**LAB # 08**

**Sharing data within multiple activities – Custom objects**

Sharing data between multiple activities is a common requirement in Android app development. When sharing custom objects, you need to ensure that the objects are serializable so they can be passed between activities using intents. Here's an example of how to share a custom object between multiple activities in Android:

Create the custom object: First, create the custom object that you want to share between activities. For example, if you want to share a User object, you could create a class called User that contains fields for the user's name, email address, and other information.

**public class User implements Serializable {**

**private String name;**

**private String email;**

**public User(String name, String email) {**

**this.name = name;**

**this.email = email;**

**}**

**public String getName() {**

**return name;**

**}**

**public String getEmail() {**

**return email;**

**}**

**}**

Note that the User class implements the Serializable interface to allow instances of the class to be serialized and passed between activities.

Pass the object between activities: To pass the User object between activities, you can create an intent and add the User object as an extra. For example, in Activity A:

**// Create a new User object**

**User user = new User("John Smith", "john@example.com");**

**// Create an intent to start Activity B**

**Intent intent = new Intent(this, ActivityB.class);**

**// Add the User object as an extra to the intent**

**intent.putExtra("user", user);**

**// Start Activity B**

**startActivity(intent);**

In Activity B, you can retrieve the User object from the intent extras:

**// Get the intent that started this activity**

**Intent intent = getIntent();**

**// Get the User object from the intent extras**

**User user = (User) intent.getSerializableExtra("user");**

**// Use the User object**

**String name = user.getName();**

**String email = user.getEmail();**

Update the object and pass it back: If you need to update the User object in Activity B and pass it back to Activity A, you can create a new intent and add the updated User object as an extra. For example, in Activity B:

**// Create a new User object with updated information**

**User updatedUser = new User("Jane Smith", "jane@example.com");**

**// Create a new intent to return the updated User object**

**Intent intent = new Intent();**

**intent.putExtra("updatedUser", updatedUser);**

**// Set the result to RESULT\_OK and add the intent as data**

**setResult(RESULT\_OK, intent);**

**// Finish the activity**

**finish();**

In Activity A, you can handle the result in the onActivityResult method:

**@Override**

**protected void onActivityResult(int requestCode, int resultCode, Intent data) {**

**super.onActivityResult(requestCode, resultCode, data);**

**if (requestCode == REQUEST\_CODE && resultCode == RESULT\_OK) {**

**// Get the updated User object from the intent extras**

**User updatedUser = (User) data.getSerializableExtra("updatedUser");**

**// Use the updated User object**

**String name = updatedUser.getName();**

**String email = updatedUser.getEmail();**

**}**

**}**

Note that this approach works best for small objects with simple data types, as larger objects or objects with complex data structures may be slow to serialize and deserialize.

**TASK 1** : Create an application which can manipulate a Clock object having hours, minutes and seconds as integer attributes. Your application must have only two activities as follows:

* 1. ***ClockActivity***

Shows the clock contents and the gives field selection by radio buttons for update. Creates new Clock object, displays it and send it to UpdateActivity for update.

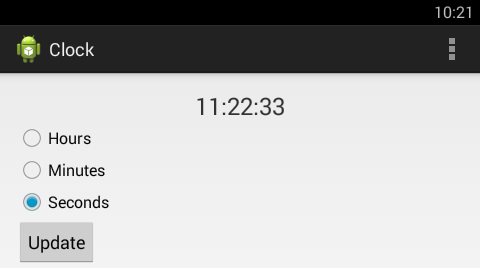
* 1. ***UpdateActivity***

Gets data for the selected field. Saves the field in the object and send back object to ClockActivity.

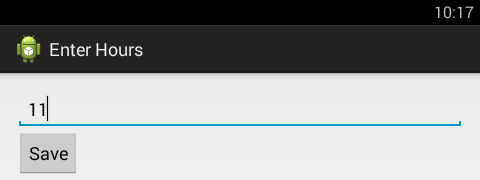
You must pass whole object from activity to activity. Remember that your class must implement Serializable interface.

<<Serializable>>

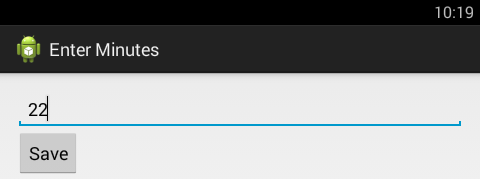
Clock



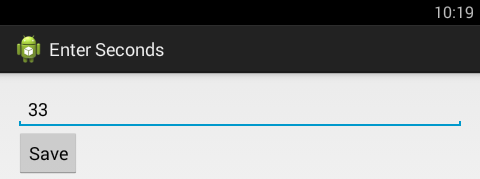
ClockActivity: Having selection of fields



UpdateActivity: After selecting hours



UpdateActivity: After selecting minutes



UpdateActivity: After selecting seconds