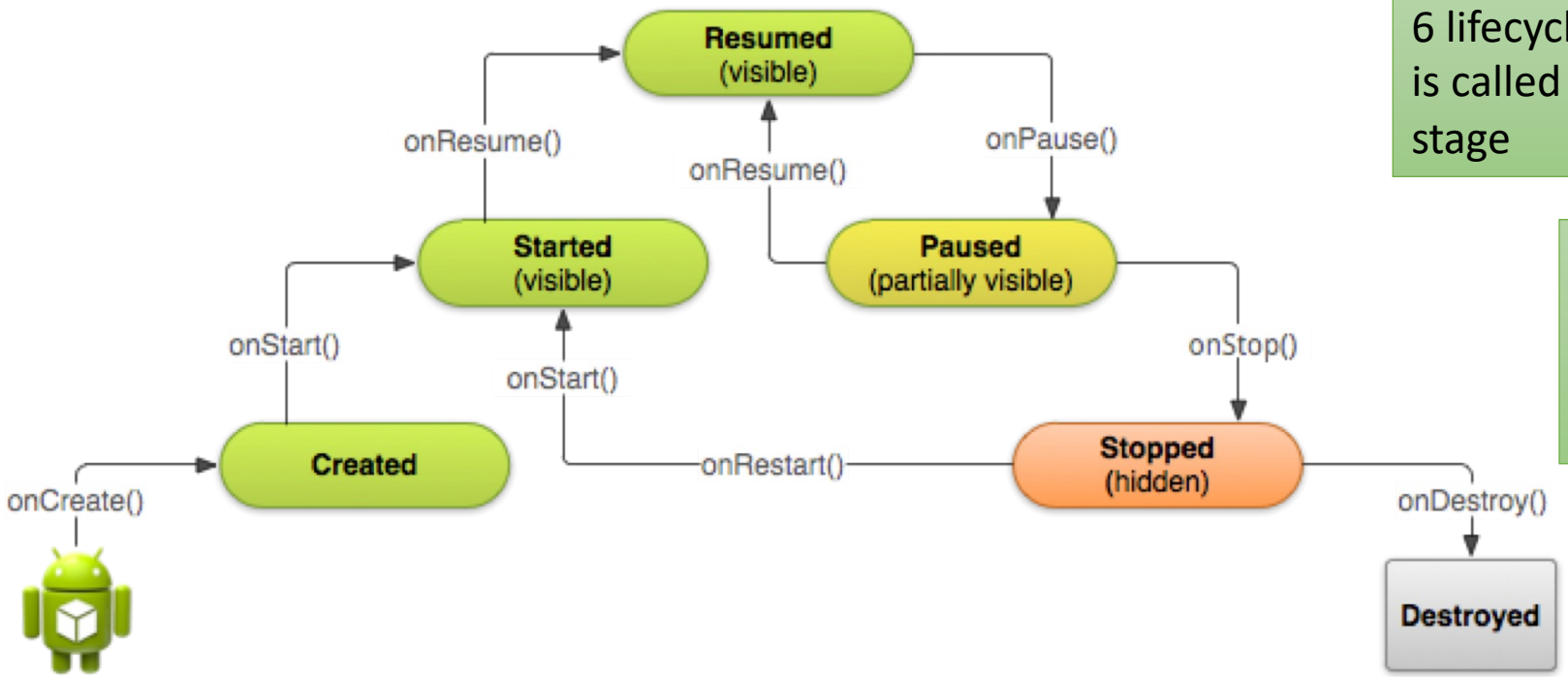
Activity Lifecycle

Mobile Computing - Android

Lifecycle Stages & Callbacks



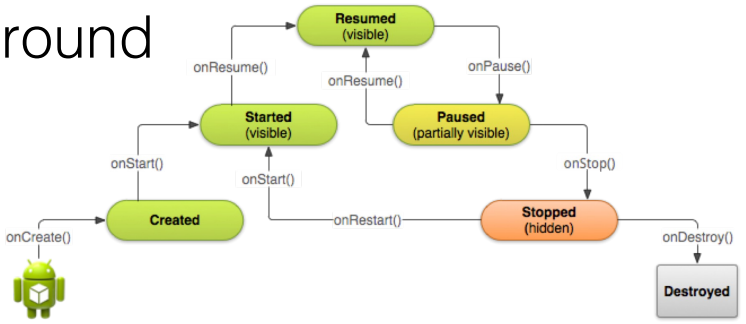
6 lifecycle stages. Each callback method is called on the transition to the next stage

Resumed = Running State
Paused = Partially visible
Stopped = Completely hidden
Destroyed = "He's dead, Jim"

Created & Started stages are transient: activity doesn't linger

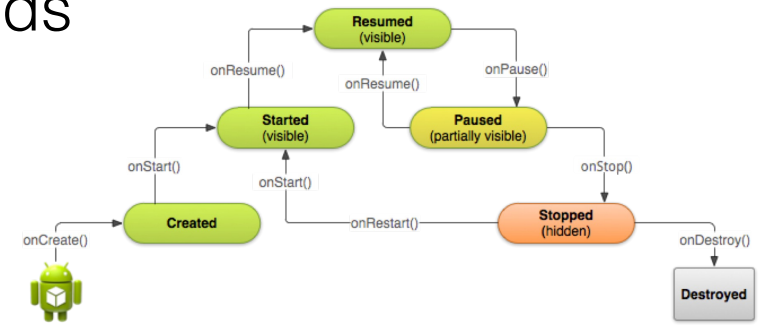
Static Lifecycle Stages

- Activities can stay in 3 stages for extended periods of time
- **Resumed** (aka Running)
  - The activity is in the foreground and the user can interact with it
- **Paused**
  - Another activity is in the foreground, but we are still visible (the other activity is either semitransparent, or does not occupy the entire screen)
  - Classic example: a dialog box
  - Cannot execute code nor receive user input
- **Stopped**
  - The activity is completely obscured, in the background
  - Cannot execute code nor receive user input
  - Still remembers its instance variables



Resuming an Activity

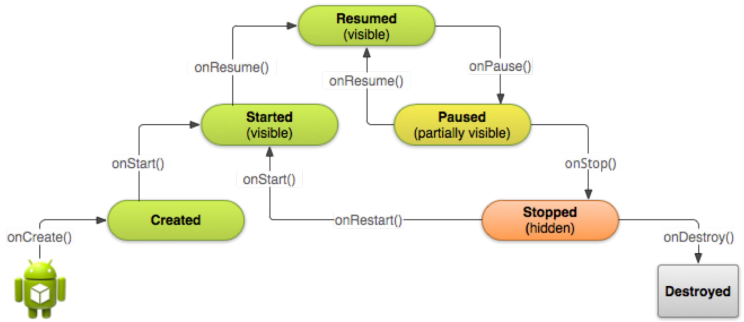
- **onResume()** is called when
  - a semi-transparent activity obscuring you ends
  - the user is coming back to your activity



- Undo the effects of onPause()
- If animations were paused -- start them
- If a GPS or camera or other resource was freed up, reacquire it

Pausing an Activity

- **onPause()** is called when
  - a semi-transparent activity obscures yours, or
  - the user is leaving your activity

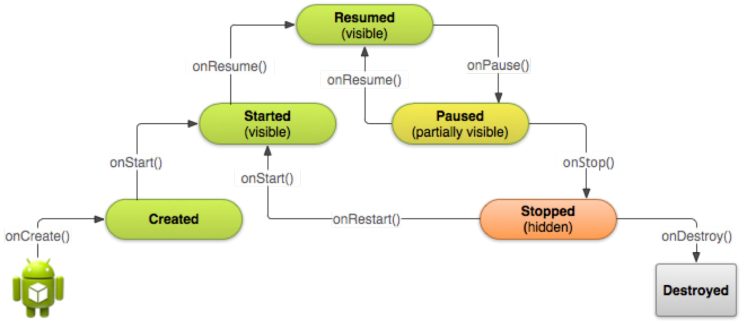


- What to do in this situation?
  - Stop animations that consume CPU
  - Save things that need to be saved (e.g., draft email)
  - Free up resources (GPS, camera) that you won't be using

Starting & Stopping Scenarios

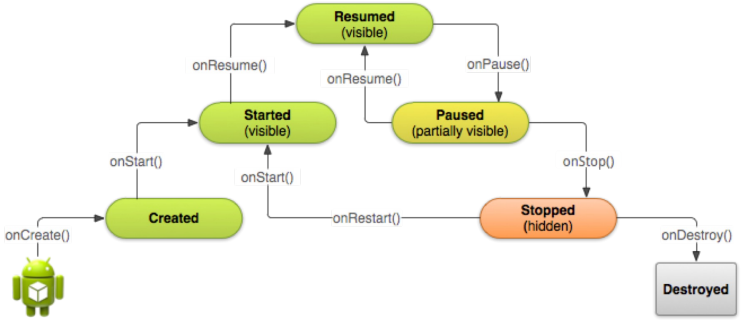
- If the user switches to another app, your activity **stops**
- If your activity starts a new activity, the current activity **stops**
- If a phone call interrupts, the current activity **stops**

- An activity may restart when the current activity ends (E.g. The user presses Back button)



Activity Destruction 😱

- An activity is **destroyed** when
  - the user clicks on the back button
  - finish() is invoked
- The user triggered the destruction of the activity, and all knowledge of it is gone. This includes any instance variables.

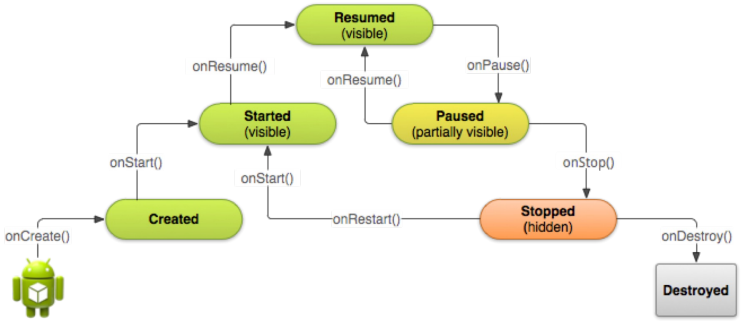


Activity Destruction 😱😱

- An activity is also destroyed **by the system** when:
  1. it needs to reclaim some resources
  2. the user *rotates* the screen. (The activity is recreated and may use a specialized layout for landscape/portrait.)
- Information in instance variables will be lost unless it is saved.
- Information entered into View objects (e.g., EditText) is automatically preserved in what is called a **Bundle**
- When the activity is recreated, the information in the Bundle is automatically repopulated in the Views
- **Only works if View objects have a unique value for android:id**

A Handy Short Cut

- If you initialize something in onCreate(), clean it up in onDestroy().
- If you initialize something in onStart(), clean it up in onStop().
- If you initialize something in onResume(), clean it up in onPause().

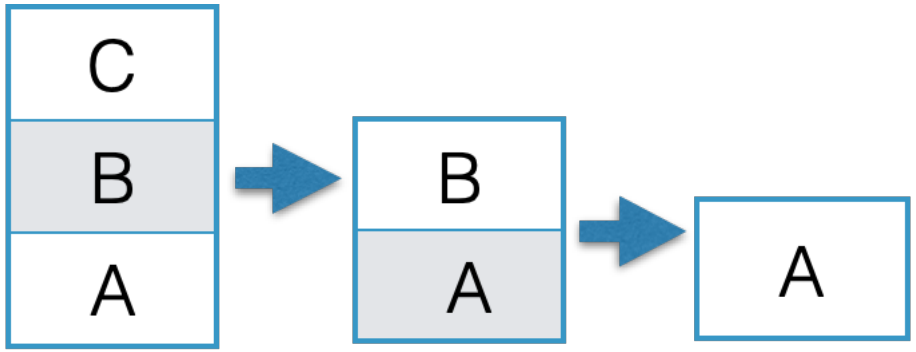


The Activity Stack

- An app is composed of a number of instances of activities all stored on a stack.
- The top activity on the stack of an active app is running.
- If the top activity is destroyed, the next item on the top will resume.
- If the last activity on the stack is destroyed, the app quits.

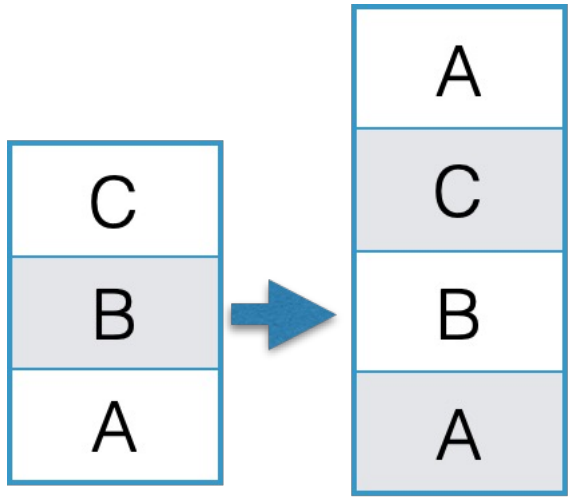
Recycling Activities

- Suppose our Activity stack is A -> B -> C; if we press back, C is destroyed and we are left with A->B
- back again and B is destroyed leaving just A.



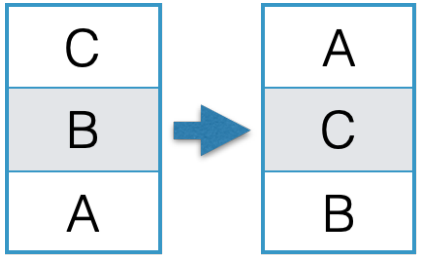
Recycling Activities

- Suppose from C we wish to navigate directly back to A (e.g., if A is the home activity, and C is the result of drilling down into a hierarchy of data.)
- startActivity() would yield A->B->C->A, i.e., two copies of A :-(, rather than just A.

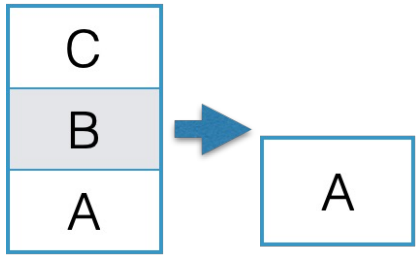


Recycling Activities Solutions

- FLAG_ACTIVITY_REORDER_TO_FRONT
  - Intent ini = new Intent(this, A.class);
  - ini.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
  - startActivity(ini); // now we would have B->C->A



- FLAG_ACTIVITY_CLEAR_TOP | FLAG_ACTIVITY_SINGLE_TOP
  - Intent ini = new Intent(this, A.class);
  - ini.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_SINGLE_TOP);
  - startActivity(ini); // this would leave just A, so hitting back would quit the app. Single top signals to use the existing instance of A; if not, A is restarted.



Preserving State

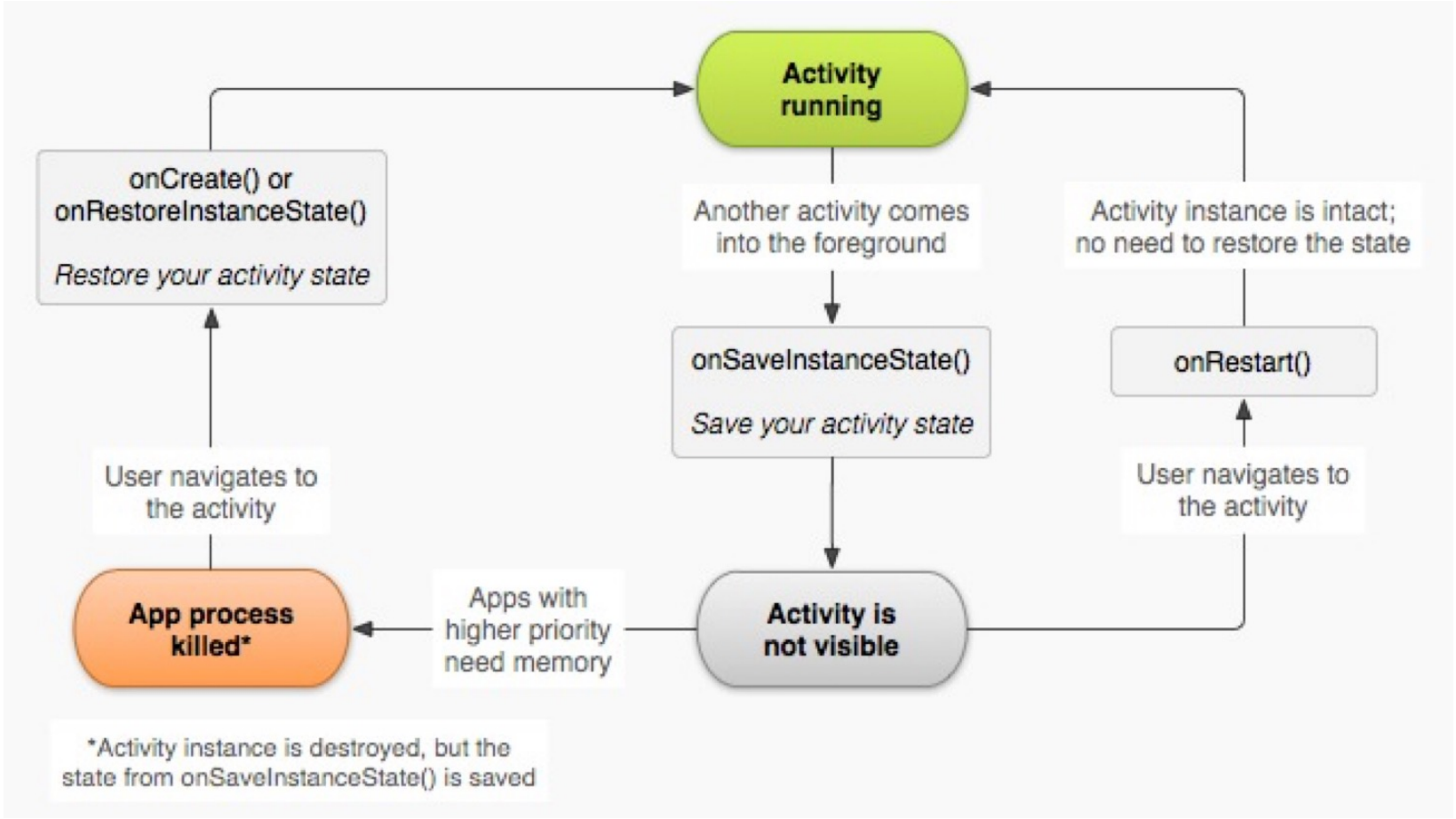
- The state of views are managed by the system, but what about other instance variables in an activity?
- Can override **onSaveInstanceState(Bundle b)**
- Called when leaving the activity, and you can store values in the Bundle that might be needed in the future.
- Uses *similar* key/value idea as Intents, with
  - put*DataType*(String key, *DataType* value)
  - *DataType* get*DataType*(String key)

Q: Why do Bundle put() methods require you specify the data type, whereas putExtra() does not?

Preserving State

- There is a more modern answer to this problem. State is kept in another class/classes which are alive even when the Activity is not. (Often called a model.) We will see this later.
- A consequence of Activity destruction is that we should never keep references to components as instance variables. They may change out from under us.

Preserving State



Example: onSaveInstanceState()

```
static final String STATE_SCORE = "playerScore";
static final String STATE_LEVEL = "playerLevel";
private int mCurrentScore;
private int mCurrentLevel;


@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    // Save the user's current game state
    savedInstanceState.putInt(STATE_SCORE, mCurrentScore);
    savedInstanceState.putInt(STATE_LEVEL, mCurrentLevel);

    // Always call the superclass so it can save the view
hierarchy state
    super.onSaveInstanceState(savedInstanceState);
}
```

Example: onCreate()

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); // Always call the superclass first

    // Check whether we're recreating a previously destroyed instance
    if (savedInstanceState != null) {
        // Restore value of members from saved state
        mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
        mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
    } else {
        // Probably initialize members with default values for a new instance
    }
    ...
}
```

Questions

- Describe the 6 major events in the lifecycle of an Activity, in the order in which they occur.
- Name the corresponding callback functions
- What is meant by a callback function?
- An activity is interrupted by a dialog box from another app, but it is still visible. Is the activity created, started, resumed, paused, stopped, or destroyed?
- A user navigates to another app (but the current activity is still in existence). Is the activity created, started, resumed, paused, stopped, or destroyed?
- An activity is stopped, but then killed by the OS. What happens when the user navigates to the activity?
- What is a Bundle and describe what common use in the Android frameworks.

Resources

- The Intent Class (Including static constants like flags)
- Android Developer - Activity Lifecycle