Platform-Based Development: Android Programming – Architecture

FTN, Novi Sad, Mini Course,
June 2019

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

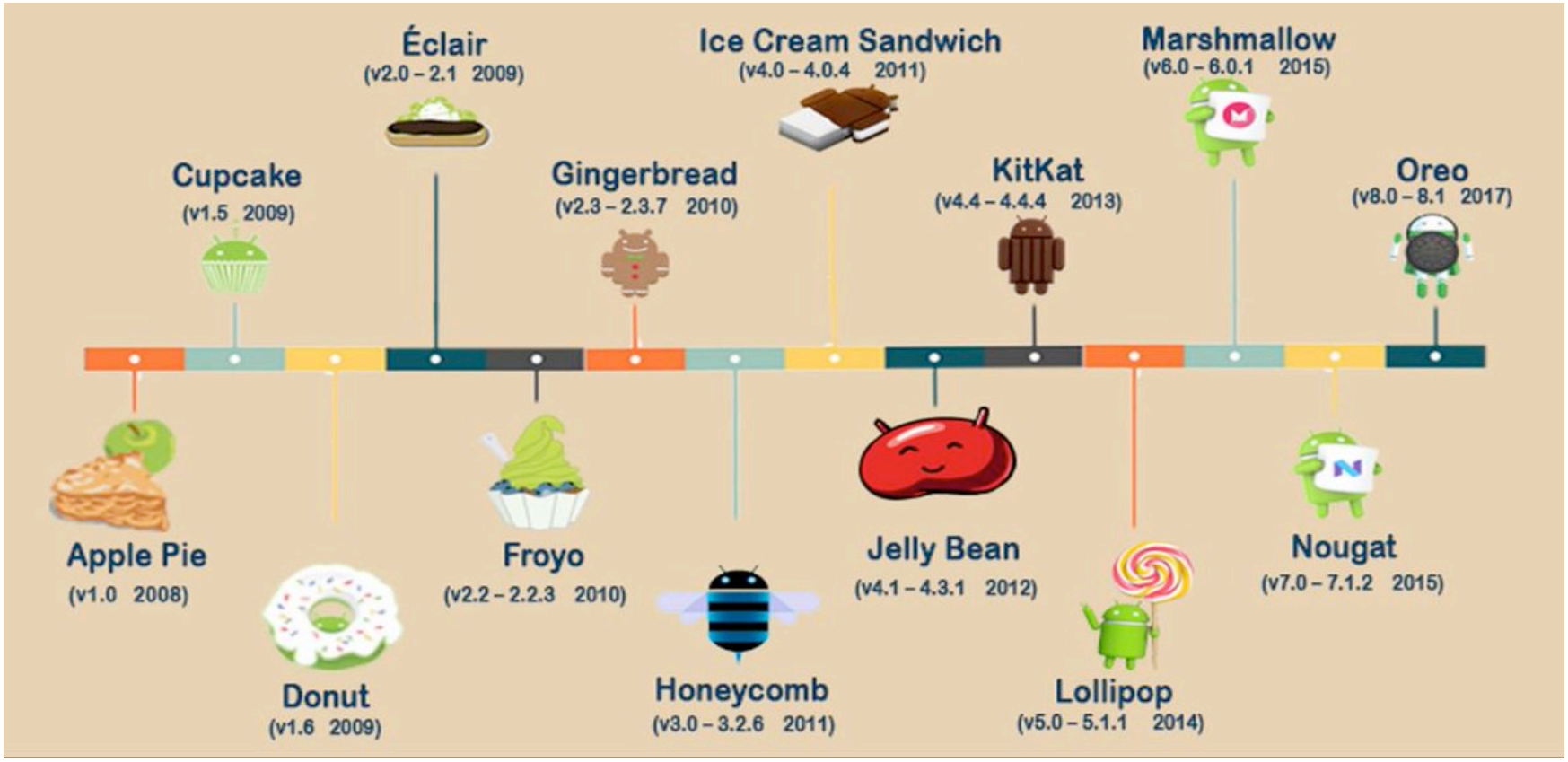
University *of Ljubljana*
Faculty *of Computer and Information Science*

Partly based on “Programming Handheld Systems”,
Adam Porter, University of Maryland

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
    - Some with OEM versions
- Market share ~ 75% worldwide
- Android SDK for creating apps
  - Lots of documentation
  - Huge community

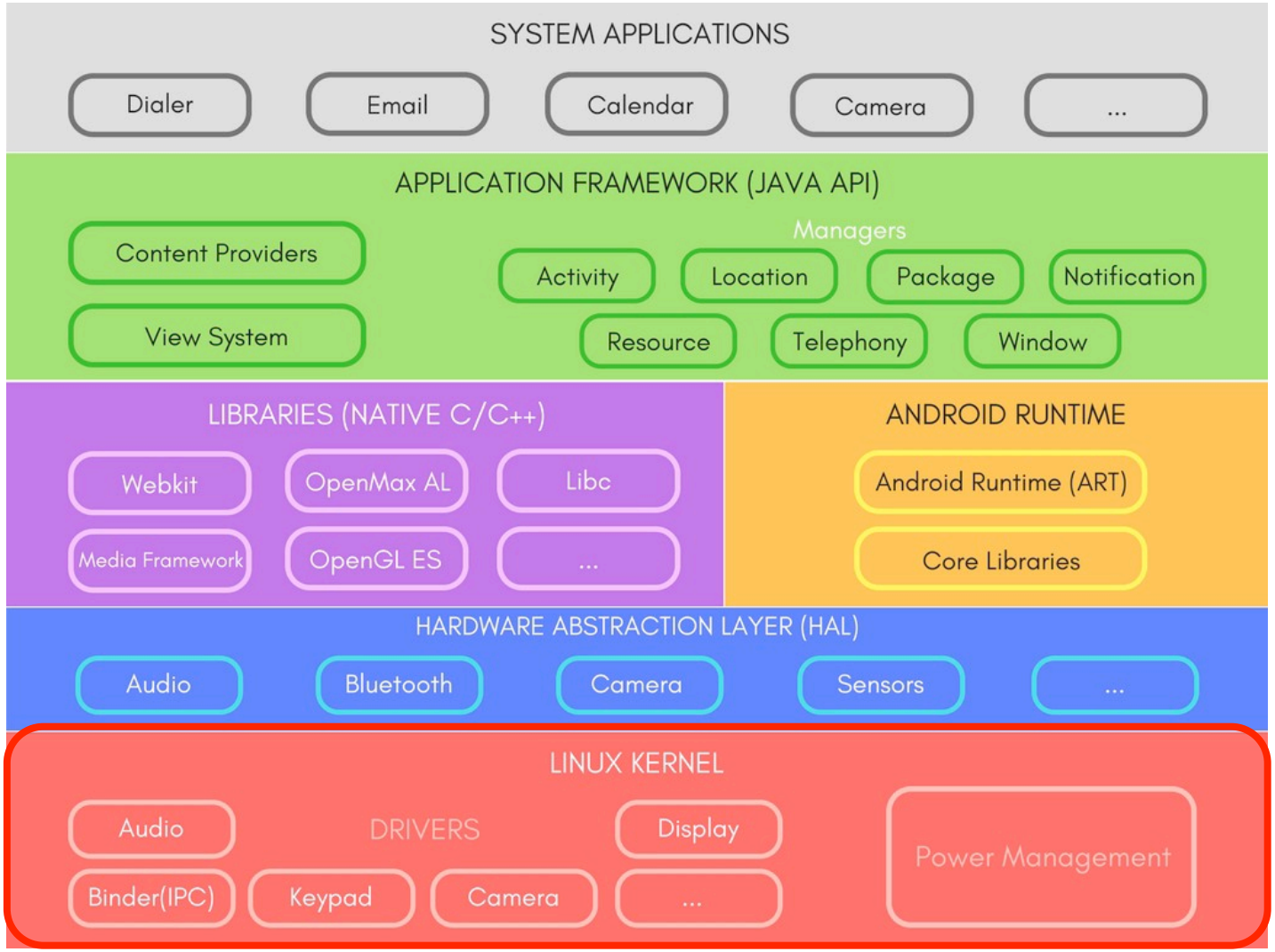
Android Versions



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

Android Architecture



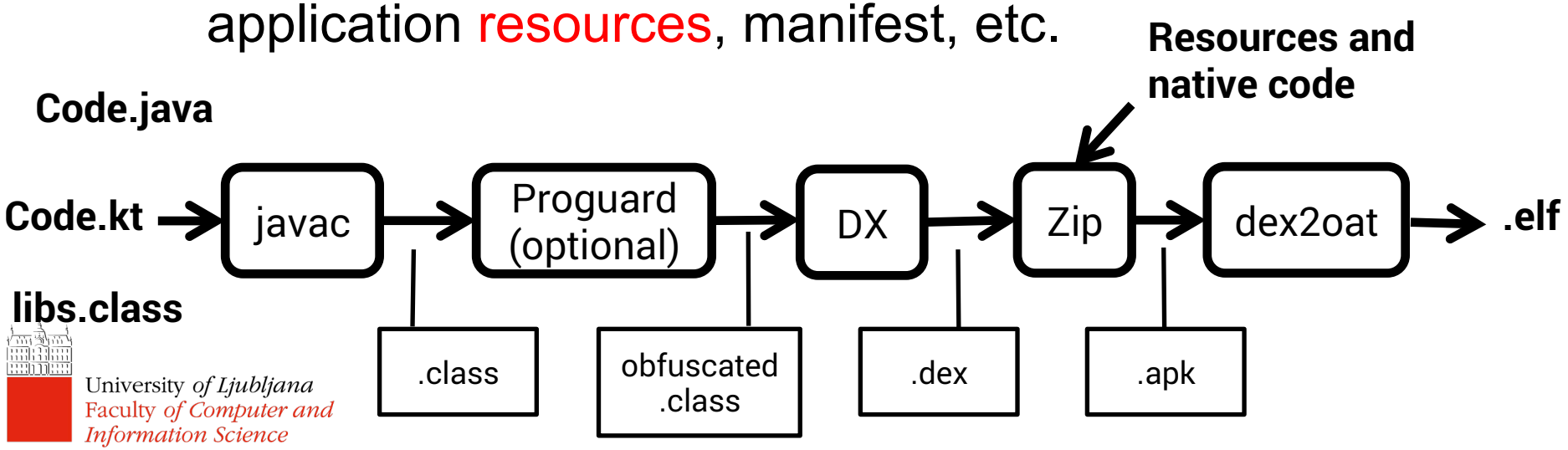
Android Runtime

- Android core libraries
  - Besides standard Java libraries for tasks such as file handling, Strings, etc., Android includes specific libraries for the mobile environment
  - basic java classes - java.*, javax.*
  - app lifecycle, db management - android.*
  - Internet/Web services - org. *
  - Unit testing - junit.*
- Process virtual machine (VM):
  - Dalvik (until Android 4.4 KitKat)
  - Android Runtime – ART (starting with 5.0 Lollypop)

Mostly wrap native libraries

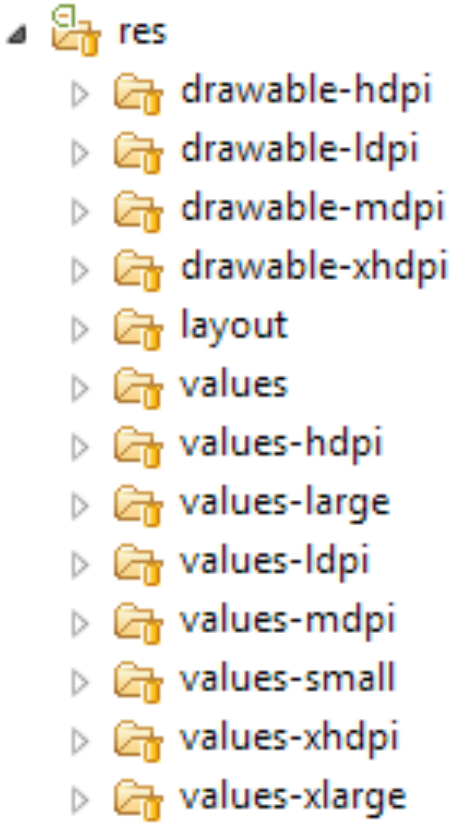
Android Runtime

- Compilation and workflow (with ART)
  - App written in Java or Kotlin
  - Compiled to Java bytecode files (i.e. .class files)
  - DX converts Java bytecode files to a single DEX bytecode file (.dex file) optimised for space
  - .apk file is generated with the dex file and all the application resources, manifest, etc.



Resources

- Android application resources:
  - Non-compiled static content of your app
  - See “res” folder created by Android Studio
  - Examples:
    - String values
    - Bitmaps (e.g. backgrounds, icons)
    - Layout files
    - Styles’ definitions
  - Programmatically accessible via the automatically-generated R file




```
String mystring = getResources()
                  .getString(R.string.mystring);
```

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
- ContentProvider
  - Encapsulates and exposes data

What is an Application?

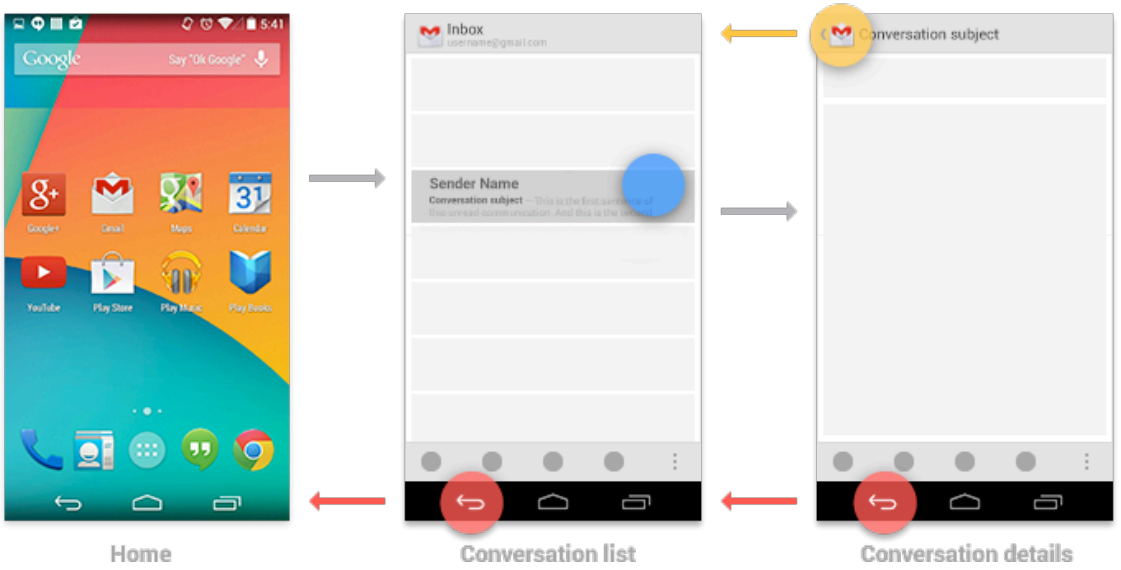
- Application
  - A collection of components that are packaged together, can be instantiated and ran as needed
  - Note that there is also Application class in Android, however, usually there is no need to use it
- .apk – is what we usually refer to when we say “application”

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

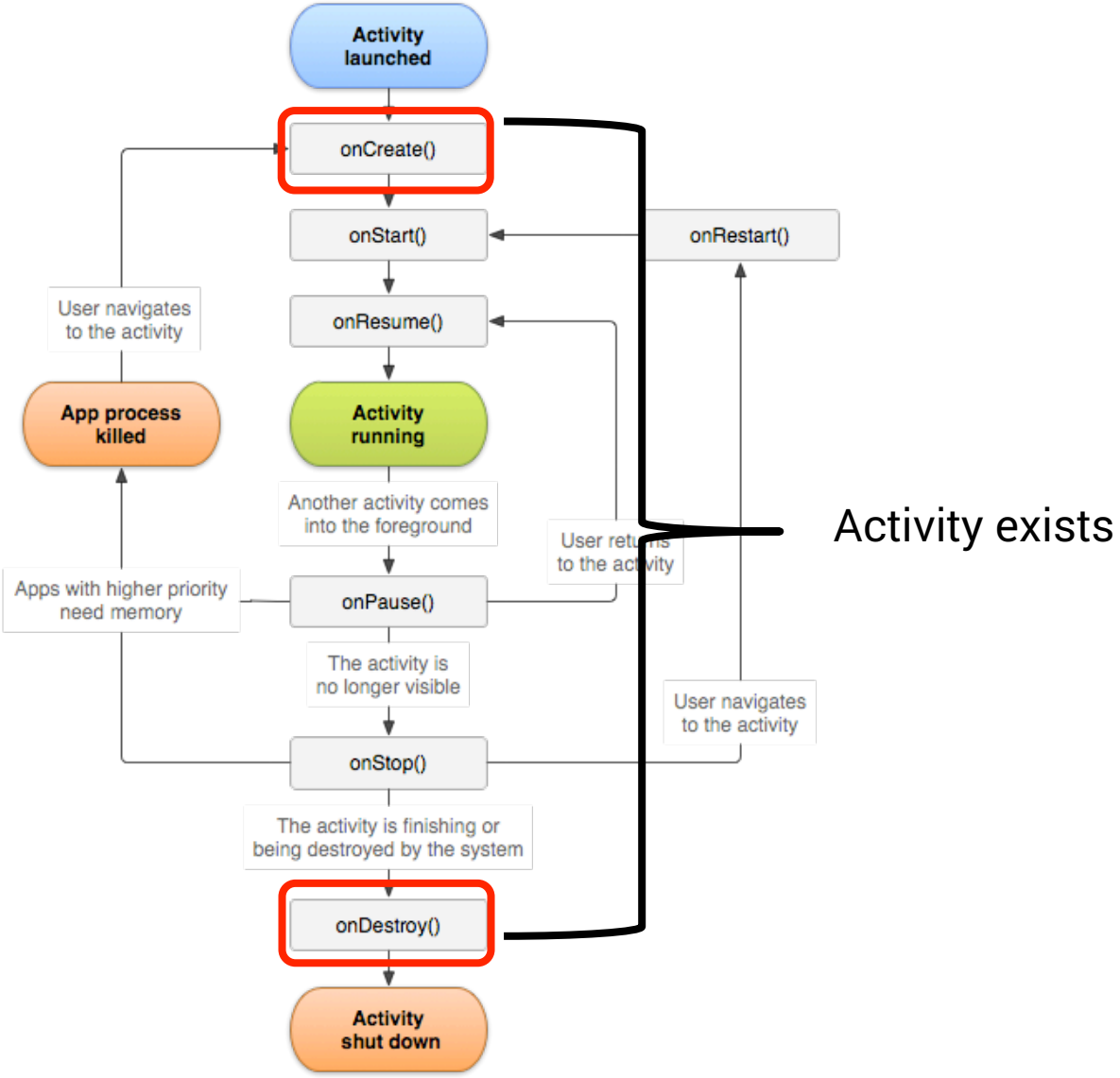
- Activity state:
  - Active/Running – in the foreground, visible, user interacting
  - Paused – lost focus but still visible, maintains state and member information
  - Stopped – completely obscured by another activity, retains state and member information, however, no longer visible; can be terminated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

Activity Lifecycle

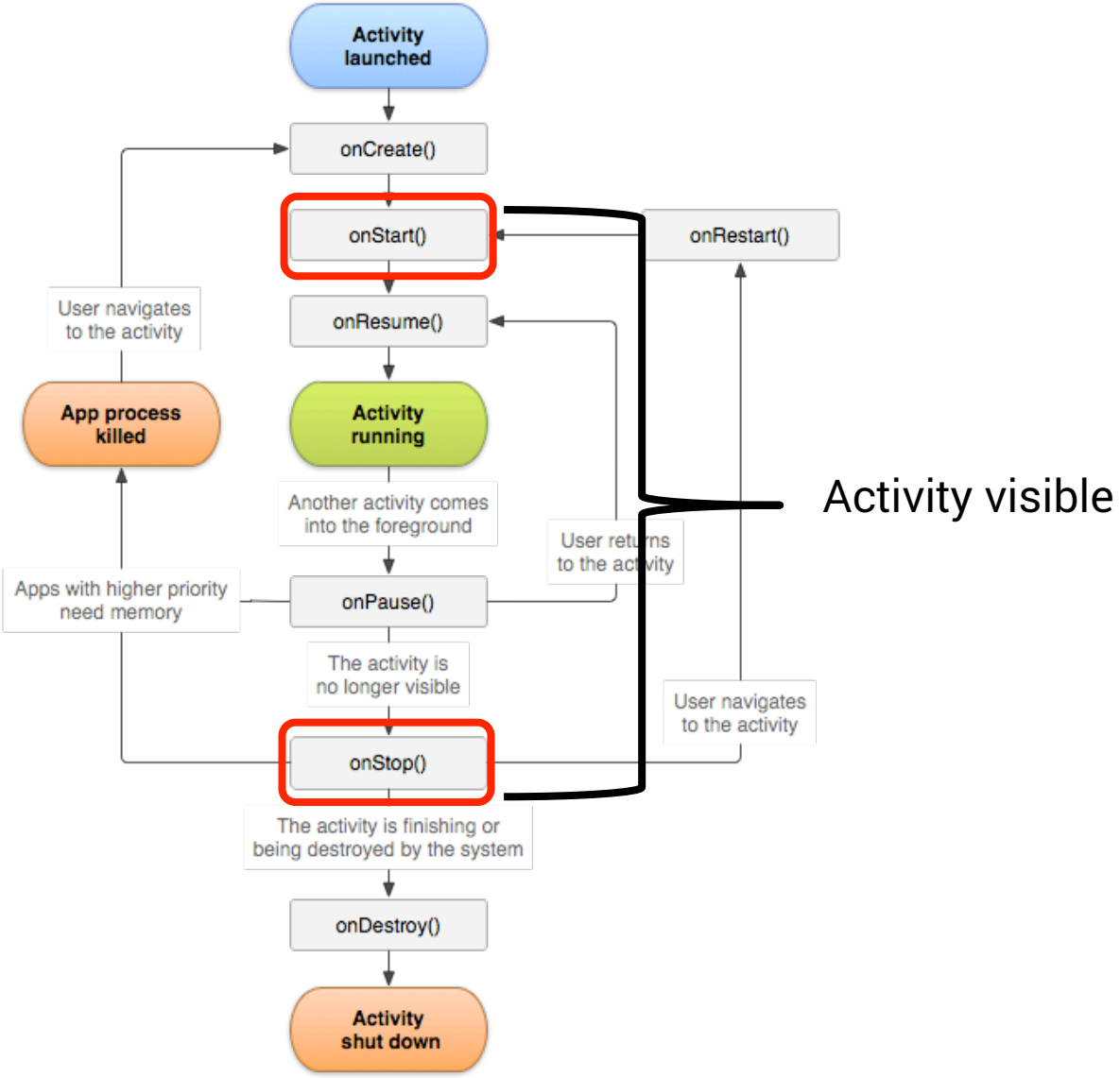
- An Activity moves through lifecycle state changes, usually as dictated by the user interaction
- Activity lifecycle state changes trigger the following activity methods:

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

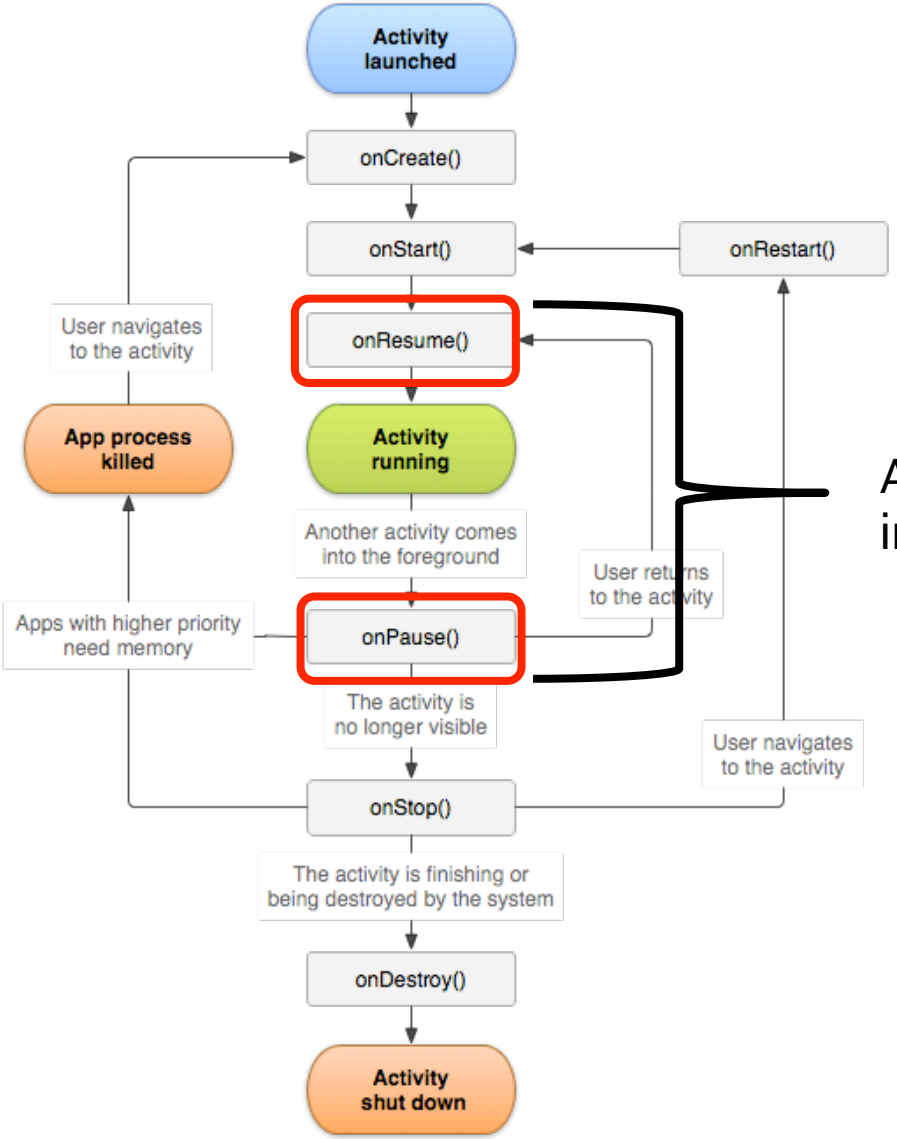
Activity Lifecycle



Activity Lifecycle



Activity Lifecycle



Activity visible and in foreground

Starting Activities

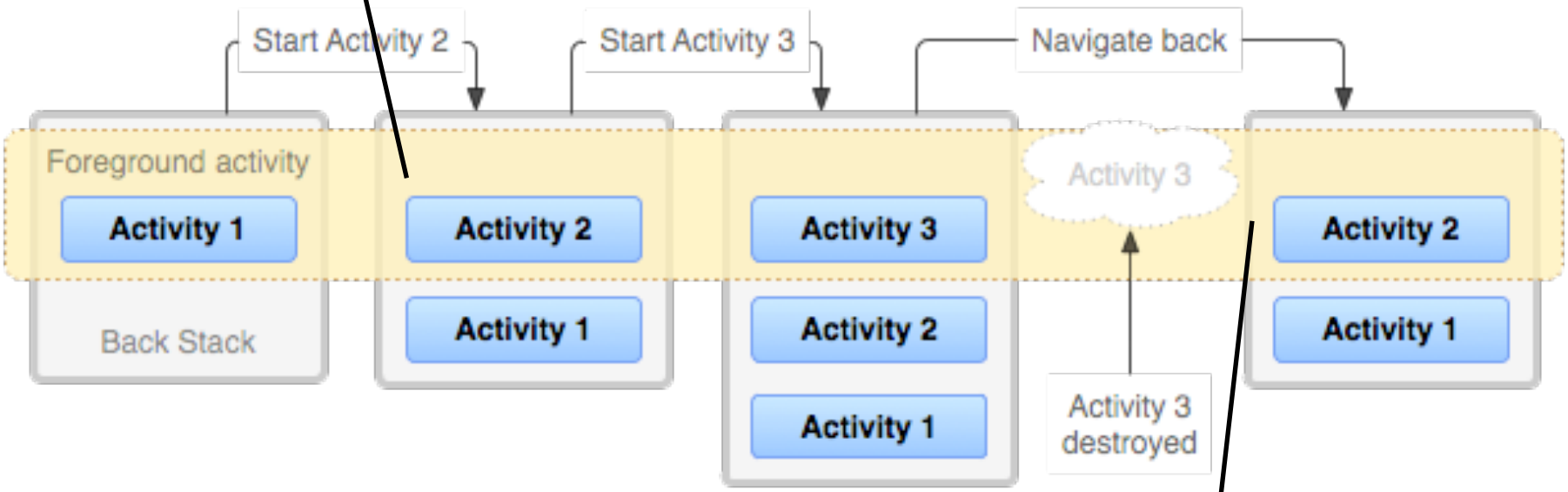
- Create an Intent specifying the Activity to start
- Pass the Intent to one of the following methods:
  - `startActivity()`
    - launches the Activity described by the Intent
  - `startActivityForResult()`
    - launches the Activity described by the Intent and expects a result that will be returned 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

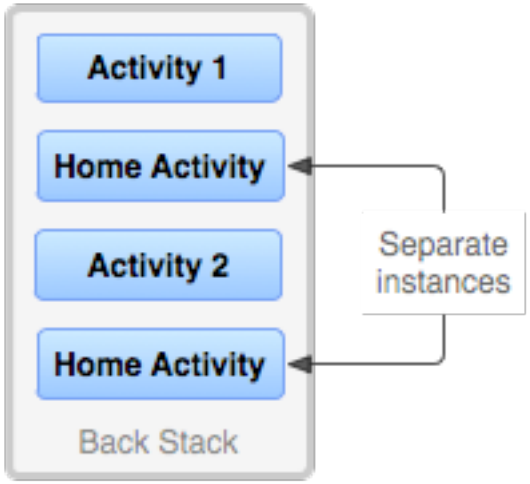
A new activity (Activity 2) is created and started, the old one (Activity 1) is stopped

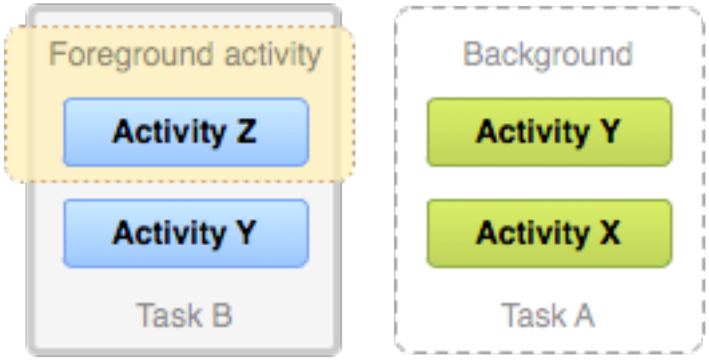


Activity 3 destroyed when the user clicked BACK, Activity 2 is started

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 activities
  - Constructed by a component that wants some work to be done
  - Received by an Activity that can perform that work
- Hold an abstract description of an action to be performed
  - Take a photo, pick a contact, show a webpage

SharedPreference

- Preserve a small amount of primitive type (int, float, String, Boolean) data on a device
  - Data are saved as key-value pairs
  - Should be read/written by your app only
  - Stored for as long as the app is installed on a device
- Common use:
  - User preferences – username, customisations, such as preferred WiFi AP, preferred theme, etc.
  - Variables for conditional app execution
    - When the app is launched for the first time set “launched” to True; next time, check if “launched” was set or not

http://developer.android.com/reference/android/content/SharedPreferences.html

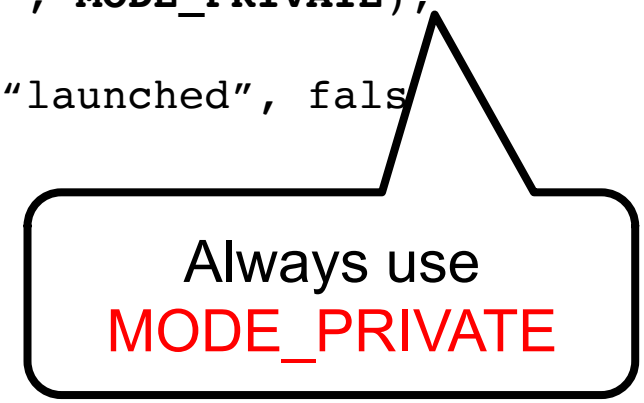
Accessing SharedPreferences

- Reading

```
SharedPreferences settings = getApplicationContext()
        .getSharedPreferences("preferences", MODE_PRIVATE);

boolean wasLaunched = settings.getBoolean("launched", fals
```

- Need to know the type of data:
  - getBoolean()
  - getString()
  - getAll() returns a Map of key-value pairs



Accessing SharedPreferences

- Writing

```
SharedPreferences settings = getApplicationContext()
        .getSharedPreferences("preferences", MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean("launched", true);
editor.commit();
```

- Different put methods for different data types
- Don't forget to save changes by calling
  - editor.commit() – synchronous (avoid calling on the main thread) or
  - editor.apply() – changes the in-memory object immediately, but writes to disk asynchronously