Object Serialization

The concept of object serialization

Java's built-in serialization support

Android has a similar capability called "Parcelable". (Parceling is faster, but more work. Serialization is automatic, but slower.)

The Activity Lifecycle

Android Chapter 3

Every activity has a lifecycle. During this lifecycle, an activity transitions between four states: Running, Paused, Stopped, and Non-existent. For each transition, there is an Activity method that notifies the activity of the change in its state.

Display and discuss Figure 3.1

Run GeoQuiz(3) to demonstrate lifecycle method calls in logcat (filer by "QuizActivity").

1. Run GeoQuiz
2. Home button
3. Recents button
4. Back button

Rotation and the Activity Lifecycle

Show that GeoQuiz has a qualified layout resource.

Show that switching between portrait and landscape orientations causes the activity to be destroyed and re-created with the appropriate resources.

Comment out QuizActivity.onSaveInstanceState() and the code on onCreate() that restores the current question index.

Demonstrate how rotating the device causes the current question index to be lost (because Android destroys and re-creates the activity).

Uncomment the code, and explain what it does.

The default onSaveInstanceState() implementation saves the state of all UI components. That's why you often don't need to override this method (i.e., the default does most of what you want). However, you must override it to preserve state that Android doesn't know about.

Primitive values can be stored in a Bundle. You can also store Serializable or Parcelable objects.

The Activity Lifecycle, Revisited

Overriding onSaveInstanceState() is not just for handling rotation. An activity can also be stashed if the user navigates away for a while and Android needs to reclaim memory.

Android will never stash a Running activity to reclaim memory, but it may stash Paused and Stopped activities.

Show updated activity lifecycle diagram (Figure 3.13). There is a new state named Stashed.

Android tracks existing activities by storing an "activity record" for each activity. When Android needs to reclaim memory, it may stash an activity by calling onSaveInstanceState() on the activity, and then storing the serialized activity state in the activity's record.

Note that when a Stashed activity is reconstituted, Android calls onCreate(Bundle), passing in the activity's serialized state. This means that you cannot count on onStop() and onDestroy() to always be called. (You can count on onPause() always being called.)

When are activity records removed from the system? When the user hits the Back button, perhaps on reboot or when an activity hasn't been used for a long time. Data you don't want to lose should be persisted in a database, etc. (e.g., in onPause), since your stashed activity state might be discarded by the OS.

More on Activities

Android Chapter 5

Starting Activities

GeoQuiz (5)

Show simplest code for starting an activity (modify example)

```
Intent i = new Intent(QuizActivity.this, CheatActivity.class);
startActivity(i);
```

Explain the Back Stack

Passing Data into Activities

Add code to pass in correct answer:

```
String EXTRA_ANSWER_IS_TRUE =
"com.bignerdranch.android.geoquiz.answer_is_true";

boolean answerIsTrue = mQuestionBank[mCurrentIndex].isAnswerTrue();
Intent i = new Intent(QuizActivity.this, CheatActivity.class);
i.putExtra(EXTRA_ANSWER_IS_TRUE, answerIsTrue);
startActivity(i);


mAnswerIsTrue = getIntent().getBooleanExtra(EXTRA_ANSWER_IS_TRUE, false);
```

Returning Results from Activities

Show final version of the code.

```
int REQUEST_CODE_CHEAT = 0;

startActivityForResult(i, REQUEST_CODE_CHEAT);
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if (resultCode != Activity.RESULT_OK) {
        return;
    }

    if (requestCode == REQUEST_CODE_CHEAT) {
        if (data == null) {
            return;
        }
        mIsCheater = CheatActivity.wasAnswerShown(data);
    }
}


private void setAnswerShownResult(boolean isAnswerShown) {
    Intent data = new Intent();
    data.putExtra(EXTRA_ANSWER_SHOWN, isAnswerShown);
    setResult(RESULT_OK, data);
}
```

Family Map Application

Starting Activities

In Main activity, Person activity is started when event info is clicked. Intent extra used to pass ID of person to be displayed.

Implemented in MapFragment

Main activity starts Search, Filter, and Settings activities when they're selected from the options menu.

Implemented in MapFragment

Child Settings activity returns a SettingsResult indicating which settings were changed.

Child Filter activity returns a FilterResult indicating if any changes were made.

In Person activity, clicking a person starts another Person activity. Intent extra used to pass ID of person to be displayed.

Implemented in PersonListItem

In Person activity, clicking an event starts a new Map activity

Implemented in EventListItem

In Map activity, Person activity is started when event info is clicked. Intent extra used to pass ID of person to be displayed.

Implemented in MapFragment

In Map and Person activities, “Go to Top” jumps back to Main activity (preserving the original activity).

For Map, implemented in MapFragment

In Settings activity, “Re-sync Data” and “Logout” jump back to Main activity (creating a new activity).

Rotation (Extra Credit)

Rotating Main (with map) or Map activities requires saving/restoring selected event ID and map camera settings.

Implemented in MapFragment

Rotating Person activity requires saving/restoring booleans indicating whether or not the person and event sections are expanded.