Homework 5: PopupQuiz Report

This report contains all the codebase of the project. Several minor changes were made for better readability:

- Reduce indent to **2 spaces.**
- Reduce max line length to **80.**
- Remove XML header tag and all attributes with `xmlns:` prefix in XML.
- Remove package and import statements in Java.

Gradle

1. /gradle/libs.version.toml

```
[versions]
jdk = "21"
jre = "8"
checkstyle = "10.17.0"
kotlin = "2.0.20"
sdk-min = "31"
sdk-target = "34"
android-plugin = "8.7.2"
androidx = "1.7.0"
androidx-lifecycle = "2.8.6"
androidx-test = "1.6.1"
retrofit = "2.11.0"

[plugins]
android-application =
  { id = "com.android.application", version.ref = "android-plugin" }

[libraries]
# lint
rulebook-checkstyle = "com.hanggrian.rulebook:rulebook-checkstyle:0.1"
# main
material =
  { module = "com.google.android.material:material", version.ref =
"androidx" }
```

```
androidx-appcompat =
  { module = "androidx.appcompat:appcompat", version.ref = "androidx" }
androidx-coordinatorlayout =
  "androidx.coordinatorlayout:coordinatorlayout:1.2.0"
androidx-lifecycle-viewmodel = {
  module = "androidx.lifecycle:lifecycle-viewmodel",
  version.ref = "androidx-lifecycle",
}
androidx-lifecycle-livedata = {
  module = "androidx.lifecycle:lifecycle-livedata",
  version.ref = "androidx-lifecycle",
}
androidx-lifecycle-extensions = "androidx.lifecycle:lifecycle-
extensions:2.2.0"
retrofit =
  { module = "com.squareup.retrofit2:retrofit", version.ref = "retrofit" }
# test
androidx-test-core =
  { module = "androidx.test:core", version.ref = "androidx-test" }
androidx-test-runner =
  { module = "androidx.test:runner", version.ref = "androidx-test" }
androidx-test-junit = "androidx.test.ext:junit:1.2.1"

robolectric = "org.robolectric:robolectric:4.13"
truth = "com.google.truth:truth:1.4.4"

[bundles]
androidx = [
  "material",
  "androidx-appcompat",
  "androidx-coordinatorlayout",
  "androidx-lifecycle-viewmodel",
  "androidx-lifecycle-livedata",
  "androidx-lifecycle-extensions",
]
androidx-test = [
  "androidx-test-core",
  "androidx-test-runner",
  "androidx-test-junit",
  "robolectric",
  "truth",
]
```

2. /settings.gradle.kts

```
pluginManagement.repositories {
  gradlePluginPortal()
  mavenCentral()
  google()
}
dependencyResolutionManagement.repositories {
  mavenCentral()
  google()
}

rootProject.name = "PopupQuiz"
```

3. /build.gradle.kts

```
val releaseGroup: String by project
val releaseArtifact: String by project
val releaseVersion: String by project

val jdkVersion = JavaLanguageVersion.of(libs.versions.jdk.get())
val jreVersion = JavaLanguageVersion.of(libs.versions.jre.get())

plugins {
  alias(libs.plugins.android.application)
  checkstyle
  kotlin("android") version libs.versions.kotlin.get() // required by some
dependencies
}

group = releaseGroup
version = releaseVersion

java.toolchain.languageVersion.set(jdkVersion)
```

```
android {
  namespace = "$releaseGroup.$releaseArtifact"
  testNamespace = "$namespace.test"
  compileSdk = libs.versions.sdk.target.get().toInt()
  defaultConfig {
    minSdk = libs.versions.sdk.min.get().toInt()
    targetSdk = libs.versions.sdk.target.get().toInt()
    version = releaseVersion
    testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    multiDexEnabled = true
      applicationId = namespace
  }
  compileOptions {
    sourceCompatibility = JavaVersion.toVersion(jreVersion)
    targetCompatibility = JavaVersion.toVersion(jreVersion)
  }
  testOptions.unitTests.isIncludeAndroidResources = true
  buildTypes {
    debug {
      enableAndroidTestCoverage = true
    }
    release {
      isMinifyEnabled = false
      proguardFiles(getDefaultProguardFile("proguard-android.txt"),
"proguard-rules.pro")
    }
  }
}

checkstyle.toolVersion = libs.versions.checkstyle.get()

dependencies {
  checkstyle(libs.rulebook.checkstyle)

  implementation(libs.bundles.androidx)
  implementation(libs.retrofit)

  testImplementation(libs.bundles.androidx.test)
}

tasks.register<Checkstyle>("checkstyle") {
  group = LifecycleBasePlugin.VERIFICATION_GROUP
  source("src")
  include("**/*.java")
  exclude("**/gen/**", "**/R.java")
  classpath = files()
}
```

XML

4. /lint.xml

```xml
<lint>
  <!-- TOML property variable is not always the latest. -->
  <issue id="GradleDependency" severity="ignore"/>

  <!-- Drawables are official Material Symbols. -->
  <issue id="VectorPath" severity="ignore"/>

  <!-- Sample application does not have an icon. -->
  <issue id="MissingApplicationIcon" severity="ignore"/>
</lint>
```

5. /src/main/AndroidManifest.xml

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
  <uses-permission android:name="android.permission.INTERNET"/>

  <application
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/Theme.Material3.DayNight.NoActionBar"
    android:usesCleartextTraffic="true">
    <activity
      android:name=".MainActivity"
      android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>
</manifest>
```

6. /src/main/res/drawable/

```
drawable/
├ btn_display.xml
├ btn_next.xml
├ ic_info.xml
└ ic_reset.xml
```

7. /src/main/res/layout/

7a. activity_main.xml

```
<androidx.coordinatorlayout.widget.CoordinatorLayout
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:animateLayoutChanges="true"
  tools:context=".MainActivity">

  <com.google.android.material.appbar.AppBarLayout
    android:id="@+id/appbarLayout"
    android:layout_width="match_parent"
    android:layout_height="192dp">

    <com.google.android.material.appbar.CollapsingToolbarLayout
      android:id="@+id/toolbarLayout"
      android:layout_width="match_parent"
      android:layout_height="match_parent"

app:expandedTitleTextAppearance="@style/TextAppearance.Material3.DisplayMe
dium"
      app:layout_scrollFlags="noScroll">
      <com.google.android.material.appbar.MaterialToolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?actionBarSize"/>
    </com.google.android.material.appbar.CollapsingToolbarLayout>
  </com.google.android.material.appbar.AppBarLayout>

  <ProgressBar
    android:id="@+id/progress"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="4"
    android:min="0"
    android:progressTint="?colorPrimary"
    app:layout_anchor="@id/appbarLayout"
    app:layout_anchorGravity="bottom"/>

  <LinearLayout
    android:id="@+id/refreshLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:animateLayoutChanges="true"
    android:clipToPadding="false"
    android:orientation="vertical"
    android:padding="16dp"
    app:layout_behavior="@string/appbar_scrolling_view_behavior">

    <com.google.android.material.textview.MaterialTextView
      android:id="@+id/text"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:width="400dp"
      android:paddingTop="8dp"
      android:paddingBottom="8dp"

android:textAppearance="@style/TextAppearance.Material3.HeadlineMedium"/>

    <RadioGroup
      android:id="@+id/radioGroup"
      android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:orientation="horizontal">

        <com.google.android.material.radiobutton.MaterialRadioButton
          android:id="@+id/trueRadio"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@string/radio_true"/>

        <com.google.android.material.radiobutton.MaterialRadioButton
          android:id="@+id/falseRadio"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_marginStart="8dp"
          android:text="@string/radio_false"/>
      </RadioGroup>


  <com.google.android.material.floatingactionbutton.ExtendedFloatingActionBu
  tton
        android:id="@+id/displayButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/btn_display"
        app:icon="@drawable/btn_display"/>

      <RatingBar
        android:id="@+id/rating"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:isIndicator="true"
        android:numStars="5"
        android:rating="0.0"
        android:stepSize="1.0"/>
    </LinearLayout>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
      android:id="@+id/nextButton"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginEnd="32dp"
      android:contentDescription="@string/btn_next_desc"
      app:layout_anchor="@id/appbarLayout"
      app:layout_anchorGravity="bottom|end"
      app:srcCompat="@drawable/btn_next" />
  </androidx.coordinatorlayout.widget.CoordinatorLayout>
```

8. /src/main/res/values/strings.xml

```
<resources>
  <string name="about">About</string>
  <string name="loading">Loading</string>
  <string name="reset">Reset</string>

  <string name="app_name">PopupQuiz</string>
  <string name="app_about">A sliding quiz application with a simple yes-
or-no answer.</string>

  <string name="radio_true">True</string>
  <string name="radio_false">False</string>

  <string name="btn_display">Display Result</string>
  <string name="btn_next_desc">Go to next question.</string>

  <string name="are_you_sure_">Are you sure?</string>
  <string name="please_wait_">Please wait…</string>
  <string name="quiz_d">Quiz #%1$d</string>
  <string name="correct_">Correct!</string>
  <string name="wrong_">Wrong!</string>
</resources>
```

9. /src/main/res/menu/activity_main.xml

```
<menu>
  <item
    android:id="@+id/reset"
    android:icon="@drawable/ic_reset"
    android:title="@string/reset"
    app:showAsAction="always"/>

  <item
    android:id="@+id/about"
    android:icon="@drawable/ic_info"
    android:title="@string/about"
    app:showAsAction="ifRoom"/>
</menu>
```

Java

11. /src/main/java/com/example/quiz/

11a. AboutDialog.java

```
/**
 * A simple dialog describing what the application does. This dialog must
be
 * attached to a {@link DialogFragment}.
 */
public class AboutDialog extends DialogFragment {
  public static final String TAG = "AboutDialog";

  @NonNull
  @Override
  public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
    return new AlertDialog.Builder(requireContext())
      .setTitle(R.string.about)
      .setMessage(R.string.app_about)
      .setPositiveButton(android.R.string.ok, (dialog, which) -> {})
      .create();
  }
}
```

11b. MainActivity.java

```
/**
 * A single screen displaying quiz question, answer toggles and rating bar
 * representing user's progress. At any point during the quiz, user may
return
 * to the initial point by selecting the <b>Reset</b> menu item.
 */
public class MainActivity extends AppCompatActivity {
  private static final long LOADING_DELAY = 1500L;
  static final Map<String, Integer> RESULT_MAP = new HashMap<>();

  static {
    RESULT_MAP.put(
      "Android's current stable OS release is Android 14.",
      R.id.trueRadio
    );
    RESULT_MAP.put(
      "An AsyncTask is tied to the life cycle of the Activity that
contains"
      + " it.",
      R.id.falseRadio
    );
    RESULT_MAP.put(
      "The last callback in the lifecycle of an activity is onDestroy()",
      R.id.trueRadio
    );
    RESULT_MAP.put(
      "To collapse / expand items use the Code -> Folding menu in AS.",
      R.id.trueRadio
    );
    RESULT_MAP.put(
      "You cannot start an Activity with an Intent.",
      R.id.falseRadio
    );
  }

  CollapsingToolbarLayout toolbarLayout;
  Toolbar toolbar;
  ProgressBar progress;

  TextView text;
  RadioGroup radioGroup;
  Button displayButton;
  RatingBar rating;
  FloatingActionButton nextButton;

  MainViewModel viewModel;
  PapademasApi api;

  private LoadingDialog loadingDialog;

  @Override
  protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);
    toolbarLayout = findViewById(R.id.toolbarLayout);
    toolbar = findViewById(R.id.toolbar);
    progress = findViewById(R.id.progress);
    text = findViewById(R.id.text);
    radioGroup = findViewById(R.id.radioGroup);
    displayButton = findViewById(R.id.displayButton);
    rating = findViewById(R.id.rating);
    nextButton = findViewById(R.id.nextButton);

    setSupportActionBar(toolbar);

    api = PapademasApi.create();
    viewModel = new ViewModelProvider(this).get(MainViewModel.class);
    viewModel.stateData.observe(
      this,
      state -> {
        switch (state) {
          case LOADING:
            reset();
            break;
          case ANSWERING:
            displayButton.setVisibility(View.VISIBLE);
            nextButton.setVisibility(View.INVISIBLE);
            radioGroup.clearCheck();
            break;
          case ANSWERED:
            displayButton.setVisibility(View.INVISIBLE);
            nextButton.setVisibility(View.VISIBLE);
            break;
          case FINISHED:
            displayButton.setVisibility(View.INVISIBLE);
            nextButton.setVisibility(View.INVISIBLE);
            break;
      }
      }
    );
      viewModel.questionIndexData.observe(
        this,
        questionIndex → {
          if (viewModel.questions.isEmpty()) {
            return;
          }
          updateText(questionIndex);
        }
      );
      viewModel.answerTallyData.observe(
        this,
        answerTally →
          rating.setRating((float) answerTally.first / answerTally.second
* 5f)
      );

      displayButton.setOnClickListener(
        v → {
          boolean isCorrect =

Objects.requireNonNull(RESULT_MAP.get(text.getText().toString()))
              == radioGroup.getCheckedRadioButtonId();
          viewModel.updateAnswerTally(isCorrect);
          viewModel.stateData.setValue(
            Objects.requireNonNull(viewModel.questionIndexData.getValue())
              == RESULT_MAP.size() - 1
              ? State.FINISHED
              : State.ANSWERED
          );

          Toast
            .makeText(
              MainActivity.this,
              isCorrect ? R.string.correct_ : R.string.wrong_,
              Toast.LENGTH_SHORT
            ).show();
        }
      );
      radioGroup.setOnCheckedChangeListener(
        (group, checkedId) → displayButton.setEnabled(checkedId ≠ -1)
      );
      nextButton.setOnClickListener(
        v → {
          viewModel.questionIndexData.setValue(
            Objects.requireNonNull(viewModel.questionIndexData.getValue())
+ 1
          );
          viewModel.stateData.setValue(State.ANSWERING);
        }
      );
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.activity_main, menu);
    return super.onCreateOptionsMenu(menu);
  }

  @Override
  public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    if (item.getItemId() == R.id.about) {
        new AboutDialog().show(getSupportFragmentManager(),
AboutDialog.TAG);
    } else if (item.getItemId() == R.id.reset) {
        new ConfirmDialog()
          .show(getSupportFragmentManager(), ConfirmDialog.TAG);
    }
    return super.onOptionsItemSelected(item);
  }

  private void updateText(int questionIndex) {
    progress.setProgress(questionIndex);
    toolbarLayout.setTitle(getString(R.string.quiz_d, questionIndex + 1));
    text.setText(viewModel.questions.get(questionIndex));
  }

  private void reset() {
    if (loadingDialog ≠ null) {
      return;
    }
    loadingDialog = new LoadingDialog();
    loadingDialog.show(getSupportFragmentManager(), LoadingDialog.TAG);
    Executors.newSingleThreadExecutor().execute(
      () → {
        try {
          viewModel.questions.clear();
          viewModel.questions.addAll(api.getRandomizedQuestions());
          new Handler(Looper.getMainLooper())
            .postDelayed(
              () → {
                loadingDialog.dismiss();
                loadingDialog = null;

                viewModel.stateData.setValue(State.ANSWERING);
                viewModel.questionIndexData.setValue(0);
                viewModel.answerTallyData.setValue(new Pair<>(0, 0));
              },
              LOADING_DELAY
            );
        } catch (IOException e) {
          loadingDialog.dismiss();
          loadingDialog = null;

          String message = e.getMessage();
          if (message == null) {
            message = "Unknown error.";
          }
          Snackbar
            .make(toolbarLayout, message, Snackbar.LENGTH_LONG)
            .show();
        }
      }
    );
  }
  public static class ConfirmDialog extends DialogFragment {
    public static final String TAG = "ConfirmDialog";

    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
      return new AlertDialog.Builder(requireContext())
        .setTitle(R.string.reset)
        .setMessage(getString(R.string.are_you_sure_))
        .setNegativeButton(android.R.string.cancel, (dialog, which) -> {})
        .setPositiveButton(
          android.R.string.ok,
          (dialog, which) -> ((MainActivity) requireActivity()).reset()
        ).create();
    }
  }

  public static class LoadingDialog extends DialogFragment {
    public static final String TAG = "LoadingDialog";

    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
      return new AlertDialog.Builder(requireContext())
        .setTitle(R.string.loading)
        .setMessage(getString(R.string.please_wait_))
        .create();
    }
  }
}
```

11c. MainViewModel.java

```
/**
 * A container of observable values controlling {@link MainActivity}
behavior.
 */
public class MainViewModel extends ViewModel {
  /**
```

```
     * The question sheet for the quiz.
     */
    @NonNull
    public final List<String> questions = new ArrayList<>();

    /**
     * @see State
     */
    @NonNull
    public final MutableLiveData<State> stateData =
      new MutableLiveData<>(State.LOADING);

    /**
     * Current index of the question being answered.
     */
    @NonNull
    public final MutableLiveData<Integer> questionIndexData =
      new MutableLiveData<>(0);

    /**
     * Number of correct and total answers, these values produce rating
  stars.
     */
    @NonNull
    public final MutableLiveData<Pair<Integer, Integer>> answerTallyData =
      new MutableLiveData<>(new Pair<>(0, 0));

    /**
     * Increment tally answers.
     *
     * @param isCorrect whether the answer is right.
     */
    public void updateAnswerTally(boolean isCorrect) {
      Pair<Integer, Integer> tally =
        Objects.requireNonNull(answerTallyData.getValue());
      answerTallyData.setValue(
        new Pair<>(
          isCorrect ? tally.first + 1 : tally.first,
          tally.second + 1
        )
      );
  }
}
```

11d. PapademasApi.java

```
/**
 * A REST API invocation site to retrieve pop quiz questions from the
Papademas website.
 */
public interface PapademasApi {
  String ENDPOINT = "http://www.papademas.net:81";

  @GET("/sample.txt")
  Call<ResponseBody> getQuestions();

  @NonNull
  default List<String> getRandomizedQuestions() throws IOException {
    List<String> questions =
      Arrays.asList(
        getQuestions()
          .execute()
          .body()
          .string()
          .split("\n")
      );
    Collections.shuffle(questions);
    return questions;
  }

  /**
   * Convenient method to instantiate builder.
   */
  @NonNull
  static PapademasApi create() {
    return new Retrofit.Builder()
      .baseUrl(ENDPOINT)
      .build()
      .create(PapademasApi.class);
  }
}
```

11e. State.java

```
/**
 * Viewing mode when answering the pop quiz.
 */
public enum State {
  /**
   * Fetching questions from server, loading dialog is blocking all
controls.
   */
  LOADING,

  /**
   * User can answer the question by clicking <b>Display Result</b>
button.
   */
  ANSWERING,

  /**
   * User can no longer answer current question because <b>Display
Result</b>
   * button is disabled, while <b>Next</b> button appears.
   */
  ANSWERED,

  /**
   * No more questions to feed, all buttons become un-clickable.
   */
  FINISHED
}
```

Tests

12. /src/test/AndroidManifest.xml

```
<manifest>
  <uses-sdk tools:overrideLibrary="androidx.test.core"/>

  <application/>
</manifest>
```

13. /src/test/java/com/example/quiz/

13a. MainActivityTest.java

```
@RunWith(RobolectricTestRunner.class)
@DoNotInstrument
public class MainActivityTest {
  private MainActivity activity;

  @Before
  public void setup() {
    activity =
Robolectric.buildActivity(MainActivity.class).setup().get();
  }

  @Test
  public void checkStates() {
    assertThat(activity.viewModel.stateData.getValue())
      .isEqualTo(State.LOADING);
    ShadowLooper.runUiThreadTasksIncludingDelayedTasks();

    for (int i = 0; i < 5; i++) {
      assertThat(activity.viewModel.stateData.getValue())
        .isEqualTo(State.ANSWERING);

      activity.radioGroup.check(R.id.trueRadio);
      activity.displayButton.performClick();

      if (i == 4) {
        assertThat(activity.viewModel.stateData.getValue())
          .isEqualTo(State.FINISHED);
        return;
      }
      assertThat(activity.viewModel.stateData.getValue())
        .isEqualTo(State.ANSWERED);
      activity.nextButton.performClick();
    }
  }

  @Test
  public void checkAnswers() {
    ShadowLooper.runUiThreadTasksIncludingDelayedTasks();
    for (int i = 0; i < 5; i++) {
      activity.radioGroup.check(
        MainActivity.RESULT_MAP.get(activity.text.getText().toString())
      );
      activity.displayButton.performClick();
      if (activity.nextButton.isShown()) {
        activity.nextButton.performClick();
      }
    }

    Pair<Integer, Integer> tally =

Objects.requireNonNull(activity.viewModel.answerTallyData.getValue());
    assertThat(tally.first)
      .isEqualTo(tally.second);
  }
}
```

13b. PapademasApiTest.java

```
public class PapademasApiTest {
  private PapademasApi api;

  @Before
  public void init() {
    api = PapademasApi.create();
  }

  @Test
  public void getQuestion() {
      try {
        assertThat(api.getQuestions().execute().body().string())
          .isEqualTo(
            "Android's current stable OS release is Android 14.\n"
              + "An AsyncTask is tied to the life cycle of the Activity
that"
              + " contains it.\n"
              + "The last callback in the lifecycle of an activity is"
              + " onDestroy()\n"
              + "To collapse / expand items use the Code → Folding menu
in"
              + " AS.\n"
              + "You cannot start an Activity with an Intent."
          );

        assertThat(api.getRandomizedQuestions().size())
          .isEqualTo(5);
      } catch (IOException e) {
        throw new RuntimeException(e);
      }
    }
}
```