

Dr. Pál László, Sapientia EMTE, Csíkszereda

MOBILESZKÖZÖK ÉS ALKALMAZÁSOK 4.ELŐADÁS



2023-2024 ősz

Felhasználói felület tervezése

GridLayout

2

- Elemek kétdimenziós megjelenítése sorokban és oszlopokban
- Hasonló a *TableLayout*-hoz, viszont jobban testre szabható
- Egy cellába akár több View elem is tehető
- Lehet cellákat egyesíteni úgy vízszintesen, mint függőlegesen

GridLayout - Tulajdonságok

3

- Sor és oszlop pozíció megadása
 - ▣ *android:layout_row, android:layout_column*
- Sor és oszlop egyesítés
 - ▣ *layout_rowSpan, layout_columnSpan*
- Space View beszúrása

```
<Space
    android:layout_row="1"
    android:layout_column="0"
    android:layout_width="50dp"
    android:layout_height="10dp" />
```

GridLayout - Példa

4

Cellák egyesítése



```
android:columnCount="3"
android:rowCount="4"
tools:context="ro.sapi.mygridlayout.MainActi

<Button android:text="Button 1" />
<Button android:text="Button 2" />
<Button android:text="Button 3" />
<Button android:text="Button 4" />
<Button android:text="Column Span 2"
    android:layout_columnSpan="2"
/>
<Button android:text="Button 6" />
<Button android:text="Row Span 2"
    android:layout_rowSpan="2"
/>
<Button android:text="Button 8" />
<Button android:text="Button 9" />
<Button android:text="Button 10"
    android:layout_width="wrap_content" />
<Button android:text="Button 11" />
<Button android:text="Button 12" />
<Button android:text="Button 13" />
```

GridLayout - Példa

5

□ Space használata

The screenshot shows a mobile application window with a blue header bar containing the text 'GridLayoutProducts'. Below the header is a form titled 'New Product Form'. The form contains three input fields: 'Product Code', 'Product Name', and 'Product Price'. The 'Product Name' field is significantly wider than the others, demonstrating the use of space in a GridLayout. At the bottom of the form are two buttons: 'ADD PRODUCT' and 'CANCEL'.

```
android:rowCount="7"
android:columnCount="2" >
<TextView
    android:layout_row="0"
    android:layout_column="0"
    android:text="New Product Form"
    android:typeface="serif"
    android:layout_columnSpan="2"
    android:layout_gravity="center_horizontal"
    android:textSize="20dip" />
<Space
    android:layout_row="1"
    android:layout_column="0"
    android:layout_width="80dp"
    android:layout_height="10dp" />
<TextView
    android:layout_row="2"
    android:layout_column="0"
    android:text="Product Code:" />
```

Más elrendezések

6

- A bemutatott elrendezéseken kívül számos elrendezést definiál az Android:
 - ▣ GridView
 - ▣ ListView
 - ▣ WebView
 - ▣ MapView
 - ▣ Stb
- Nagyobb mennyiségű adat megjelenítésénél, amelyet még görgetni is kell a *GridView* hatékonyabb, mint a *GridLayout*

Tervezési minták (Design patterns)

7

- Egyszerű és elegáns megoldása az objektumorientált tervezés során felmerülő sajátos problémáknak
- Többszöri újratervezés és újrakódolás után alakultak ki
- Rugalmasság és újrafelhasználhatóság jellemzi
- Egymással kommunikáló osztályok és objektumok leírása, amelyek egy általános tervezési feladat megoldására alkalmasak

Tervezési minták osztályozása

8

□ Osztályozási szempontok:

▣ Hatókör szerint (scope)

- Osztály
- Objektum

▣ Cél szerint (purpose)

- Létrehozási (creational)
- Szerkezeti (structural)
- Viselkedési (behavioral)

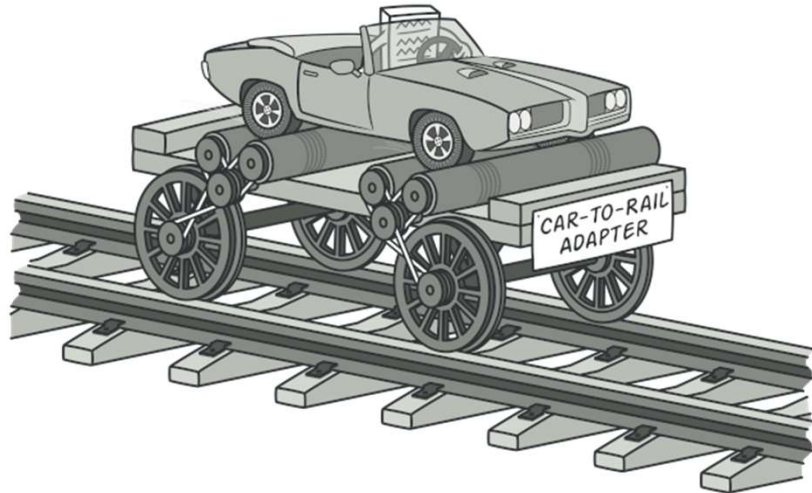
		Purpose		
		Creational	Structural	Behavioral
Scope	Class	Factory Method	Adapter	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Flyweight (195) Observer State Strategy Visitor

		Cél		
		Létrehozási	Szerkezeti	Viselkedési
Hatókör	Osztály	Gyártófüggvény	(Osztály)illesztő	Értelmező Sablonfüggvény
	Objektum	Elvont gyár Építő Prototípus Egyke	(Objektum)illesztő Híd Összetétel Díszítő Homlokzat Pehelysúlyú Helyettes	Felelősséglánc Parancs Bejáró Közvetítő Emlékeztető Megfigyelő Állapot Stratégia Látogató

Adapter tervezési minta

9

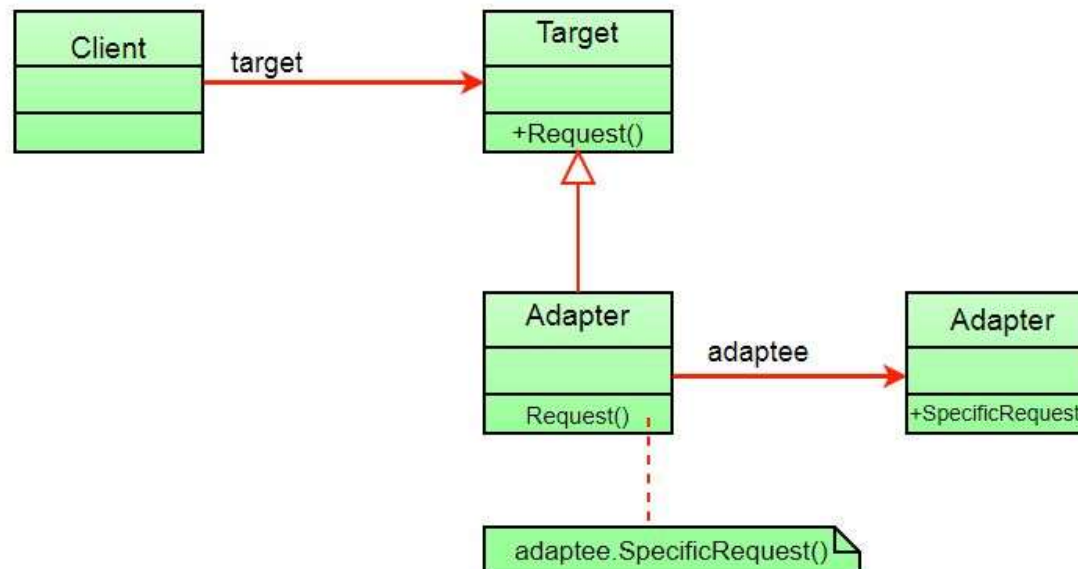
- ❑ Adapter (illesztő): lehetővé teszi olyan osztályok együttműködését, amelyek az inkompatibilis interfészeik miatt normálisan nem tudnának együttműködni
- ❑ A stratégia megvalósítása *örökléssel* vagy *aggregációval* történik



Adapter tervezési minta

10

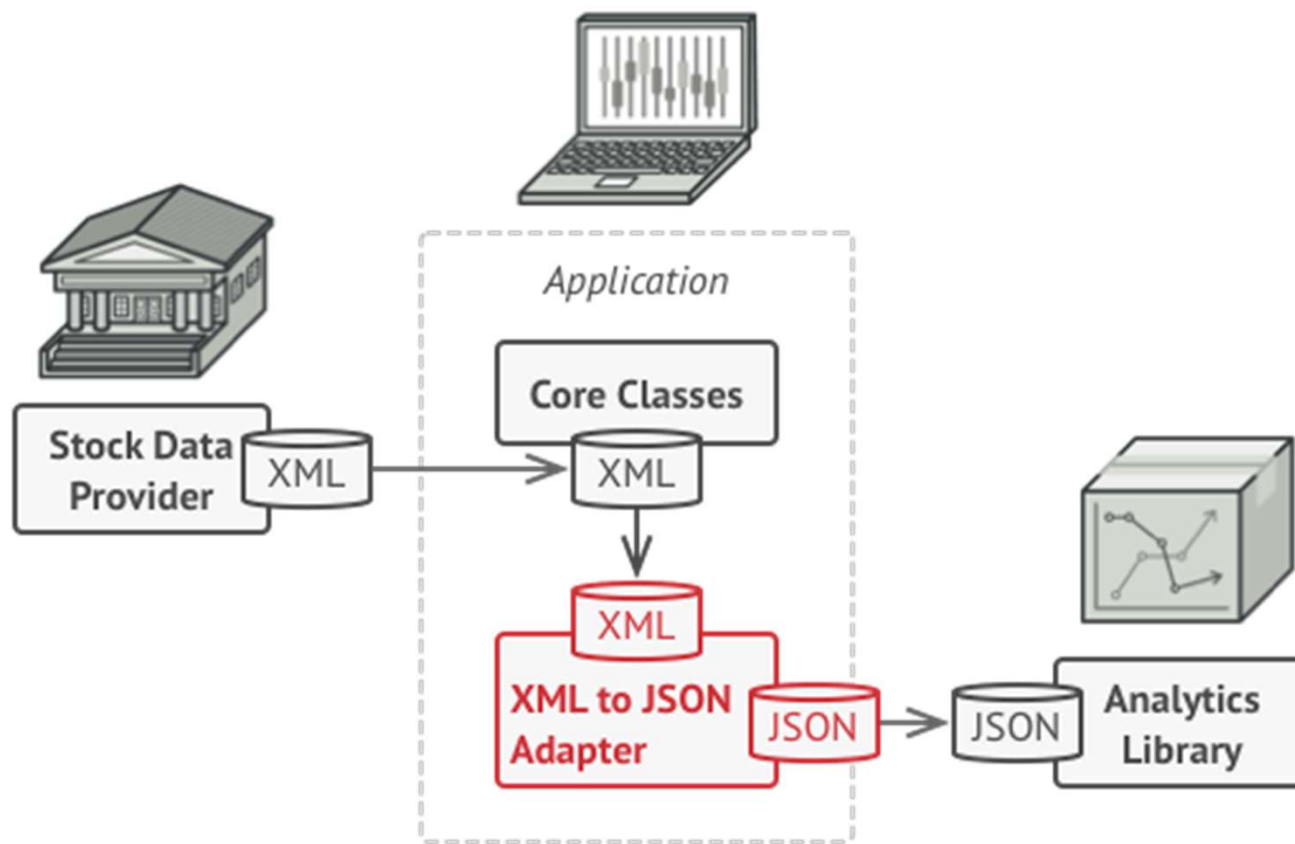
- A kliens csak a cél interfészt látja és nem az adaptert
- Az Adapter megvalósítja a cél interfészt
- Az Adapter továbbít minden kérést



Adapter - Példa

11

- XML konverzió JSON típusra



Adapter tervezési minta

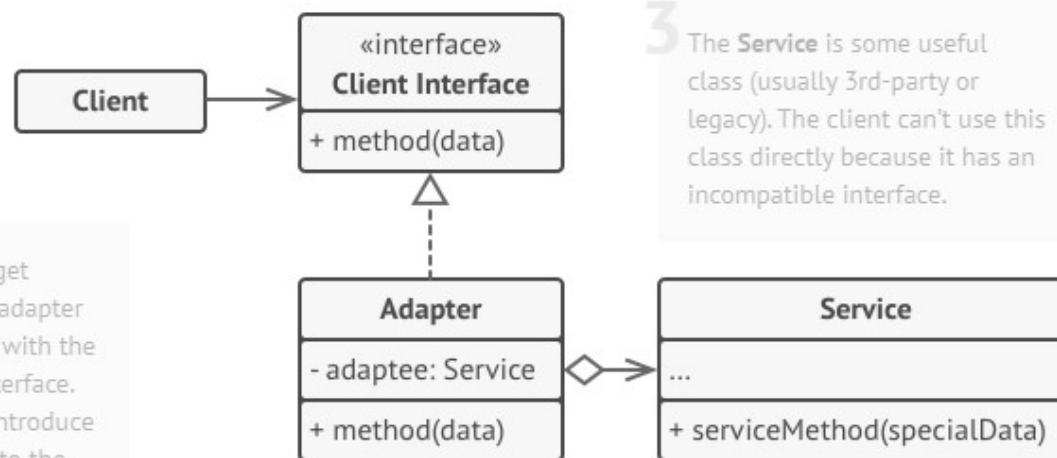
12

1 The **Client** is a class that contains the existing business logic of the program.

2 The **Client Interface** describes a protocol that other classes must follow to be able to collaborate with the client code.

3 The **Service** is some useful class (usually 3rd-party or legacy). The client can't use this class directly because it has an incompatible interface.

5 The client code doesn't get coupled to the concrete adapter class as long as it works with the adapter via the client interface. Thanks to this, you can introduce new types of adapters into the program without breaking the existing client code. This can be useful when the interface of the service class gets changed or replaced: you can just create a new adapter class without changing the client code.



```
specialData = convertToServiceFormat(data)
return adaptee.serviceMethod(specialData)
```

4 The **Adapter** is a class that's able to work with both the client and the service: it implements the client interface, while wrapping the service object. The adapter receives calls from the client via the adapter interface and translates them into calls to the wrapped service object in a format it can understand.

Adapter - Példa

13

```
interface Bird
{
    // birds implement Bird interface that allows
    // them to fly and make sounds adaptee interface
    public void fly();
    public void makeSound();
}

class Sparrow implements Bird
{
    // a concrete implementation of bird
    public void fly()
    {
        System.out.println("Flying");
    }
    public void makeSound()
    {
        System.out.println("Chirp Chirp");
    }
}
```

```
interface ToyDuck
{
    // target interface
    // toyducks dont fly they just make
    // squeaking sound
    public void squeak();
}

class PlasticToyDuck implements ToyDuck
{
    public void squeak()
    {
        System.out.println("Squeak");
    }
}
```

Feladat: rávenni a „Bird”-et, hogy úgy viselkedjen, mint egy „ToyDuck”

Adapter - Példa

14

```
interface Bird
{
    // birds implement Bird interface that allows
    // them to fly and make sounds adaptee interface
    public void fly();
    public void makeSound();
}

class Sparrow implements Bird
{
    // a concrete implementation of bird
    public void fly()
    {
        System.out.println("Flying");
    }
    public void makeSound()
    {
        System.out.println("Chirp Chirp");
    }
}
```

```
interface ToyDuck
{
    // target interface
    // toyducks dont fly they just make
    // squeaking sound
    public void squeak();
}

class PlasticToyDuck implements ToyDuck
{
    public void squeak()
    {
        System.out.println("Squeak");
    }
}
```

Feladat: rávenni a „Bird”-et, hogy úgy viselkedjen, mint egy „ToyDuck”

Adapter - Példa

15

```
class BirdAdapter implements ToyDuck
{
    // You need to implement the interface your
    // client expects to use.
    Bird bird;
    public BirdAdapter(Bird bird)
    {
        // we need reference to the object we
        // are adapting
        this.bird = bird;
    }

    public void squeak()
    {
        // translate the methods appropriately
        bird.makeSound();
    }
}
```

```
class Main
{
    public static void main(String args[])
    {
        Sparrow sparrow = new Sparrow();
        PlasticToyDuck toyDuck = new PlasticToyDuck();

        // Wrap a bird in a birdAdapter so that it
        // behaves like toy duck
        ToyDuck birdAdapter = new BirdAdapter(sparrow);

        System.out.println("Sparrow...");
        sparrow.fly();
        sparrow.makeSound();

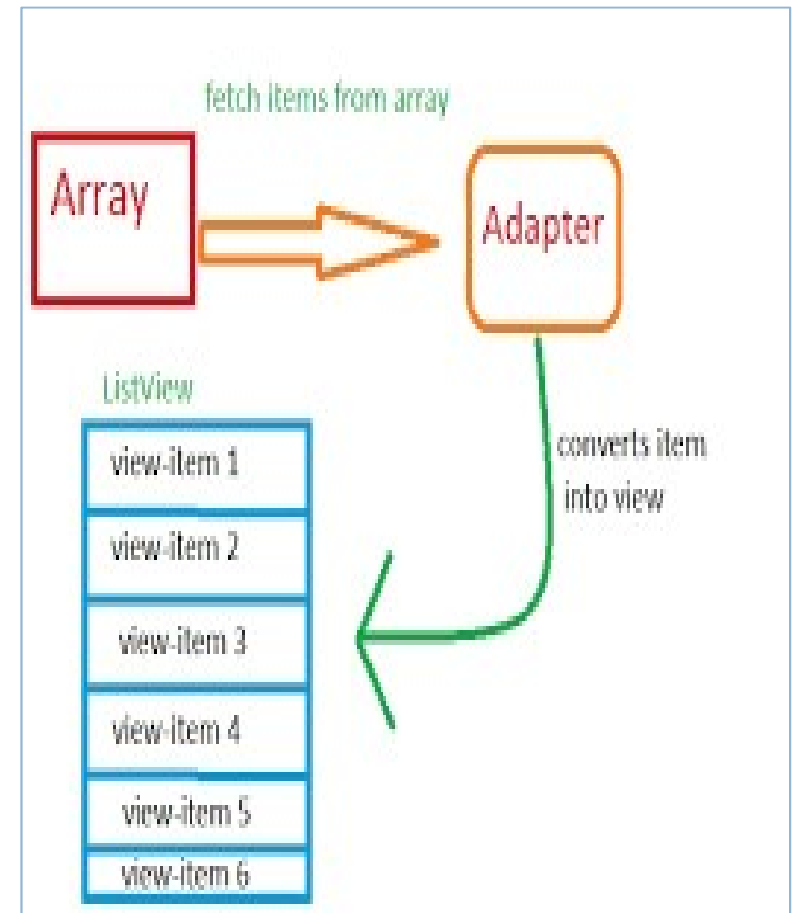
        System.out.println("ToyDuck...");
        toyDuck.squeak();

        // bird behaving like a toy duck
        System.out.println("BirdAdapter...");
        birdAdapter.squeak();
    }
}
```

ListView

16

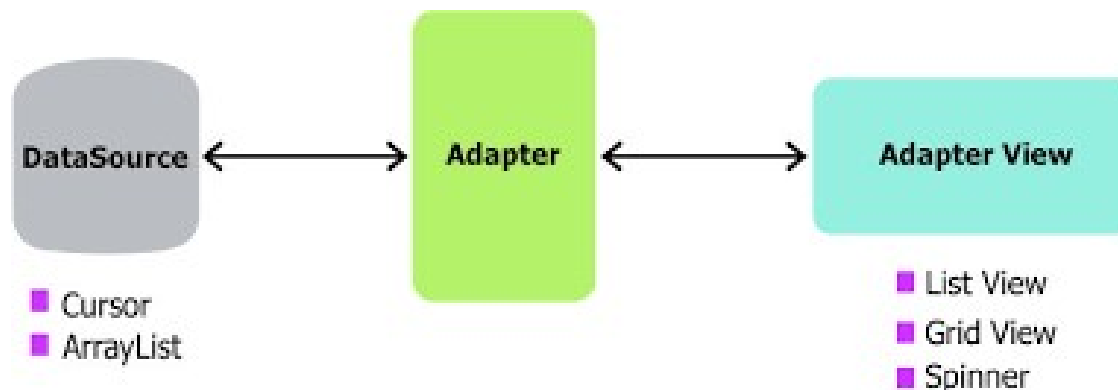
- Elemek egy görgethető listáját jeleníti meg
- A lista elemei automatikusan kerülnek beszűrésre az Adapter osztály segítségével, mely a tartalmat egy *tömbből*, vagy *adatbázisból* nyeri és a kinyert adatokat átalakítva a listában megjeleníthető view-ba konvertálja



Adapter

17

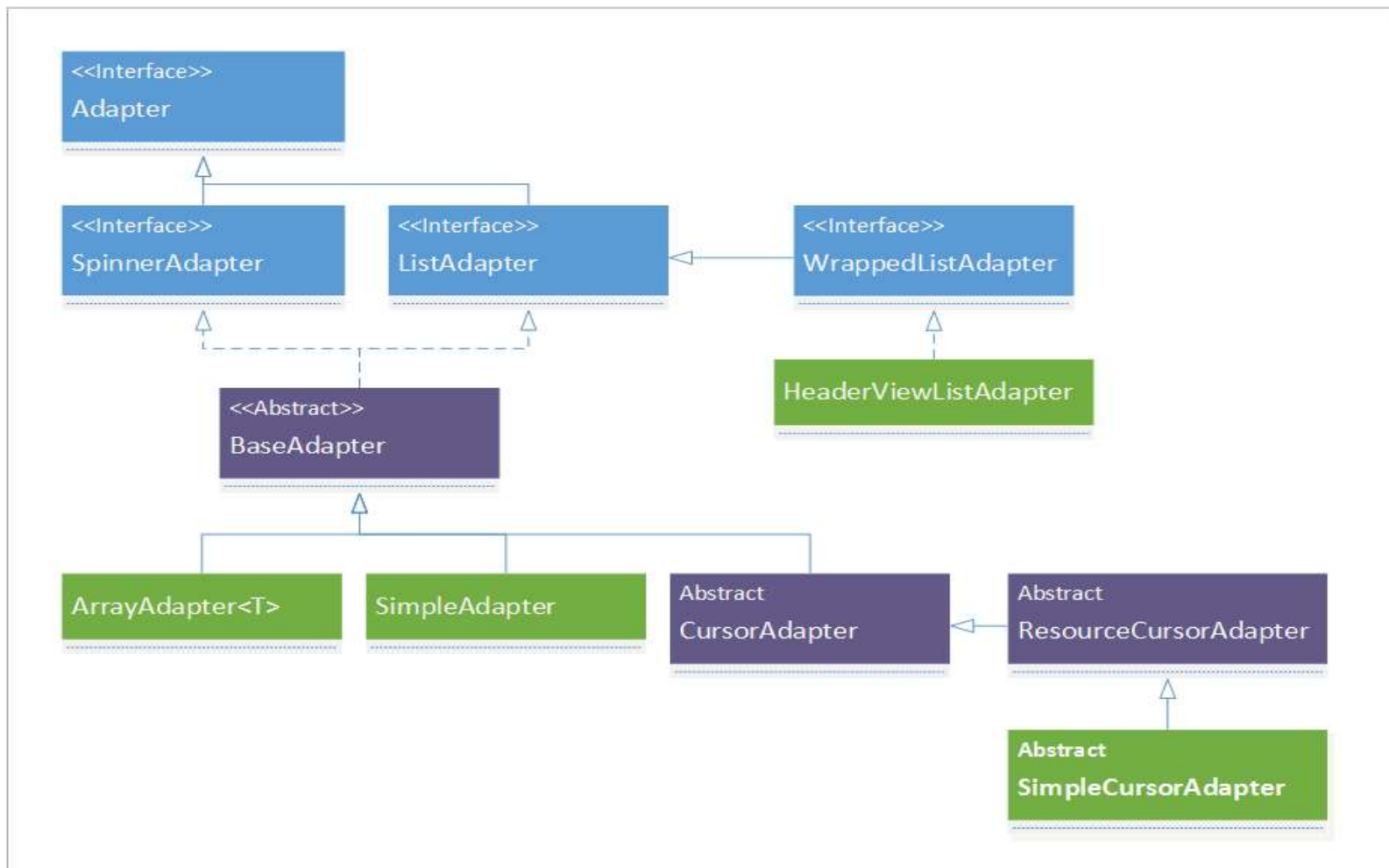
- Interface, amely összekapcsolja az adatot a megjelenítéssel
- Ahhoz, hogy a ListView-val tudjuk használni, implementálni kell a saját osztályunkat
 - ▣ SDK-ban sok alapértelmezett implementáció. Pld. BaseAdapter, ArrayAdapter<T>



Adapter

18

□ Adapter osztályok hierarchiája:



BaseAdapter osztály

19

- Absztrakt osztály
- A konkrét megvalósítása az alábbi metódusokat kell implementálja:
 - ▣ `int getCount()`: lista mérete
 - ▣ `Object getItem(int position)`: lista egy elemét adja vissza
 - ▣ `long getItemId(int position)`: sorszámot adja vissza
 - ▣ `View getView(int position, View convertView, ViewGroup parent)`: aktuális listaelem nézetét adja vissza

SimpleAdapter, CursorAdapter

20

- SimpleAdapter:
 - ▣ Listák feltöltésére használjuk valamilyen statikus erőforrásból
- CursorAdapter:
 - ▣ Listák feltöltésére használjuk valamilyen Cursor objektumot használva (pld. feltöltés adatbázisból)

ArrayAdapter

21

- Abban az esetben használjuk, ha az adatok egy tömbben tárolódnak:
- Pld: egy sztring tömb esetén használt adapter:

```
ArrayAdapter adapter = new ArrayAdapter<String>(this,  
                                              layout, stringarray);
```

ahol

- ▣ this: alkalmazás context
- ▣ layout: a nézet, amely az xml fájlban van definiálva
- ▣ stringarray: a sztringeket tartalmazó tömb (a lista elemei)
- Adapter listához való rendelése: *list.setAdapter(adapter)*

ArrayAdapter - Példa

22

- Sztring típusú lista kezelése:

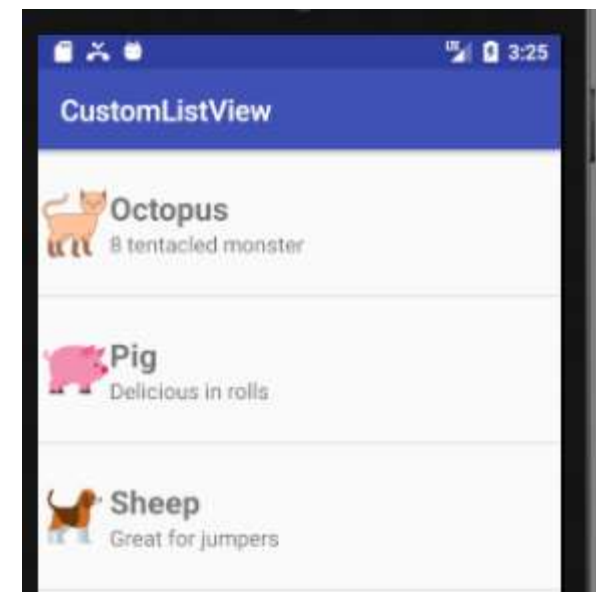
```
String[] mobileArray = {"Android", "iPhone", "WindowsMobile", "Blackberry",  
    "WebOS", "Ubuntu", "Windows7", "Max OS X"};  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    final ListView listView ;  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    ArrayAdapter adapter = new ArrayAdapter<String>(this,  
        R.layout.list_row, mobileArray);  
  
    listView = (ListView) findViewById(R.id.mobile_list);  
    listView.setAdapter(adapter);  
}
```

SimpleListView
Android
iPhone
WindowsMobile
Blackberry
WebOS
Ubuntu
Windows7
Max OS X

Custom ArrayAdapter

23

- Mi történik akkor, ha a listaelem jóval összetettebb, mint az előző példában?
 - ▣ Tartalmaz képet
 - ▣ Tartalmaz különböző szövegrészeket, stb
- Megoldás:
 - ▣ Az *ArrayAdapter* osztály kiterjesztése



Custom ArrayAdapter

24

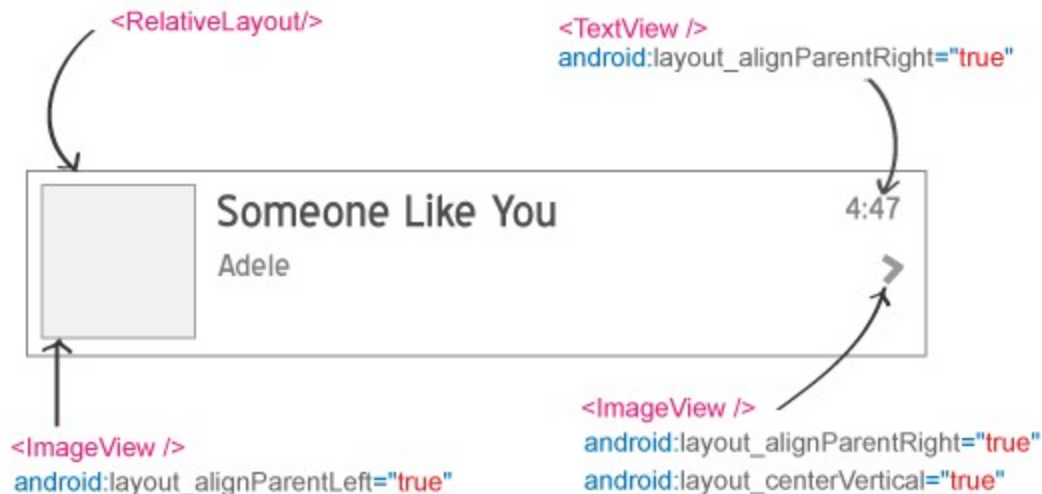
- Az adapter kell elkészítse a nézetet minden egyes sor esetén
- A *ListView* példány hívja az adapter *getView()* metódusát, amely elkészíti a lista egy sorát
- A nézet kirajzolásáért egy *LayoutInflater* osztály felel, amely erőforrást használ a kirajzolásra
- Majd az egyes nézeteket megkeresve feltöltjük adatokkal

Custom ArrayAdapter

25

- Egy listaelem (sor) gyökér nézete egy ViewGroup (layout manager) elem, amely különböző View elemeket tartalmaz (TextView, ImageView, stb)

ListView with Image and Text



Custom ArrayAdapter - Példa

26

- MainActivity layout: lista (ListView) elhelyezése

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="ro.sapi.customlistview.MainActivity">

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

</LinearLayout>
```

CustomListView

Item 1
Sub Item 1

Item 2
Sub Item 2

Item 3
Sub Item 3

Item 4
Sub Item 4

Item 5
Sub Item 5

Item 6
Sub Item 6

Item 7
Sub Item 7

Item 8
Sub Item 8

Custom ArrayAdapter - Példa

27

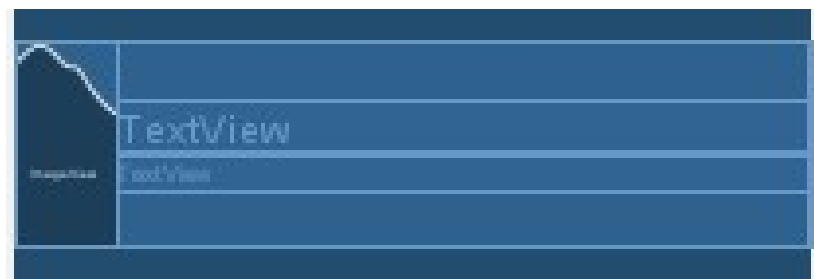
□ Lista elemek: listview_row.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/imageView1ID"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        app:srcCompat="@drawable/octopus" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center_vertical"
        android:layout_weight="1"
        android:gravity="center_vertical"
        android:orientation="vertical">

        <TextView
            android:id="@+id/nameTextViewID"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="TextView"
            android:textSize="22sp"
            android:textStyle="bold" />
    </LinearLayout>
</LinearLayout>
```



Custom ArrayAdapter

28

□ MainActivity.java: adapter osztály meghívása

```
CustomListAdapter whatever = new CustomListAdapter(this, nameArray, infoArray, imageArray);  
listView = (ListView) findViewById(R.id.ListviewID);  
listView.setAdapter(whatever);
```

```
String[] nameArray = {"Octopus", "Pig", "Sheep", "Rabbit",  
    "Snake", "Spider", "Spider", "Spider", "Spider" };
```


```
String[] infoArray = {  
    "8 tentacled monster",  
    "Delicious in rolls",  
    "Great for jumpers",  
    "Nice in a stew",  
    "Great for shoes",  
    "Scary",  
    "Scary",  
    "Scary",  
    "Scary"  
};
```

```
Integer[] imageArray = {R.drawable.cat,  
    R.drawable.disznyo,  
    R.drawable.dog,  
    R.drawable.giraffe,  
    R.drawable.horse,  
    R.drawable.lion,  
    R.drawable.octopus3,  
    R.drawable.rabbit,  
    R.drawable.sheep  
};
```

Custom ArrayAdapter - Példa

29

- CustomListAdapter.java: az adapter osztály
 - ▣ Mezők és konstruktor megadása

```
public class CustomListAdapter extends ArrayAdapter {  
      
    //to reference the Activity  
    private final Activity context;  
  
    //to store the animal images  
    private final Integer[] imageIDarray;  
  
    //to store the list of countries  
    private final String[] nameArray;  
  
    //to store the list of countries  
    private final String[] infoArray;  
  
    public CustomListAdapter(Activity context, String[] nameAr  
  
        super(context, R.layout.listview_row , nameArrayParam);  
        this.context=context;  
        this.imageIDarray = imageIDArrayParam;  
        this.nameArray = nameArrayParam;  
        this.infoArray = infoArrayParam;
```

Custom ArrayAdapter - Példa

30

- ❑ CustomListAdapter.java: az adapter osztály
 - ❑ getView metódus megírása:
 - A lista egy elemének (position) az elkészítése: a tömbbeli elemek hozzárendelése a megfelelő View komponensekhez

```
public View getView(int position, View view, ViewGroup parent) {  
    LayoutInflater inflater=context.getLayoutInflater();  
    View rowView=inflater.inflate(R.layout.listview_row, null,true);  
  
    //this code gets references to objects in the listview_row.xml file  
    TextView nameTextField = (TextView) rowView.findViewById(R.id.nameTextViewID);  
    TextView infoTextField = (TextView) rowView.findViewById(R.id.infoTextViewID);  
    ImageView imageView = (ImageView) rowView.findViewById(R.id.imageView1ID);  
  
    //this code sets the values of the objects to values from the arrays  
    nameTextField.setText(nameArray[position]);  
    infoTextField.setText(infoArray[position]);  
    imageView.setImageResource(imageIDarray[position]);  
  
    return rowView;  
}
```

GridView

31

- Egy ViewGroup elem, amely 2 dimenziós, görgethető formában jelenít meg elemeket
- A rács elemeit egy Adapter (a ListView elemhez hasonlóan) osztály szolgáltatja
- Fontosabb tulajdonságok:
 - ▣ android:columnWidth
 - ▣ android:gravity
 - ▣ android:horizontalSpacing
 - ▣ android:numColumns
 - ▣ android:stretchMode
 - ▣ android:verticalSpacing

GridView - Példa

32

□ Egyszerű adapter megvalósítás:

```
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridView1"
    android:numColumns="auto_fit"
    android:gravity="center"
    android:columnWidth="50dp"
    android:stretchMode="columnWidth"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
</GridView>
```

```
gridView = (GridView) findViewById(R.id.gridView1);

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    R.layout.list_item, numbers);

gridView.setAdapter(adapter);
```

```
static final String[] numbers = new String[] {
    "A", "B", "C", "D", "E",
    "F", "G", "H", "I", "J",
    "K", "L", "M", "N", "O",
    "P", "Q", "R", "S", "T",
    "U", "V", "W", "X", "Y", "Z"};
```



GridView - Példa

33

- Kép és szöveg megjelenítése: összetett adapter osztály
 - ▣ grid item

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

    <ImageView
        android:id="@+id/android_gridview_image"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_marginTop="15dp" />

    <TextView
        android:id="@+id/android_gridview_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:textSize="12sp" />
```



GridView - Példa

34

- Az adapter osztály elkészítése: a *BaseAdapter*-ből származtatunk:

```
public class CustomAdapter extends BaseAdapter {  
  
    private Context mContext;  
    private final String[] gridViewString;  
    private final int[] gridViewImageId;  
  
    public CustomAdapter(Context context, String[] gridViewString, int[] gridViewImageId) {  
        mContext = context;  
        this.gridViewImageId = gridViewImageId;  
        this.gridViewString = gridViewString;  
    }  
}
```

GridView - Példa

35

- Az adapter osztály elkészítése: a *BaseAdapter*-ből származtatunk:
 - ▣ *getView()* metódus implementálása

```
public View getView(int i, View convertView, ViewGroup parent) {
    View gridViewAndroid;
    LayoutInflater inflater = (LayoutInflater) mContext
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

    if (convertView == null) {
        //gridViewAndroid = new View(mContext);
        gridViewAndroid = inflater.inflate(R.layout.grid_item, null);
        TextView textViewAndroid = (TextView) gridViewAndroid.findViewById(R.id.android_gridview_text);
        ImageView imageViewAndroid = (ImageView) gridViewAndroid.findViewById(R.id.android_gridview_image);
        textViewAndroid.setText(gridViewString[i]);
        imageViewAndroid.setImageResource(gridViewImageId[i]);
    } else {
        gridViewAndroid = (View) convertView;
    }

    return gridViewAndroid;
}
```

GridView - Példa

36

□ A fő Activity:

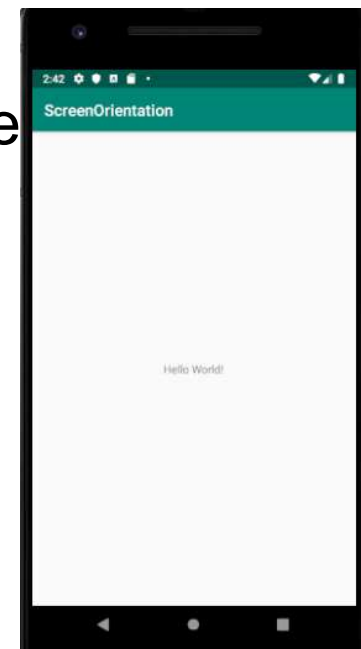
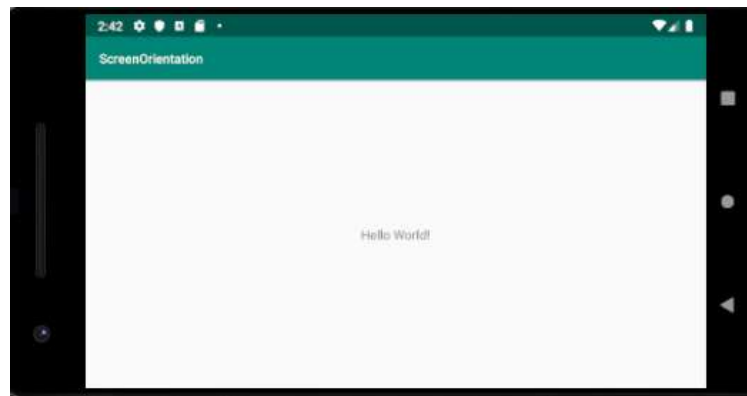
```
CustomAdapter adapterViewAndroid = new CustomAdapter(MainActivity.this, gridViewString, gridViewImageId);
androidGridView = (GridView) findViewById(R.id.grid_view_image_text);
androidGridView.setAdapter(adapterViewAndroid);
androidGridView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
```

```
String[] gridViewString = {
    "Alarm", "Android", "Mobile", "User", "Settings", "Email"
} ;
int[] gridViewImageId = {
    R.drawable.alarm, R.drawable.android, R.drawable.mobile, R.drawable.user, R.drawable.settings,
};
```

Képernyő elforgatás kezelése

37

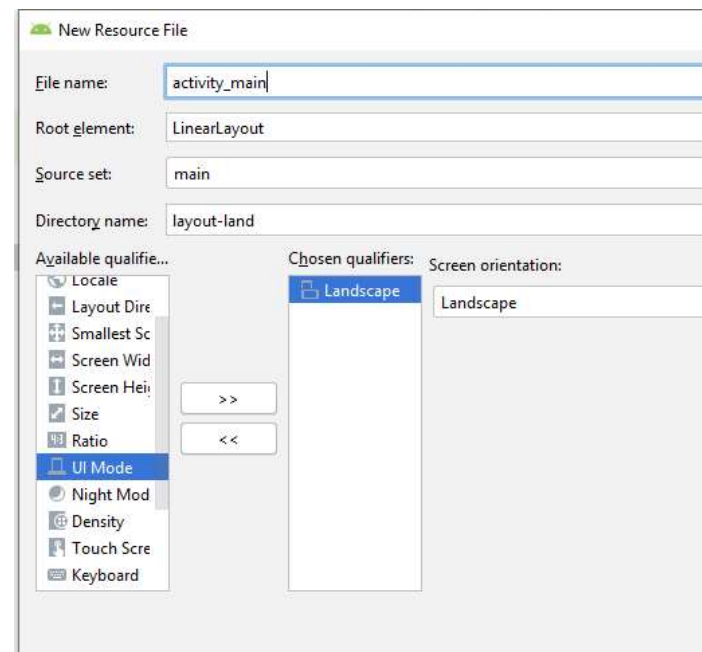
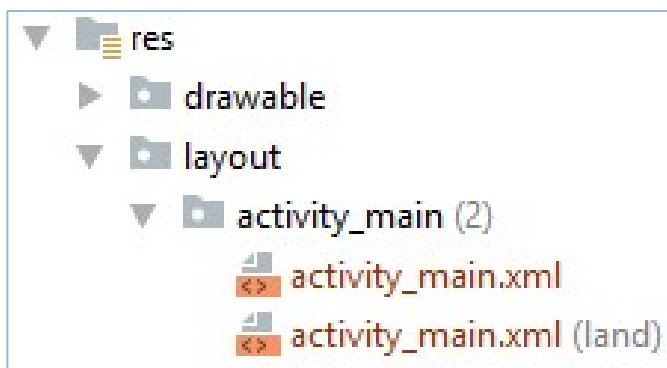
- Két mód: álló (*portrait*) és fekvő (*landscape*)
- Alapesetben:
 - ▣ Activity portrait módban van
 - ▣ Elforgatáskor az Activity újraindul és újra betöltődik a nézet
 - Ezért kell az esetleges beállításokat menteni (`onSaveInstanceState`)
 - Példa:



Képernyő elforgatás kezelése

38

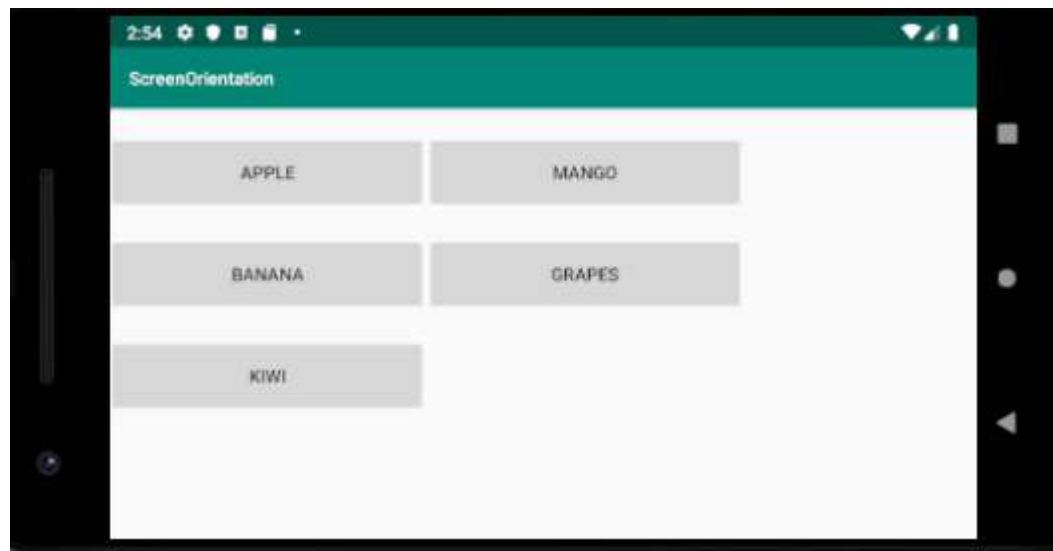
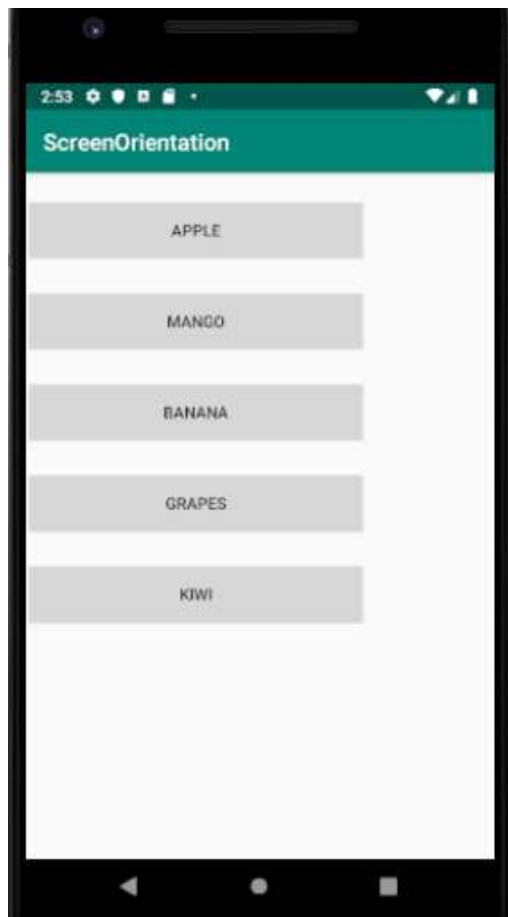
- Egy Activity-hez az alapértelmezett *portrait* mód mellett, egy *landscape* mód is csatolható
- Új nézet hozzáadása:
 - ▣ Layout mappa->New->Layout resource file->Orientation



Képernyő elforgatás kezelése

39

□ Példa:



Képernyő elforgatás kezelése

40

- ActivityInfo: osztály amely az Activity különböző beállításait tartalmazza (ezek állíthatóak be a Manifest fájlban)
- Képernyő elforgatásával kapcsolatos beállítások:
 - ▣ ActivityInfo.SCREEN_ORIENTATION_UNSPECIFIED
 - ▣ ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE
 - ▣ ActivityInfo.SCREEN_ORIENTATION_PORTRAIT
 - ▣ ActivityInfo.SCREEN_ORIENTATION_USER
 - ▣ ActivityInfo.SCREEN_ORIENTATION_BEHIND
 - ▣ ActivityInfo.SCREEN_ORIENTATION_SENSOR
 - ▣ *Stb.*

Képernyő elforgatás kezelése

41

- android:configChanges
 - ▣ Manifest-ben állítjuk be
 - ▣ Jelzi hogy manuálisan kezeljük az elforgatásokat: az Activity-t nem törli a memóriából
 - ▣ Elforgatáskor az *onConfigurationChanged* függvény hívódik meg
 - ▣ Példa:

```
<activity
    android:name=".MainActivity"
    android:configChanges="orientation|screenSize|keyboardHidden">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Képernyő elforgatás kezelése

42

- Képernyő mód rögzítése:
 - Manifest fájlban
 - *android:screenOrientation="landscape"* vagy
 - *android:screenOrientation="portrait"*
- *Példa:*

```
<activity
    android:name=".MainActivity"
    android:screenOrientation="landscape"
    android:configChanges="orientation|screenSize|keyboardHidden">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Képernyő elforgatás kezelése

43

- Képernyő mód rögzítése: kódból
 - ▣ *onConfigurationChanged* lekezelése
- *Példa:*

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    // Checks the orientation of the screen for landscape and portrait and set
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT) {
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    }
}
```

Képernyő elforgatás kezelése

44

- Képernyő mód lekérdezése: kódból
 - ▣ *onConfigurationChanged* lekezelése
- *Példa:*

```
public void onConfigurationChanged(Configuration newConfig) {  
    super.onConfigurationChanged(newConfig);  
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {  
        Toast.makeText( context: this, text: "landscape", Toast.LENGTH_SHORT).show();  
    } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT) {  
        Toast.makeText( context: this, text: "portrait", Toast.LENGTH_SHORT).show();  
    }  
}
```

Eseménykezelés

45

- Eseményfigyelő interfészek legfontosabb callback metódusai:
 - ▣ `onClick()` (`View.OnClickListener` interfész) – akkor hívódik meg, ha a felhasználó megérinti az adott elemet
 - ▣ `onLongClick()` (`View.OnLongClickListener` interfész) – akkor hívódik meg, ha a felhasználó megérinti az adott elemet, és továbbra is nyomva tartja azt
 - ▣ `onFocusChange()` (`View.OnFocusChangeListener` interfész) – akkor hívódik meg, ha a felhasználó az adott elemre, vagy arról elnavigál.

Eseménykezelés

46

- Eseményfigyelő interfészek legfontosabb callback metódusai:
 - ▣ `onKey()` (`View.OnKeyListener` interfész) - akkor hívódik meg, ha a fókus az adott elemen van, és a felhasználó megnyom, vagy felenged egy hardveres gombot
 - ▣ `onTouch()` (`View.OnTouchListener`) – akkor hívódik meg, ha a felhasználó végrehajt egy érintési eseményt, ami lehet lenyomás, felengedés, vagy egyéb mozgatóssal járó kézmozdulat

Esemény regisztrálás

47

- 3 módszer:
 - ▣ Belső osztály
 - ▣ Listener interfész megvalósítása az Activityben
 - ▣ Az xml fájlban direkt adjuk meg az eseménykezelőt
- Belső osztály:
 - ▣ Példa:

```
button1 = (Button) findViewById(R.id.button1);
button1.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View arg0) {
        text.setText("Inner Listener Class Way!");
        Toast.makeText(getApplicationContext(), "Into the ListenerActivity Class",
            Toast.LENGTH_SHORT).show();
    }
});
```

Esemény regisztrálás

48

- Listener interfész megvalósítása az Activityben
- Példa: események regisztrálása

```
public class MainActivity extends Activity implements OnClickListener {  
  
    private TextView text;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        text = (TextView) findViewById(R.id.text);  
  
        Button listenerBtn = (Button) findViewById(R.id.buttonMain);  
        listenerBtn.setOnClickListener(this);  
  
        Button nextBtn = (Button) findViewById(R.id.nextBtn);  
        nextBtn.setOnClickListener(this);  
    }  
}
```


Esemény regisztrálás

49

- Listener interfész megvalósítása az Activityben
- Példa: több vezérlő kezelése

```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.buttonMain:
            text.setText("Listener implementation in the Activity!");
            Toast.makeText(this, "Implements onClickListener",
                Toast.LENGTH_SHORT).show();
            break;

        case R.id.nextBtn:
            try{
                Intent intent = new Intent(this, ListenerActivity.class);
                startActivity(intent);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
            break;
    }
}
```

Esemény regisztrálás

50

- Az xml fájlban direkt adjuk meg az eseménykezelőt

- ```
<Button
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginTop="10dp"
 android:id="@+id/button2"
 android:onClick="clickButton2"
 android:text="onClick Button" />
```

```
public void clickButton2(View view) {
 text.setText("onClick - XML Way!");
 Toast.makeText(getApplicationContext(), "Function that was declared in XML",
 Toast.LENGTH_SHORT).show();
}
```

# Esemény kezelés – ListView példa

51

- Példa: *onItemClickListener* esemény, belső osztály megvalósítással

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
 @Override
 public void onItemClick(AdapterView<?> parent, View view, int position,
 long id) {
 Intent intent = new Intent(MainActivity.this, DetailActivity.class);
 String message = nameArray[position];
 intent.putExtra("animal", message);
 startActivity(intent);
 }
});
```

# Kérdések

52

- ❑ TableLayout vs GridLayout?
- ❑ Mit értünk tervezési mintán?
- ❑ Tervezési minták osztályozása?
- ❑ Mi a szerepe az Adapter (Illesztő) tervezési mintának?
- ❑ ListView megvalósítás lépései Adapter-el?
- ❑ Simple ArrayAdapter vs. Custom ArrayAdapter?
- ❑ ListView vs. GridView?
- ❑ Esemény regisztrálás lehetőségei?

# Szakirodalom

53

- Ekler Péter és társszerzői, Android alapú szoftverfejlesztés, Szak kiadó, 2012.
- <https://refactoring.guru/design-patterns/adapter>