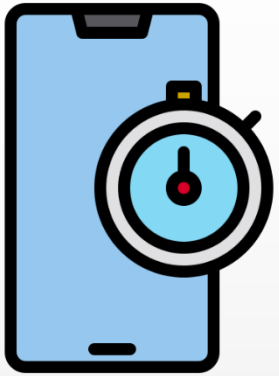
Intro to Android Game Development

Dmytro Zubov, PhD
dmytro.zubov@ucentralasia.org

Naryn, Kyrgyzstan, 1:56 pm, November 13, 2022

Lessons learnt last time

- An Example of the Java Class
- Java Inheritance
- Java Interfaces
- Java Polymorphism
- Java Abstract Class
- Java Encapsulation
- final Keyword
- static Class Members
- Java OOP Constructor



What we gonna discuss today?

- Scope and lifetime of variables
- Intro to file I/O
- A simple Star Wars quiz
- In-class activity
- 2D game in Android Studio (optional)

Scope and Lifetime of Variables

- The **scope of a variable** defines the section of the code in which the variable is visible. In general, variables that are defined inside a block are not accessible outside this block.
- The **lifetime of a variable** refers to how long the variable exists before it is destroyed. Destroying variables refers to deallocating the memory that was allotted to the variables when declaring it.

Scope and Lifetime of Variables (cont.)

MyApp1ScopeLifetime > ScopeLifetimeVariable > src > main > java > com > example > zubov > myapp1 > FullscreenActivity

activity_fullscreen.xml × | strings.xml × | FullscreenActivity.java ×

FullscreenActivity | onButtonClick()

```
122
123     public void onButtonClick(View view) {
124         {
125             String s1 = "Result:\r\n";
126             outer:
127             for (int i = 0; i < 3; i++)
128                 for (int j = 0; j < 2; j++)
129                     if (i == 1 && j == 1)
130                         continue outer;
131                     else
132                         s1 = s1 + "i=" + i + ", j=" + j + "\r\n";
133             s1 = s1 + "Both loops terminated.";
134         }
135         textView.setText(s1);
136     }
137
```

Cannot resolve symbol 's1'

- This code is wrong!
- When we want to output `s1`, the variable does not exist anymore!

What should we do?

Intro to File I/O

- Android offers a few structured ways to store data:
  - Shared Preferences
  - Internal Storage
  - External Storage (e.g., SD card)
  - SQLite Storage
  - Storage via Network Connection (on a cloud)

https://www.journaldev.com/9383/android-internal-storage-example-tutorial

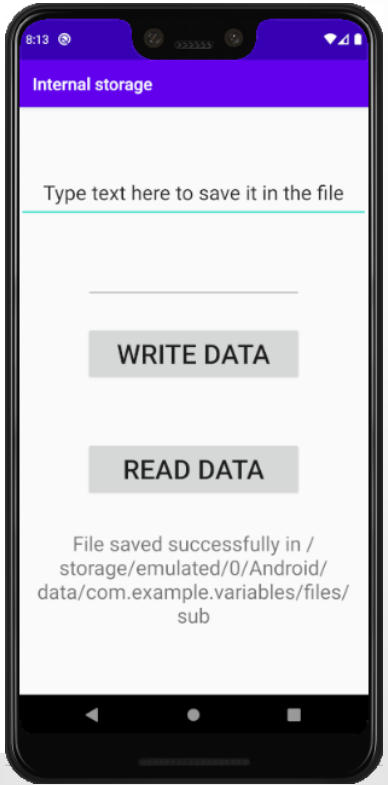
Android Internal Storage

- **Android Internal storage** is the storage of the private data on the device memory. By default, saving and loading files to the internal storage are private to the application and other applications will not have access to these files.
- When the user uninstalls the applications, the internal stored files associated with the application are also removed. However, some users root their Android phones, gaining superuser access. These users will be able to read and write whatever files they wish.

Android Internal Storage (cont.)

- An example: Write the text to the file

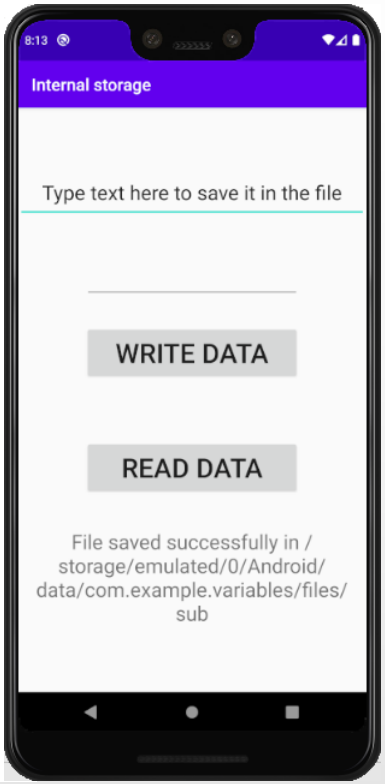
```
try {
    String FILENAME = "mytextfile.txt";  // The file name
    String FOLDERNAME = "sub"; // The folder name
    // The path to the file:
    String folder = getApplicationContext().getExternalFilesDir( type: null).getAbsolutePath()+ File.separator + FOLDERNAME;
    File subFolder = new File(folder);  // Here, we define a path to the file
    if (!subFolder.exists()) subFolder.mkdirs(); // If folder doesn't exists, we create this folder
    // FileOutputStream creates a file output stream to write to the file with the specified name
    FileOutputStream outputStream = new FileOutputStream(new File(subFolder, FILENAME));
    // An OutputStreamWriter is a bridge from character streams to byte streams:
    // Characters written to it are encoded into bytes using a specified charset.
    // The charset that it uses may be specified by name or may be given explicitly,
    // or the platform's default charset may be accepted.
    OutputStreamWriter outputWriter=new OutputStreamWriter(outputStream);
    outputWriter.write(editText.getText().toString()); // Here, we write the text to the file
    outputWriter.close(); // Here, we close the file

    //display file saved message
    textView.setText("File saved successfully in " + subFolder);

} catch (Exception e) {
    e.printStackTrace();
    textView.setText("File was NOT saved successfully");
}
```



Android Internal Storage (cont.)

- An example: Read the text from the file

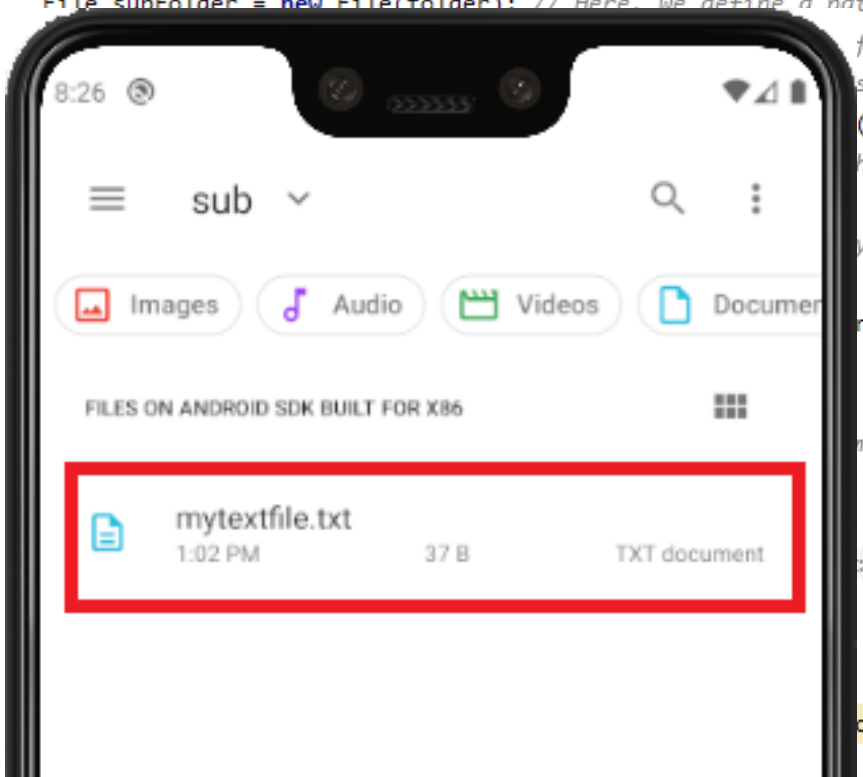
```
try {
    int READ_BLOCK_SIZE = 1;
    String FILENAME = "mytextfile.txt"; // The file name
    String FOLDERNAME = "sub"; // The folder name
    // The path to the file:
    String folder = getApplicationContext().getExternalFilesDir( type: null).getAbsolutePath()+ File.separator + FOLDERNAME;
    File subFolder = new File(folder); // Here, we define a path to the file
    // A FileInputStream obtains input bytes from a file in a file system.
    //FileInputStream is meant for reading streams of raw bytes such as image data.
    FileInputStream inputStream = new FileInputStream(new File(subFolder, FILENAME));
    // An InputStreamReader is a bridge from byte streams to character streams:
    // It reads bytes and decodes them into characters using a specified charset.
    // The charset that it uses may be specified by name or may be given explicitly,
    // or the platform's default charset may be accepted.
    InputStreamReader InputRead= new InputStreamReader(inputStream);
    char[] inputBuffer= new char[READ_BLOCK_SIZE];
    String s=""; int charRead;
    // Here, we call read method to read one character at a time from the file
    while ((charRead=InputRead.read(inputBuffer))>0) {
        // char to string conversion
        String readstring=String.copyValueOf(inputBuffer, offset: 0,charRead); s +=readstring;
    }
    InputRead.close();
    editText.setText(s);
    textView.setText("File was read successfully from " + subFolder);
} catch (Exception e) {
    e.printStackTrace();
    textView.setText("File was NOT read successfully!");
}
```

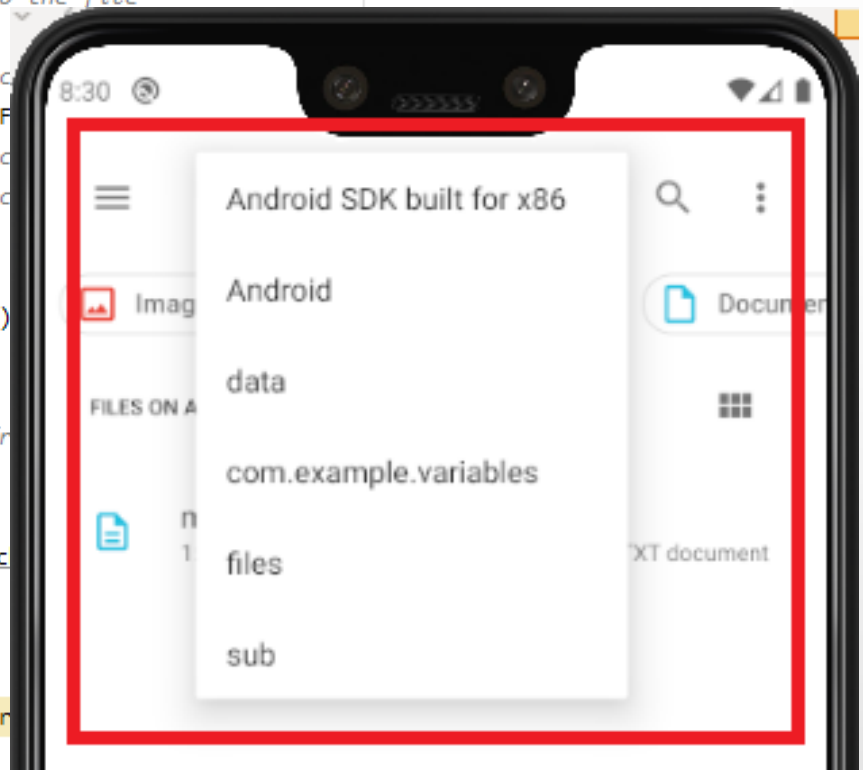


Android Internal Storage (cont.)

- Location of the file

```
try {
    String FILENAME = "mytextfile.txt"; // The file name
    String FOLDERNAME = "sub"; // The folder name
    // The path to the file:
    String folder = getApplicationContext().getExternalFilesDir( type: null).getAbsolutePath()+ File.separator + FOLDERNAME
    File subFolder = new File(folder); // Here, we define a path to the file
```





Intro to File I/O

- Internal storage: Multiple lines

Code … ?

Intro to File I/O

https://www.journaldev.com/9412/android-shared-preferences-example-tutorial

- **Android Shared Preferences** overview
  - Shared Preferences allows activities and applications to keep preferences, in the form of key-value pairs like a Map that will persist even when the user closes the application
  - Android stores Shared Preferences settings as XML file in shared_prefs folder under DATA/data/{application package} directory
  - SharedPreferences is application specific, i.e., the data is lost on performing one of the following options:
    - on uninstalling the application
    - on clearing the application data (through Settings)
  - *The primary purpose of Shared Preferences is to store user-specified configuration details, such as user specific settings, keeping the user logged into the application*

Intro to File I/O

- **Android SQLite** is a lightweight database which comes with Android OS. Android SQLite combines a clean SQL interface with a very small memory footprint and decent speed. For Android, SQLite is “baked into” the Android runtime, so every Android application can create its own SQLite databases.
- SQLite is a typical relational database, containing tables (which consists of rows and columns), indexes, etc.

https://www.journaldev.com/9438/android-sqlite-database-example-tutorial

A simple Star Wars quiz

https://www.youtube.com/watch?v=8Qn_spdM5Zg

- Let’s see what you know about “Star Wars” :)
  - Star Wars: Episode IX – The Rise of Skywalker: Final Trailer - 2019

A simple Star Wars quiz

https://www.youtube.com/watch?v=XHk5kCIiGoM

- Let’s see what you know about “Star Wars” :)
  - Star Wars: Episode IV – A New Hope. Original Trailer (Restored) - 1976

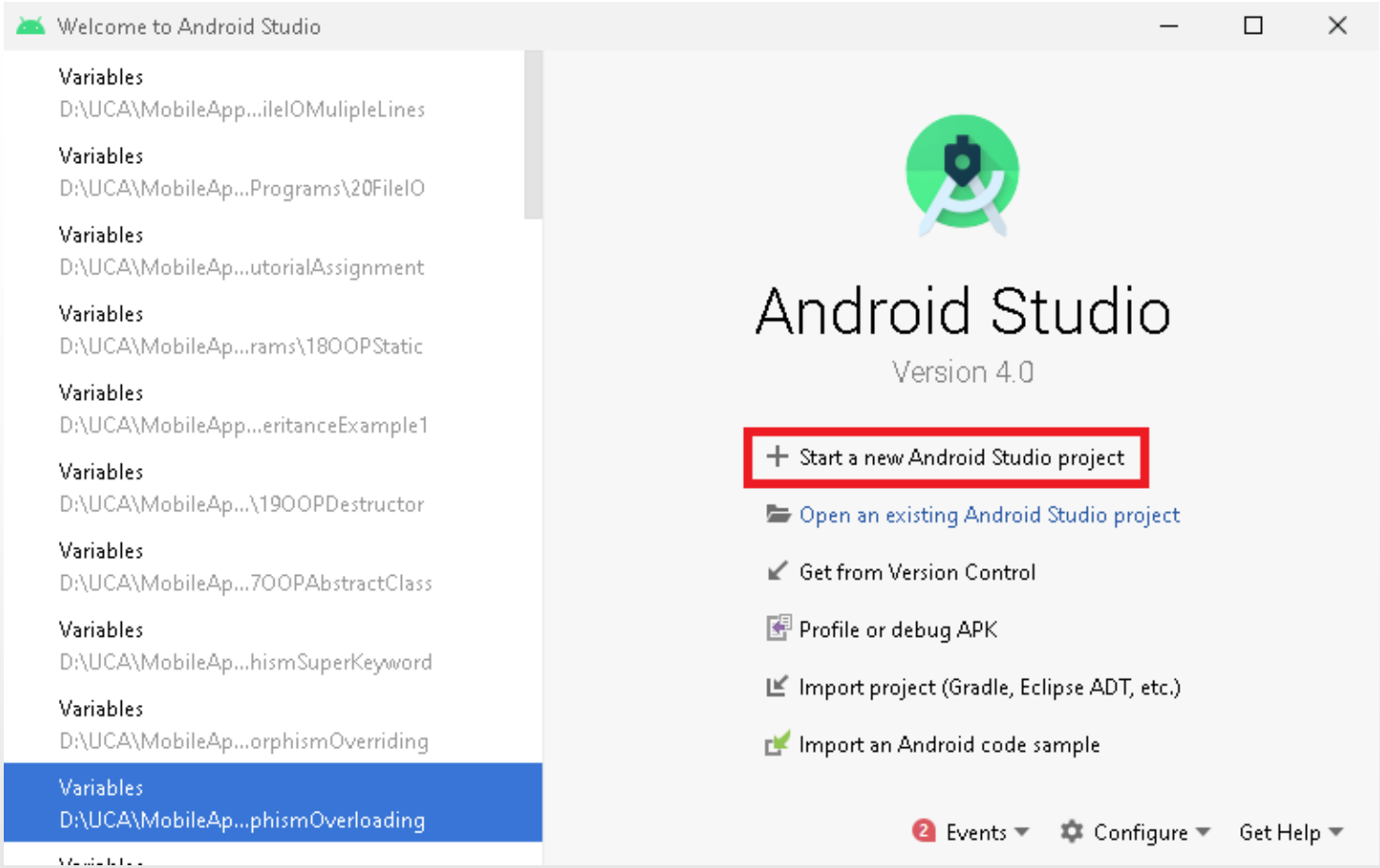
A simple Star Wars quiz

- Let’s see what you know about “Star Wars” :)

Are they different? (1977 and 2019 :)

A simple Star Wars quiz

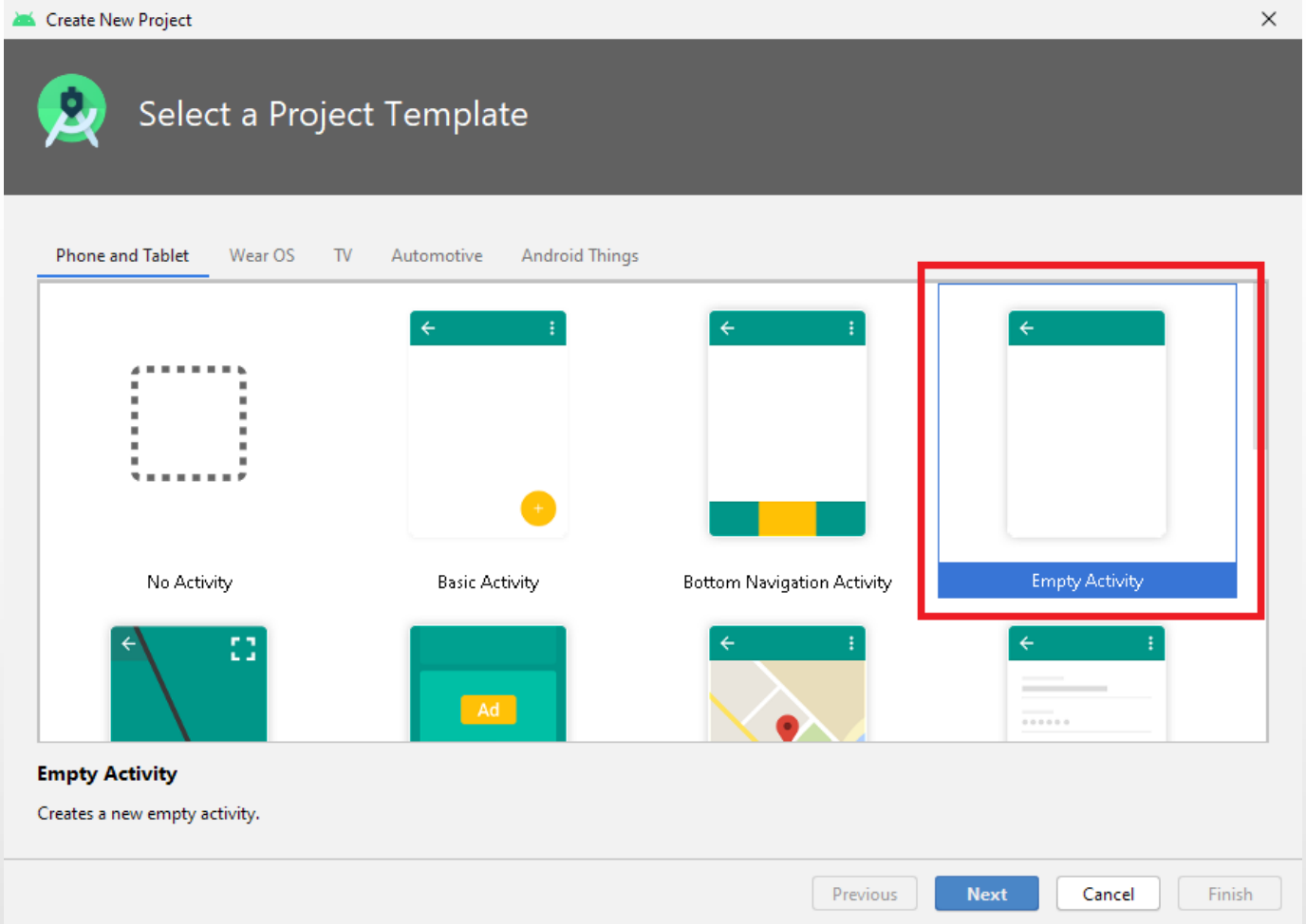
- Start a new Android Studio project:



https://www.androidauthority.com/make-a-star-wars-quiz-828959/

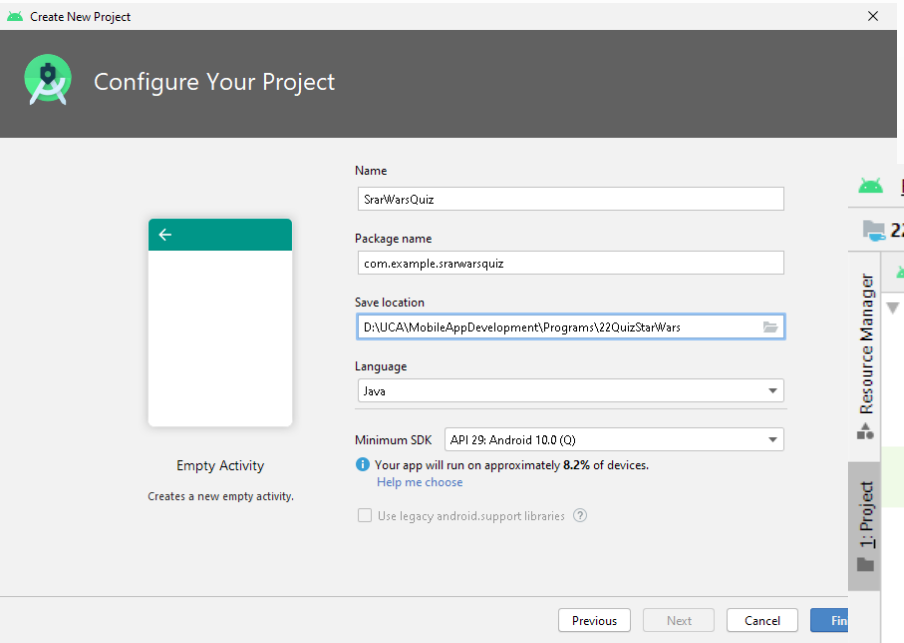
A simple Star Wars quiz

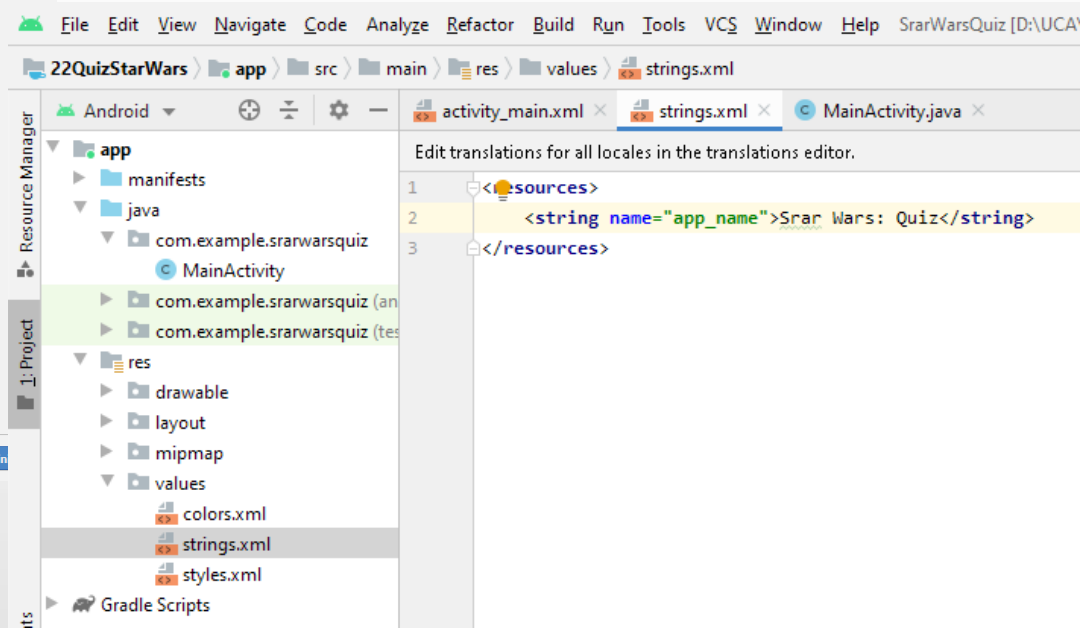
- Select a project template “Empty Activity”:



A simple Star Wars quiz

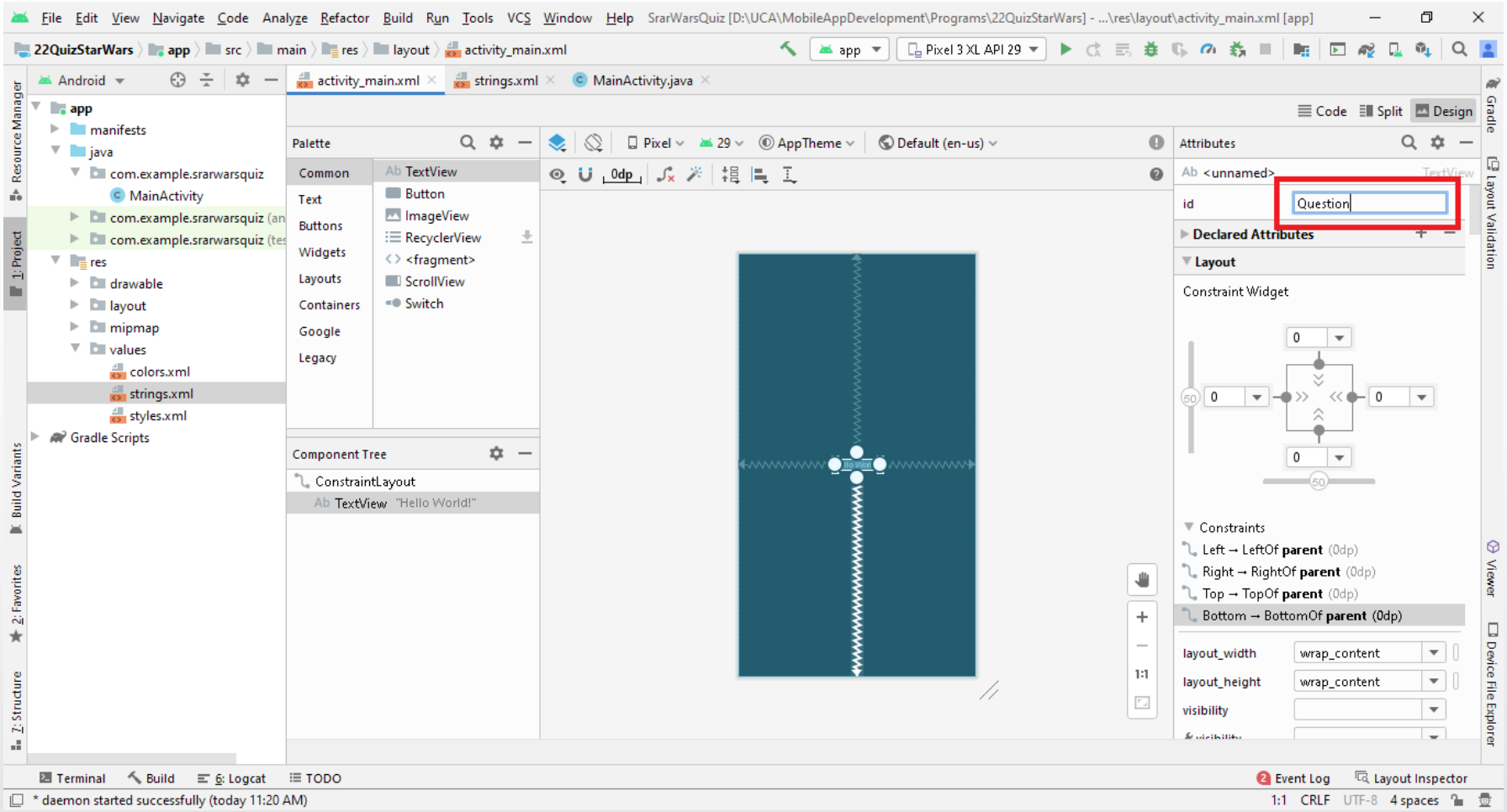
- Configure the project and change the name of the app in accord with your preferences:





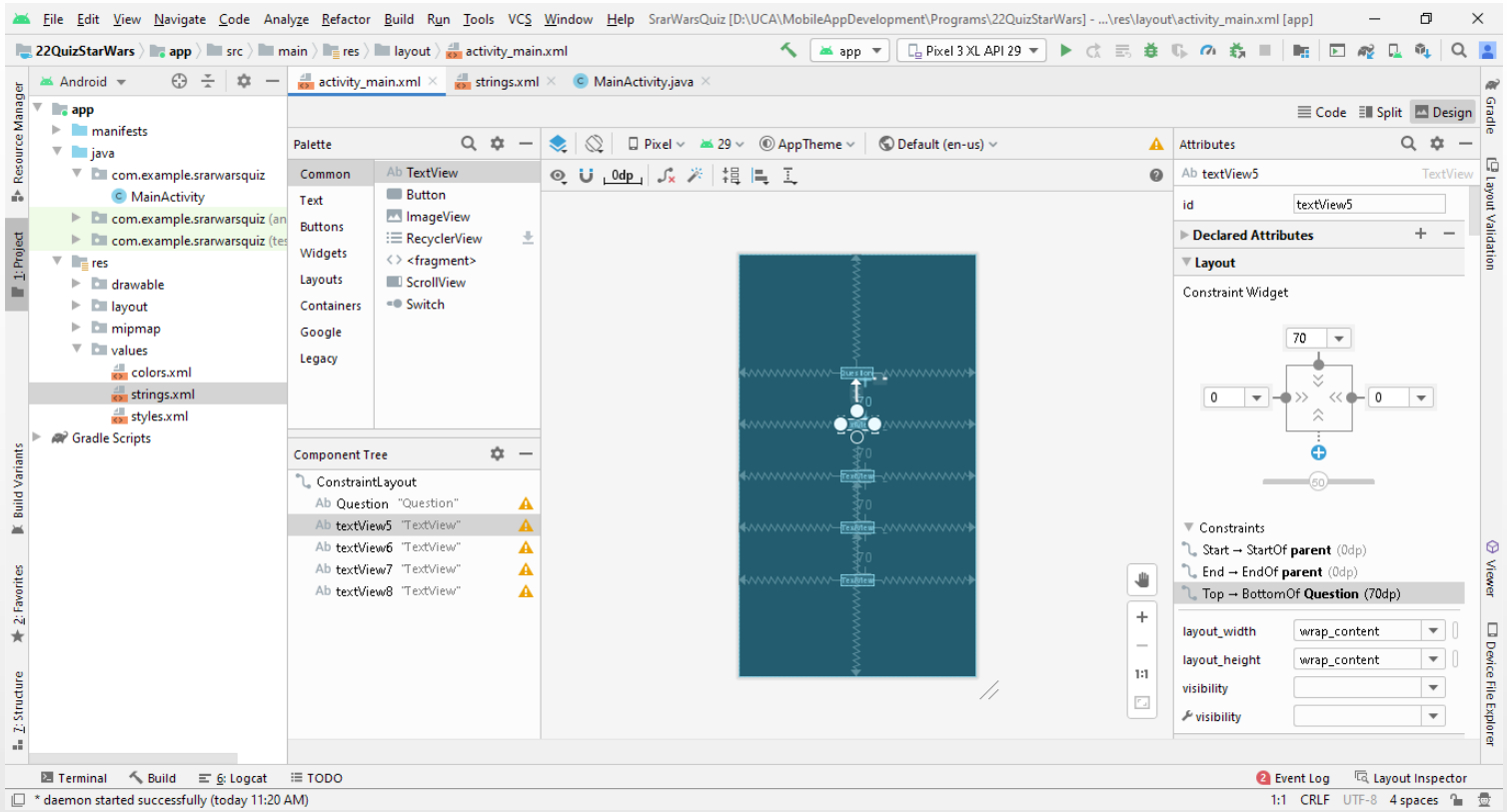
A simple Star Wars quiz

- Change the ID of the TextView element to "`Question`":



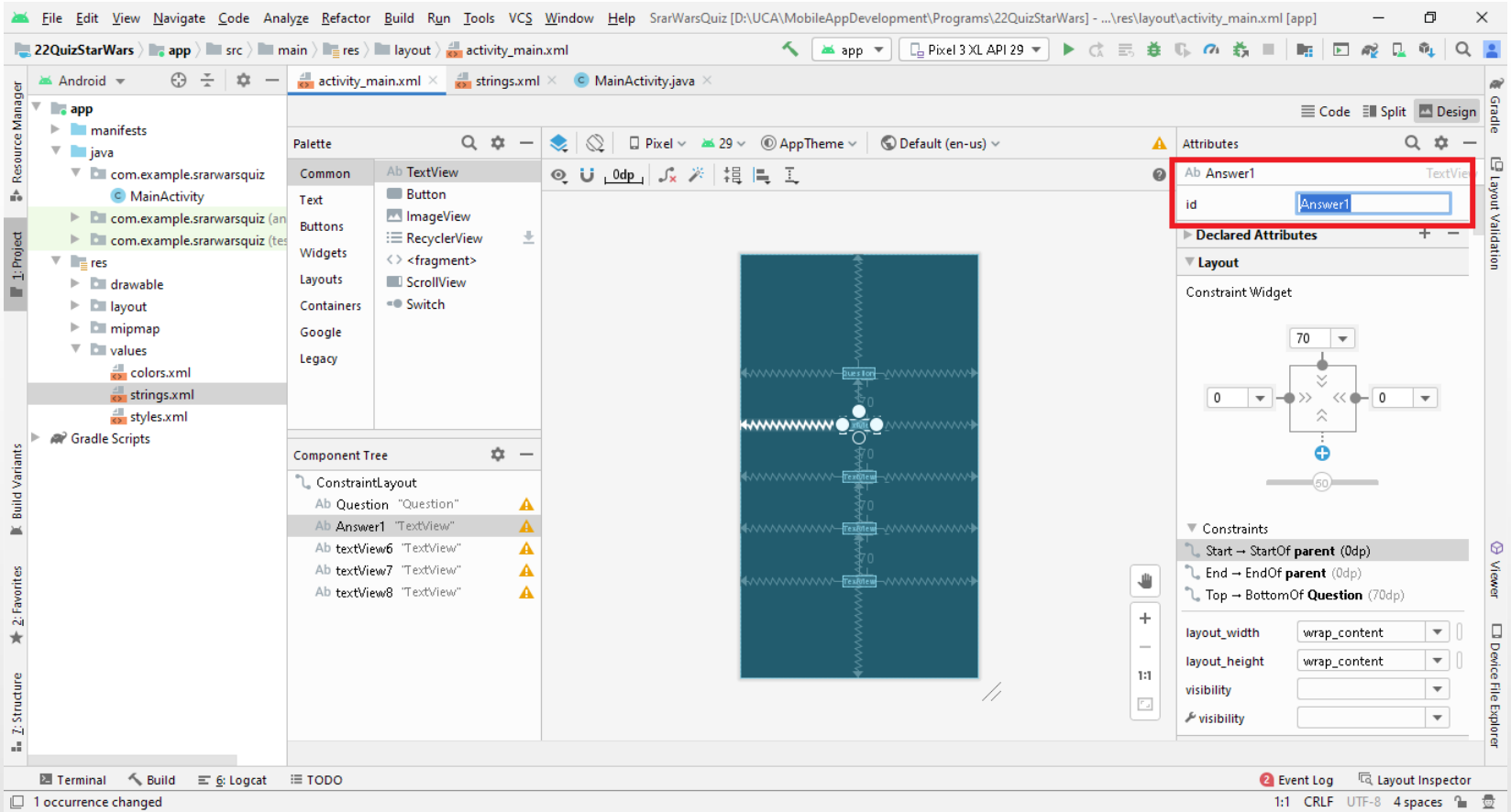
A simple Star Wars quiz

- Drag and drop four TextView elements and change the layout in accord with your preferences:



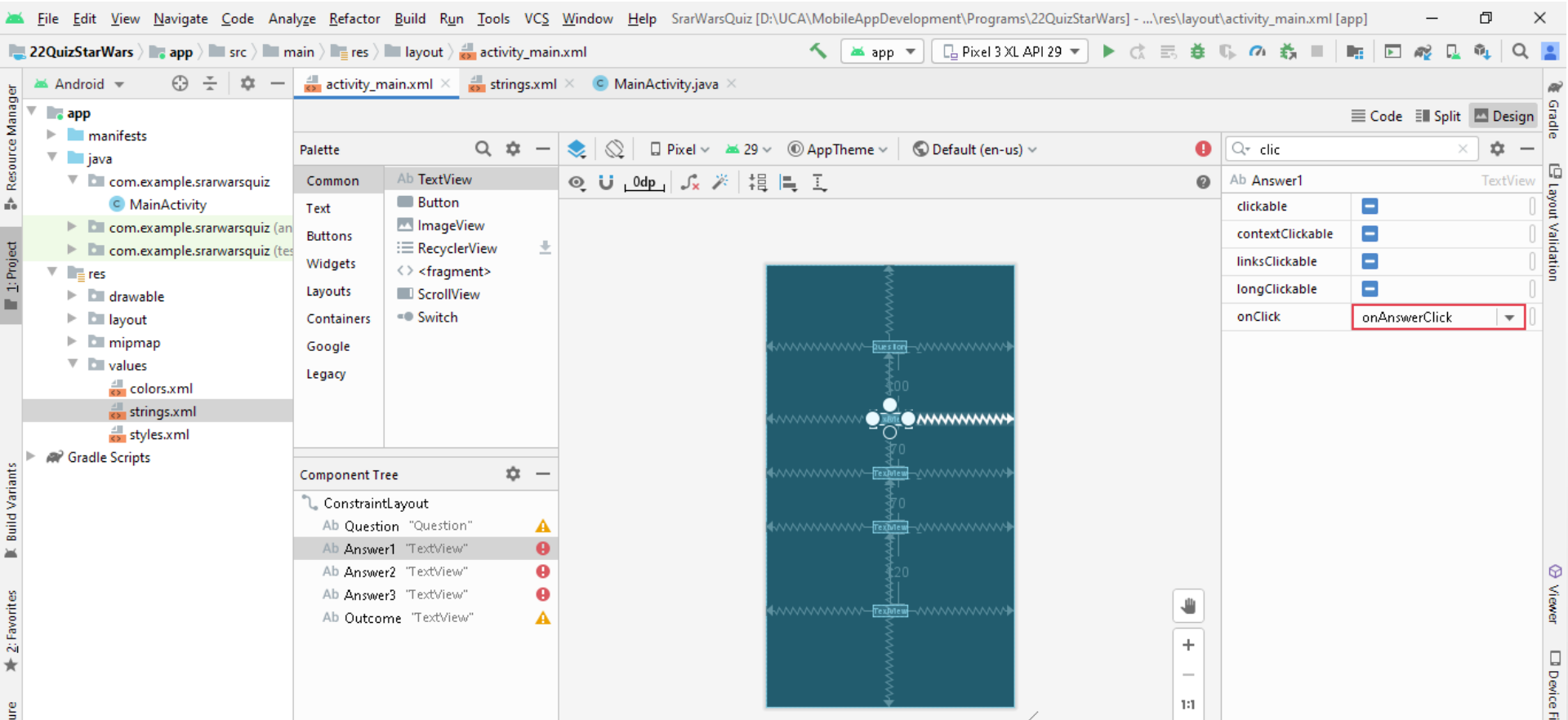
A simple Star Wars quiz

- Change IDs of new TextView elements to `Answer1`, `Answer2`, `Answer3`, and `Outcome`:



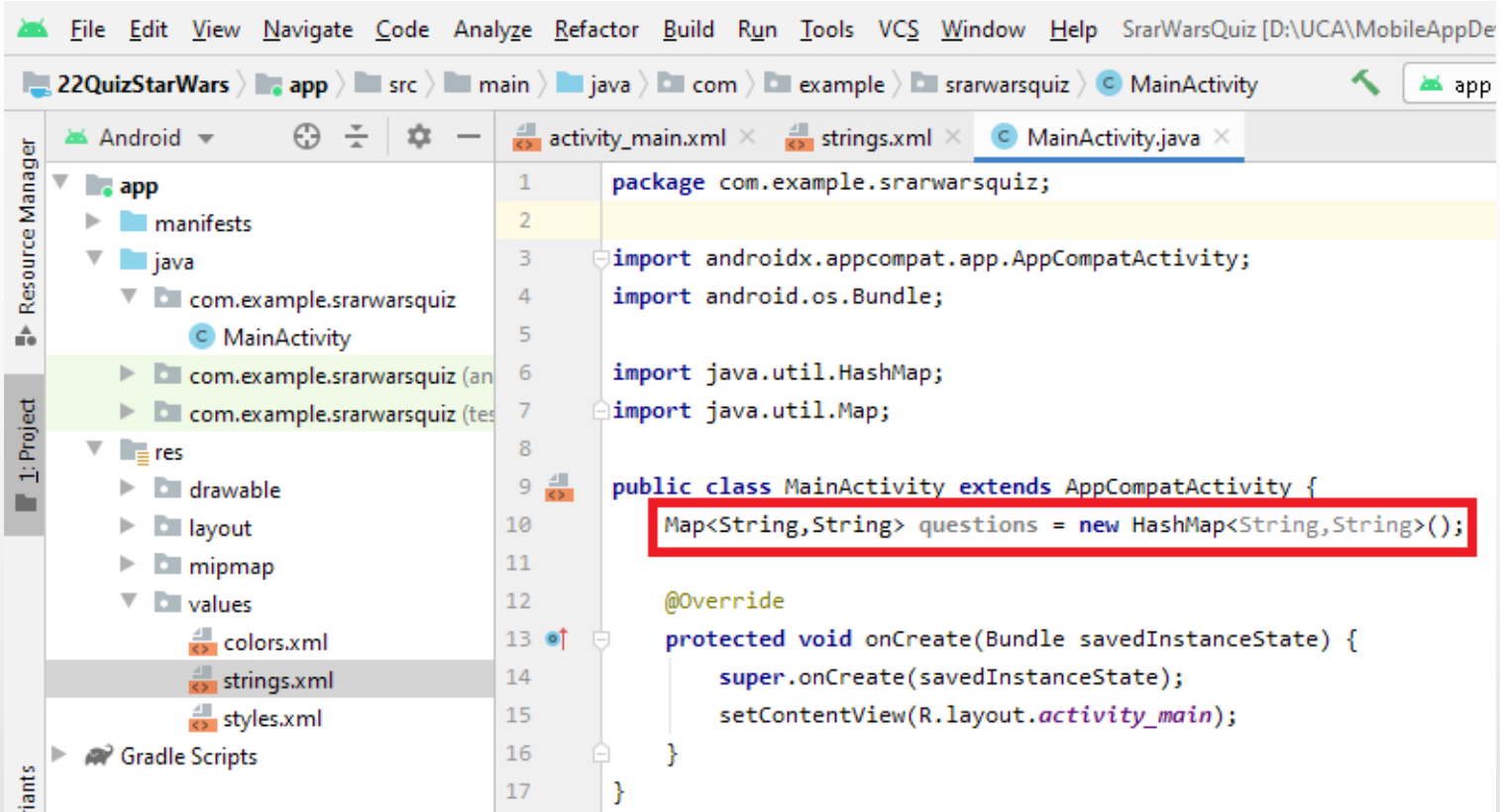
A simple Star Wars quiz

- Add the method `OnAnswerClick` to elements `Answer1`, `Answer2`, and `Answer3`:



A simple Star Wars quiz

- Declare a variable `questions` of type `Map`
  - A map stores multiple pairs using ‘key’ and ‘value’. In our case, ‘key’ and ‘value’ are of `String` type:



A simple Star Wars quiz

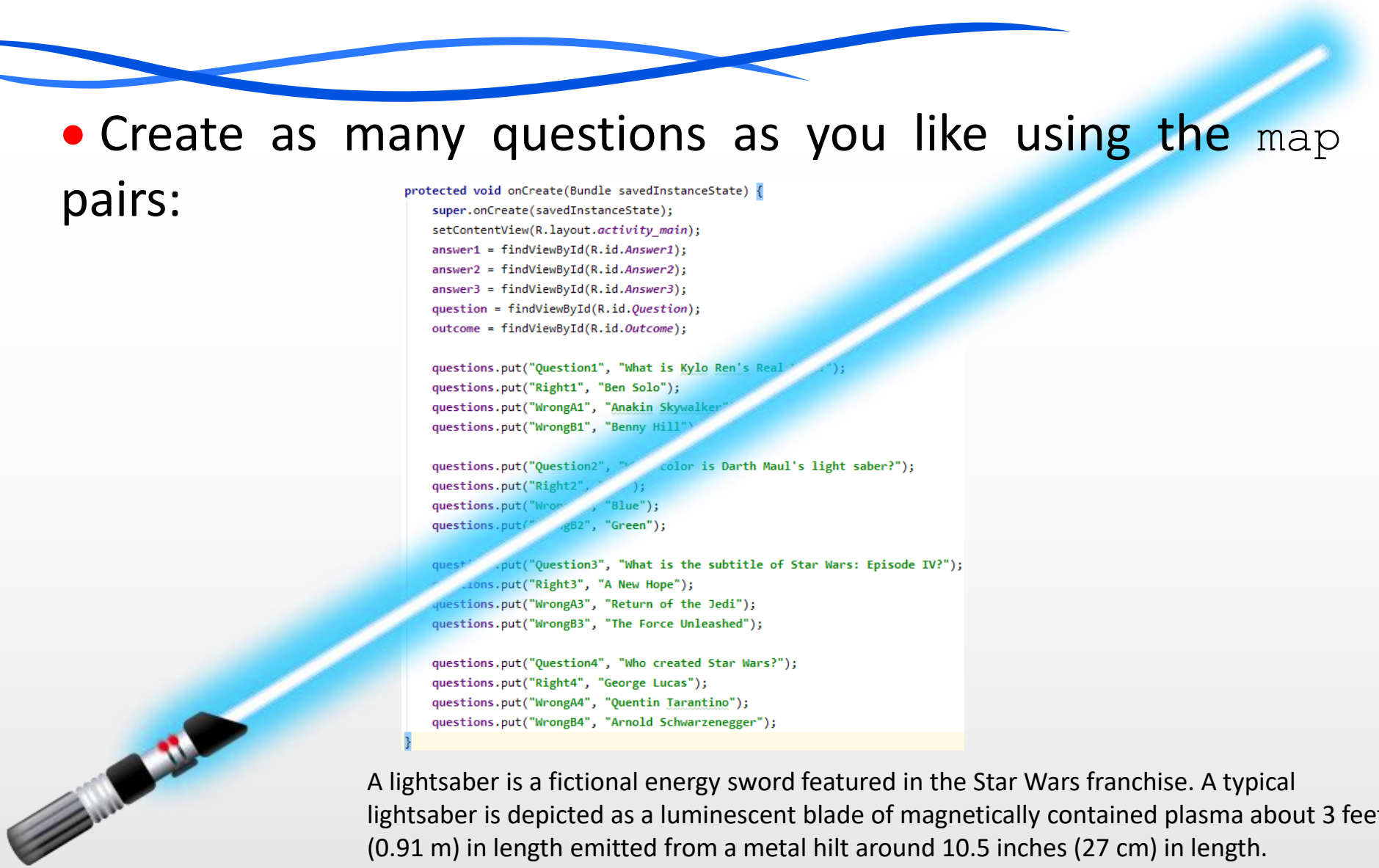
- Declare an integer variable `questionNo`, the current number of the question we work with, and an integer variable `NoOfQuestions`, the total number of questions, as well as write several findViewById:

```
9   public class MainActivity extends AppCompatActivity {
10      Map<String,String> questions = new HashMap<String,String>();
11      int questionNo = 1; // Number of the question that we work with
12      int NoOfQuestions = 4; // A total number of questions we work with
13      TextView question;
14      TextView answer1;
15      TextView answer2;
16      TextView answer3;
17      TextView outcome;
18
19      @Override
20      protected void onCreate(Bundle savedInstanceState) {
21          super.onCreate(savedInstanceState);
22          setContentView(R.layout.activity_main);
23          answer1 = findViewById(R.id.Answer1);
24          answer2 = findViewById(R.id.Answer2);
25          answer3 = findViewById(R.id.Answer3);
26          question = findViewById(R.id.Question);
27          outcome = findViewById(R.id.Outcome);
```

A simple Star Wars quiz

- Create as many questions as you like using the `map` pairs:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    answer1 = findViewById(R.id.Answer1);
    answer2 = findViewById(R.id.Answer2);
    answer3 = findViewById(R.id.Answer3);
    question = findViewById(R.id.Question);
    outcome = findViewById(R.id.Outcome);

    questions.put("Question1", "What is Kylo Ren's Real [illegible] ");
    questions.put("Right1", "Ben Solo");
    questions.put("WrongA1", "Anakin Skywalker"[illegible]
    questions.put("WrongB1", "Benny Hill"[illegible]

    questions.put("Question2", "[illegible] color is Darth Maul's light saber?");
    questions.put("Right2", [illegible] );
    questions.put("Wron[illegible], "Blue");
    questions.put("[illegible]gB2", "Green");

    quest[illegible].put("Question3", "What is the subtitle of Star Wars: Episode IV?");
    [illegible]ions.put("Right3", "A New Hope");
    questions.put("WrongA3", "Return of the Jedi");
    questions.put("WrongB3", "The Force Unleashed");

    questions.put("Question4", "Who created Star Wars?");
    questions.put("Right4", "George Lucas");
    questions.put("WrongA4", "Quentin Tarantino");
    questions.put("WrongB4", "Arnold Schwarzenegger");
}
```



A lightsaber is a fictional energy sword featured in the Star Wars franchise. A typical lightsaber is depicted as a luminescent blade of magnetically contained plasma about 3 feet (0.91 m) in length emitted from a metal hilt around 10.5 inches (27 cm) in length.

A simple Star Wars quiz

- At the end of the callback `onCreate`, set the color of the text in TextView elements in accord with your preferences, as well as invoke method `setQuestion()`:

```
question.setTextColor(Color.MAGENTA);
answer1.setTextColor(Color.GRAY);
answer2.setTextColor(Color.GRAY);
answer3.setTextColor(Color.GRAY);

setQuestion();
```

A simple Star Wars quiz

- Method `setQuestion()` generates the random order of the answers to the current question:

```
void setQuestion(){ // This method generates the random order of the answers to the current question
    question.setText(questions.get("Question" + questionNo));
    Random r = new Random();
    int RandomIntValue = r.nextInt( bound: 3);
    if (RandomIntValue == 0) { // We have 3 different sets of the questions' orders that are chosen randomly
        answer1.setText(questions.get("Right" + questionNo));
        answer1.setTag("Correct");answer2.setTag("Wrong");answer3.setTag("Wrong");
        answer2.setText(questions.get("WrongA" + questionNo));
        answer3.setText(questions.get("WrongB" + questionNo));
    }
    if (RandomIntValue == 1) {
        answer2.setText(questions.get("Right" + questionNo));
        answer2.setTag("Correct");answer1.setTag("Wrong");answer3.setTag("Wrong");
        answer1.setText(questions.get("WrongA" + questionNo));
        answer3.setText(questions.get("WrongB" + questionNo));
    }
    if (RandomIntValue == 2) {
        answer3.setText(questions.get("Right" + questionNo));
        answer3.setTag("Correct");answer1.setTag("Wrong");answer2.setTag("Wrong");
        answer2.setText(questions.get("WrongA" + questionNo));
        answer1.setText(questions.get("WrongB" + questionNo));
    }
}
```

A simple Star Wars quiz

- The method `onAnswerClick` analyzes the choice of the user:

```
public void onAnswerClick(View v){
// This method analyzes the choice of the user:
// Correct choice - "Well done!"
// Wrong choice - "You can play again :)"
// If all questions were answered, the message "You successfully finished the quiz!\nClick me to exit :)"
// is shown. The user has to tap this message to exit the program.
    if (v.getTag()=="Correct"){
        questionNo++;
        if (questionNo>NoOfQuestions) {
            outcome.setTextColor(Color.GREEN);
            outcome.setText("You successfully finished the quiz!\nClick me to exit :)");
            question.setVisibility(View.INVISIBLE); // Here, we hide question
            answer1.setVisibility(View.INVISIBLE); // Here, we hide answer1
            answer2.setVisibility(View.INVISIBLE); // Here, we hide answer2
            answer3.setVisibility(View.INVISIBLE); // Here, we hide answer3
        }
        else {
            outcome.setTextColor(Color.GREEN);
            outcome.setText("Well done!");
            setQuestion();
        }
    } else{
        outcome.setTextColor(Color.RED);
        outcome.setText("You can play again :)");
    }
}
```

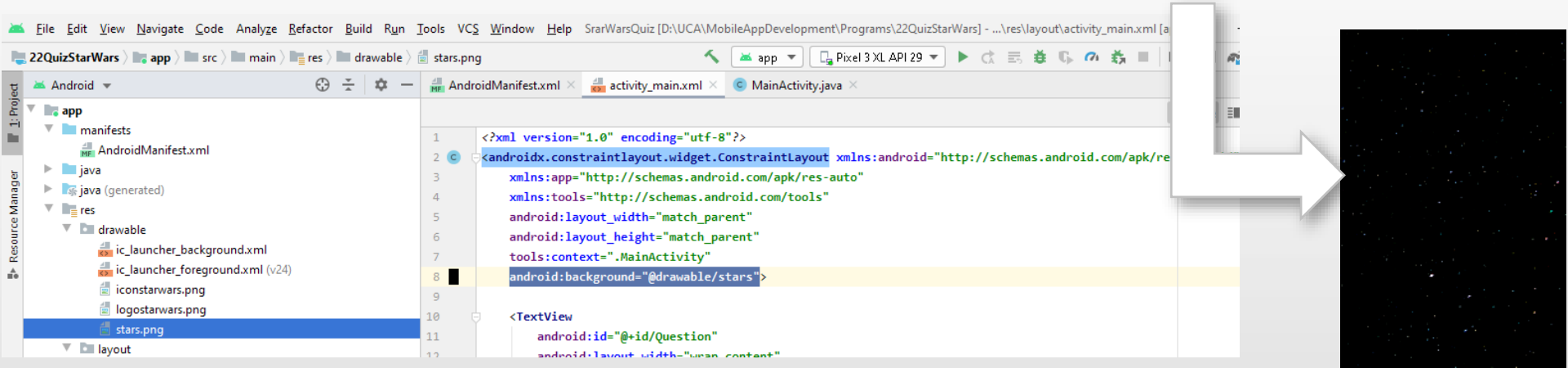
A simple Star Wars quiz

- Add the method `onClickFinish` to the element `Outcome` to terminate the program when the end-user taps the message “"You successfully finished the quiz!”:

```
public void onClickFinish(View v) {
    if (outcome.getText().toString().contains("successfully")==true) {finish(); System.exit( status: 0);}
}
```

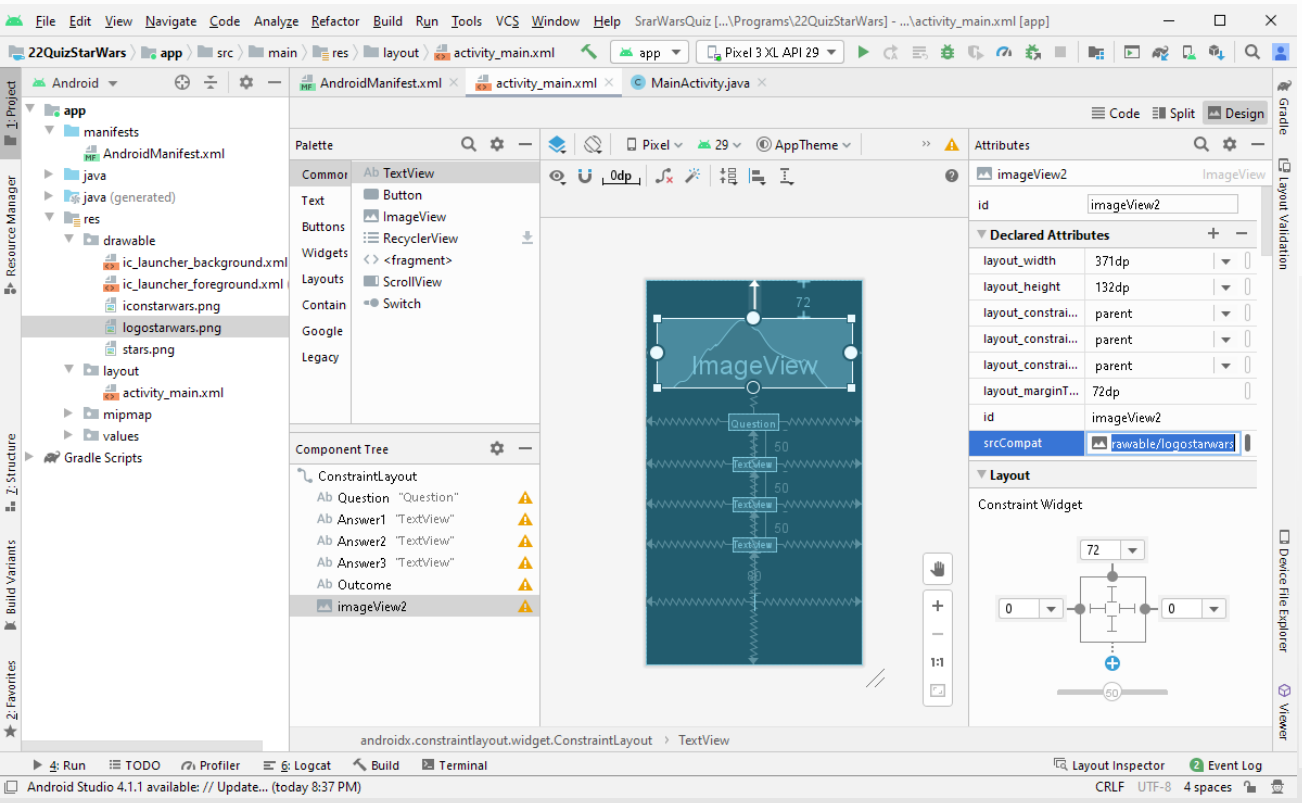
- Add background image in accord with your preferences:

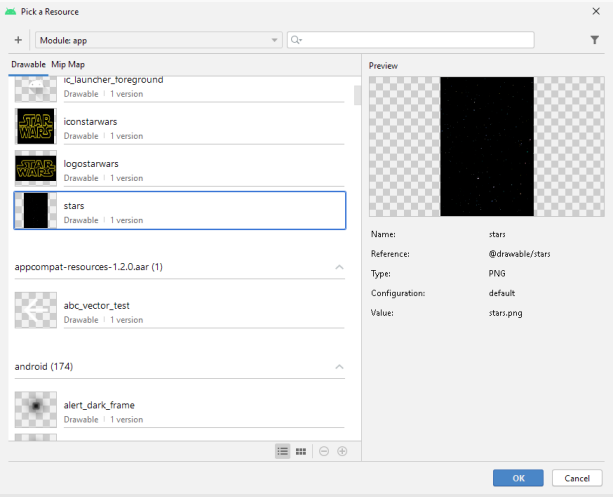
`android:background="@drawable/stars"`



A simple Star Wars quiz

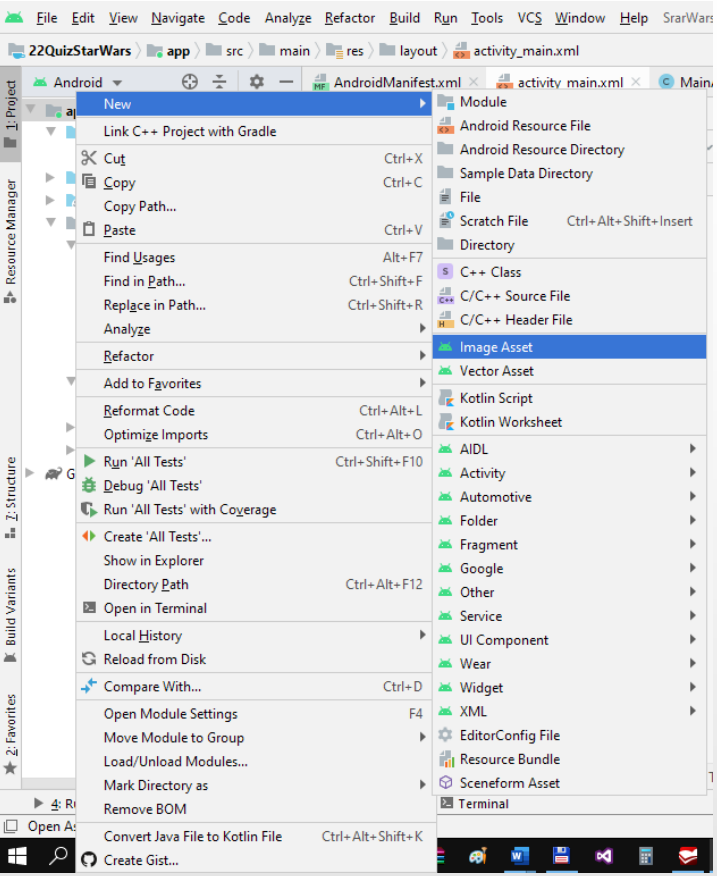
- Drag and drop an ImageView element and associate it with an image `logostarwars.png`. Change the layout in accord with your preferences:

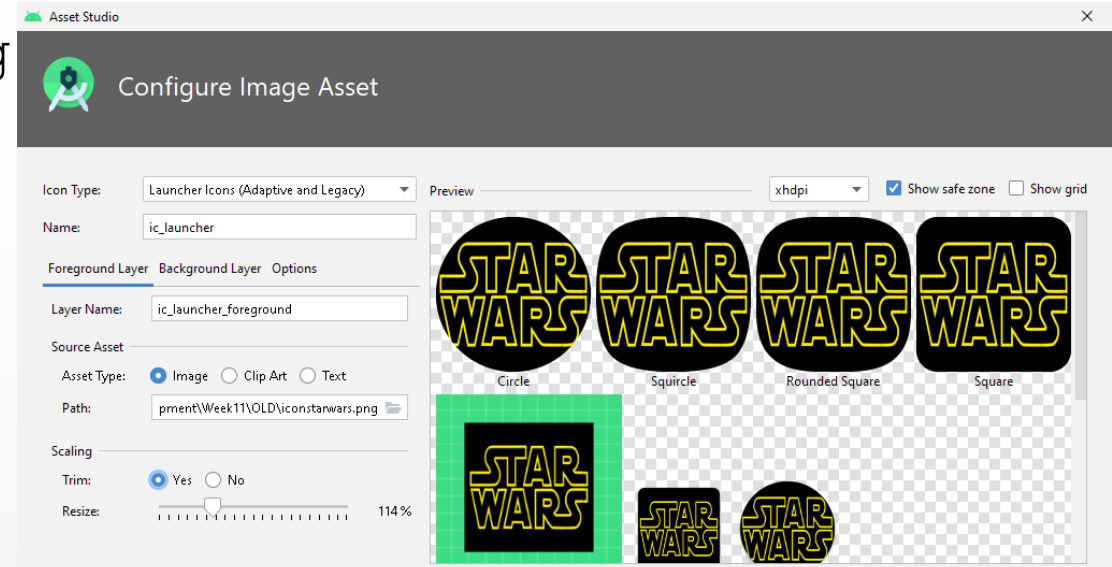


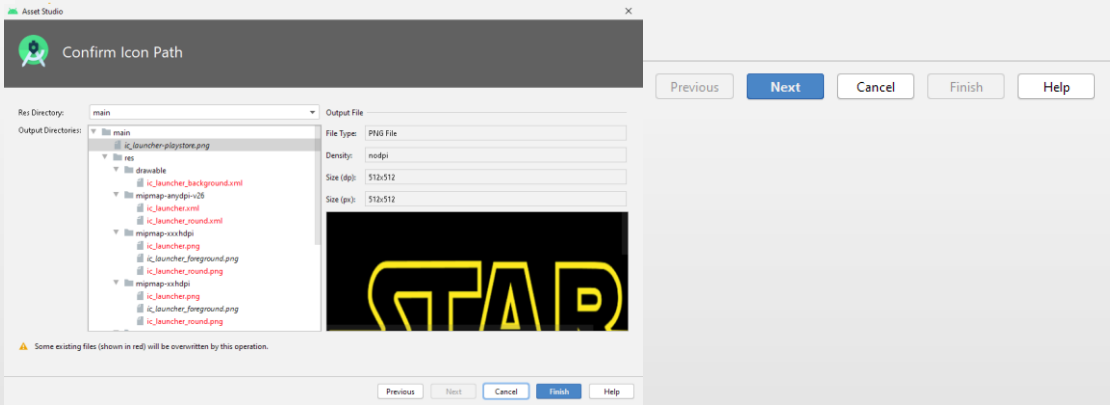


A simple Star Wars quiz

- Change the app icon using the Android Image Asset and the `logostarwars.png`



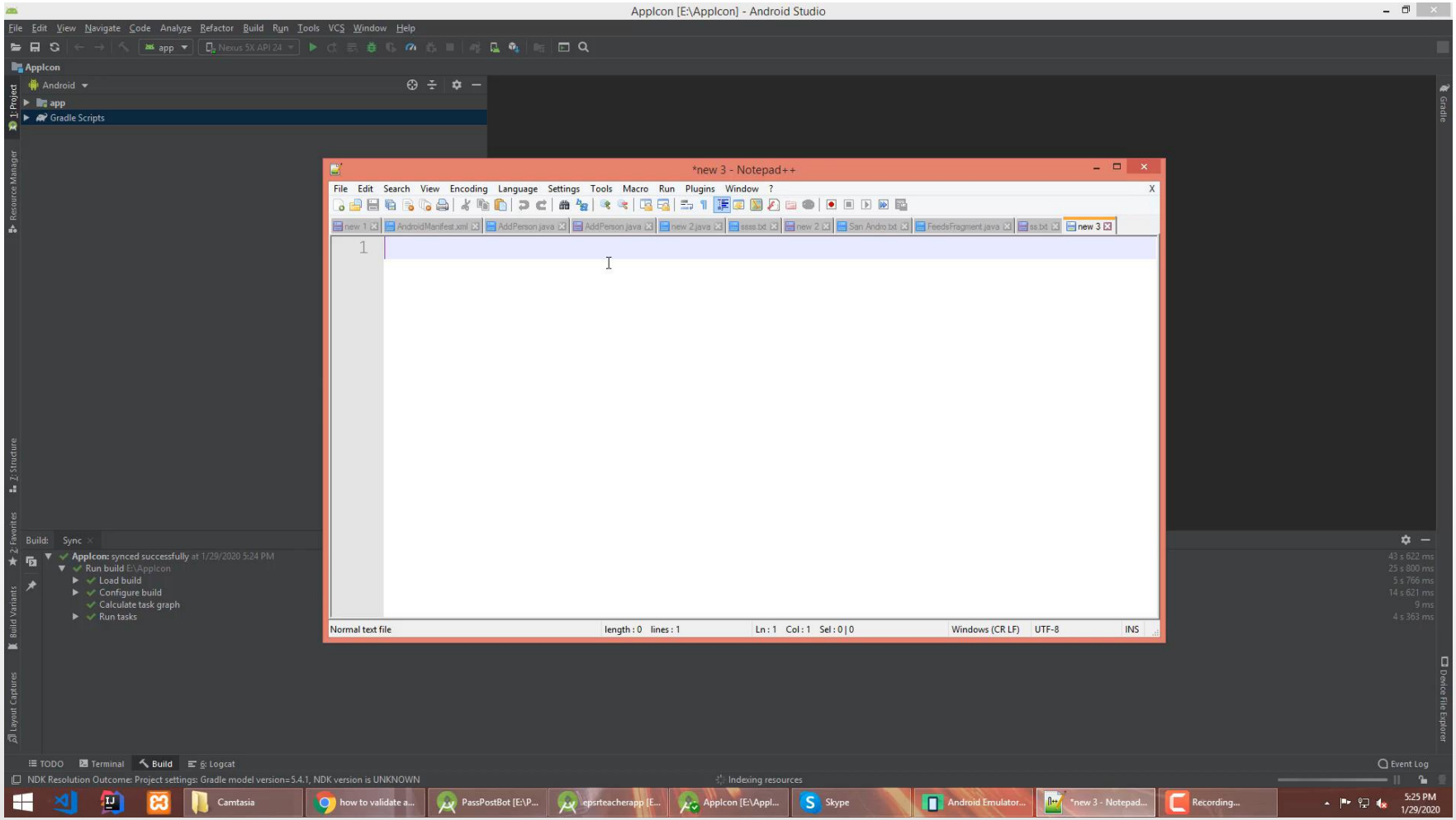




A simple Star Wars quiz

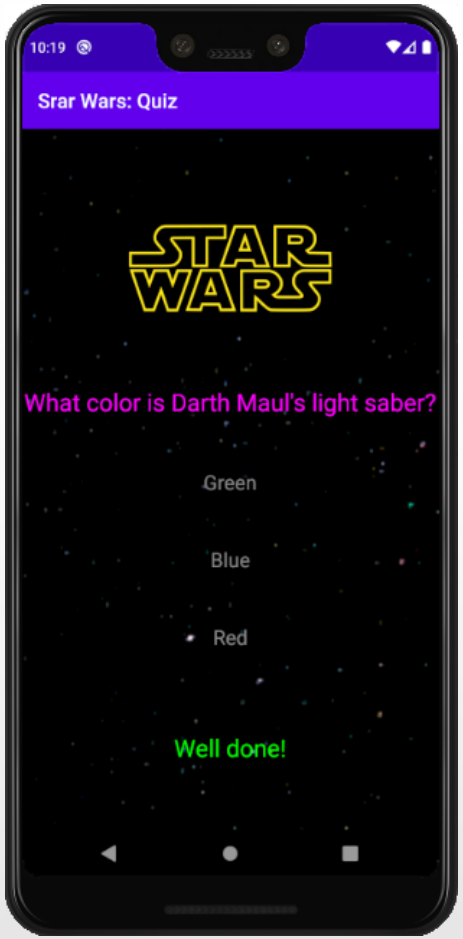
https://www.youtube.com/watch?v=zuAhhliMLak

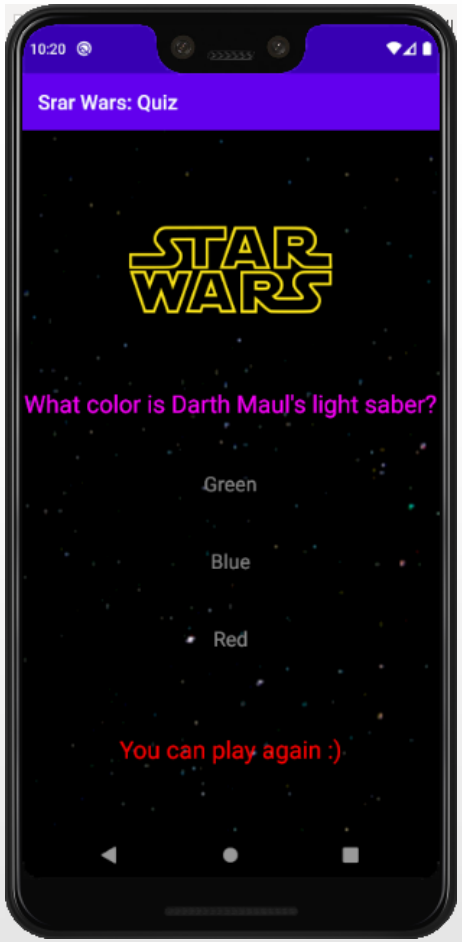
- How to change the app icon in Android Studio:



A simple Star Wars quiz

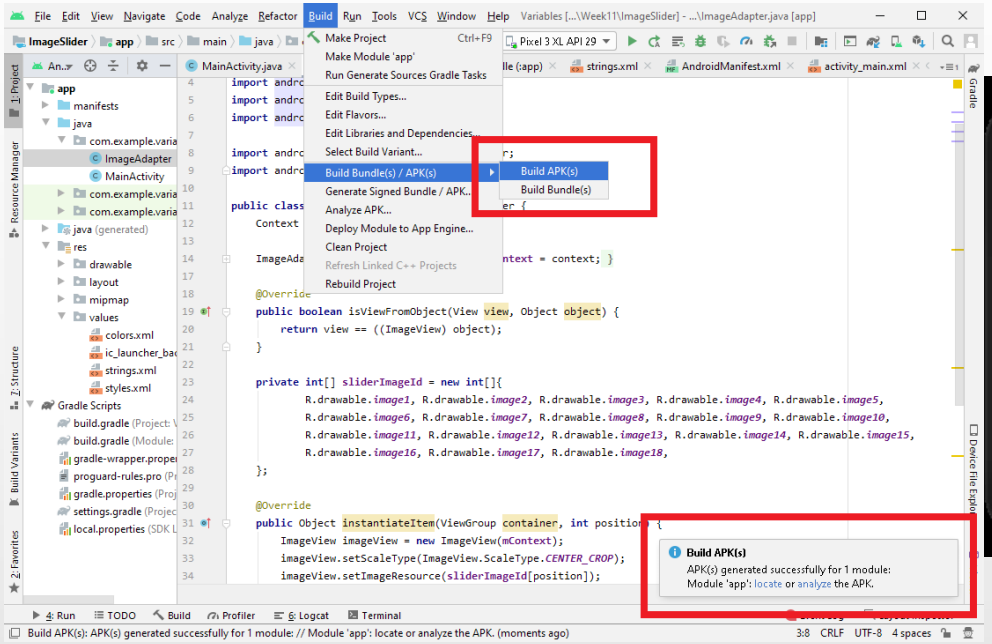
- Let’s answer the quiz :)





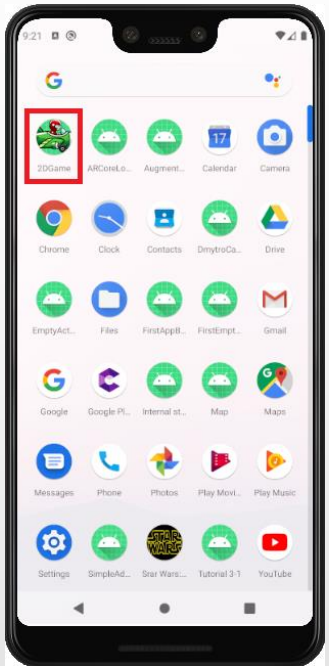
In-class activity

- Develop a Java Android app to display photos. Run this app on the Android physical/virtual device from the Android Studio or copy .apk file onto the Android smartphone.



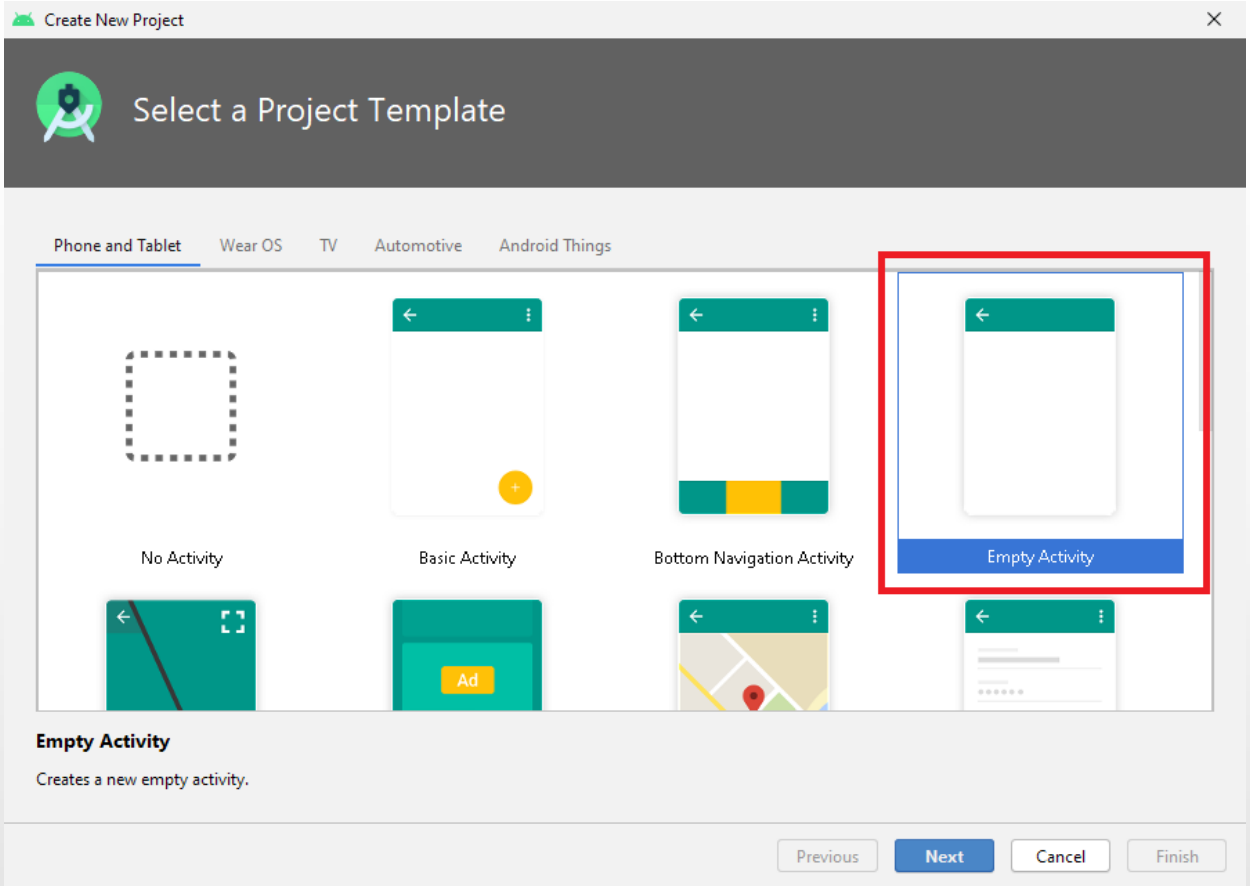
2D game in Android Studio

- We will develop a 2D game on the Java Android platform
  - We will be creating a game where a flight has to shoot some angry birds. If the flight misses shooting any bird, then the game is over. If the bird hits the flight, then the game is over as well.



2D game in Android Studio

- Start a new Android Studio project and select a project template “Empty Activity”:



2D game in Android Studio

- Development of 2D game on the Java Android platform includes five steps:

1. Start a new Android Studio project using a project template “Empty Activity”

https://www.youtube.com/watch?v=aTT4GfojkHA

2. Moving background:

https://www.youtube.com/watch?v=RQoT9BUsl1Q

3. Flight is moving up and down:

https://www.youtube.com/watch?v=EBJDo9a1q-o

4. Animated bullets and shots:

https://www.youtube.com/watch?v=tFGtx7Ikfbo
https://www.youtube.com/watch?v=5q8CkErKNyg

5. Final steps:

https://www.youtube.com/watch?v=5W3rVBDYFjE

Do you have any questions or comments?

Thank you
for your attention !

In this presentation:

- Some icons were downloaded from flaticon.com and iconscout.com