Mobile Application Lab 7

Instructors: Legand Burge, Asad Ullah Naweed
Date: 08/23/2018
Topics: Fragments and SharedPreferences

Be sure to click on the links in this doc to view more information about certain topics. These may or may not help you in your lab, but can always provide some extra context and information.

Initial Setup

For this lab, you'll be using the ViewPagerDemo project from the in-class demo for Lecture 6. You can download it here. Make sure that you can build and run the app without any errors before you start work.

In a second Android Studio window, also bring up your solution for Lab 5. You will need it moving forward.

Swiping Quiz Questions

You will now transform this ViewPager sample into a quiz app where the questions can be answered and then swiped left and right to navigate between questions.

The Model layer

You will first create a `Question` class to represent your Model layer. This will be nearly identical to the `Question` class you wrote for Lab 5. However, you will need to add a new private String field called `selectedAnswer`. Add a getter and a setter for this field, and initialize it to `null` in the constructor.

QuestionViewFragment

You now need to turn the `ContactViewFragment` into a `SingleQuestionFragment`. This will be a fragment created for a single question. You should have a static method on the class to create a fragment for a given `Question` with the associated Bundle arguments, like so:

```
static SingleQuestionFragment createFragmentFromQuestion(
    Question question, int questionIndex) {
  Bundle fragArgs = new Bundle();
  fragArgs.putString(ARG_QUESTION_TEXT, question.getQuestion());
  fragArgs.putString(ARG_CORRECT_ANSWER, question.getCorrectAnswer());
  fragArgs.putString(ARG_WRONG_ANSWER, question.getWrongAnswer());
  fragArgs.putInt(ARG_QUESTION_INDEX, questionIndex);
  fragArgs.putString(ARG_SELECTED_ANSWER, question.getSelectedAnswer());

  SingleQuestionFragment frag = new SingleQuestionFragment();
  frag.setArguments(fragArgs);
  return frag;
}
```

Inside the `onCreateView` method, use the `getArguments()` method to get the arguments Bundle and obtain the various Strings that were set when the fragment was created. Also retrieve and save the `questionIndex` as a private variable. We will need this later.

Copy-paste the layout you created for `QuizFragment` in Lab 5, and use it as the layout for `SingleQuestionFragment`. Initialize all the UI components, as well as their callbacks. However, remove the "Next" button from both the layout and the code. You will not need it because you'll be navigating questions by swiping.

In your MainActivity, modify the `ContactFragmentAdapter` to be a `QuestionFragmentAdapter`. Remove the code for instantiating the list of contacts, and implement the `getItem` and `getCount` methods as follows:

```
private static final class QuestionFragmentAdapter
    extends FragmentStatePagerAdapter {

  public ContactFragmentAdapter(FragmentManager fm) {
    super(fm);
  }

  @Override
  public Fragment getItem(int position) {
    Question question = new Question(
        "What is the capital of Kenya?", "Nairobi", "Mombasa");
    return SingleQuestionFragment.createFragmentFromQuestion(question, position);
  }

  @Override
  public int getCount() {
    return 1;
```

```
  }
}
```

Before proceeding further, make sure your app can build and run, and that your fragment is displayed correctly.

Once your app is building and running correctly, modify the adapter to show multiple questions instead of just a single one.

Fragment -> Activity communication

Now, create the following interface inside `SingleQuestionFragment`:

```
interface OnQuestionAnsweredListener {
  void onQuestionAnswered(String selectedAnswer, int questionId);
}
```

Now, make your `MainActivity` class implement this interface:

```
public class MainActivity extends AppCompatActivity
    implements OnQuestionAnsweredListener {
  // Existing code...

  @Override
  public void onQuestionAnswered(String selectedAnswer, int questionId) {
    Log.d(TAG, "The " + selectedAnswer + " button was pressed!");
  }
}
```

Now, override the `onAttach` method in your Fragment as follows:

```
public class SingleQuestionFragment extends Fragment {
  // Existing code...

  private OnQuestionAnsweredListener answerListener;

  @Override
  public void onAttach(Context context) {
    try {
      answerListener = (OnQuestionAnsweredListener) context;
    } catch (ClassCastException ex) {
      throw new ClassCastException(
```

```
            "The Context did not implement OnQuestionAnsweredListener!");
        }
    }
}
```

Now, inside the callback handlers for the answer buttons in `SingleQuestionFragment`, be sure to call `answerListener.onQuestionAnswered` with the text of the question that was answered and the ID of the question.

Before proceeding, build and run your app. Make sure that you can see the Logcat output from the `onQuestionAnswered` method inside `MainActivity` whenever you press an answer button.

ScoreActivity

Just like in Lab 5, you will need to display a ScoreActivity with a final score when the user has answered all the questions. To do that, you will need to fully implement the `onQuestionAnswered` method in `MainActivity` instead of the simple log statement that currently exists there.

You will need to:

1. Keep track of all the answers given, and
2. Launch the ScoreActivity with the user's score when all questions have been answered.

High Scores

Now, you will also need to keep track of the high score of every user on the device. This should be done inside `ScoreActivity` using `SharedPreferences`.

When `ScoreActivity` receives a user's score, it should check for the current high score in the `SharedPreferences` you create for this purpose. If the user beats the high score, you should display a Toast message with a congratulatory message, and the new high score.