Introduction to Android Programming

IJS, November 2019

Dr Veljko Pejović
Veljko.Pejovic@fri.uni-lj.si

Dr David Jelenc
David.Jelenc@fri.uni-lj.si

Faculty of Computer and Information Science University of Ljubljana

University *of Ljubljana*
Faculty *of Computer and*
Information Science

REPUBLIKA SLOVENIJA
MINISTRSTVO ZA GOSPODARSKI
RAZVOJ IN TEHNOLOGIJO

The investment is co-financed by the Republic of Slovenia and the European Union through the European Regional Development Fund.
The TETRAMAX project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement number 761349.

Instructors

- Dr Veljko Pejović
  - PhD in resource-efficient wireless networks (UC Santa Barbara, USA)
  - Postdoc working on mobile sensing, human behaviour inference (Uni. Birmingham, UK)
  - Assistant professor at Faculty of Computer and Information Science, University of Ljubljana
  - Current projects and research interests:
    - Resource-efficient approximate mobile computing
    - Modelling user behaviour using mobile sensors
    - Analysing mobile wireless measurements
  - Veljko.Pejovic@fri.uni-lj.si

Instructors

- Dr David Jelenc
  - PhD in trust management (University of Ljubljana)
  - Teaching assistant at Faculty of Computer and Information Science, University of Ljubljana
  - David.Jelenc@fri.uni-lj.si

The World of Android

- The Android Platform
  - A mobile operating system + libraries + application frameworks + key apps
  - Based on Linux
  - Open source
  - Runs on a range of devices
- Market share ~ 85% worldwide
- Android SDK for creating apps
  - Lots of documentation
  - Huge community

Android Versions

API 21

API 22

API 23

API 24

API 25

API 26

API 27

API 28

API 29

Android 10

University *of Ljubljana*
Faculty *of Computer and Information Science*

Key Android Features

- Process management specifically tailored for battery-powered devices
  - When an app is not used, it gets suspended by Android
- Process management specifically tailored for low-memory devices
  - When the memory is low, suspended apps are terminated
- Support for direct manipulation interfaces
  - Touchscreen gestures, sensors, notifications
- Open ecosystem of applications
  - Support for developing and distributing Android apps

So, how do I start programming for Android?

Getting Started with Android Programming

- Android Software Development Kit (SDK)
  - Libraries
  - Debugger
  - Android device emulator
  - Communication between the SDK and the device via Android Debug Bridge (adb)
- Android Studio (IDE)
  - http://developer.android.com/studio
  - Code editor
  - Compilation, running, emulation control

Android Application Form

User Interface (UI)

- What the user sees and can interact with
- Written in XML
- Standard elements: Layouts, Buttons,
- TextViews, etc.
- In “res” folder

Interaction and navigation (with limited processing)

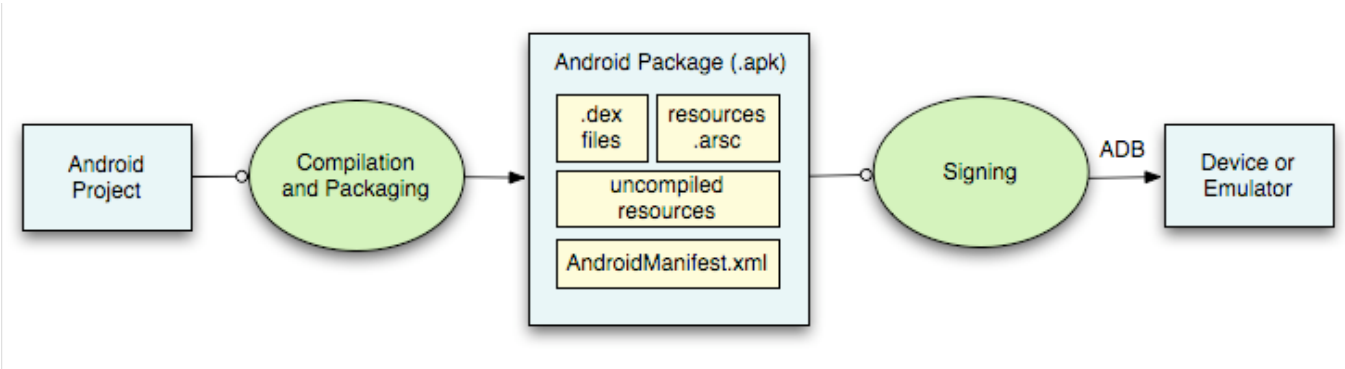
- What happens when a user interacts with the app?
- Written in Java or Kotlin
- In “src” folder

Background processing

- Long-running and heavy computation
- Data storage and network communication
- Written in Java or Kotlin
- In “src” folder

Android Application Content

- Application
  - A collection of components that are packaged together, can be instantiated and ran as needed
  - Non-compiled resources that the application needs
    - Images, Strings, Media files
- Building the application:



App Manifest File

- AndroidManifest.xml
- Declares essential information about your app
  - Name, version
  - Required permissions
  - Required SDKs
  - App components
    - Defines which component to start at launch

Basic Application Components

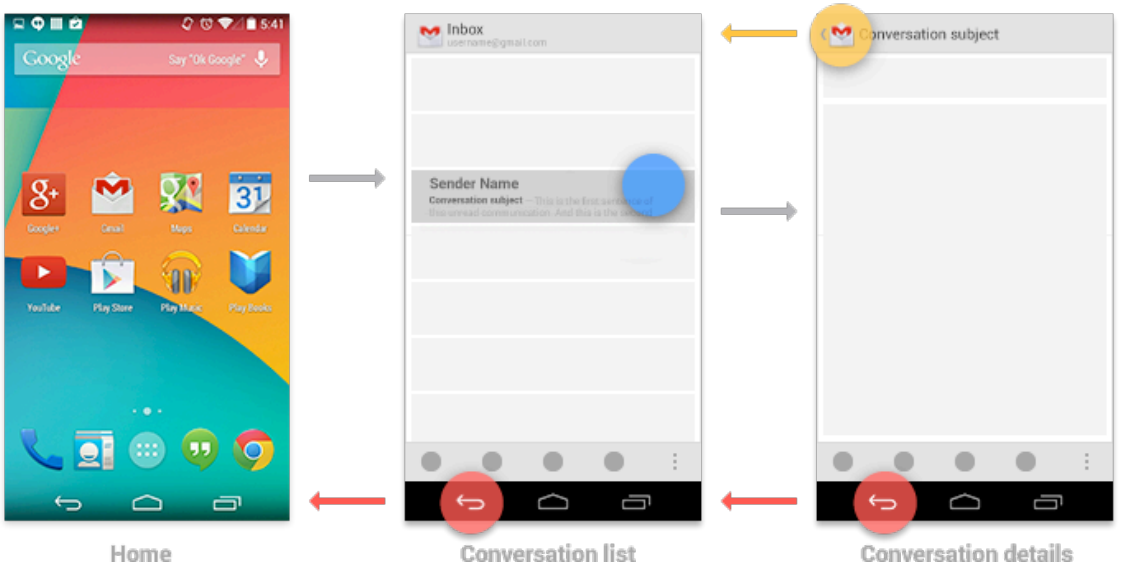
- Activity
  - Has a graphical user interface (GUI)
- Service
  - Performs background processing
- BroadcastReceiver
  - Subscribes to events of interest
- Intent
  - Communicates an intention to perform an action

Activity

- The primary class for managing user interaction
- One Activity usually implements a single focused task a user can do:
  - Log-in screen
  - Select a contact to write a message to
  - “Compose message” window
- Usually more than one Activity per application
- Activity interface itself is usually defined in a separate layout file, an XML file in the resources

Activity

- A user's interaction influences the activity that is going to be shown
  - Activity launching/parking via Intents in the code
  - Using "Up", "Back", "Home", "Menu/Recent apps" buttons, swipes



Activity Lifecycle

- Mobile devices have limited resources
  - Battery charge
  - Computing power
  - Screen real estate
- Activities are kept active only when a user can interact with them
- Activities are stopped in the background when not used
- Activities may be destroyed when the OS needs resources

Activity Lifecycle

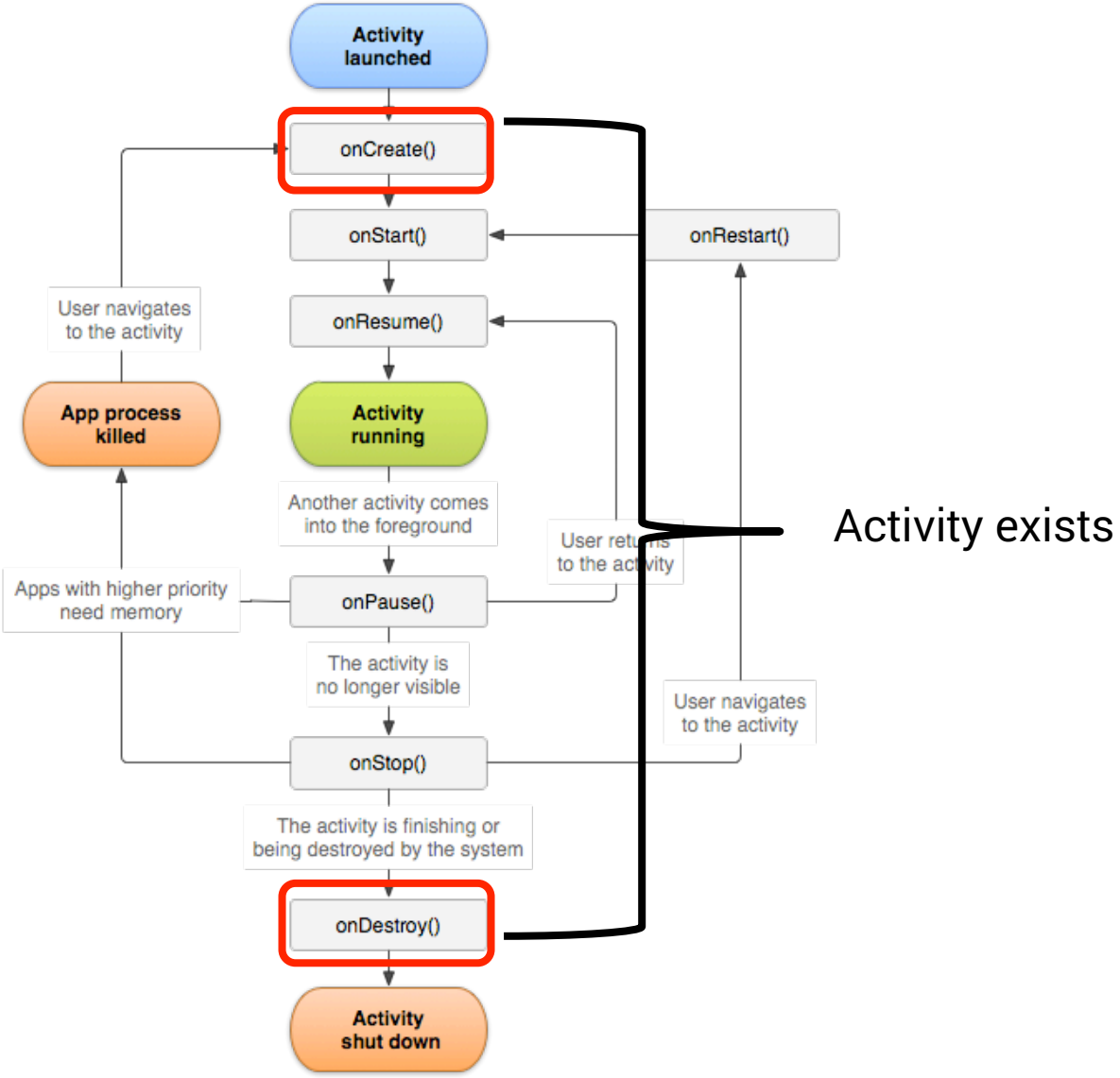
- Activity state:
  - Active/Running – in the foreground, visible, user interacting
  - Paused – lost focus but still visible, maintains state and member information
  - Stopped – completely obscured by another activity, retains state and member information, however, no longer visible; can be terminated by the OS when needed

Activity Lifecycle

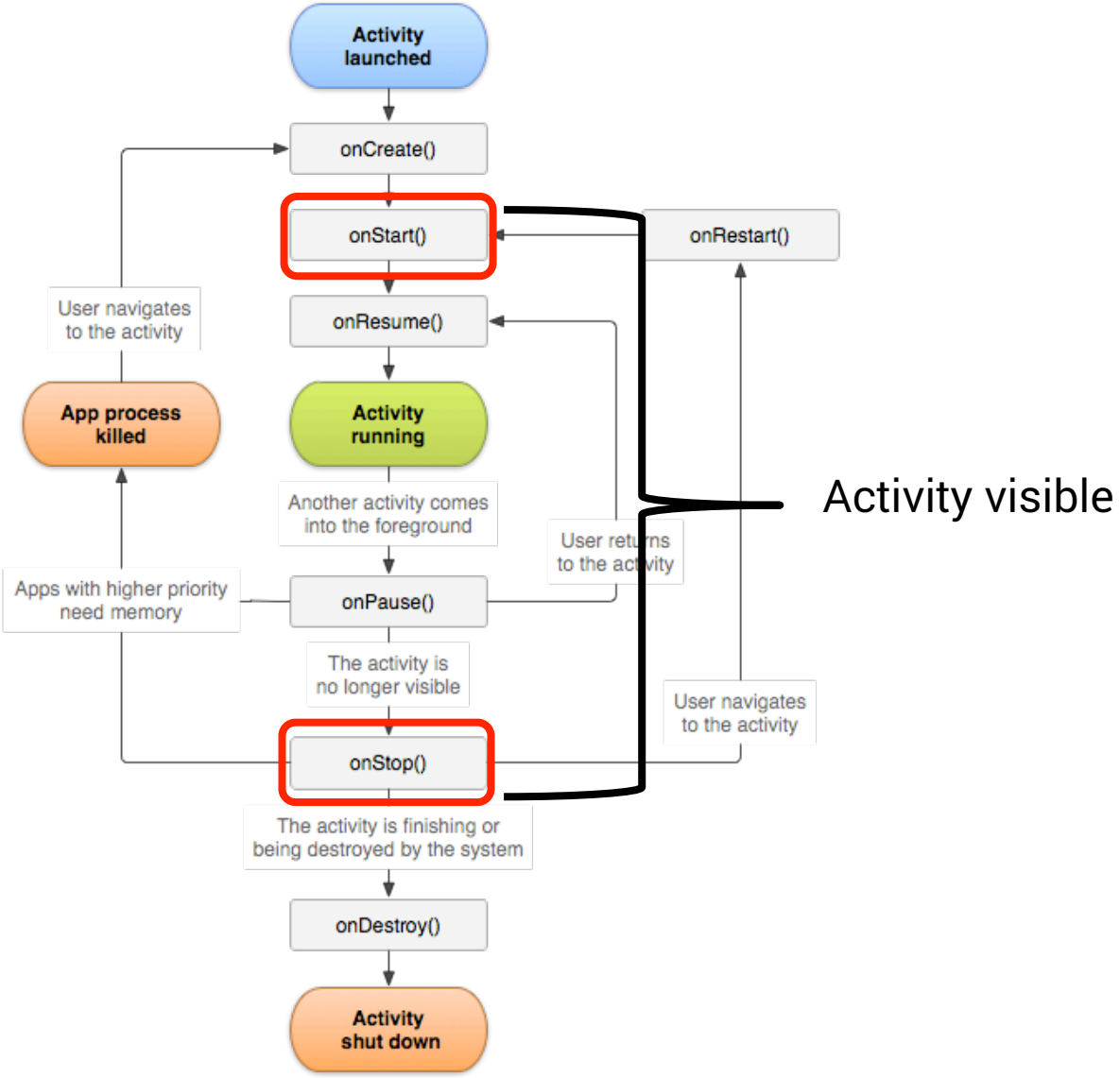
- An Activity moves through lifecycle state changes, usually as dictated by the user interaction
- Activity lifecycle state changes trigger the following activity methods:

```
protected void onCreate (Bundle savedInstanceState)
protected void onStart()
protected void onResume()
protected void onPause()
protected void onRestart()
protected void onStop()
protected void onDestroy()
```

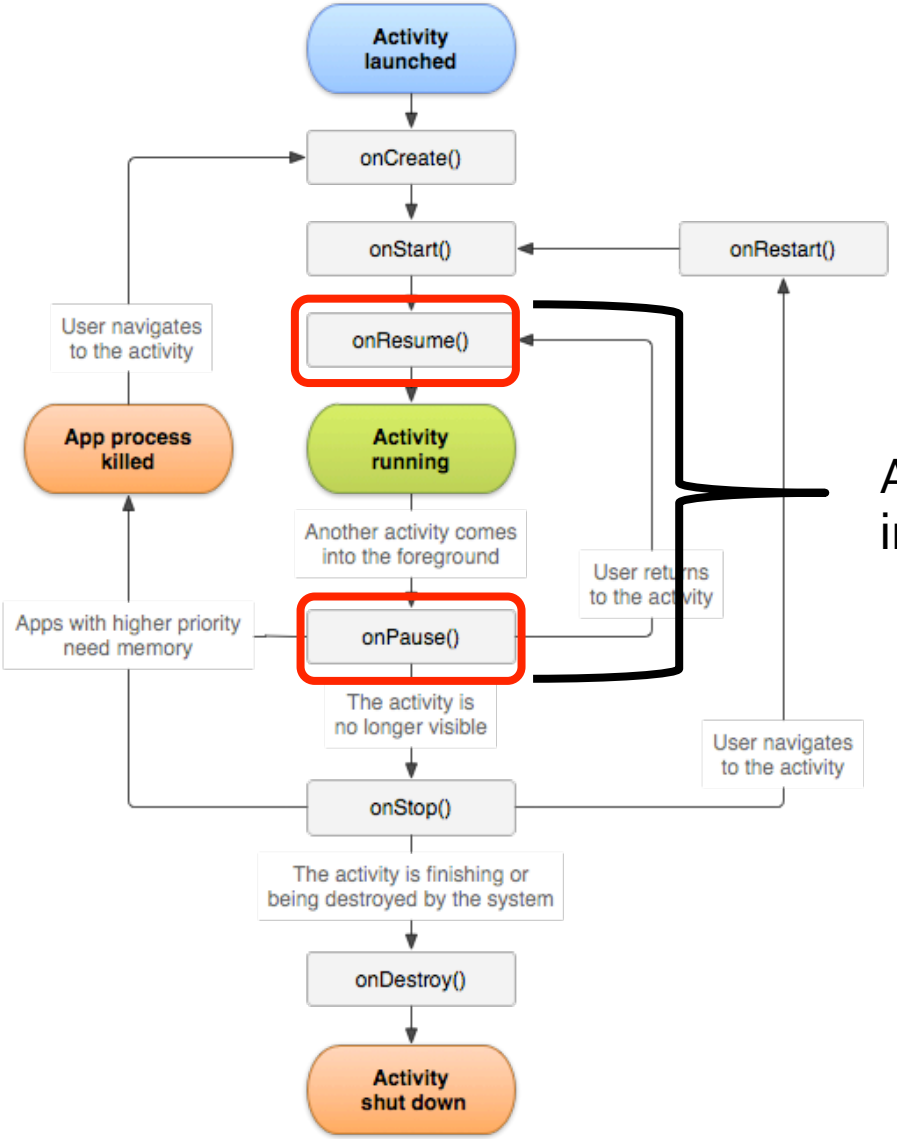
Activity Lifecycle



Activity Lifecycle



Activity Lifecycle



Activity visible and in foreground

onCreate()

- Called when the activity is first created
- Sets up the initial state:
  - Create and configure views
  - Set the Activity's content view, i.e. instruct the Activity to show something to the user
  - Bind data to lists
  - super.onCreate() – hides some complex code that must be called in order to instantiate the Activity properly
- onCreate() also gets a Bundle with the Activity's previous state

onStart()

- Called when the activity is becoming visible
- Setup state relevant for visible-only behaviour, for example:
  - Register certain BroadcastReceivers
- Load persistent application state

onRestart()

- Called if the activity is becoming visible, after being stopped
- Perform special processing needed only after having been stopped

onResume()

- Called when the activity is visible and is about to start interacting with the user
- Start foreground-only activities
  - For example, get user location and show it on the map

onPause()

- Called when the Activity loses focus, and another activity is about to start
- Use it to commit unsaved changes to persistent data, stop animations, CPU-intensive processing
- Processing in this method should be done quickly, because the next activity will not start until this method returns
  - Alternatively, run a parallel thread from onPause()

onStop()

- Called when the Activity is no longer visible
  - Another Activity is being started, an existing one is being brought in front of this one, or this one is being destroyed
- Note: may not be called if Android kills your application

onDestroy()

- Called when the Activity is about to get destroyed
  - Happens when `finish()` is called
  - Happens when the OS calls it
- Use it to release resources such as Threads that area associated with the Activity
- Note: may not be called if Android kills your application

Starting Activities

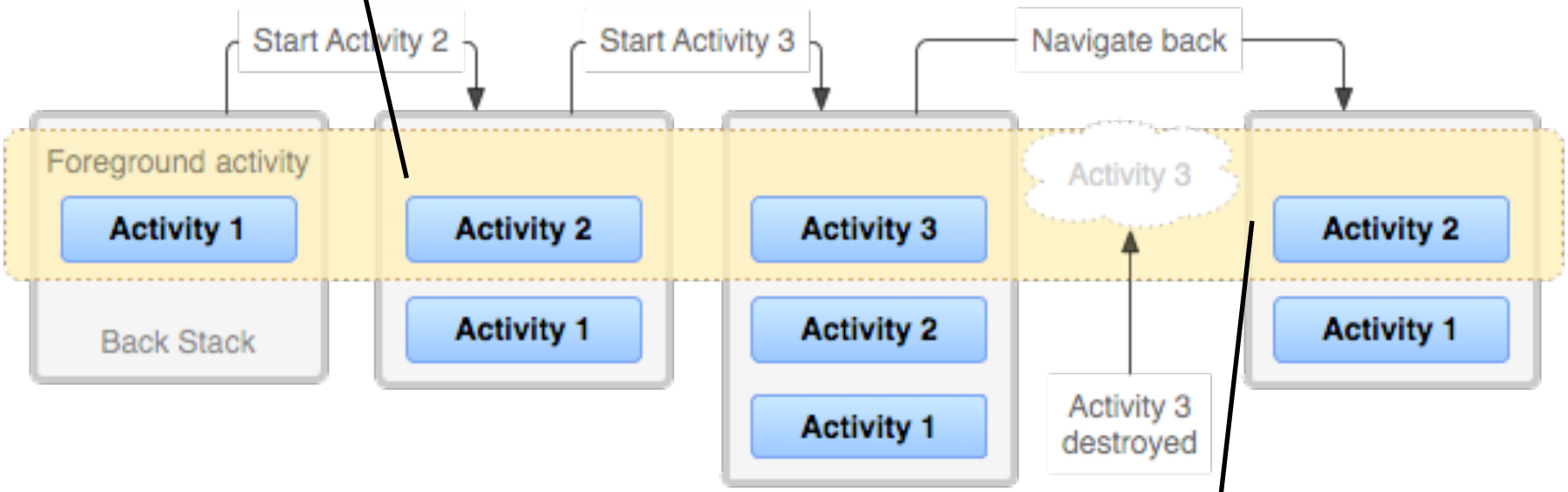
- Create an Intent specifying the Activity to start
- Pass the Intent to one of the following methods:
  - `startActivity()`
    - launches the Activity described by the Intent
  - `startActivityForResult()`
    - launches the Activity described by the Intent and expects a result that will be returned via onActivityResult
    - the called activity can set result via setResult() method

Task

- A task is a collection of Activities that users interact with when performing a certain job
- The Activities need not be from the same application (although usually they are)
- Backstack: the activities are arranged in a stack in the order in which each activity is opened
  - When launched the activity goes on top of the backstack
  - When destroyed it is popped of the backstack

Backstack

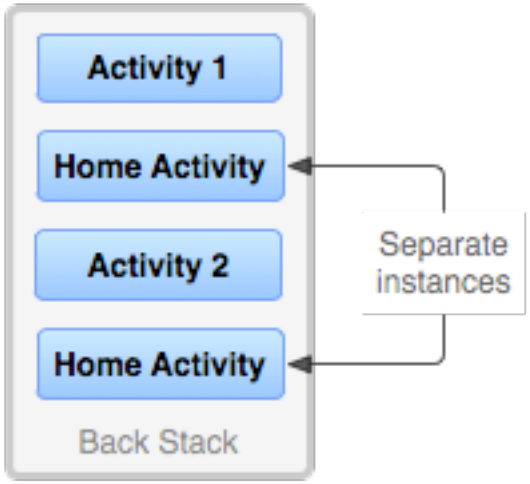
A new activity (Activity 2) is created and started, the old one (Activity 1) is stopped

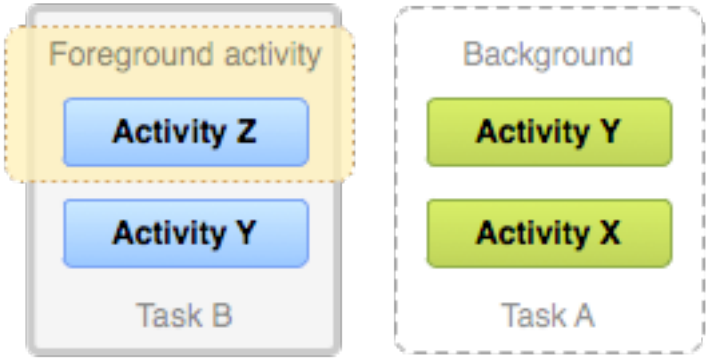


Activity 3 destroyed when the user clicked BACK, Activity 2 is started

Backstack

- More than one instance of an Activity can be on the backstack
  - This behaviour can be changed via Intent options or in the Manifest file
- When HOME is pressed, the current activity is stopped, its task goes into the background.
- If the user later resumes the task, the activity at the top of the stack is started





Intent

- A data structure representing:
  - An operation to be performed or
  - An event that has occurred
- Intents serve as a glue between components
  - Constructed by a component that wants some work to be done
  - Received by a component that can perform that work
- Hold an abstract description of an action to be performed
  - Take a photo, pick a contact, show a webpage

Intent Fields

- Action
- Data
- Category
- Type
- Component
- Extras

Explicit Intents specify the component to be launched (Action, data, etc.) become irrelevant.

Implicit Intents do not specify the component; instead, they must include enough information for the system to determine which of the available components is best to run for that intent. PackageManager is queried to find the right component.

Service

- Activities run on the UI (main) thread and have a UI attached (layout)
  - Processing-heavy functions on the main thread impact the responsiveness
- Services can run on either the Main or separate threads and do not have a UI attached
  - Run outside UI, for long-running operations
- Services are often more convenient than custom Threads for tasks than need to be “independent” and run even when the Activity is destroyed

Background and Foreground Service

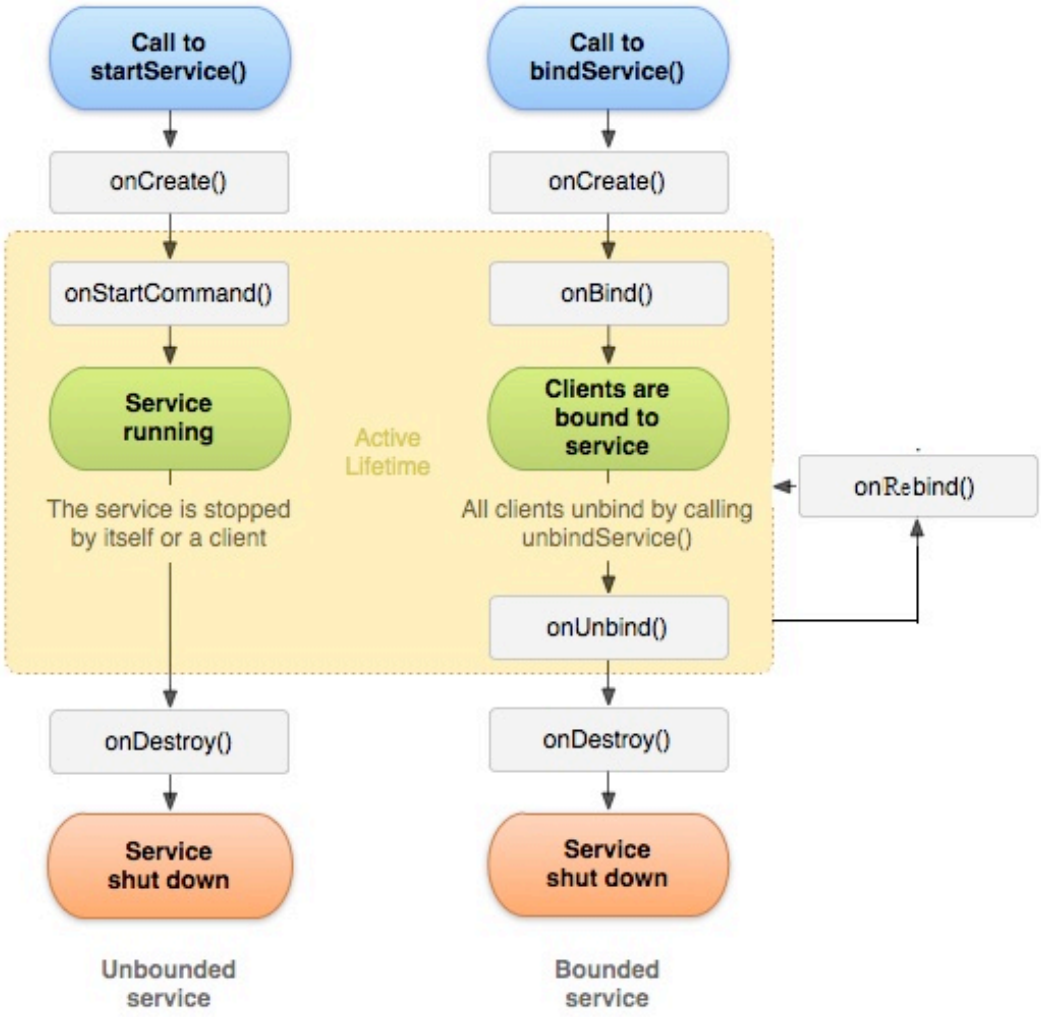
- Background Service
  - For actions that do not have to be noticed by the user (e.g. sensing a user's physical activity)
- Foreground Service
  - For actions that the user needs be aware of and should the control of (e.g. a music player app)
  - A foreground service must show a notification in the notification bar

Starting/Stopping a Service

- Services can be created:
  - Explicitly using Context.startService()
  - Implicitly, if not already running, when a client requests connection to a Service via Context.bindService()
- Services can be stopped:
  - From within the Service with stopSelf()
  - From another component with Context.stopService()

Services

- Multiple startService calls do not nest – you only have one service; however, onStartCommand() will be called repeatedly
- Service will be stopped only once with Context.stopService() or stopSelf()



Services – Bound

- Bound Services – like servers in a client-server paradigm
- Services started through binding, do not call onStartCommand()
- Return IBinder object from onBind(Intent) so that connected clients can call the Service
- The service remains running as long as the connection is established

Broadcast

- Messages sent from other components of your app, other apps or from the Android system
- Messages are wrapped in Intents

```
Intent intent = new Intent();
intent.setAction(ACTION);
intent.putExtra(STOP_SERVICE_BROADCAST_KEY, RQS_STOP_SERVICE);
sendBroadcast(intent);
```

- Send broadcasts
  - System sends certain broadcasts when an event happens, e.g. ACTION_BOOT_COMPLETED
  - Send custom broadcasts via sendBroadcast()

Broadcast

- Broadcasts are captured in an app/component if a BroadcastReceiver is registered in the code:
  - Create a BroadcastReceiver and impl. onReceive()

```
public class NotifyServiceReceiver extends BroadcastReceiver{

    @Override
    public void onReceive(Context arg0, Intent arg1) {
                …
        }
    }
}
```

  - Register for receiving certain kinds of Intents

```
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(ACTION);
registerReceiver(notifyServiceReceiver, intentFilter);
```

Broadcast

- Broadcasts are captured in an app/component if a BroadcastReceiver is registered in AndroidManifest.XML and onReceive() is implemented in the code:

```
<receiver android:name=".MyBroadcastReceiver" android:exported="true">
 <intent-filter>
   <action android:name="android.intent.action.BOOT_COMPLETED"/>
   <action android:name="android.intent.action.INPUT_METHOD_CHANGED"/>
 </intent-filter>
</receiver>
```

```
public class MyBroadcastReceiver extends BroadcastReceiver {
     @Override
     public void onReceive(Context context, Intent intent) {
….
```

BroadcastReceiver

- Receive events announced by other components
- Events announced via Intents
  - Not the same Intent as the one starting an Activity: this one remains in the background
- Events can be announced within your app or publicly to every app on the phone
  - Announce via sendBroadcast()
- Events captured if the receiver is registered:
  - onReceiverRegistered() and then onReceive()

Practical Lab: Mobile Music Player

University *of Ljubljana*
Faculty *of Computer and Information Science*

Conclusions

- Mobile app dev ≠ desktop app dev
  - OS can kill your app
  - Limited resources, especially energy
  - Dynamic ecosystem thanks to app stores
    - You have only one chance to capture the users!
- UI, interaction handling, and background processing are usually considered separately
- Basic components include Activity, Service, Intent, and BroadcastReceiver
  - Design your app around them

Thank you!

Dr Veljko Pejović
Veljko.Pejovic@fri.uni-lj.si

Dr David Jelenc
David.Jelenc@fri.uni-lj.si

Faculty of Computer and Information Science University of Ljubljana

University *of Ljubljana*
Faculty *of Computer and*
Information Science

REPUBLIKA SLOVENIJA
MINISTRSTVO ZA GOSPODARSKI
RAZVOJ IN TEHNOLOGIJO

The investment is co-financed by the Republic of Slovenia and the European Union through the European Regional Development Fund.
The TETRAMAX project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement number 761349.