Programowanie urządzeń mobilnych

dr inż. Andrzej Grosser
na podstawie wykładu
dr inż. Juliusza Mikody

Kontrolka – lista


```
<!-- row_txt.xml  -->

<?xml version="1.0" encoding="utf-8"?>
<TextView
  xmlns:android="http://schemas.android.com
                              /apk/res/android"
  android:layout_height="wrap_content"
  android:layout_width="fill_parent"
  android:textSize="30dp">
</TextView>
```

```
public class ListaShow extends ListActivity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ArrayAdapter<String> aa
       = new ArrayAdapter<String>(this,
           R.layout.row_txt,
            new String[]{"Słońce", "Merkury", "Wenus",
              "Ziemia", "Mars", "Jowisz", "Saturn",
              "Uran", "Neptun", "Pluton"}
         );
    setListAdapter(aa);
  }
}
```

Kontrolka – lista


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/
            /pk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/lista"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

```
public class ListaShow extends ListActivity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayAdapter<String> aa
            = new ArrayAdapter<String>(this,
                  R.layout.row_txt,
                   new String[]{"Słońce", ..., "Pluton"}
                );
        ListView l = (ListView)findViewById(R.id.lista);
        l.setAdapter(aa);
    }
  }
}
```

ListView – podział wierszy

- android:divider – element drawable służący do rozdzielenia wierszy listy, setDivider(Drawable divider), getDivider(),
- android:dividerHeight - wysokość elementu rozdzielającego, setDividerHeight(int height), getDividerHeight(),
- android:footerDividersEnabled – widoczny dolny element rozdzielajacy,
- android:headerDividersEnabled – widoczny górny element rozdzielający.

ListView – nagłówek i stopka

- public void addHeaderView (View v, Object data, boolean isSelectable) – ustawienie nagłówka listy,
- removeHeaderView(View v) – usunięcie nagłówka z listy
- public void addFooterView (View v, Object data, boolean isSelectable) – ustawienie stopki dla listy
- removeFooterView(View v) – usunięcie stopki z listy

Interface – Cursor

- Interfejs pozwala na losowy dostęp do wyniku zapytania do „bazy danych”.
- Implementacja interface pozwala na odczyt:
  - odczyt pozycji: getCount(), getPosition()
  - poruszanie się po odczytanych rekordach : move(int offset), moveToFirst(), moveToLast(), moveToNext(), moveToPosition(int position), moveToPrevious(),
  - określenia pozycji: isAfterLast(), isBeforeFirst(), isClosed(), isFirst(), isLast()

Interface – Cursor

- Implementacja interface pozwala na odczyt:
  - informacji o kolumnach krotki (rekordu): getDouble(int columnIndex), getFloat(int columnIndex), getInt(int columnIndex), getLong(int columnIndex), getShort(int columnIndex), getString(int columnIndex)
  - isNull(int column)
- Podklasy: AbstractCursor, AbstractWindowedCursor, CrossProcessCursor, CursorWrapper, MatrixCursor, MergeCursor, MockCursor, SQLiteCursor

Interface Cursor - podklasy

- MatrixCursor – klasa pozwalająca zapisać dane w postaci tabeli. addRow(Object[] columnValues).
- MergeCursor – Pozwala na powiązanie kilku kursorów w jeden ciągły. Konstrukcja: MergeCursor(Cursor[] cursors).
- SQLiteCursor – kursor przeznaczony do operacji na bazie danych.

Adapter kursorów

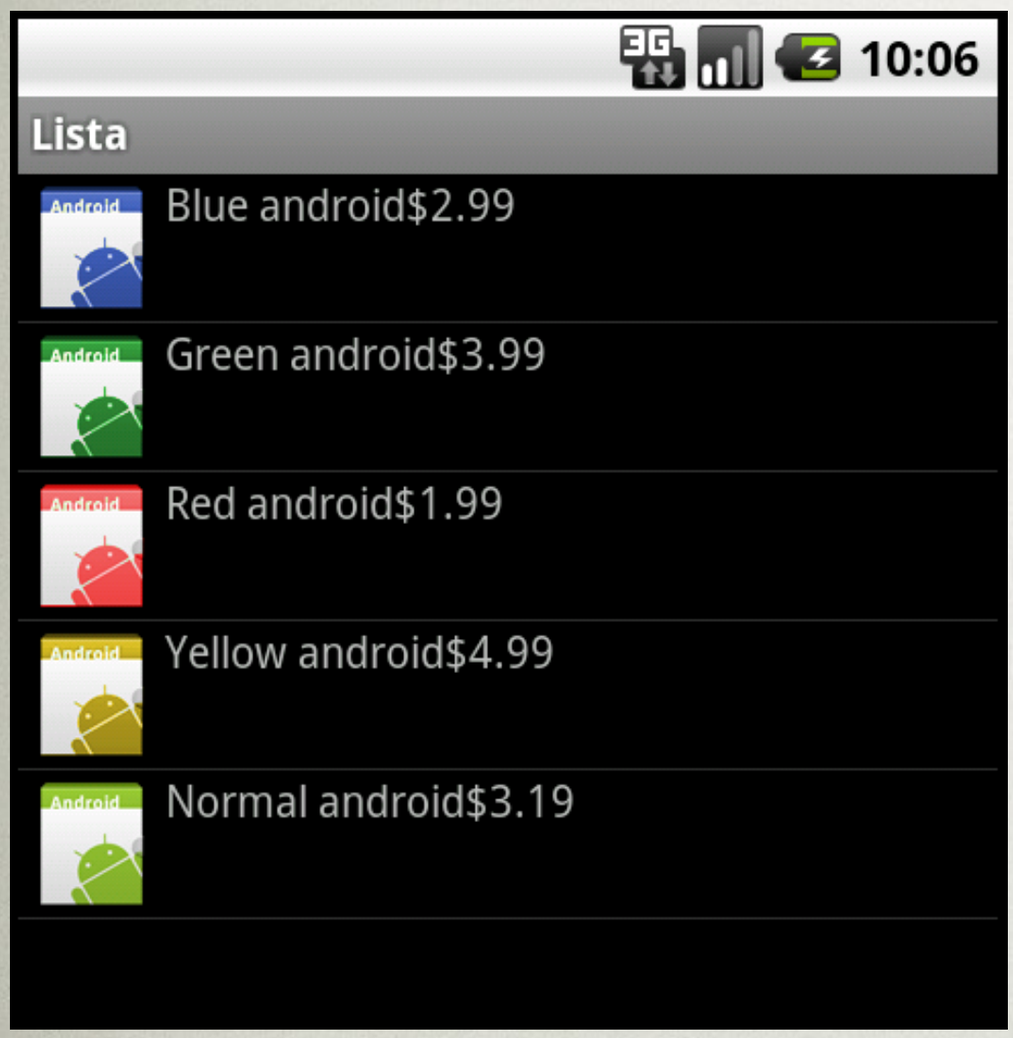
- BaseAdapter – klasa bazowa abstrakcyjna adapterów (dla komponentów ListView, GridView ...),
  - **ArrayAdapter** – adapter prostych list danych,
  - CursorAdapter – klasa bazowa abstrakcyjna operująca na źródle danych typu cursor,
    - ResourceCursorAdapter – j.w.
      - **SimpleCursorAdapter** – adapter źródła danych typu cursor,
  - **SimpleAdapter** – adapter prostych list - map danych : List<? extends Map<String, ?>> data

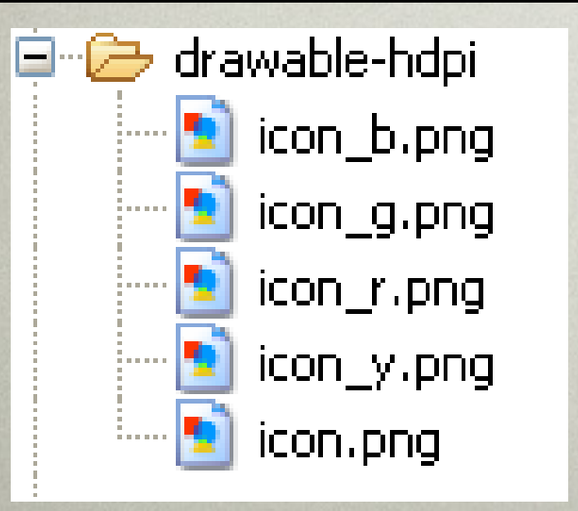
Kontrola - siatka


```xml
<!-- grid.xml -->
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android=
        "http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:padding="10px"
    android:verticalSpacing="10px"
    android:horizontalSpacing="10px"
    android:numColumns="2"
    android:stretchMode="columnWidth"
    android:gravity="center"
    android:id="@+id/dataGrid">
</GridView>
```

```java
public class Grid extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.grid);
        ArrayAdapter<String> aa = new ArrayAdapter<String>(this,
             R.layout.row_txt,
             new String[]{"Słońce", "Merkury", "Wenus", "Ziemia",
                    "Mars", "Jowisz", "Saturn", "Uran", "Neptun", "Pluton"}
            );
        GridView gv = (GridView) this.findViewById(R.id.dataGrid);
        gv.setAdapter(aa);
    }
}
```

Kontrolka – lista rozszerzenia



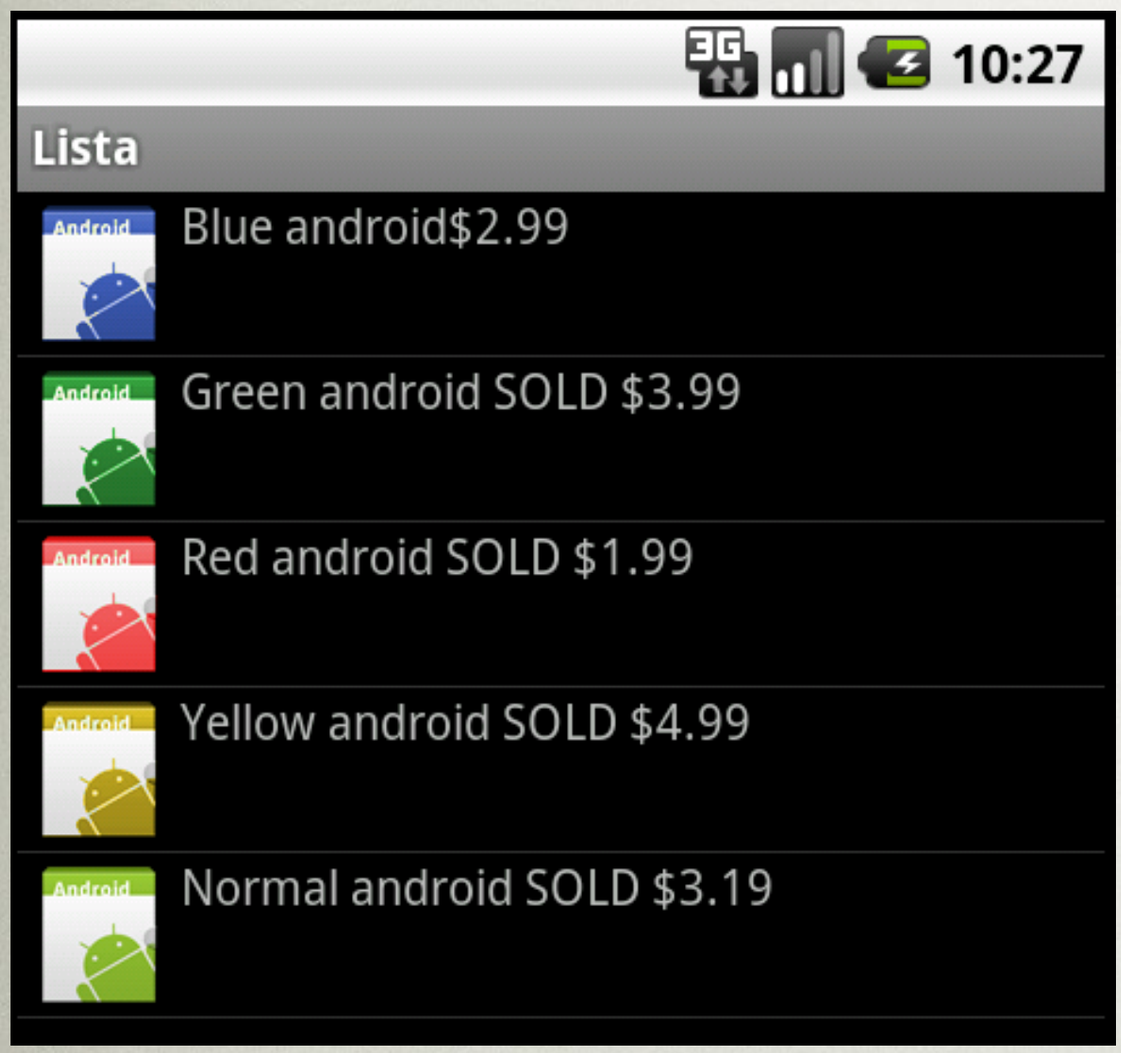



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  xmlns:android="http://schemas.android.com
                              /apk/res/android">
    <ImageView
      android:id="@+id/icon"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

    <TextView
      android:id="@+id/item"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

    <TextView
      android:id="@+id/price"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

</LinearLayout>
```

Kontrolka – lista rozszerzenia

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

    String[] menuCols = new String[] {"_id" , "icon", "item", "price" };
    int[] to = new int[] { -1, R.id.icon, R.id.item, R.id.price };

    MatrixCursor menuCursor = new MatrixCursor(menuCols);
    startManagingCursor(menuCursor);

    menuCursor.addRow(new Object[] {1, R.drawable.icon_b,
                "Blue android", "$2.99" });
    menuCursor.addRow(new Object[] {2, R.drawable.icon_g,
                "Green android", "$3.99" });
    menuCursor.addRow(new Object[] {3, R.drawable.icon_r,
                "Red android", "$1.99" });
    menuCursor.addRow(new Object[] {4, R.drawable.icon_y,
                "Yellow android", "$4.99" });
    menuCursor.addRow(new Object[] {5, R.drawable.icon,
                "Normal android", "$3.19" });

    SimpleCursorAdapter menuItems = new SimpleCursorAdapter(
                this, R.layout.menu_row, menuCursor, menuCols, to);
    setListAdapter(menuItems);
}
```

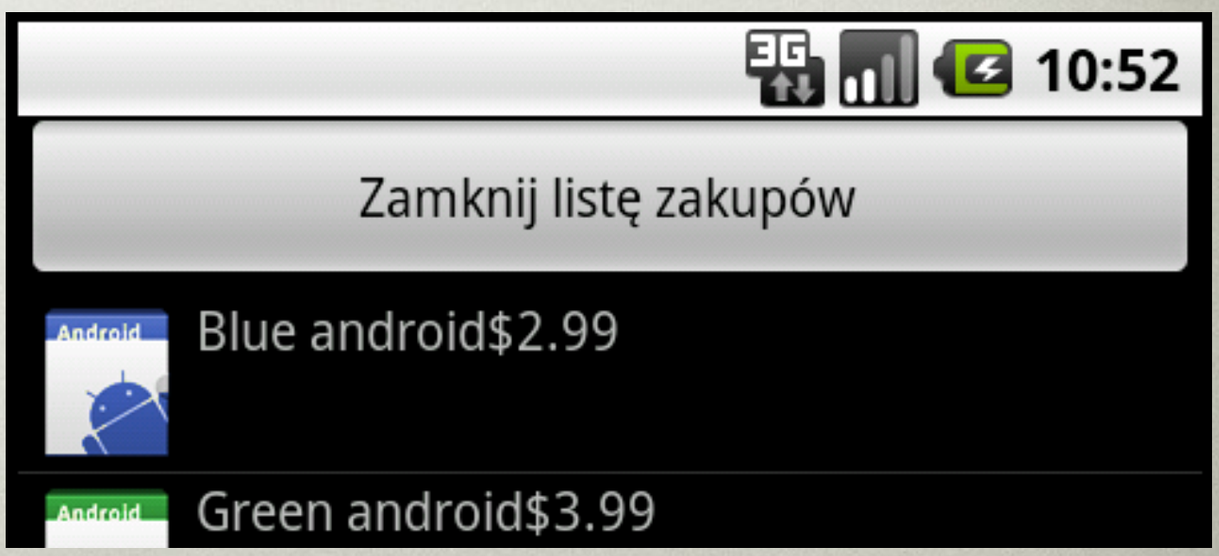
Kontrolka – lista rozszerzenia




```
protected void onListItemClick(
      ListView parent, View view,
      int position, long id) {

  super.onListItemClick(
      parent, view, position, id);

  if (view.isEnabled()) {
    TextView text =
      (TextView)
        view.findViewById(R.id.price);
    text.setText(" SOLD "
      + text.getText());
    view.setEnabled(false);
  }
}
```

Kontrolka – lista niestandardowa

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_height="fill_parent"
  android:layout_width="fill_parent" >
  <Button
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:onClick="zamknijOkno"
    android:text="Zamknij listę zakupów" />
  <ListView
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:id="@+id/lista" />
</LinearLayout>
```



Kontrolka – lista niestandardowa

```
public class ListaShow extends Activity
            implements OnItemClickListener {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.list);
        // ... tworzenie kursora ... MatrixCursor menuCursor = new ...
        ListView lv = (ListView) findViewById(R.id.lista);
        lv.setAdapter(menuItems);
        lv.setOnItemClickListener(this);
    }

    public void zamknijOkno(View view) {
        finish();
    }

    public void onItemClick(AdapterView<?> parent, View view,
            int position, long id) {
        if (view.isEnabled()) {
            TextView text = (TextView) view.findViewById(R.id.price);
            text.setText(" SOLD " + text.getText());
            view.setEnabled(false);
        }
    }
}
```

BaseAdapter – własny adapter

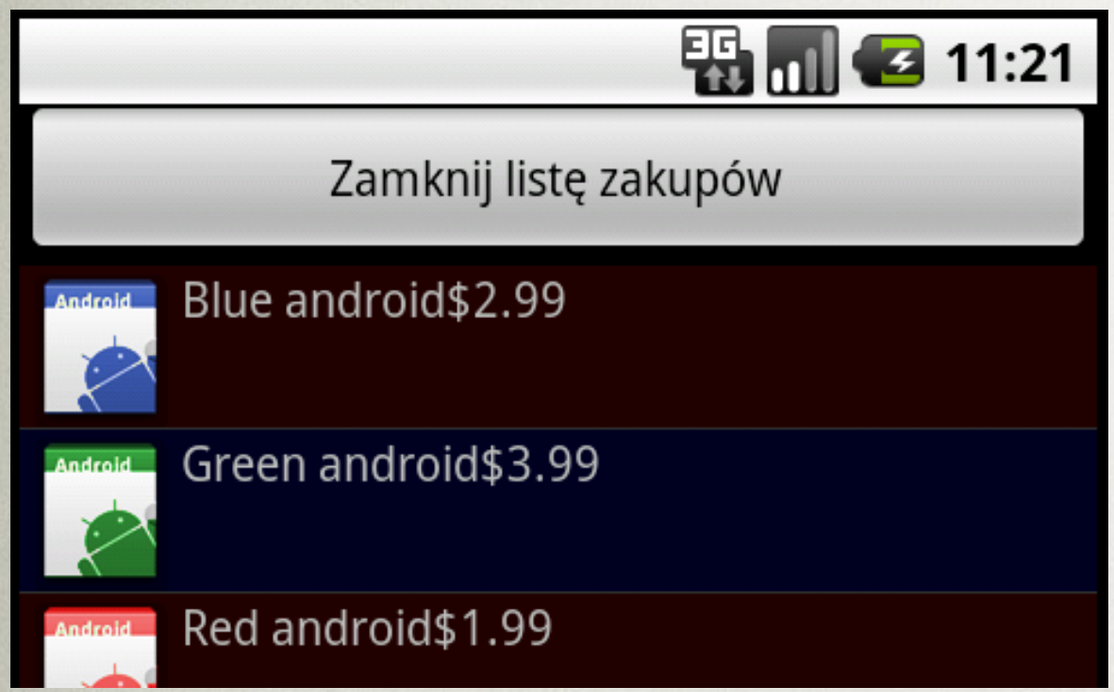
- Własny adapter można uzyskać po zdefiniowaniu metod abstrakcyjnych klasy BaseAdapter. Klasa BaseAdapter implementuje wybrane metody interfejsów:
  - ListAdapter – klasa przeznaczona dla komponentów tworzonych na podstawie zestawu danych (ListView, GridView)
  - SpinnerAdapter – zestaw metod przeznaczonych dla komponentu Spinner

BaseAdapter – własny adapter

Metody które trzeba zaimplementować:

- `public int` getCount() - liczba elementów wyświetlanych na liście,
- `public` Object getItem(`int` position) – dane powiązane z danym wierszem listy,
- `public long` getItemId(`int` position) – id danego elementu listy
- `public` View getView(`int` position, View convertView, ViewGroup parent) – tworzenie widoku elementów na liście:
  - position – pozycja na liście
  - convertView – element zachowany
  - parent – rodzic - lista

Kontrolka – lista niestandardowa




```
<!-- strings.xml -->
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- ..........  -->
    <drawable name="darkred">
            #200000</drawable>
    <drawable name="darkblue">
            #000020</drawable>
</resources>
```

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    setContentView(R.layout.list);

    ListView lv = (ListView) findViewById(R.id.lista);
    lv.setAdapter(new ListMenuAdapter(this));
    lv.setOnItemClickListener(this);
}
```

Kontrolka – lista niestandardowa

```
public class ListMenuAdapter extends BaseAdapter {

  protected LayoutInflater mInflater;
  protected int[] images;
  protected String[] texts;
  protected String[] prices;

  public ListMenuAdapter(Context context) {
    mInflater = LayoutInflater.from(context);
    images = new int[]{R.drawable.icon_b, R.drawable.icon_g,
            R.drawable.icon_r, R.drawable.icon_y, R.drawable.icon};
    texts = new String[]{"Blue android", "Green android",
            "Red android", "Yellow android", "Normal android"};
    prices = new String[]{ "$2.99", "$3.99", "$1.99", "$4.99", "$3.19"};
  }

  public int getCount() {
    return 5;
  }
  public Object getItem(int arg0) {
    return null;
  }
  public long getItemId(int arg0) {
    return arg0;
  }
```

Kontrolka – lista niestandardowa

```
public class ListMenuAdapter extends BaseAdapter {
  protected LayoutInflater mInflater;
  protected int[] images;
  protected String[] texts;
  protected String[] prices;

  // ..................

  public View getView(int pos, View view, ViewGroup parent) {

    if (view == null) view = mInflater.inflate(R.layout.menu_row, null);
    if (pos < getCount()) {
      TextView text = (TextView) view.findViewById(R.id.item);
      TextView price = (TextView) view.findViewById(R.id.price);
      ImageView img = (ImageView) view.findViewById(R.id.icon);
      text.setText(texts[pos]);
      price.setText(prices[pos]);
      img.setImageResource(images[pos]);
    }

    if (pos % 2 == 0)   view.setBackgroundResource(R.drawable.darkred);
    else                view.setBackgroundResource(R.drawable.darkblue);

    return view;
  }
}
```

Fragmenty

- Fragment przedstawia część interfejsu użytkownika w aktywności.
- Można połączyć kilka fragmentów w pojedynczej aktywności.
- Fragment posiada własny cykl życia.
- Wprowadzone w Android 3.0 (z myślą o tabletach).

Tworzenie fragmentów

- Klasa fragmentu musi dziedziczyć bezpośrednio lub pośrednio po klasie biblioteczne *Fragment.*
- Zawiera metody zwrotne:
  - onCreate() - wywoływane, gdy system tworzy fragment
  - onCreateView() - wywoływane, gdy istniej potrzeba odrysowania interfejsu użytkownika
  - onPause() - wywoływane, jako pierwsza oznaka, tego że użytkownik zakończył pracę z fragmentem.

Klasy fragmentów

- DialogFragment – wyświetla pływający dialog.
- ListFragment – wyświetla listę elementów.
- PreferenceFragment – wyświetla hierarchię obiektów Preference w postaci listy

Dodawanie fragmentów

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="horizontal"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
    <fragment
       android:name="com.example.news.ArticleListFragment"
       android:id="@+id/list"
       android:layout_weight="1"
       android:layout_width="0dp"
       android:layout_height="match_parent" />
    <fragment
       android:name="com.example.news.ArticleReaderFragment"
       android:id="@+id/viewer"
       android:layout_weight="2"
       android:layout_width="0dp"
       android:layout_height="match_parent" />
</LinearLayout>
```

Dodawanie programowe

- Można dodawać fragmenty, w każdej chwili, w której jest uruchomiona aktywność, wystarczy jedynie określić ViewGroup, w którym ma być umieszczony fragment.
- Operacje na fragmentach (takie jak ich dodawanie, usuwanie, zastępowanie) są wykonywane za pośrednictwem FragmentTransaction.
- Obiekt transakcji uzyskuje się za pośrednictwem FragmentManagera.

Dodawanie programowe

```
FragmentManager fragmentManager =
getFragmentManager();

FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
```

- Dodawanie fragmentu można zrealizować za pomocą metody add()

```
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container,
                                        fragment);
fragmentTransaction.commit();
```

- Metoda commit() zatwierdza zmiany.

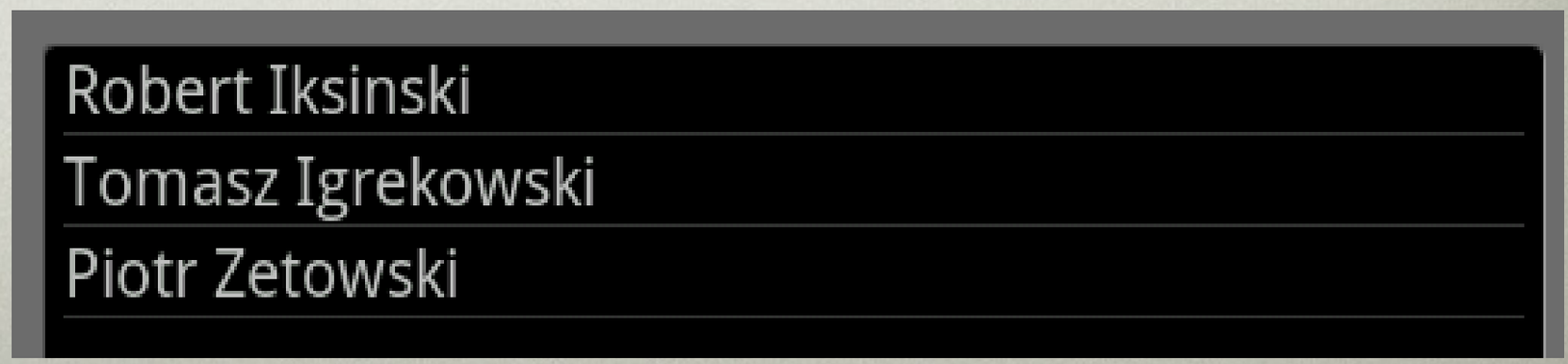
Dostawcy treści

- Zarządzanie przepływem informacji w systemie android.
- Pełna lista dostawców dostępna jest pod http://developer.android.com/reference/android/provider/package-summary.html
- Contacts: .People, Phones, Photos, Groups
- MediaStore: Audio (Albums, Artists, Media, Playlists), Images (Media, Thumbnails), Video (Media, Thumbnails)
- Settings

Dostawcy treści – kontakty 1

- Odczytanie listy kontaktów

```
// Zezwolenie odczytu listy kontaktów w AndroidManifest.xml
<uses-permission android:name="android.permission.READ_CONTACTS"/>

public class Dostawcy extends ListActivity {
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    Cursor cursor = getContentResolver()
      .query(ContactsContract.Contacts.CONTENT_URI,
        null, null, null, null);
    String[] menuCols = new String[]{
      ContactsContract.Contacts.DISPLAY_NAME };
    int[] to = new int[] { R.id.text };
    SimpleCursorAdapter menuItems = new SimpleCursorAdapter(this,
        R.layout.text, cursor, menuCols, to);
    setListAdapter(menuItems);
  }
}
```



Dostawcy treści – kontakty 2

- ContentResolver getContentResolver(); - Klasa ta zapewnia dostęp do dostawców treści.
- Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder):
  - uri – adres dostawcy treści,
  - projection – lista kolumn zawracanych,
  - selection – warunek WHERE (bez słowa kluczowego WHERE),
  - selectionArgs – argumenty podmienia wartości '?',
  - sortOrder – sortowanie wyniku (wartości za klauzulą ORDER BY).

Dostawcy treści – kontakty 3

```
Cursor cursor = getContentResolver()
  .query(ContactsContract.Contacts.CONTENT_URI,
    null, null, null, null);
while (cursor.moveToNext()) {
  String contactId = cursor.getString(
    cursor.getColumnIndex(ContactsContract.Contacts._ID));
  String name = cursor.getString(
    cursor.getColumnIndex(
      ContactsContract.Contacts.DISPLAY_NAME));

  Log.v("ContactsContract", contactId + " " + name);

  int hasPhone = cursor.getInt(
    cursor.getColumnIndex(
      ContactsContract.Contacts.HAS_PHONE_NUMBER));
  // ....
}
cursor.close();
```

Dostawcy treści – kontakty 4

```
if (hasPhone > 0) {
  Cursor phones = getContentResolver().query(
    ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
    ContactsContract.CommonDataKinds.Phone.CONTACT_ID
      + " = " + contactId,
    null, null);
  while (phones.moveToNext()) {
    String phoneNumber = phones.getString(
      phones.getColumnIndex(
        ContactsContract.CommonDataKinds.Phone.DATA));
    Log.v("ContactsContract tel", phoneNumber);
  }
  phones.close();
}

LOG:
VERBOSE/ContactsContract: 1 Robert Iksinski
VERBOSE/ContactsContract: 2 Tomasz Igrekowski
VERBOSE/ContactsContract tel: 23654789
VERBOSE/ContactsContract: 3 Piotr Zetowski
VERBOSE/ContactsContract tel: 123
```

Dostawcy treści – kontakty 5

- Odczytanie kontaktu, konkretnej pozycji z pomocą standardowej listy kontaktów.

```
public class Dostawcy extends Activity {

  protected static int PICK_CONTACT = 1;

  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);

    Intent intentContact = new Intent(Intent.ACTION_PICK,
      ContactsContract.Contacts.CONTENT_URI);
    startActivityForResult(intentContact, PICK_CONTACT);
  }

// ...
```

Dostawcy treści – kontakty 6

```
public void onActivityResult(int requestCode,
  int resultCode, Intent intent)
{
  if (requestCode == PICK_CONTACT) {
    getContactInfo(intent);
  }
}
protected void getContactInfo(Intent intent)
{
    Cursor cursor =  managedQuery(intent.getData(),
      null, null, null, null);
    while (cursor.moveToNext()) {
      String contactId = cursor.getString(
        cursor.getColumnIndex(ContactsContract.Contacts._ID));
      String name = cursor.getString(
        cursor.getColumnIndexOrThrow(
           ContactsContract.Contacts.DISPLAY_NAME));
      Log.v("ContactsContract", contactId + " " + name);
    }
    cursor.close();
  }
}
```

Dostawcy treści – identyfikator URI

- Identyfikator URI jednoznacznie określa dostawcę treści,
- Jego struktura przypomina identyfikatory URI protokołu HTTP,
- ContactsContract.Contacts.CONTENT_URI
  - content://com.android.contacts/contacts
- ContactsContract.CommonDataKinds.Phone.CONTENT_URI
  - content://com.android.contacts/data/phones
- ContactsContract.CommonDataKinds.Email.CONTENT_URI
  - content://com.android.contacts/data/emails

Identyfikator URI - budowa

- content://authoriy-name/path-segment/...
- content – człon określający dostawcę treści,
- authoriy-name – niepowtarzalny identyfikator upoważnienia używany do zlokalizowania dostawcy w rejestrze dostawców,
- path-segment – człon ten określa ścieżkę dostępu do danych (inną dla każdego dostawcy)
- Człon path-segment może być powtarzany wielokrotnie

Identyfikator URI - budowa

- Wywołanie listy kontaktów – identyfikator jednego kontraktu:
  - content://com.android.contacts/contacts/lookup/0n293F33292B314F2929292929/2

Identyfikator URI -budowa

- Własny wybór elementu listy kontaktów:
  - content://com.android.contacts/contacts/1
- Dla dostawców wbudowanych (com.android) nie trzeba używać całego identyfikator, wystarczy wskazać odpowiednie słowo:
- content://contacts/contacts/1