Creating a Stopwatch, Pt 2

Reviewing the Stopwatch

What questions do you have from yesterday?

Maydm

Activity Lifecycle Review

onCreate() runs immediately after activity is launched.

Activity spends most of its life in this stage.

onDestroy() runs before it's destroyed for final cleanup.

Maydm

Activity Lifecycle Review

We've been using `onCreate()` whenever we start an app. Now we've added the `onSaveInstanceState()` method.

When does `onSaveInstanceState()` get called?

How does it work with `onCreate()`?

How do these methods relate to screen rotation?

Maydm

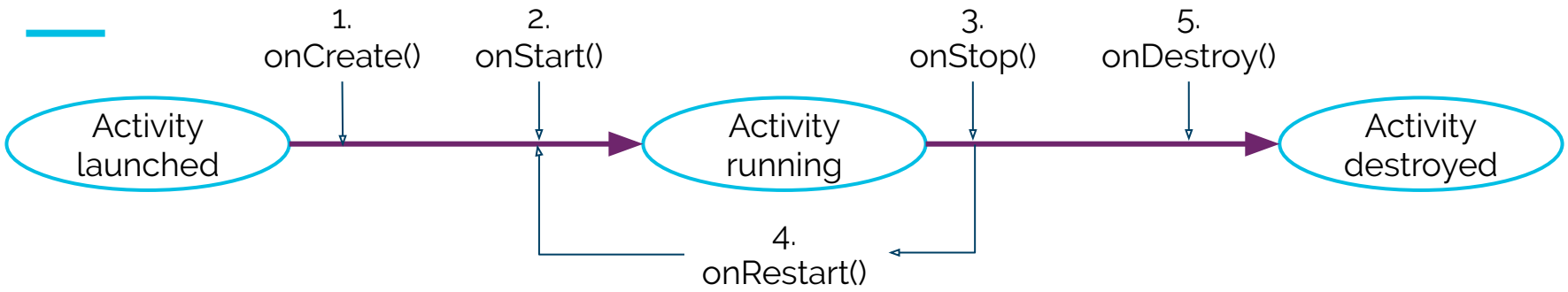
Lifecycle Methods

We've only used a few of these methods so far. Now we're going to add `onStart()` and `onStop()`.

- `onCreate(Bundle)`
- `onStart()`
- `onRestart()`
- `onResume()`
- `onPause()`
- `onStop()`
- `onDestroy()`
- `onSaveInstanceState()`
- `startActivity(Intent)`
- `findViewById(Int)`
- `setContentView(View)`

Maydm

Activity Lifecycle



1 - Activity launched, onCreate() runs.

2 - onStart() runs when activity is about to be visible.

3 - onStop() runs when activity stops being visible to user.

4 - If activity becomes visible again, onRestart() runs then onStart().

5 - onDestroy() runs and activity is destroyed.

Maydm

Adding onStop() and onStart()

We're going to add two more lifecycle methods to the stopwatch. onStop() will stop the timer from running when the app isn't visible. onStart() will restart the timer once the app is visible again.

We will start with onStop().

Adding onStop()

Whenever we Override a lifecycle method, we need to call super so that activity can perform any other actions in the superclass method, otherwise Android will throw an exception. Below the onSaveInstanceState() method, add an onStop() method:

```
@Override
protected void onStop() {
    super.onStop();
    running = false;
}
```

Maydm

Adding onStop()

Now we want to check if the stopwatch was running before onStop() was called. If it was running, we want the stopwatch to start again when the activity regains focus.

How can we implement that?

We can create another Boolean that will store the results of running! We can save it in the onSaveInstanceState(), along with the other variables.

Adding wasRunning Variable

Create private boolean wasRunning at the top of the activity and add it to the onCreate() method.

```
private boolean wasRunning;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_stopwatch);
    if (savedInstanceState != null) {
        seconds = savedInstanceState.getInt( key: "seconds");
        running = savedInstanceState.getBoolean( key: "running");
        wasRunning = savedInstanceState.getBoolean( key: "wasRunning");
```

Maydm

Adding wasRunning Variable

Save wasRunning in the onSaveInstanceState() method.

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    savedInstanceState.putInt("seconds", seconds);
    savedInstanceState.putBoolean("running", running);
    savedInstanceState.putBoolean("wasRunning", wasRunning);
}
```

Maydm

Adding wasRunning Variable

In the onStop() method, set wasRunning to the value of running, so we know what it was before onStop() was called.

```
@Override
protected void onStop() {
    super.onStop();
    wasRunning = running;
    running = false;
}
```

Maydm

Adding onStart()

Now that we have a variable that tells us whether the stopwatch was running before the activity was hidden, we can restart the stopwatch. We'll do that by add an onStart() method that will check wasRunning. Add this after onStop():

```
@Override
protected void onStart() {
    super.onStart();
    if (wasRunning) {
        running = true;
    }
}
```

Maydm

Test the App!

Does it work like you expect it to?

If no, what's happening?

Back Button Breakage

You might have noticed that the stopwatch reset when coming back to it after using the back button.

This happens because Android keeps a stack of activities (called the *back stack*). When a user hits the back button, Android knocks the last activity off the top of the stack and destroys it.

Back Button Breakage

Whenever the back button is used, the activity lifecycle methods happen differently than when a user closes the app:

- onStop() is not called
- onSaveInstanceState() is not called
- onDestroy() is called

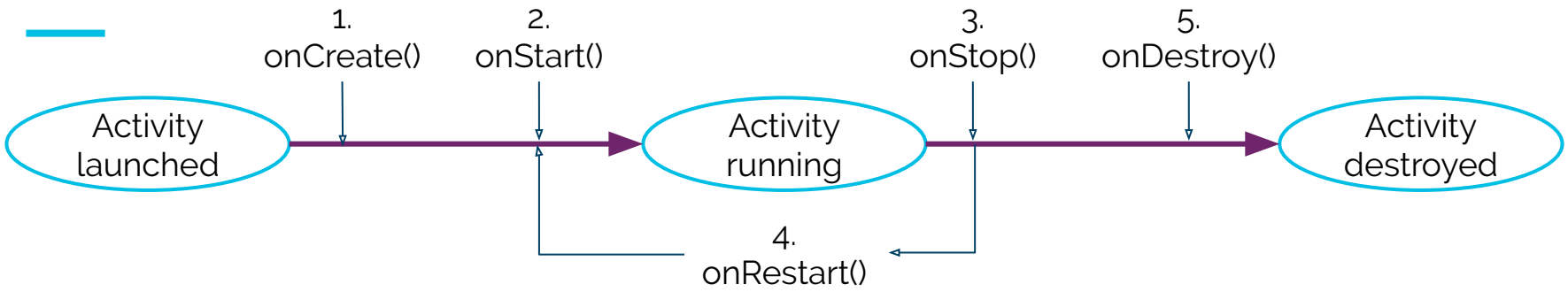
This means the state of the app will not be saved. :(

Maydm

Discussion

What is the difference between paused and destroyed states?

Activity Lifecycle Review



1 - Activity launched, onCreate() runs.

2 - onStart() runs when activity is about to be visible.

3 - onStop() runs when activity stops being visible to user.

4 - If activity becomes visible again, onRestart() runs then onStart().

5 - onDestroy() runs and activity is destroyed.

Maydm

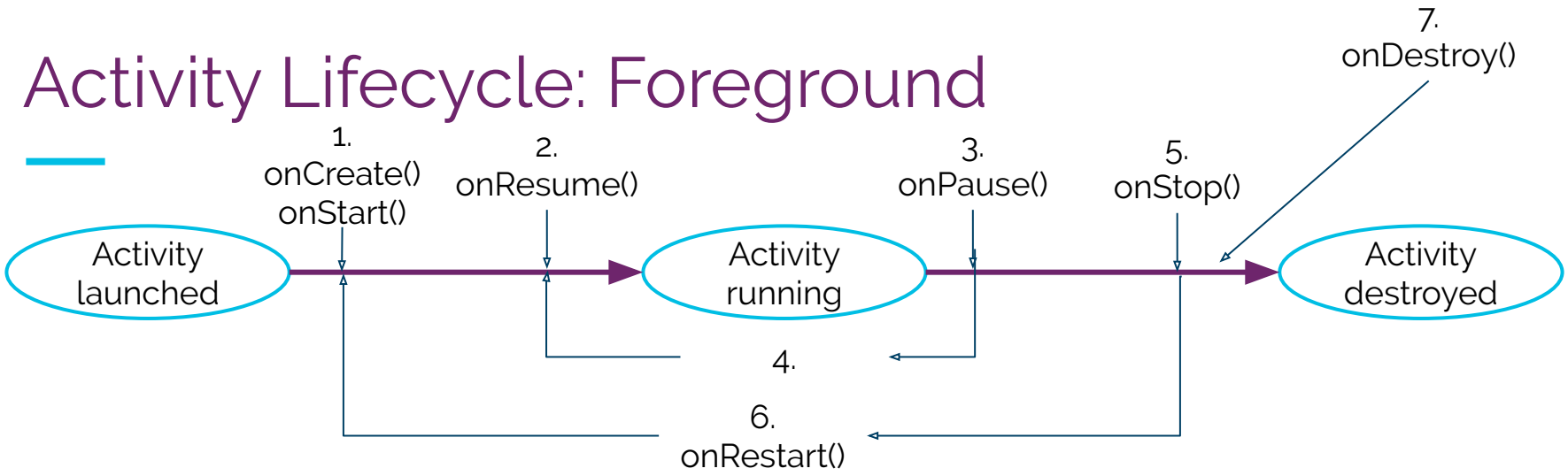
What if the app is only partially visible?

Another state we have to consider is when an activity is visible but doesn't have the focus, like if an alert pops up on the screen and takes the focus away from the app.

Whenever this happens, the activity underneath is paused. It is still alive and maintains its state information.

We will use the onPause() and onResume() methods to handle this case.

Activity Lifecycle: Foreground



1 - Activity launched, onCreate() and onStart() methods run.

2 - onResume() runs when activity is about to move into foreground.

3 - onPause() method runs when activity stops being in foreground.

4 - If activity moves back to foreground onResume() is called again.

5-6 - If activity stops being visible, onStop() gets called. If it's visible again, onRestart() is called.

Maydm

Adding onResume() Method

The onResume() method is called after onStart(), whether the activity is resumed or started. This means we can simply change the onStart() method to the onResume() method. Remember to also change the super!

```
@Override
protected void onStart() {
    super.onStart();
    if (wasRunning) {
        running = true;
    }
}
```

```
@Override
protected void onResume() {
    super.onResume();
    if (wasRunning) {
        running = true;
    }
}
```

Maydm

Adding onPause() Method

The onPause() method is called just before onStop(), whether the activity is paused or stopped. This means we can do the same thing with onStop() and onPause() that we just did with onStart() and onResume()!

```
@Override
protected void onStop() {
    super.onStop();
    wasRunning = runn
    running = false;
}
```

```
@Override
protected void onPause() {
    super.onPause();
    wasRunning = running;
    running = false;
}
```

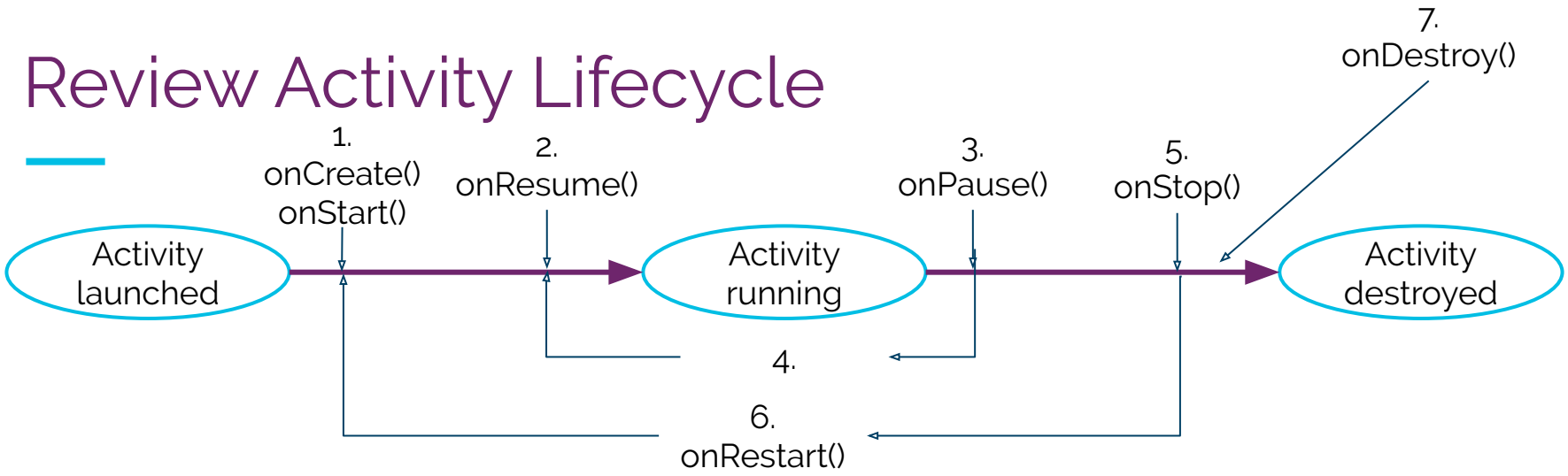
Maydm

Review Code & Fix Bugs

Any bugs in your app? Review the code and work on squashing those bugs.

Can't find a bug? Ask for help!

Review Activity Lifecycle



1 - Activity launched, onCreate() and onStart() methods run.

2 - onResume() runs when activity is about to move into foreground.

3 - onPause() method runs when activity stops being in foreground.

4 - If activity moves back to foreground onResume() is called again.

5-6 - If activity stops being visible, onStop() gets called. If it's visible again, onRestart() is called.

Maydm