Platform-Based Development: Android Programming – Activity, Lifecycle, Intents

BS UNI studies, Spring 2019/2020

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

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?

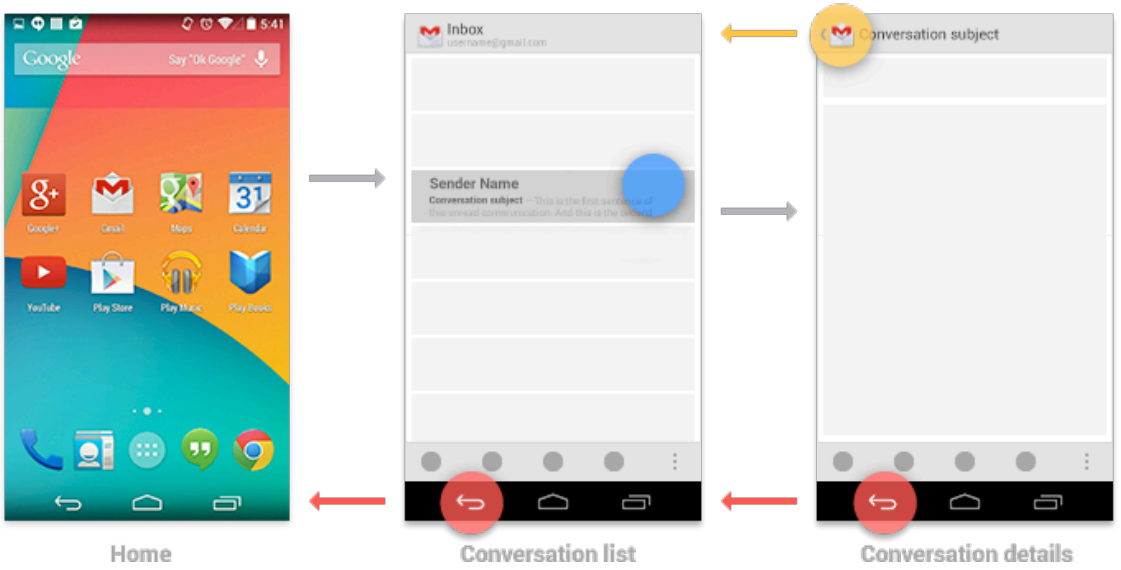
- Application
  - A collection of components that are packaged together, can be instantiated and ran as needed
  - Note that there is also Application class in Android, however, usually there is no need to use it
- .apk – is what we usually refer to when we say “application”

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

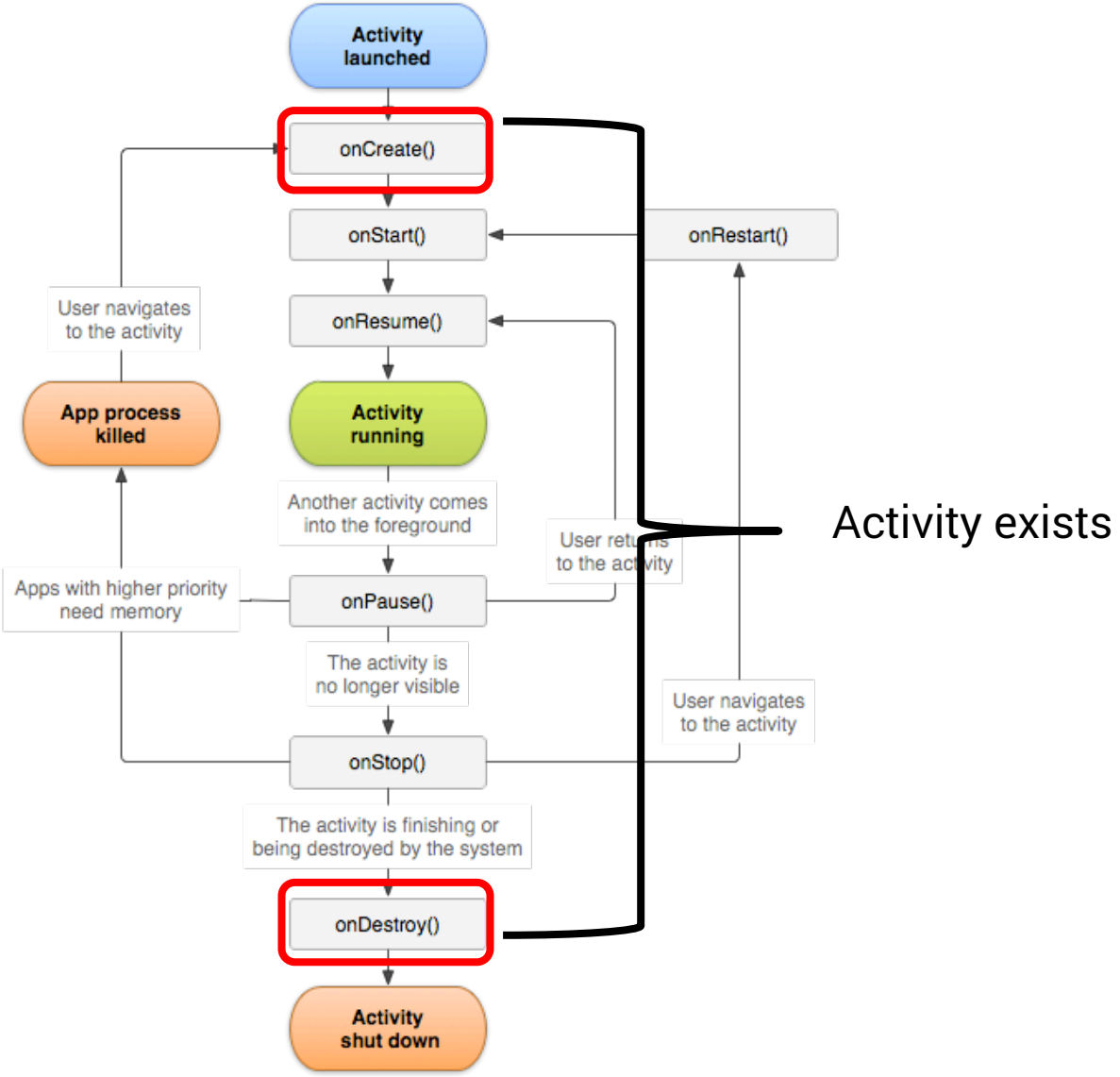
- Activity state:
  - Active/Running – in the foreground, visible, user interacting
  - Paused – lost focus but still visible, maintains state and member information
  - Stopped – completely obscured by another activity, retains state and member information, however, no longer visible; can be terminated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

Activity Lifecycle

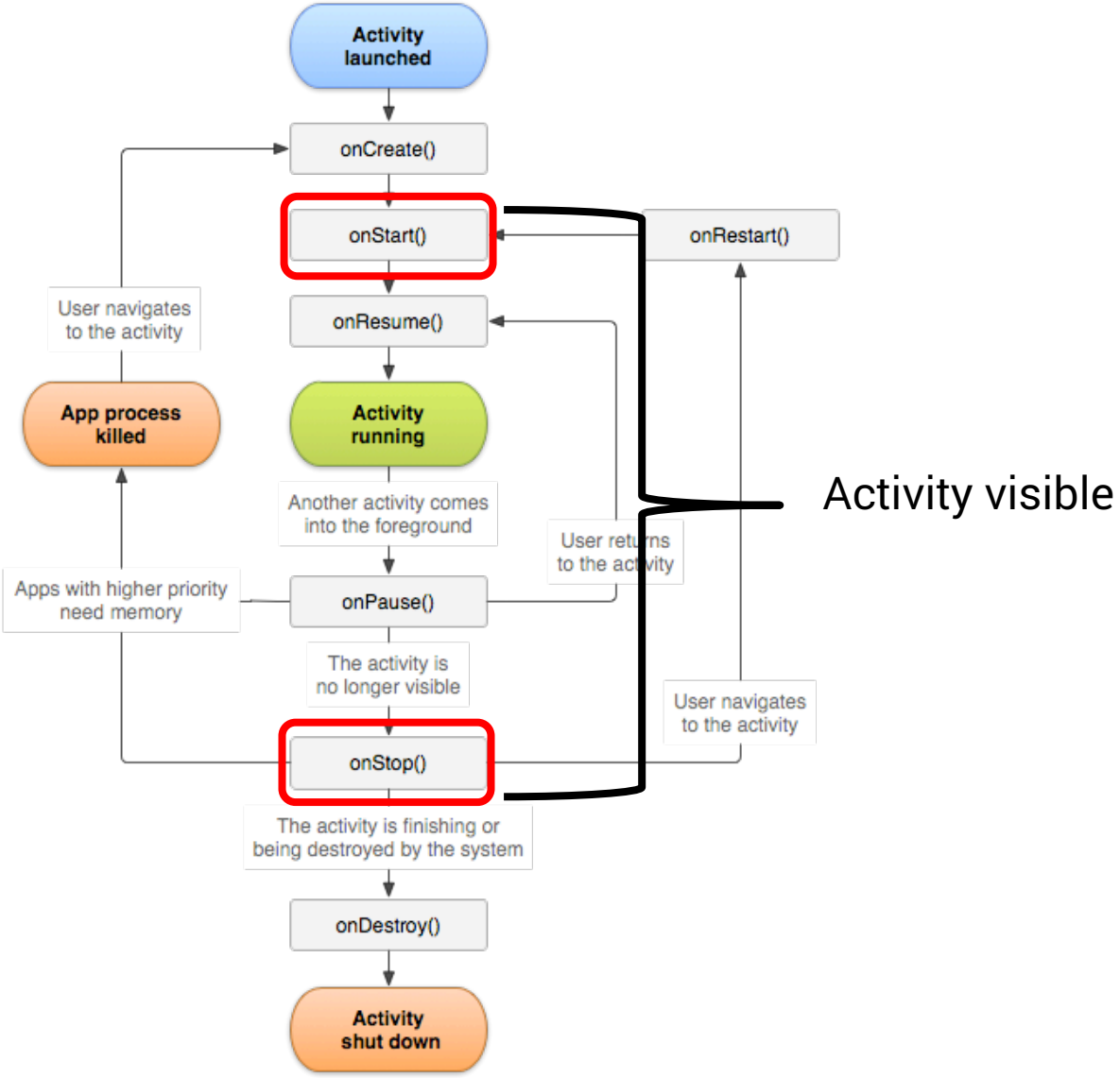
- An Activity moves through lifecycle state changes, usually as dictated by the user interaction
- Activity lifecycle state changes trigger the following activity methods:

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

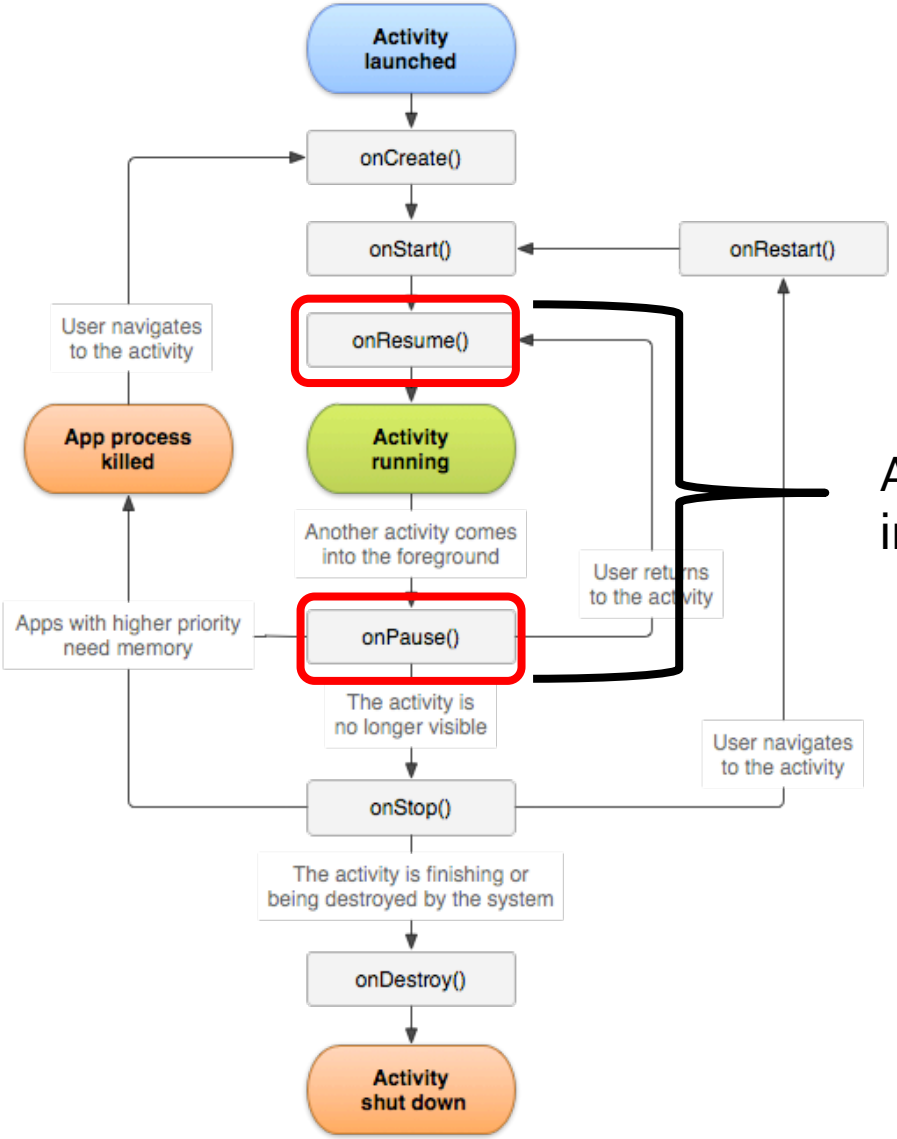
Activity Lifecycle



Activity Lifecycle



Activity Lifecycle



Activity visible and in foreground

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
- Will be followed by onStart() and onResume()

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
- Release resources that are not needed while the activity is not visible
- Perform CPU-heavy shutdown operations
- Your activity still exists, but is not connected to Window Manager
  - Returning (via onRestart()) will set the Activity state back to the last seen value

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

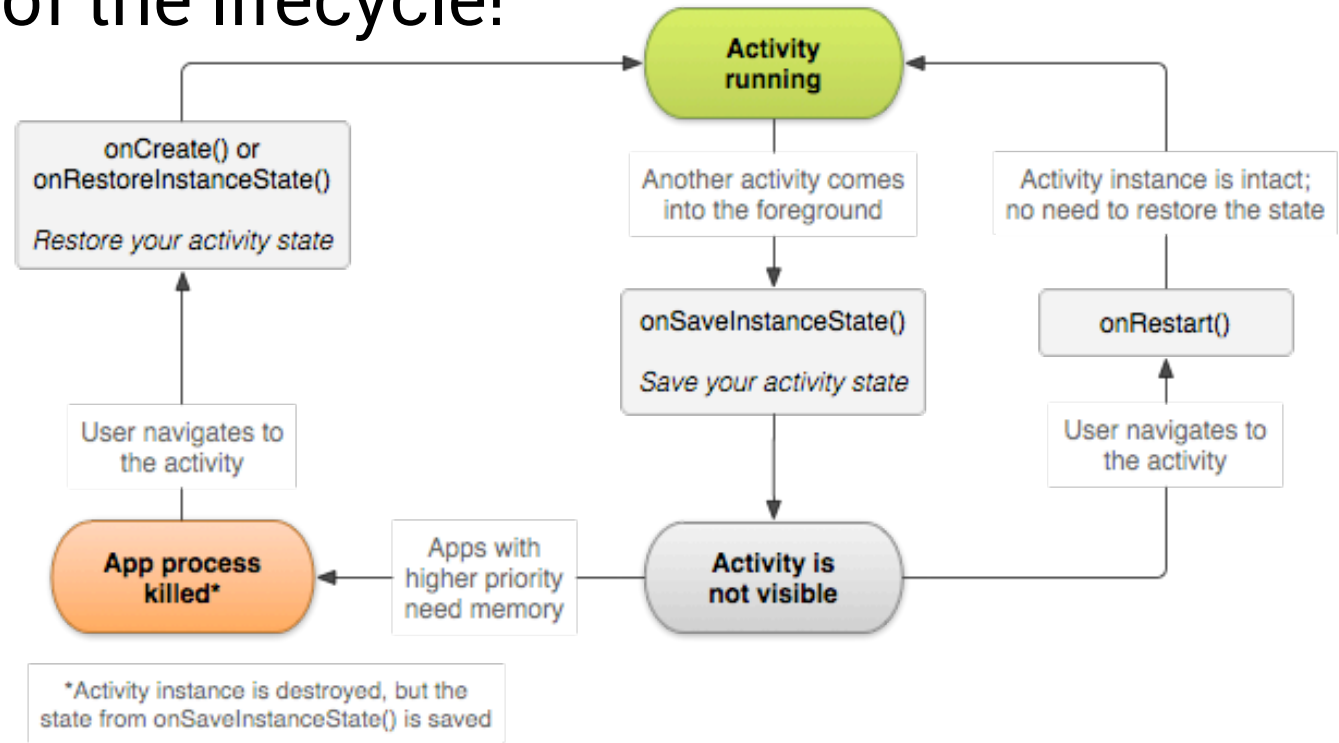
Design Decisions – Placing App Functionalities

- Instantiate member variables of the class
  - onCreate()
- (Un)register listeners for certain events
  - Location listener for showing a user's location on a map in a navigation app
    - onResume()/onPause()
  - Location listener for tracking a user's loc… fitness activity monitoring app
    - onStart()/onStop()
- Kill threads that the activity has spawned
  - onDestroy()

Background sensing is even better for this purpose

Saving Activity State

- `onSaveInstanceState()` – called when the Activity gets stopped to store state in a Bundle
- Not a part of the lifecycle!



from developer.android.com/guide/components/activities.html#SavingActivityState

Saving Activity State

- `onSaveInstanceState()`
  - By default saves transient information about the state of the activity's view hierarchy
    - Text in EditText, scroll position of a ListView, etc.
  - Add other key-value pairs you want to persist
  - Only “light” values should be stored in the Bundle – do not store big objects
- Restore in onRestoreInstanceState() or in onCreate()
- Uses: restore the state after the app’s process gets killed while in the background

Starting Activities

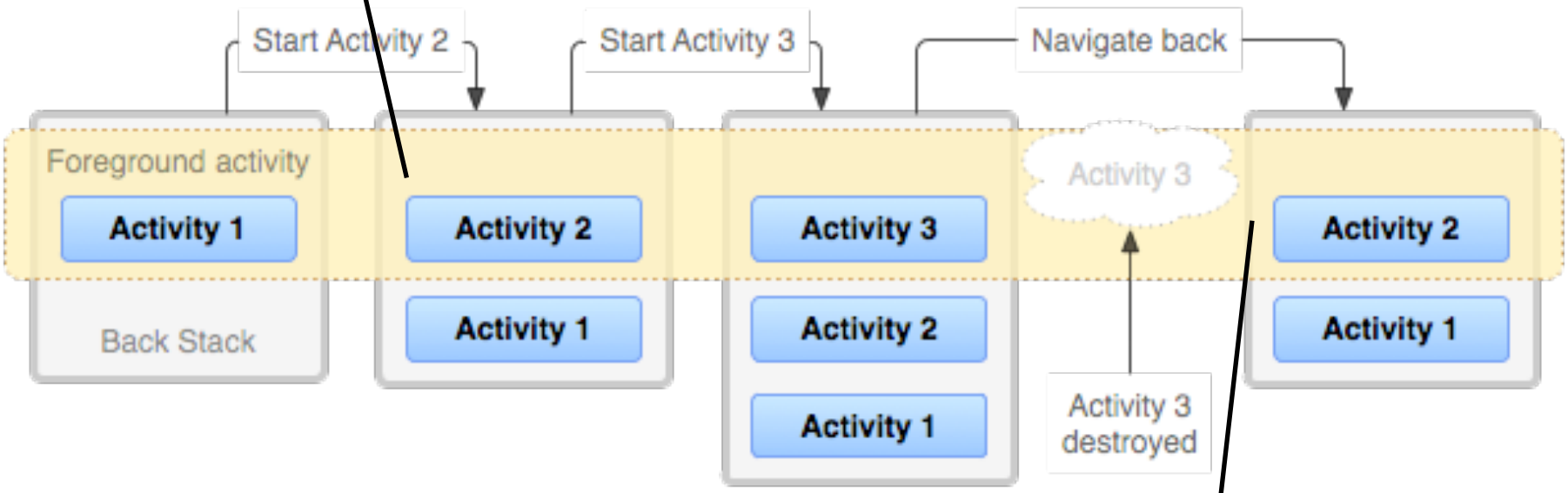
- Create an Intent specifying the Activity to start
- Pass the Intent to one of the following methods:
  - `startActivity()`
    - launches the Activity described by the Intent
  - `startActivityForResult()`
    - launches the Activity described by the Intent and expects a result that will be returned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

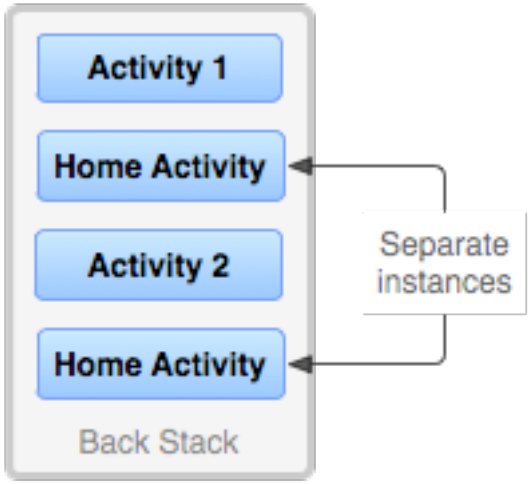
A new activity (Activity 2) is created and started, the old one (Activity 1) is stopped

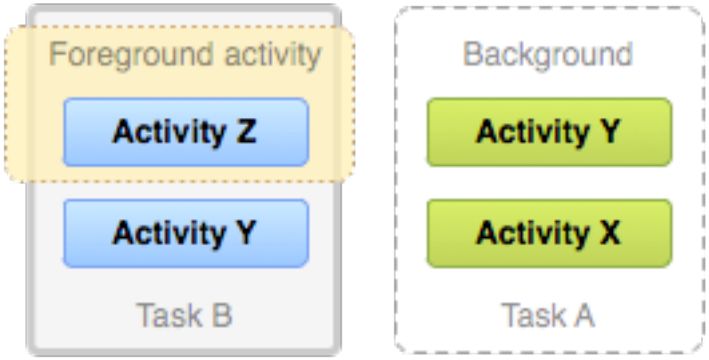


Activity 3 destroyed when the user clicked BACK, Activity 2 is started

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

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.

Action Field

- String representing a desired operation
- Examples
  - ACTION_DIAL – dial a number
  - ACTION_EDIT – start a component that can edit a certain piece of data (to be defined)
  - ACTION_GET_CONTENT – start a component that can get a certain piece of data (to be defined)
- Setting the action:

```
Intent newIntent = new Intent(Intent.ACTION_DIAL)
```

or

```
Intent newIntent = new Intent();
newIntent.setAction(Intent.ACTION_DIAL);
```

Data Field

- Data that the Intent should operate on
  - Formatted as a Uniform Resource Identifier (URI)
- Example:
  - Uri.parse(“content://contacts/people/1”) contact of a person “1”.
- Setting the data field (show location on a map):

```
Intent newIntent = new Intent(Intent.ACTION_VIEW,
Uri.Parse("geo:0,0?q=Ljubljana, Slovenia, 1000"))
```

or

```
Intent newIntent = new Intent(Intent.ACTION_VIEW);
newIntent.setData("geo:0,0?q=Ljubljana, Slovenia,
1000");
```

Category Field

- Gives additional information about the component that can handle an intent
- Examples:
  - CATEGORY_BROWSABLE – can be invoked by a browser to display data referenced by a URI
  - CATEGORY_LAUNCHER – can be the initial activity of a task and is listed in the top-level app launcher

Type Field

- Specifies the MIME (Internet standard) type of the Intent data
- Examples:
  - image/*, image/png, image/jpeg
  - text/html, text/plain
- If unspecified, Android will try to infer the type
- Setting the type

```
Intent.setType(String type)
OR
Intent.setDataAndType(Uri data, String type)
```

Component Field

- Specifies an explicit name of a component class to use for the Intent
- Component to be launched is often determined by matching the intent fields (the action, data/type, and categories) with components that might handle it
- If component field is set then the implicit matching is skipped, and the specified component is launched. In this case all of the other Intent attributes become optional.

Component Field

- Setting the component:

```
Intent newIntent = Intent (Context packageContext,
                           Class<?> cls)
```

OR

```
Intent newInt = new Intent ();
```

and one of:

```
setComponent(), setClass(), or setClassName()
```

Extras Field

- A Bundle of additional information associated with the Intent
- Example (email recipients):

```
Intent newInt = new Intent(Intent.ACTION_SEND);
newInt.putExtra(android.content.Intent.EXTRA_EMAIL,
  new String[]{ “bob@yahoo.com”, “alice@microsoft.com”}
);
```

- Usage
  - Different method depending on the data type:
    - putExtra(String name, String value);
    - putExtra(String name, float[] value);
    - etc.

Determining which Component Should be Started

- If explicitly named in the component field, execute the named component
- Otherwise – Intent Resolutions:
  - Intent needs to describe a desired operation
    - Action
    - Data (URI and Type)
    - Category
  - Components need to have IntentFilters which describe operations they can handle
    - Specified in AnodridManifest.XML

Specifying IntentFilters

- Activity handles sending text data

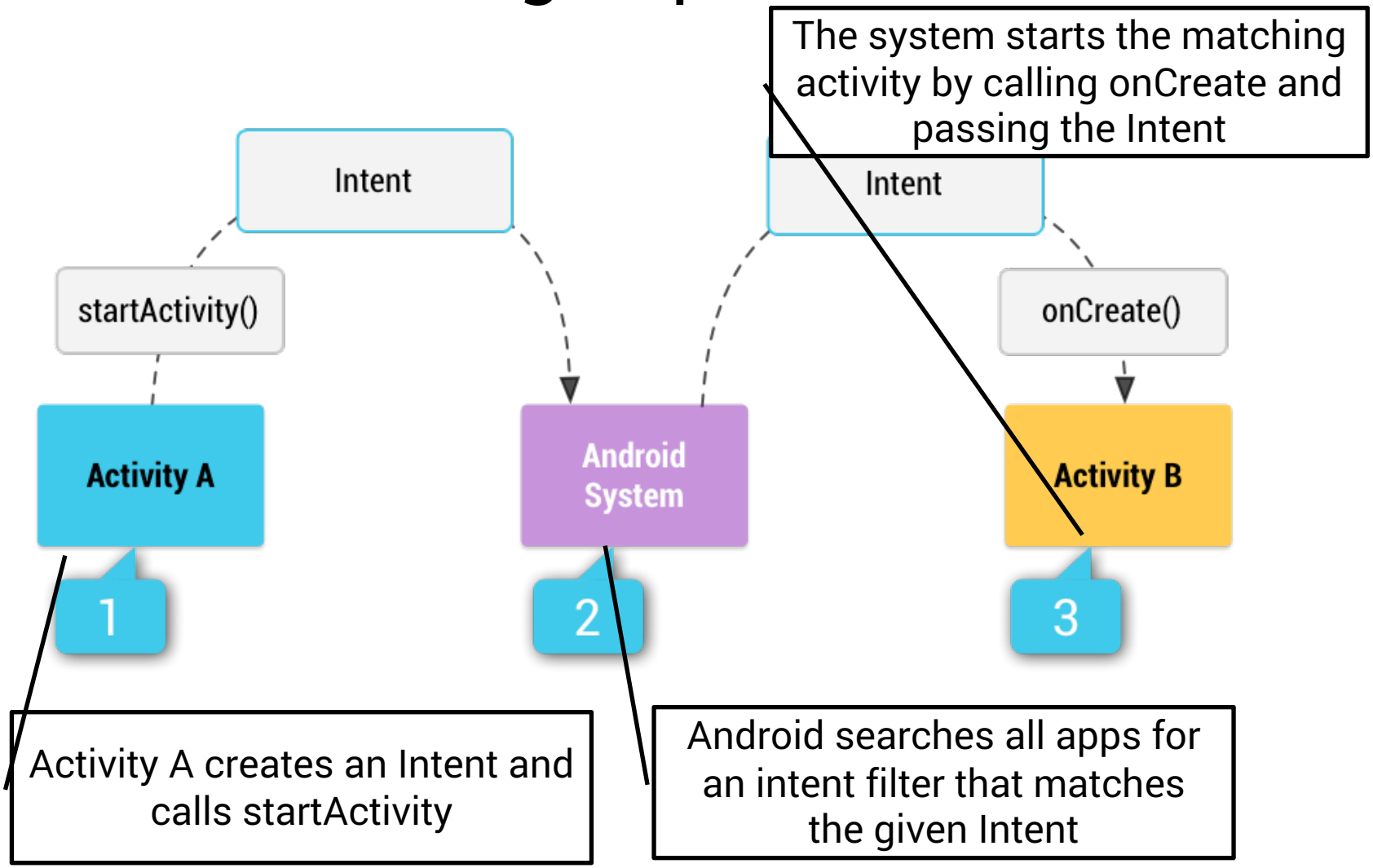
```
<activity android:name="ShareActivity">
    <!-- This activity handles "SEND" actions with text data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category
android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

Specifying IntentFilters

- Activity handles sending text and media

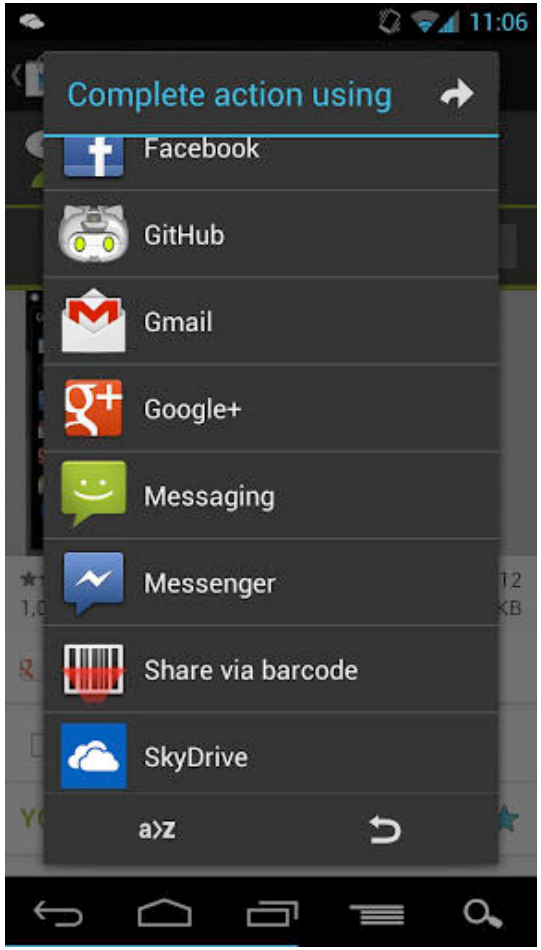
```
<activity android:name="ShareActivity">
    <!-- This activity also handles "SEND" and "SEND_MULTIPLE" with media
data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <action android:name="android.intent.action.SEND_MULTIPLE"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="application/vnd.google.panorama360+jpg"/>
        <data android:mimeType="image/*"/>
        <data android:mimeType="video/*"/>
    </intent-filter>
</activity>
```

Resolving Implicit Intents



Resolving Implicit Intents

- Resolving ties:
  - Present the user with a choice
  - Set the priority for the components:
    - `android:priority` in AndroidManifest.XML
      - An integer where higher values represent higher priorities



Flags

- Specify how Intent should be handled
- Examples:
  - FLAG_ACTIVITY_NO_HISTORY – don't keep the activity on the history task
  - FLAG_ACTIVITY_NEW_TASK – the activity is started in a new task
- Note: these flags can change the default backstack behaviour, and can result in unintuitive applications – use carefully!

Flags

- Setting flags:

```
Intent newInt = new Intent(Intent.ACTION_SEND);
newInt.setFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
```

- Intent flags are very similar to `android:launchMode` [`"multiple"`|`"singleTop"`|`"singleTask"`|`"singleInstance"`] from AndroidManifest.XML, but not quite the same