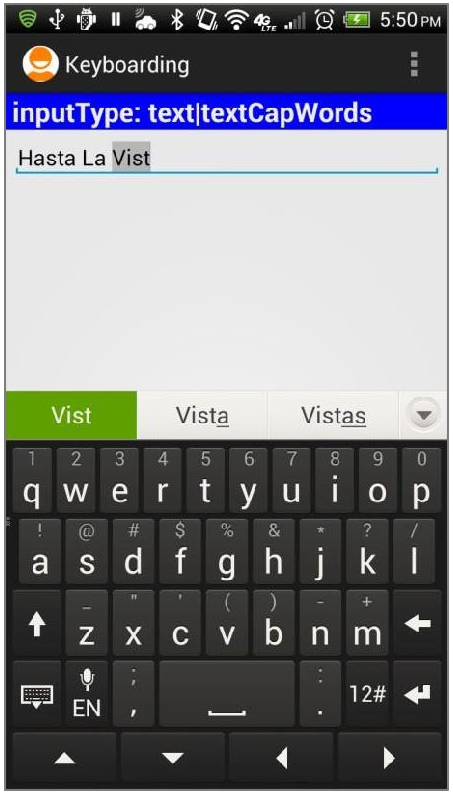
Przykład 10: Użycie android:inputType= *" text|textCapWords"*



Po kliknięciu na EditText pokazywana jest wirtualna klawiatura z wielkimi literami.



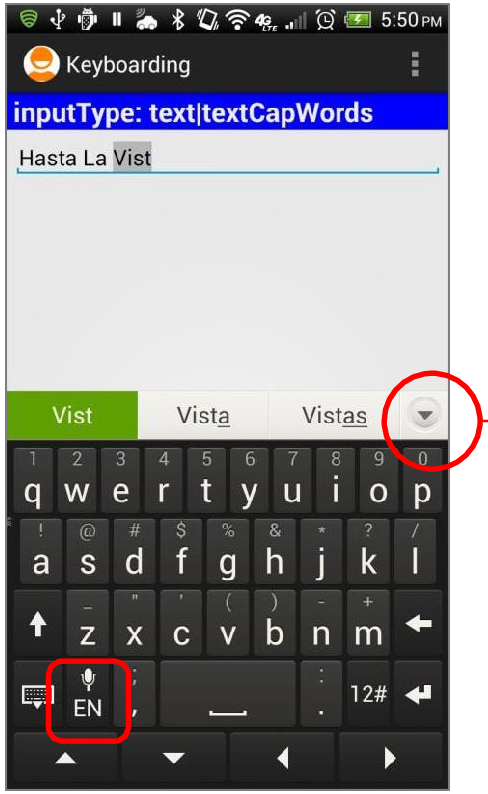
Po wprowadzeniu pierwszej litery klawiatura przechodzi w tryb małych liter.



Po wprowadzeniu spacji cykl się powtarza.

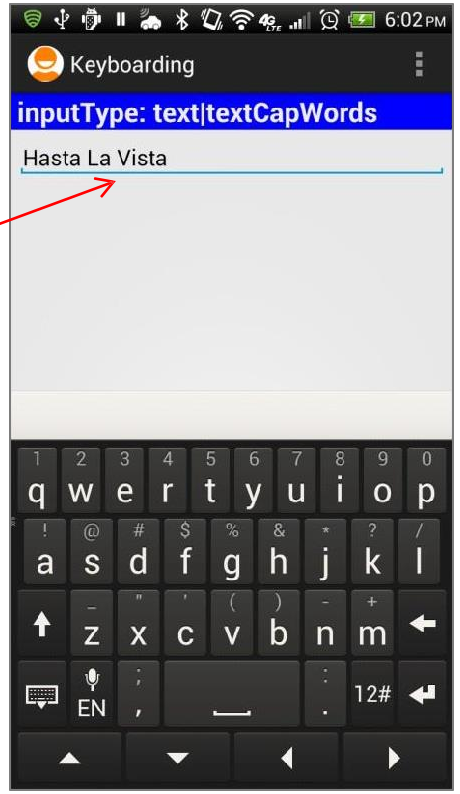
Dodatek B: Pola tekstowe i klawiatury

Przykład 10: Użycie `android:inputType="text|textCapWords"`



Angielski i Hiszpański są wybrane jako dostępne języki wprowadzania danych.

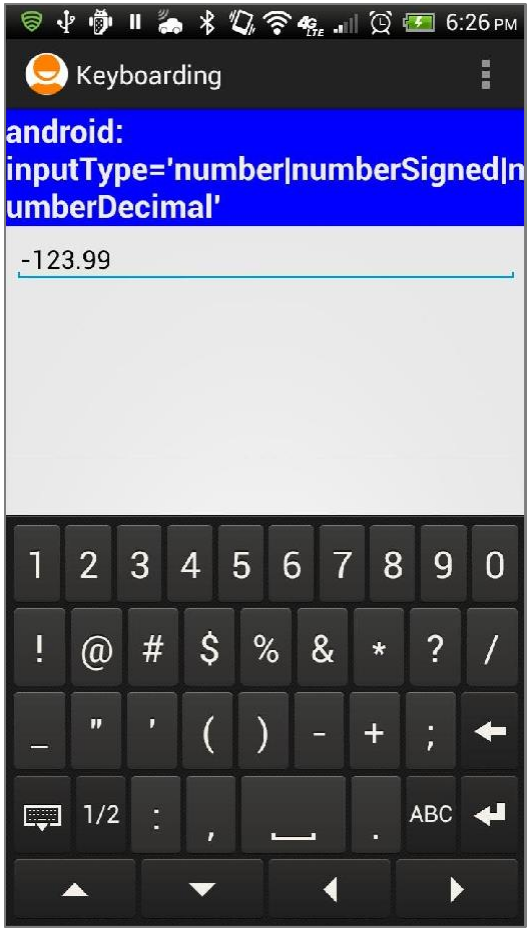
Można przyspieszyć wprowadzenie danych korzystając z listy sugestii (dwujęzyczne)



Wybrane słowo wprowadzane jest do pola tekstowego.

Dodatek B: Pola tekstowe i klawiatury

Przykład 11: Wykorzystanie
`android:inputType="number|numberSigned|numberDecimal"`



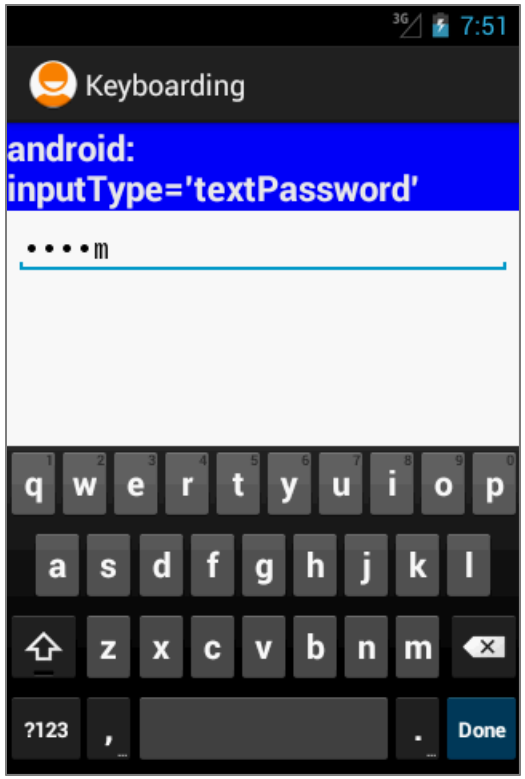
1. Klawiatura wyświetla liczby.
2. *Nie-numeryczne* klawisze (jak !@#$%&*?/_) mogą być widoczne ale są nieaktywne.
3. Tylko poprane wyrażenia numeryczne mogą być wprowadzone.
4. Typ **number|numberSigned** akceptuje liczby całkowite.
5. Typ **numberDecimal** akceptuje liczby rzeczywiste.

Przy założeniu, że pole tekstowe ma nazwę **editTextBox** ustawienie maski wprowadzania podczas działania programu odbywa się przez:

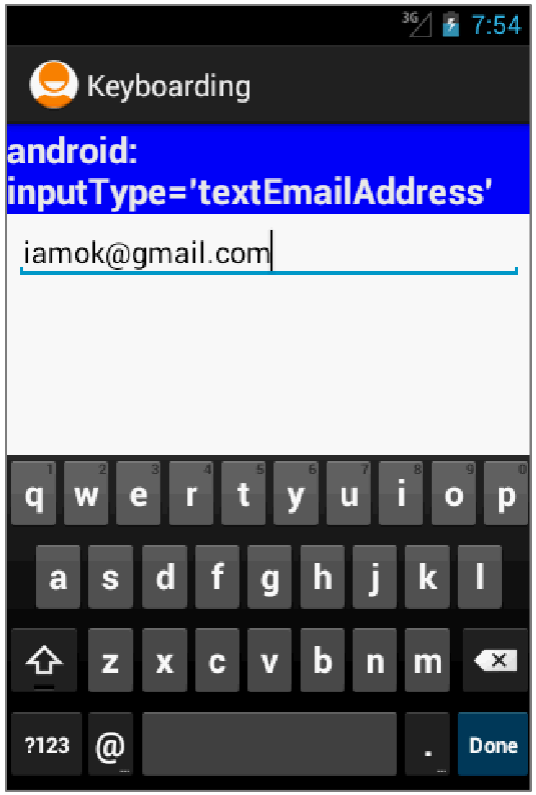
```
editTextBox.setInputType(
  android.text.InputType.TYPE_CLASS_NUMBER |
  android.text.InputType.TYPE_NUMBER_FLAG_SIGNED);
```

Dodatek B: Pola tekstowe i klawiatury

Przykład 12: Użycie android:inputType=*"textPassword"*



Przykład 13: Użycie android:inputType=*"textEmailAddress"*



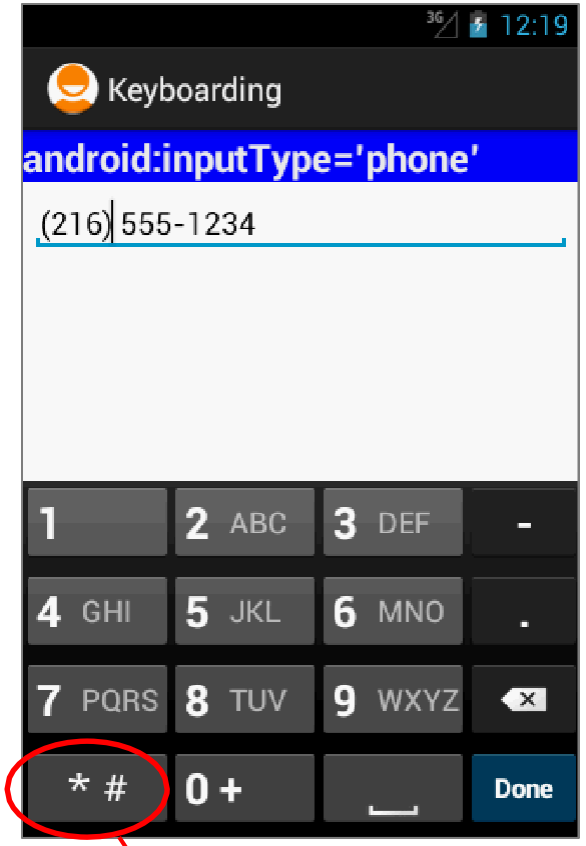
Wirtualna klawiatura pokazuje klawisze używane w komunikacji mailowej (jak litery, **@,** kropka**).**

Wybierz [**?123**] by wyświetlić pozostałe symbole.

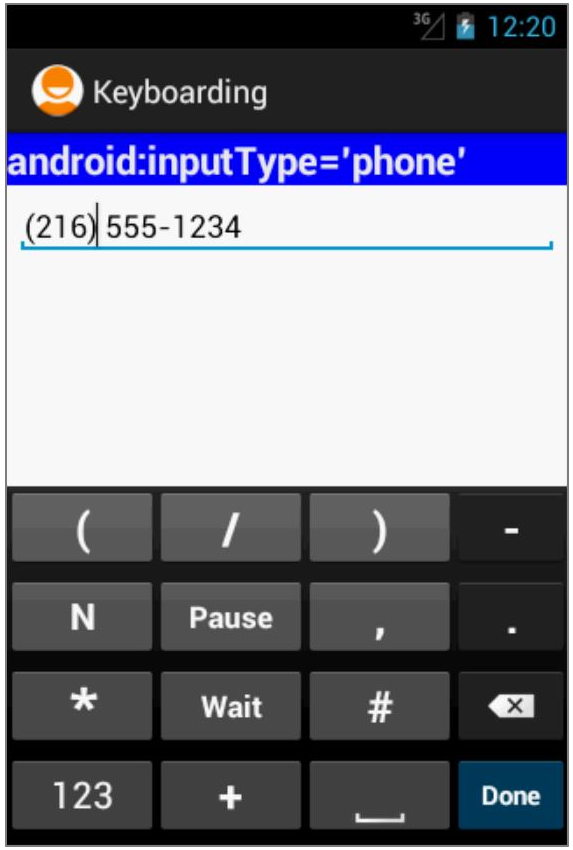
- Klawiatura wyświetla wszystkie klawisze.
- Aktualny znak prezentowany jest na chwilę dla referencji.
- Aktualny znak jest ukrywany za znakiem gwiazdki.

Dodatek B: Pola tekstowe i klawiatury

Przykład 14: Użycie `android:inputType="phone"`



Dodatkowe symbole



Wirtualna klawiatura prezentuje układ typowy dla telefonu plus dodatkowe symbole jak: **( ) . / # - +**

Dodatek B: Pola tekstowe i klawiatury

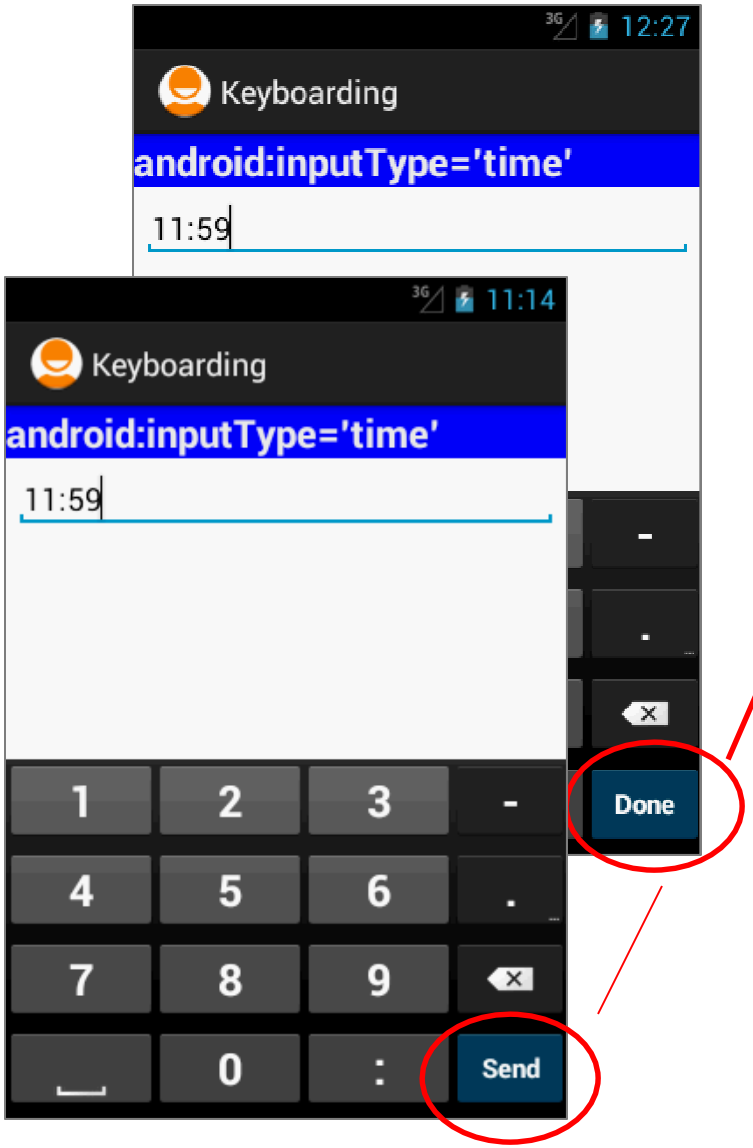
Przykład 15: Użycie `android:inputType="time"`

Wirtualna klawiatura prezentuje układ numeryczny.

Tylko numery oraz "**:**" mogą zostać użyte.

Dodatek B: Pola tekstowe i klawiatury

Przykład 16: Użycie `android:inputType="time"`



Gdy kliknięto, przycisk **dodatkowy DONE** klawiatura zostaje zamknięta.

Można zmienić opis przycisku oraz ustawić nasłuchiwacz by zdefiniować reakcję na zdarzenie kliknięcia przycisku. Należy wówczas dodać do szablonu XML następującą deklarację:

```
android:imeAction="actionSend"
```

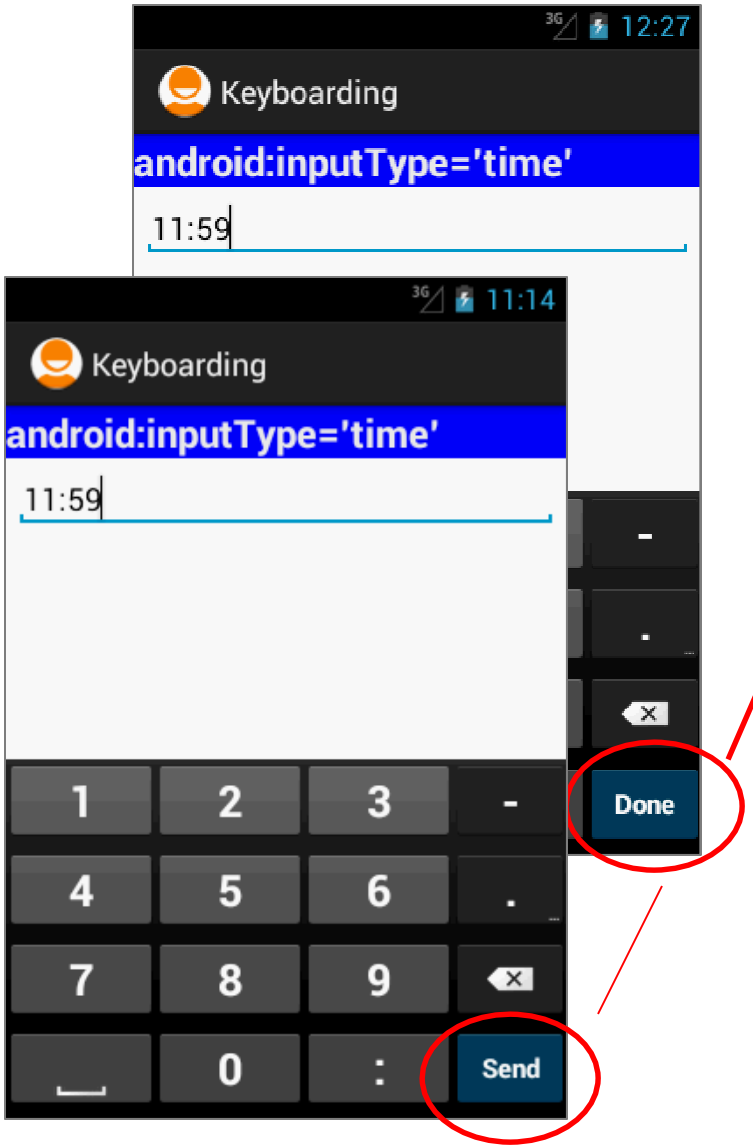
Należy także podać implementację metody:

```
editTextBox
   .setOnEditorActionListener()
```

by zdefiniować reakcję na to zdarzenie.

Dodatek B: Pola tekstowe i klawiatury

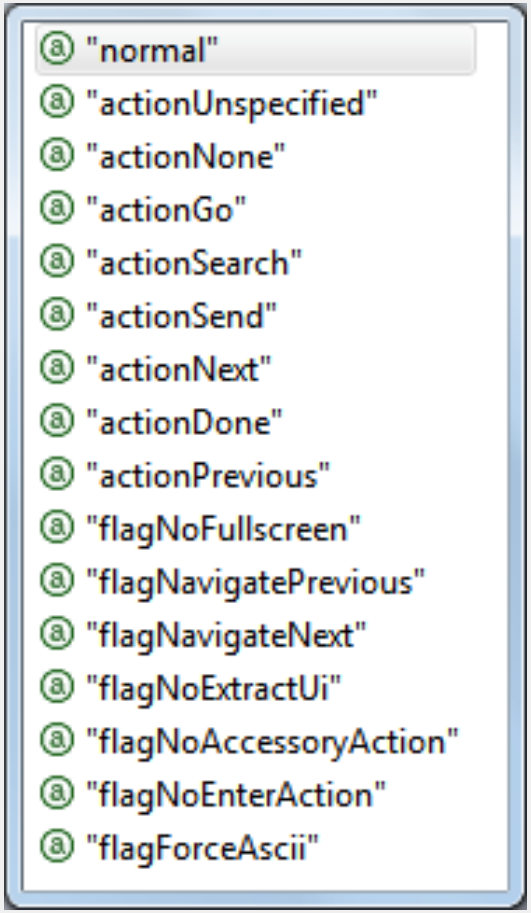
Przykład 16: Użycie android:inputType=*"time"*



Inne opcje dla

```
android:imeAction=“. . .”
```

to



Dodatek B: Pola tekstowe i klawiatury

Przykład 17: Użycie `android:inputType="datetime"`

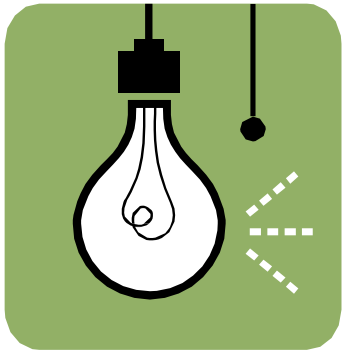
Wirtualna klawiatura prezentuje układ numeryczny.

Jedynie liczby i symbole występujące w dacie/czasie są akceptowalne:

Przykłady poprawnych wyrażeń:

```
12/21/2012  12:12
12/31/2011
12:30
```

Dodatek B: Pola tekstowe i klawiatury



Wyłączenie wirtualnej klawiatury dla pola tekstowego

- By **wyłączyć** pokazywanie wirtualnej klawiatury dla pola tekstowego należy ustawić jego input type na null:

```
editTextBox.setInputType( InputType.TYPE_NULL );
```

- By tymczasowo **ukryć wirtualną klawiaturę można wywołać metodę**:

```
public void hideVirtualKeyboard() {
 Context context = getActivity().getApplicationContext();
 ((InputMethodManager) context
            .getSystemService(Activity.INPUT_METHOD_SERVICE))
            .toggleSoftInput(InputMethodManager.SHOW_IMPLICIT, 0);
}
```

- By **wyświetlić** wirtualną klawiaturę należy użyć:

```
public void showVirtualKeyboard() {
 Context context = getActivity().getApplicationContext();
 ((InputMethodManager) context
            .getSystemService(Activity.INPUT_METHOD_SERVICE))
            .toggleSoftInput(InputMethodManager.SHOW_IMPLICIT, 0);
}
```