Systemy operacyjne na platformach mobilnych

Wykład 3

Grzegorz Jabłoński, Piotr Perek
Katedra Mikroelektroniki i Technik Informatycznych

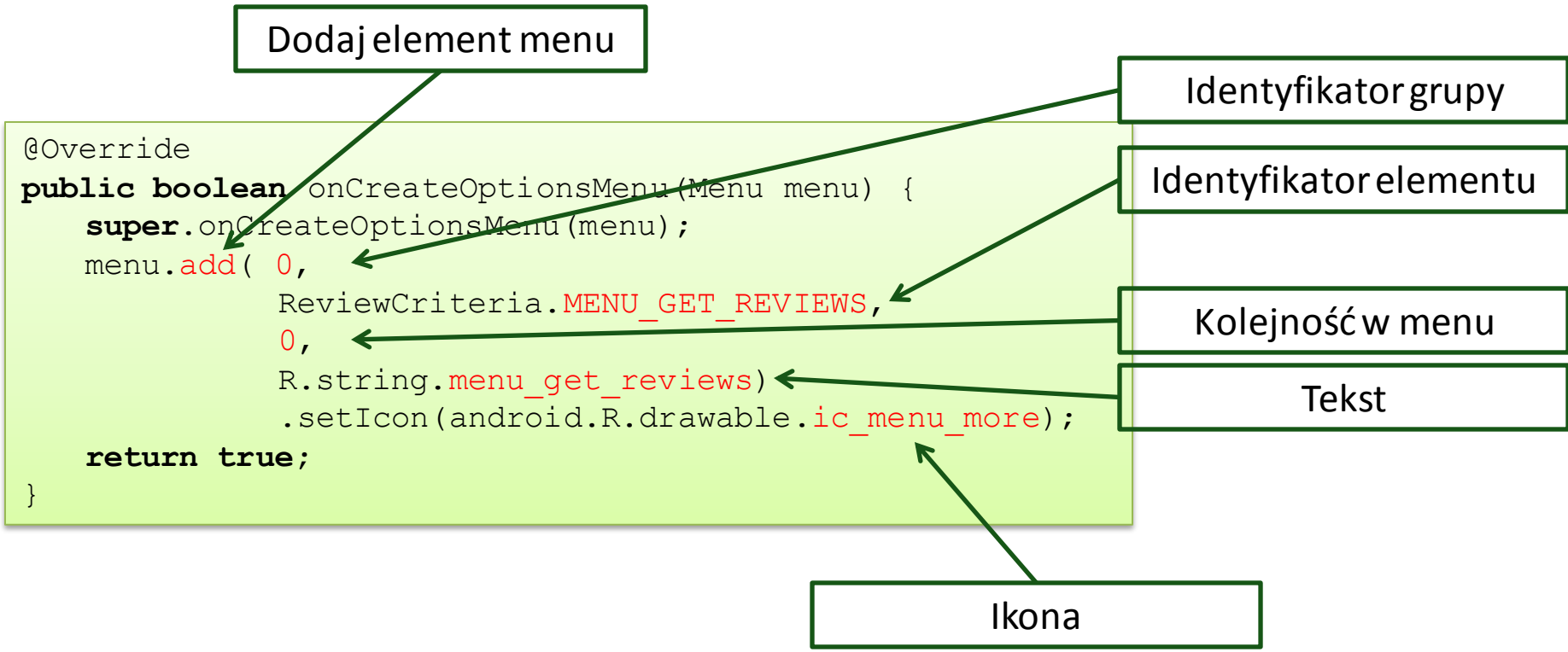
Zagadnienia wykładu

- Menu opcji
- ListActivity
  - własny widok
  - własny adapter
- BroadcastReceiver
- Serwis

Tworzenie menu

- Podobnie jak w przypadku widoków, menu może być zdefiniowane na dwa sposoby:
  - w języku Java
  - w formie pliku XML

Tworzenie menu - Java




```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    menu.add( 0,
              ReviewCriteria.MENU_GET_REVIEWS,
              0,
              R.string.menu_get_reviews)
              .setIcon(android.R.drawable.ic_menu_more);
    return true;
}
```

Tworzenie menu - Java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {

    menu.add(Menu.NONE, ID_MENU_OPTION_A, 0, "Option A");
    menu.add(Menu.NONE, ID_MENU_OPTION_B, 1, "Option B");
    menu.add(Menu.NONE, ID_MENU_OPTION_C, 2, "Option C");

    SubMenu submenu = menu.addSubMenu(Menu.NONE, ID_MENU_SUBMENU, 3, "Submenu");
    submenu.add(Menu.NONE, ID_SUBMENU_OPTION_A, 0, "Option A");
    submenu.add(Menu.NONE, ID_SUBMENU_OPTION_B, 1, "Option B");
    submenu.add(Menu.NONE, ID_SUBMENU_OPTION_C, 2, "Option C");

    return true;
}
```

Tworzenie menu - XML

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/option_a" android:title="Option A"/>
    <item android:id="@+id/option_b" android:title="Option B"></item>
    <item android:id="@+id/option_c" android:title="Option C"></item>
    <item android:title="Submenu">
        <menu>
          <item android:id="@+id/submenu_option_a" android:title="Option A"/>
          <item android:id="@+id/submenu_option_b" android:title="Option B"/>
          <item android:id="@+id/submenu_option_c" android:title="Option C"/>
        </menu>
    </item>
</menu>
```

Tworzenie menu - XML

MenuInflater tworzy obiekty języka Java na podstawie pliku XML

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    //Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
```

Plik XML opisujący menu

Obsługa zdarzenia wyboru menu

Sprawdzenie który element menu został wybrany

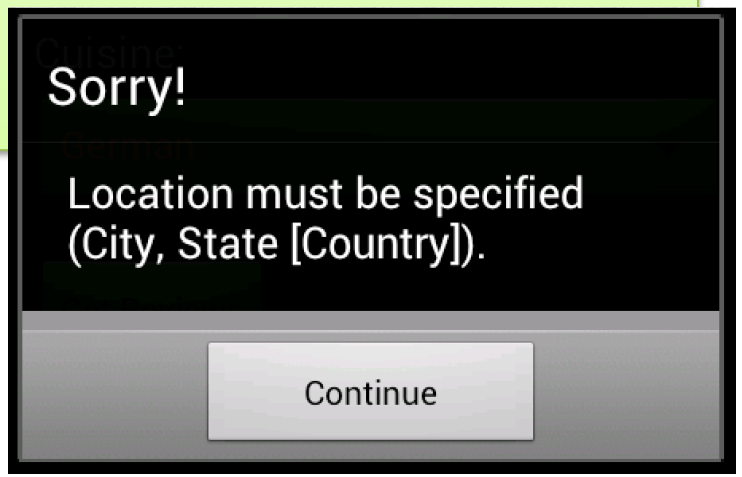
```
@Override
public boolean onMenuItemSelected(intfeatureId, MenuItem item) {
    switch(item.getItemId()) {
        case MENU_GET_REVIEWS:
            handleGetReviews();
            return true;
    }

    return super.onMenuItemSelected(featureId, item);
}
```

Funkcja obsługująca daną opcję

Okno dialogowe

```
private boolean validate() {
     .........
     if(!valid)
     {
          new AlertDialog.Builder(this)
               .setTitle(getResources().getString(R.string.alert_label))
               .setMessage(validationText.toString()).setPositiveButton(
               "Continue",
                   new android.content.DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int arg1) { }
                   })
               .show();
          validationText = null;
     }
     return valid;
}
```

Klasa tworząca okno dialogowe



ListActivity

- Służy do wyświetlenia zbioru danych np. tablicy w postaci przewijanej listy
- Umożliwia obsługę wyboru poszczególnych elementów listy
- Wymaga, aby layout definiujący widok zawierał obiekt ListView o identyfikatorze @android:id/list

ListActivity - implementacja

Dziedziczenie po ListActivity

```
public class MainActivity extends ListActivity {

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

        String[] items = new String[] { "Item 1", "Item 2", "Item 3", "Item
4", "Item 5", "Item 6", "Item 7", "Item 8", "Item 9", "Item 10"};

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

        setListAdapter(adapter);
    }

    …
```

Podstawowy adapter konwertujący tablicę na zestaw widoków dla ListActivity

Własny layout wierszy

```
<?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="horizontal" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="4"
        android:text=""
        android:textColor="@android:color/holo_red_light"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text=""
        android:textColor="@android:color/holo_blue_light" />

</LinearLayout>
```

Wiersz składa się z dwóch pól tekstowych: textView1 i textView2

Własny layout wierszy

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

    ArrayList<HashMap<String, String>> values = new
                                    ArrayList<HashMap<String,String>>();

    for(int i = 0; i < 100; i++)
    {
        HashMap<String, String> map = new HashMap<String, String>();
        map.put("id", String.valueOf(i));
        map.put("text", "item" + String.valueOf(i));

        values.add(map);
    }

    SimpleAdapter adapter = new SimpleAdapter(this, values,
                R.layout.row_layout, new String[]{"id", "text"},
                new int[]{R.id.textView1, R.id.textView2});

    setListAdapter(adapter);
}
```

Nazwa pliku opisującego layout dla pojedynczego wiersza

Mapowanie kluczy mapy na widoki

Własny adapter

```
public class MySimpleAdapter extends ArrayAdapter<MyContact>{
    private final Context context;
    private final MyContact[] contacts;

    public MySimpleAdapter(Context context, MyContact[] objects) {
        super(context, R.layout.row_layout, objects);
        this.context = context;
        this.contacts = objects;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View rowView = inflater.inflate(R.layout.row_layout, parent, false);

        TextView textView1 = (TextView) rowView.findViewById(R.id.textView1);
        TextView textView2 = (TextView) rowView.findViewById(R.id.textView2);
        textView1.setText(String.valueOf(contacts[position].getName()));
        textView2.setText(contacts[position].getPhone());

        return rowView;
    }
}
```

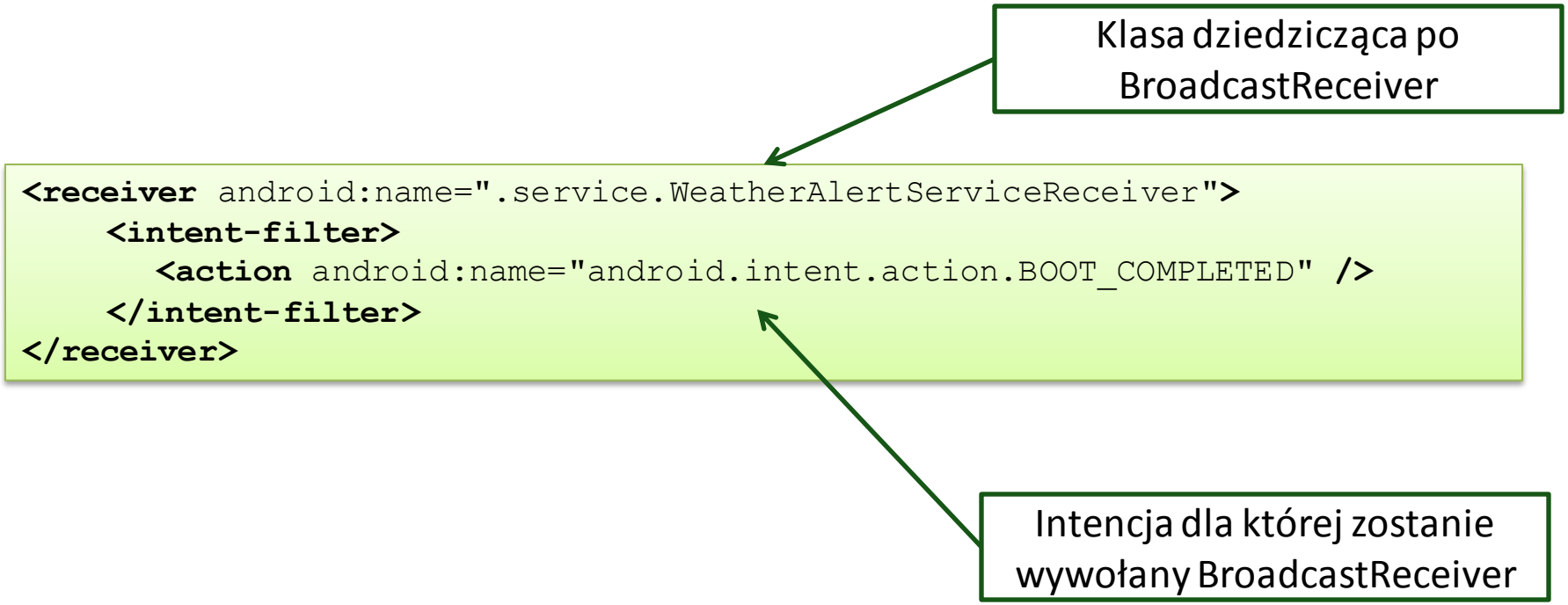
Własny adapter

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

    MyContact contacts[] = new MyContact[10];
    contacts[0] = new MyContact("Contact 1", "600-100-200");
    . . .

    MySimpleAdapter adapter = new MySimpleAdapter(this, contacts);

    setListAdapter(adapter);
}
```

BroadcastReceiver

- Umożliwia odbiór intencji rozsyłanych do wszystkich aplikacji w systemie za pomocą funkcji sendBroadcast()
- Może być zdefiniowany w pliku AndroidManifest.xml (tag `<receiver>`) lub tworzony i rejestrowany dynamicznie z użyciem funkcji registerReceiver()
- Dostarcza metodę onReceive(), która jest wywoływana w momencie odebrania intencji
- Obiekty klasy BroadcastReceiver mają krótki cykl życia i po wywołaniu metody onReceive() mogą zostać usunięte przez system
- W związku z powyższym w klasach BroadcastReceiver nie można wykonywać operacji asynchronicznych (uruchamiać wątków, wyświetlać okien)

BroadcastReceiver




```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
```

Odbiór rozgłaszanych intencji

```
public classWeatherAlertServiceReceiver extends BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent) {
    if(intent.getAction().equals(Intent.ACTION_BOOT_COMPLETED)) {
        context.startService(newIntent(context, WeatherAlertService.class));
    }
}}
```

Service

- Działa w tle
- Jest używany do wykonywania długotrwałych operacji (np. ściąganie danych z internetu, sprawdzanie czy pojawiły się nowe dane)
- Nie posiada interfejsu do bezpośredniej interakcji z użytkownikiem
- Działa z wyższym priorytetem niż nieaktywne i niewidoczne aktywności
- Domyślnie jest uruchamiany w głównym wątku aplikacji, dlatego zalecane jest utworzenie nowego wątku do wykonywania zadań wymagających dużej ilości zasobów
- Serwisy działające w głównym wątku aplikacji są nazywane serwisami lokalnymi

Service - implementacja

AndroidManifest.xml

```
<service
    android:name="com.example.calcservice.CalculatorService" >
    <intent-filter >
        <action android:name="com.example.calcservice.ICalc"/>
    </intent-filter>
</service>
```

Service - implementacja

- Serwis

```
public class CalculatorService extends Service {

    IBinder mBinder = new LocalBinder();

    public class LocalBinder extends Binder {
        public CalculatorService getServerInstance() {
            return CalculatorService.this;
        }
    }

    public double calculate(char op_code, double arg_a, double arg_b) throws
RemoteException {
        . . .
        }
        return result;
    }

    public CalculatorService() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }
}
```

Service - implementacja

- Aktywność

```
protected void onStart() {
        super.onStart();
        if(!bound)
        {
            bindService(new Intent(this, CalculatorService.class), connection,
Context.BIND_AUTO_CREATE);
        }
    }
```

Service - implementacja

- Aktywność

```
private ServiceConnection connection = new ServiceConnection(){

        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            LocalBinder mLocalBinder = (LocalBinder)service;
            mService = mLocalBinder.getServerInstance();
            Toast.makeText(CalcSimple.this, "Connected to service",
Toast.LENGTH_LONG).show();
            bound = true;
        }

        @Override
        public void onServiceDisconnected(ComponentName name) {
            mService = null;
            Toast.makeText(CalcSimple.this, "Disconnected from service",
Toast.LENGTH_LONG).show();
            bound = false;
        }

    };
```