Laborator 04
Structura unei Aplicații (II)

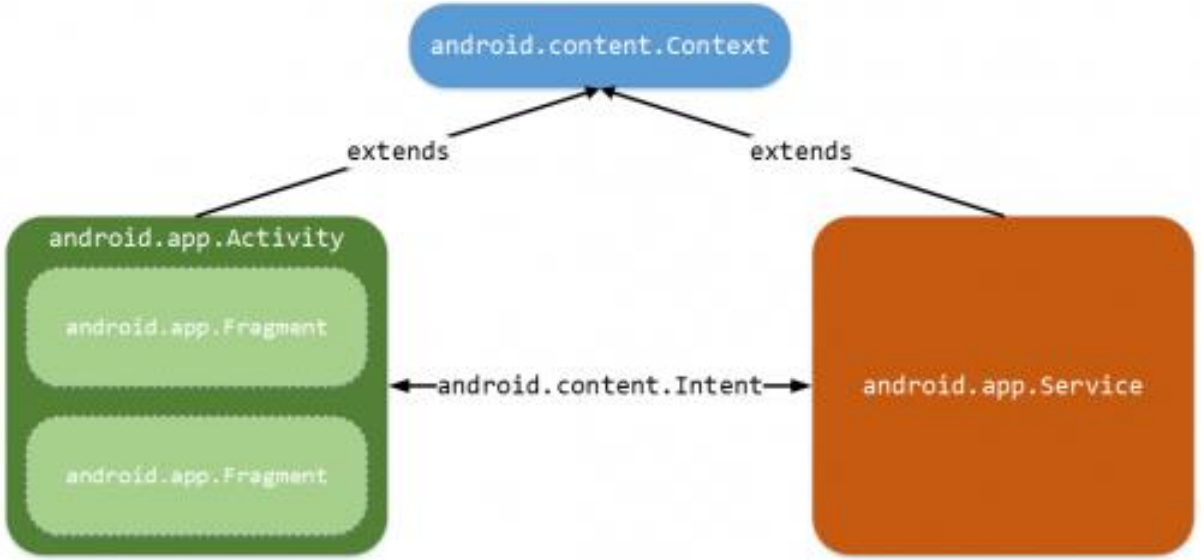
Elemente de Informatică Mobilă
Semestrul de Primăvară 2016

Agenda

- Intenții
  - Structura
  - Mecanisme de Invocare
  - Fluxuri Informaționale între Componente
  - Intenții cu Difuzare
- Fragmente
  - Structura
  - Ciclul de Viață al unui Fragment
  - Lucrul cu Fragmente
  - Interacțiunea între Fragmente

Intenții

- mecanism de comunicație inter-proces în sistemul de operare Android, asigurând
  - extensibilitate
  - flexibilitate
- intenție = acțiune + date
  - transmisă ca mesaj asincron
  - metodă preferată pentru execuția unei acțiuni asupra unor date
- sistemul de operare Android – colecție de componente slab cuplate



Funcționalitatea unei Intenții

- invocarea
  - unei activități – `startActivity()`, `startActivityForResult()` / `onActivityResult()`
    - în cadrul aceleiași aplicații Android
    - din cadrul altei aplicații, rezidente în contextul sistemului de operare Android
  - unui serviciu – `startService()`
- transmiterea unui mesaj cu difuzare
  - propagate la nivelul sistemului de operare Android
  - gestiunea evenimentelor: apeluri telefonice / mesaje, nivelul de energie, conectivitatea
- poate încapsula date sub forma unui obiect de tipul `android.os.Bundle`

Structura unei Intenții

- precizată în fișierul `AndroidManifest.xml` pentru fiecare activitate / serviciu = `<intent-filter>`
- folosită de mecanismul de rezoluție al intențiilor
- componente
  - **`action`** (android:name) – `MAIN`, `VIEW`, `DIAL`, `CALL`, `ANSWER`, `SEND`, `SENDTO`, `INSERT`, `EDIT`, `DELETE`, `SEARCH`, `WEB_SEARCH`
    - nu se află în relație 1:1 cu o componentă (activitate, serviciu)!!!
    - convenție de nume în Java
  - `category` (android:name) – `LAUNCHER`, `DEFAULT`, `ALTERNATIVE`, `SELECTED_ALTERNATIVE`, `BROWSABLE`, `HOME`
  - **`data`** – sub forma unui URI
    - `android:host`, `android:port`
    - `android:mimeType`
    - `android:path`
    - `android:scheme` – `tel:`, `content://contacts/people`, `geo:`, `http://`, `file://`, `mailto:`
  - `type`
  - `component` – intenții implicite / explicite
  - `extra`
    - date transmise sub forma unui obiect de tip `android.os.Bundle`
    - `putExtras(Bundle)`, `getExtras()`

Mecanisme de Invocare a unei Intenții

- specificarea clasei Java încărcate
  - corespunzătoare unei activități din cadrul aceleiași aplicații Android
  - corespunzătoare unei activități în contextul sistemului de operare Android
- precizarea acțiunii realizate (și a datelor)
- indicarea unui URI (Uniform Resource Identifier)

Invocarea unei Intenții prin Specificarea Clasei Java Încărcate

- **intenții explicite**
- în fișierul `AndroidManifest.xml`, în secțiunea `<intent-filter>` trebuie indicate
  - acțiunea
    - predefinită
    - definită de utilizator - se folosește un identificator unic, folosind de regulă denumirea pachetului și clasa care deservește activitatea
  - categorie: `android.intent.category.LAUNCHER`, `android.intent.category.DEFAULT`

```
<activity
  android:name="ro.pub.cs.systems.eim.lab04.MainActivity"
  android:label="@string/app_name" >
  <intent-filter>
    <action android:name="ro.pub.cs.systems.eim.lab04.intent.action.MainActivity" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

Invocarea unei Intenții prin Specificarea Clasei Java Încărcate

- tipuri de constructori
  - corespunzătoare unei activități din cadrul aceleiași aplicații Android

```
Intent intent = new Intent(this, MainActivity.class)
```

  - corespunzătoare unei activități în contextul sistemului de operare Android

```
Intent intent =
        new Intent("ro.pub.cs.systems.eim.lab04.intent.action.MainActivity")
```

- lansarea în execuție – `startActivity(intent)`
  - componenta este plasată în vârful stivei
  - sunt invocate metodele de callback corespunzătoare
- terminarea – `finish()`
- `getIntent()` – obținerea intenției prin intermediul căreia a fost invocată componenta

Invocarea unei Intenții prin Precizarea Acțiunii Realizate (și a Datelor)

- **intenții implicite**
  - nu este cunoscută componenta care va fi invocată
  - sistemul de operare Android este responsabil pentru a identifica o componentă care poate deservi acțiunea precizată asupra datelor respective
- **algoritmul de rezoluție a intenției**
  1. se construiește lista cu filtrele de intenții pentru toate componentele Android
  2. sunt eliminate componentele cu filtre de intenții necorespunzătoare
     a) **nici una** dintre acțiuni nu este corespunzătoare intenției de rezolvat
     b) nu conține **toate** categoriile din cadrul intenției de rezolvat
  3. comparație câmpurilor din cadrul secțiunii de date
  4. alegerea unei componente de către utilizator, în situația în care sunt identificate mai multe

Invocarea unei Intenții prin Precizarea Acțiunii Realizate (și a Datelor)


```
Intent intent = new Intent(...);
PackageManager packageManager = new PackageManager();
ComponentName componentName = intent.resolveActivity(packageManager);
if (componentName == null) {
  Intent marketIntent = new Intent(Intent.ACTION_VIEW,
                                   Uri.parse("market://search?q=pname:..."));
  if (marketIntent.resolveActivity(packageManager) != null) {
    startActivity(marketIntent);
  } else {
    Toast.makeText(getApplicationContext(),
                   "Google Play Store is not available",
                   Toast.LENGTH_LONG)
         .show();
  }
} else {
  startActivity(intent);
}
```

Invocarea unei Intenții prin Precizarea Acțiunii Realizate (și a Datelor)

- `queryIntentActivities()` – lista tuturor acțiunilor disponibile pentru un set de date
  - `intent`
    - `data`
    - `category`: `Intent.CATEGORY_SELECTED_ALTERNATIVE`
  - `flags` – `PackageManager.MATCH_DEFAULT_ONLY`
- o componentă are acces la
  - intenția care a invocat-o: `getIntent()`
  - proprietățile intenției: `getAction()`, `getData()`, `getExtras()`
- transferul de responsabilitate
  - `onNewIntent()` – metodă de callback apelată automat când o componentă este invocată după ce a fost creată
  - `startNextMatchingActivity()` – lansarea în execuție a următoarei componente identificată în procesul de rezoluție a intenției

Invocarea unei Intenții prin Precizarea Acțiunii Realizate (și a Datelor)


```
Intent intent = new Intent(Intent.ACTION_...);
intent.setData(Uri.parse("..."));
```

- `Intent.ACTION_VIEW` – vizualizarea conținutului în secțiunea `data`, în funcție de protocol (`tel`, `http`, `geo`, `content`)
- `Intent.ACTION_INSERT`, `Intent.ACTION_EDIT`, `Intent.ACTION_DELETE`
- `Intent.ACTION_SEARCH`, `Intent.ACTION_WEB_SEARCH`
- `Intent.ACTION_DIAL`
- `Intent.ACTION_CALL` – permisiunea `android.permission.CALL_PHONE`
- `Intent.ACTION_ALL_APS`
- `Intent.ACTION_PICK`
- `Intent.ACTION_SEND`, `Intent.ACTION_SENDTO`

Invocarea unei Intenții prin Indicarea unui URI (Uniform Resource Identifier)

- se folosește
  - acțiunea `Intent.ACTION_VIEW`
  - un URI pentru care se indică o schemă și o gazdă (restul informațiilor sunt ignorate)

```
Intent intent = new Intent(Intent.ACTION_VIEW,
                           Uri.parse("myprotocol://mynamespace/myactivity"));
startActivity(intent);
```

- în secțiunea `<intent-filter>`, componenta trebuie să indice eticheta `data` cu atributele `android:scheme` și `android:host`

```
<intent-filter>
    <action android:name="ro.pub.cs.systems.eim.lab04.intent.action.AnotherActivity" />
    <category android:name="android.intent.category.DEFAULT" />
    <data
      android:scheme="myprotocol"
      android:host="mynamespace" />
</intent-filter>
```

Fluxuri Informaționale între Componente

- transfer de date unidirecțional
  - componenta care invocă → componenta invocată
  - câmpul `extra` conține asocieri în cadrul unui obiect `android.os.Bundle`
    - `putExtras(Bundle)` – persistența valorilor existente, `getExtras()`
    - `put<Type>Extra(key, value)`, `get<Type>Extra(key)`
      - `key` – identificator unic (eventual prefixat de denumirea pachetului)
      - `value` – tip de dată `android.os.Parcelable`
- componentele care comunică au o existență autonomă!!!
  - `startActivity()`
  - `startActivityForResult()` / `onActivityResult()`

Fluxuri Informaționale între Componente

ACTIVITATEA PĂRINTE	ACTIVITATEA COPIL

- `startActivityForResult(Intent, int)`
  - `intent` – activitate invocată (acțiune, date, inclusiv câmpul `extra`)
  - `requestCode` – identificarea componentei
- `onActivityResult(int, int, Intent)`
  - `requestCode` – identificarea componentei
  - `resultCode` – rezultat
    - `Activity.RESULT_OK` = -1 !!!
    - `Activity.RESULT_CANCELED` = 0
  - `Intent` – pentru informații suplimentare

- `getIntent()` – informații cu privire la activitatea care a invocat
  - `getAction()`
  - `getData()`
  - `getExtra()`
- `setResult(int, Intent)`
  - `resultCode` – rezultat
    - `Activity.RESULT_OK` = -1 !!!
    - `Activity.RESULT_CANCELED` = 0
  - `Intent` – pentru informații suplimentare
- `finish()` – marchează faptul că activitatea a fost terminată

Fluxuri Informaționale între Componente

ACTIVITATEA PĂRINTE

```
@Override
protected void onCreate(Bundle state) {
  // ...
  Intent intent = new Intent("AnotherActivity");
  intent.putExtra("someKey", someValue);
  startActivityForResult(intent, 1);
}
@Override
public void onActivityResult(int requestCode,
                    int resultCode, Intent intent) {
  switch(requestCode) {
    case 1:
      String anotherValue =
                    intent.getExtra("anotherKey");
      break;
  }
}
```

ACTIVITATEA COPIL

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

  // intent from parent
  Intent intentFromParent = getIntent();
  Bundle data = intentFromParent.getExtras();
  String someValue = intentFromParent
                          .getExtra("someKey");

  // intent to parent
  Intent intentToParent = new Intent();
  intentToParent.putExtra(
                    "anotherKey",
                    anotherValue);
  setResult(RESULT_OK, intentToParent);
  finish();
}
```

Intenții cu Difuzare

- propagate la nivelul tuturor componentelor din cadrul sistemului de operare Android
  - notifică un eveniment (starea dispozitivului mobil, aplicații) – folosit inclusiv de implementarea sistemului de operare
  - procesate de către obiecte ascultător dedicate
    - pot fi activate / dezactivate în anumite condiții particulare de lucru
    - `setComponentEnabledSetting()` – `COMPONENT_ENABLED_STATED_ENABLED`, `COMPONENT_ENABLED_STATE_DISABLED`
- tipuri
  1. globale
  2. locale (`LocalBroadcastManager`) – apel sincron, la nivelul componentelor din cadrul aplicației

Intenții cu Difuzare

- trimiterea
  - `Intent intent = new Intent(action);`
    - acțiuni predefinite și definite de utilizator
    - descriere completă: categorie, date (inclusiv câmpul `extra`)
  - `sendBroadcast(intent)`
- primirea – componente specializate, înregistrate în acest sens
  1. fișierul `AndroidManifest.xml` – nu este necesar ca aplicația Android să ruleze pentru ca mesajul să fie procesat
  2. programatic – este necesar ca aplicația Android să se afle în execuție pentru ca mesajul să fie procesat

Procesarea unei Intenții cu Difuzare

- fișierul `AndroidManifest.xml`

```
<manifest ... >
  <application ... >
    <receiver
      android:name=".SomeEventBroadcastReceiver">
      <intent-filter>
        <action android:name="ro.pub.cs.systems.eim.lab04.SomeAction.SOME_ACTION" />
      </intent-filter>
    </receiver>
  </application>
</manifest>
```

Procesarea unei Intenții cu Difuzare

- înregistrarea / deînregistrarea pe metodele de callback

```
private SomeEventBroadcastReceiver someEventBroadcastReceiver = new SomeEventBroadcastReceiver();
private IntentFilter intentFilter = new IntentFilter(SOME_ACTION);
@Override
protected void onResume() {
  super.onResume();
  registerReceiver(someEventBroadcastReceiver, intentFilter);
}
@Override
protected void onPause() {
  super.onPause();
  unregisterReceiver(someEventBroadcastReceiver);
}
```

Procesarea unei Intenții cu Difuzare


```
public class SomeEventBroadcastReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    // ...
  }
}
```

- metoda `onReceive()` este apelată în mod automat la momentul identificării unei intenții cu difuzare
  - executată pe firul de execuție principal al aplicației Android
  - trebuie să se termine în maxim 5 minute
- NotificationManager – serviciu de notificare responsabil cu transferul intenției

Tipuri Particulare de Intenții cu Difuzare

- **ordonate**
  - o intenție cu difuzare este procesată secvențial, de mai mulți ascultători
  - `sendOrderedBroadcast()`
  - pot specifica
    - o permisiune
    - o prioritate – ordinea în care este procesată intenția cu difuzare
- **persistente**
  - reținerea celei mai recente valori (nivelul de încărcare al bateriei, conectivitate la Internet)
  - `sendStickyBroadcast()`
  - trebuie să dețină permisiunea `android.permission.BROADCAST_STICKY`

Intenții cu Difuzare Native

- ACTION_BATTERY: LOW, OKAY, CONNECTED, DISCONNECTED, CHANGED (EXTRA_STATUS: BATTERY_STATUS_CHARGING, BATTERY_STATUS_FULL)
- ACTION_BOOT_COMPLETED
- ACTION_CAMERA_BUTTON
- ACTION_DATE_CHANGED / ACTION_TIME_CHANGED
- ACTION_DOCK_EVENT – EXTRA_DOCK_STATE (car, desk)
- ACTION_MEDIA_EJECT
- ACTION_MEDIA_MOUNTED, ACTION_MEDIA_UNMOUNTED
- ACTION_NEW_OUTGOING_CALL – EXTRA_PHONE_NUMBER
- ACTION_SCREEN_OFF, ACTION_SCREEN_ON
- ACTION_TIMEZONE_CHANGED

Fragmente

- **componentă atomică** a unei interfețe grafice
  - o activitate ⇔ unul sau mai multe fragmente
  - reutilizabilă, asigură flexibilitatea aplicației pe mai multe suprafețe de afișare
  - ciclu de viață propriu
  - partajează controalele grafice
- best practice pentru proiectarea și dezvoltarea interfețelor grafice (> API 11), disponibile în cadrul bibliotecilor de suport

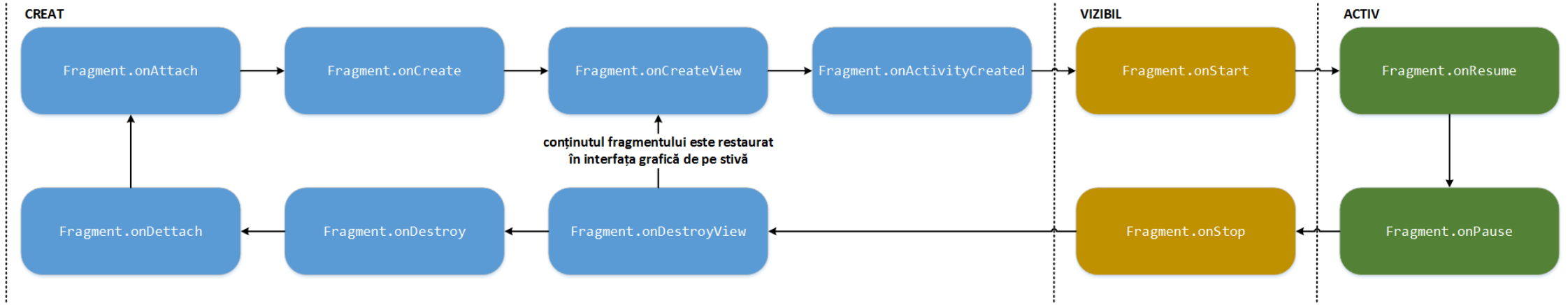
Structura unui Fragment

- interfața grafică descrisă într-un fișier XML din directorul `/res/layout`
- logica aplicației conținută într-o clasă `android.app.Fragment`
  - `onCreateView(LayoutInflater, ViewGroup, Bundle)` – încărcarea interfeței grafice
  - `inflate(int, ViewGroup, boolean)`
    - `resource` – referință către fișierul XML care conține descrierea interfeței grafice
    - `root` – container-ul din care face parte
    - `attachToRoot` – controalele interfeței grafice sunt atașate container-ului sau sunt preluate doar configurațiile de tip `LayoutParams`

Ciclul de Viață al unui Fragment

- `onAttach(Activity)` – fragmentul este atașat activității
  - obținerea unei referințe către context
  - alte operații de inițializare
- `onCreate(Bundle)`
- `onCreateView(LayoutInflater, ViewGroup, Bundle)` – utilizată pentru a încărca interfața grafică
- `onActivityCreated(Bundle)` – metoda `onCreate()` a activității a fost terminată, pot fi accesate elemente ale interfeței grafice
- `onStart()`
- `onResume()`
- `onPause()`
- `onStop()`
- `onDestroyView()` – utilizată pentru a descărca interfața grafică
- `onDestroy()`
- `onDetach()` – fragmentul este detașat activității

Ciclul de Viață al unui Fragment



Lucrul cu Fragmente

- **static**, folosind eticheta `<fragment>` în fișierul XML care conține structura interfeței grafice
- **dinamic**
  - se utilizează containere (de regulă, `FrameLayout`) cu diferite vizibilități
  - operațiile sunt realizate în cadrul unor tranzacții
    - asigurarea persistenței în cazul distrugerii / (re)creării activității
    - implementarea de tranziții la trecerea prin fragmente – `setTransition()`, `setFragmentAnimation()`
    - menținerea stărilor anterioare pe stivă
    - mai multe operații realizate atomic
    - operații: `add()`, `remove()`, `replace()`, `findFragmentById()`, `findFragmentByTag()`

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
fragmentTransaction.add(R.id.frame1, new Fragment1(), Constants.FRAGMENT1_TAG);
Fragment fragment2 = fragmentManager.findFragmentByTag(Constants.FRAGMENT2_TAG);
fragmentTransaction.remove(fragment2);
fragmentTransaction.addToBackStack(null);
fragmentTransaction.commit();
```

Interacțiunea dintre Fragmente

- folosind activitatea ca intermediar: `getActivity()` + `findViewById()`
  - asigură decuplarea fragmentelor
  - alternativă la obținerea unei referințe către celălalt fragment prin intermediul obiectelor `FragmentManager`, `FragmentTransaction`
- utilizând o interfață `OnEventProducedListener`
  - definește callback-ul de tratare al evenimentului `onEventProduced(Event)`
  - implemetată de clasa activitate, responsabilă pentru impactul asupra interfeței grafice