Let's Improve The Quizzler!

Android and MVC

- **Model-View-Controller**
  - Application development pattern that divides an app into three parts
- **Classes**
  - A template that describes the behavior of an object
- **Objects**
  - An instance of a class that has a state and behavior

Improving The Quizzler

A one question quiz app is not very challenging.

We'll create a class called "Question" that will instantiate an array of Question objects that MainActivity will manage.

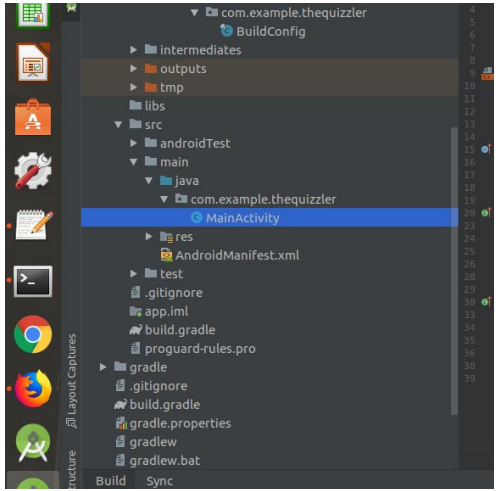
Open Android Studio

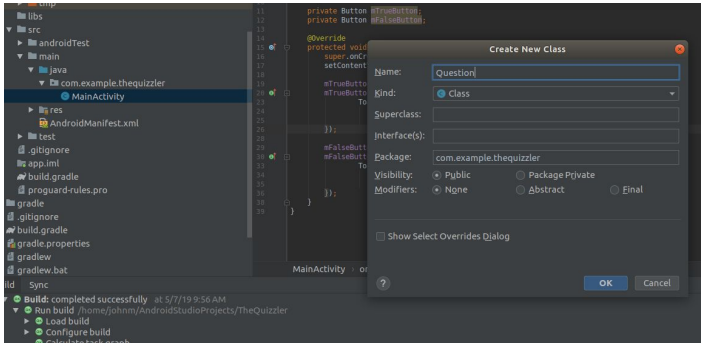
and Open an existing Android Studio Project if the Quizzler app doesn't open automatically

Maydm

Create a New Class

Right click on MainActivity in the Project view and select New -> Java Class
Name the class Question





Maydm

Edit Question.java

The previous action will create a new Class called Question.
Now we will create two new private variables and a Question constructor.

```
package com.example.thequizzler;

public class Question {
    private int mTextResId;
    private boolean mAnswerTrue;

    public Question(int textResId, boolean answerTrue) {
        mTextResId = textResId;
        mAnswerTrue = answerTrue;
    }
}
```

Maydm

What did we just write?

We have created a Question class that contains two pieces of data.
question text and the question answer. The answer is a boolean, true or false.

You may have noticed that `mTextResId` is an integer and not a String.
That's because Android will use the resource ID (an integer) to serve the user the question (a string).

We'll create question resource lists in a little while.

Getters and Setters

- **Getters and Setters are methods used to manipulate private variables**
  - Get methods returns variable values
  - Set methods set values

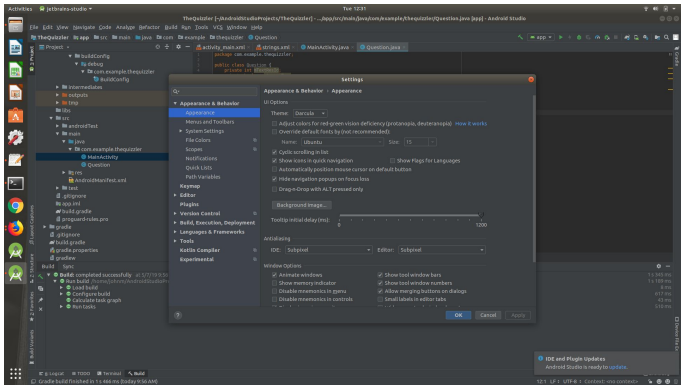
- **These are attributes of Encapsulation**
  - Allows better control of class methods and attributes
  - We can create read-only or write-only Class variables by omitting `set` or `get` methods, respectively
  - Programmers can change one part of legacy code without affecting other parts
  - Increases data security

Maydm

“m” is for member

Did you notice the “m” prefix in the private variables? That is a prefix that stands for “member”. We can configure Android Studio to understand the “m” prefix as member variables.

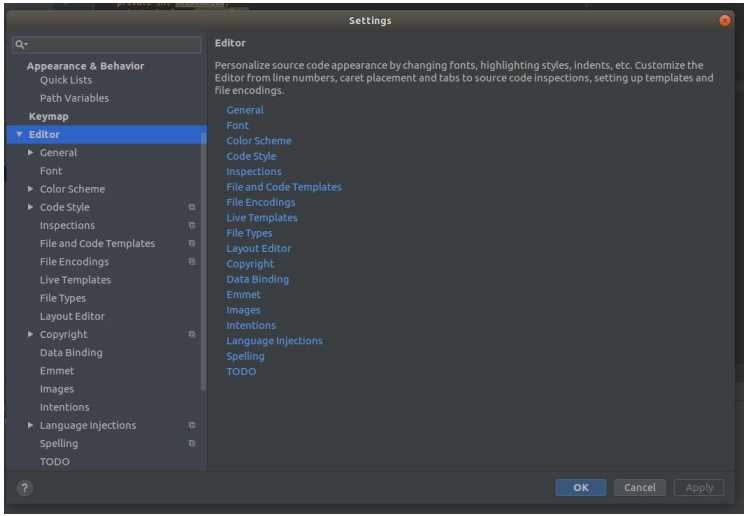
- **Open Android Studio preferences by clicking on File -> Settings on Windows or the “Android Studio” menu on Macs**

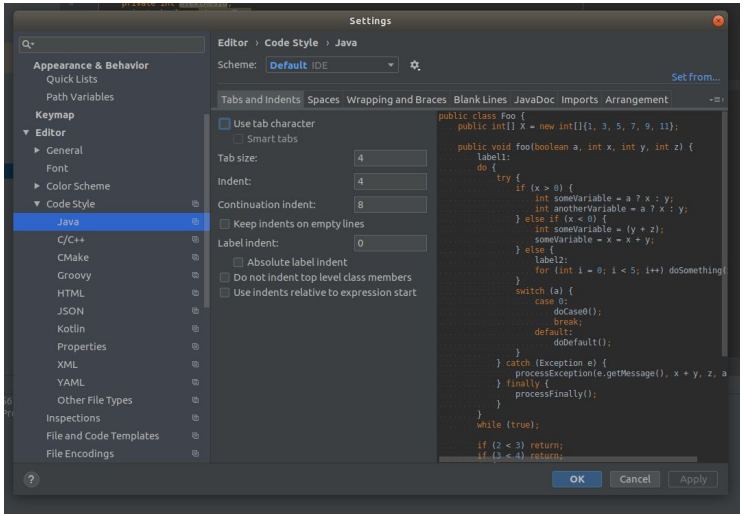


Maydm

Set style preferences

- **Expand Editor, then Code Style, then select Java**

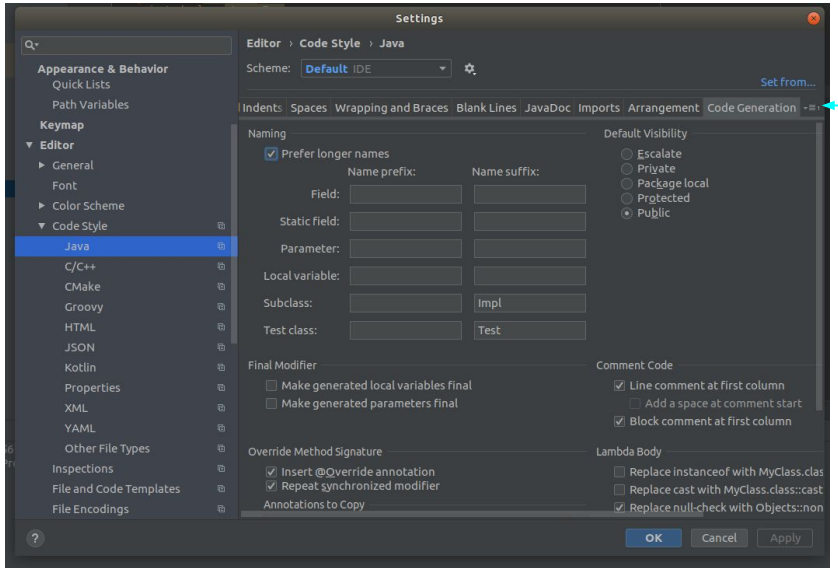




Maydm

Set style preferences

- **Select the “Code Generation” Tab**
- **It is probably hiding to the right of the “Arrangement” tab**

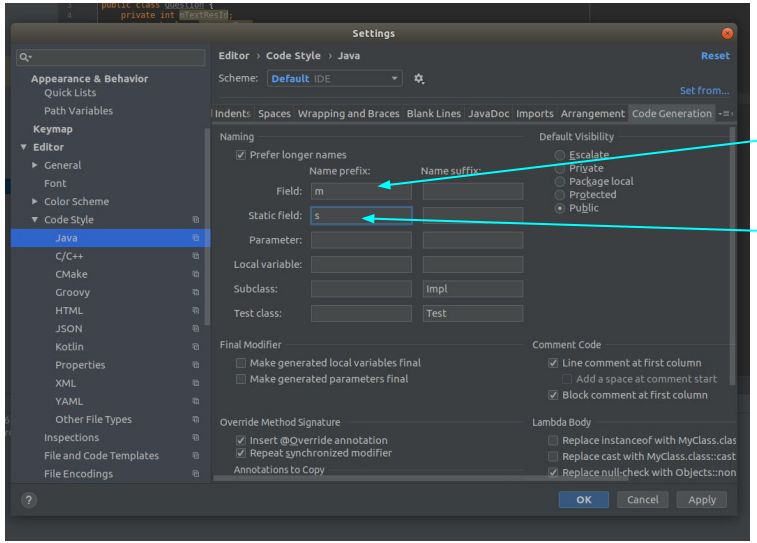


To find “Code Generation” Tab click this button.

Maydm

Set style preferences

- **Instruct Android Studio to recognize the “m” prefix**
- **Click “Apply” and then “OK”**



Add “m” in the Field Name Prefix Box

Add “s” in the Static field Name Prefix Box

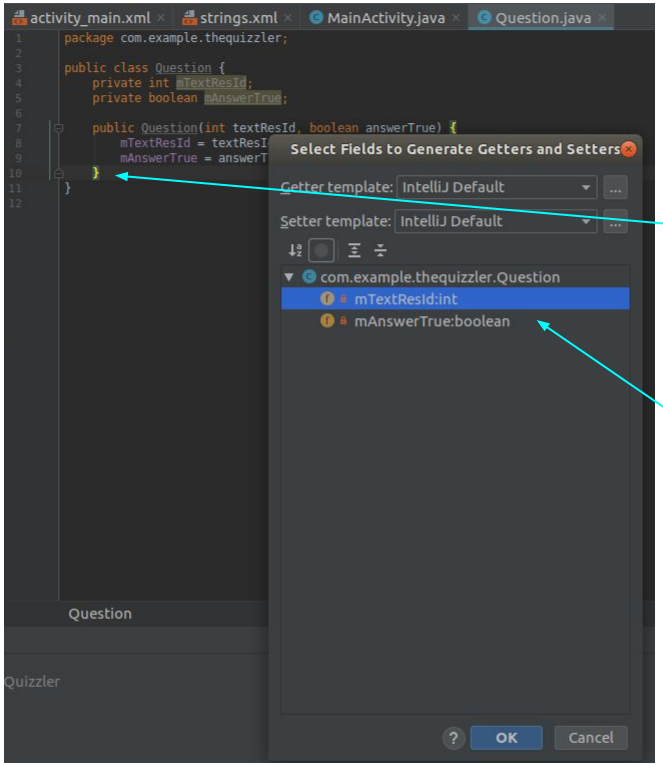
Maydm

What Prefixes do for Programmers

- **Remember the code we wrote in Questions.java?**
- **Now when we ask Android to generate a getter for `mTextRedId` it will produce `getTextResId()` instead of `getMTextResId()`**

Maydm

Automatically Generate Getters and Setters



- Make Android Studio work for you
- Go to Question.java and right-click at the end of the constructor closing squiggly bracket.
- Select “Generate” and then “Getter and Setter”
- Select both “mTextRedId:int” and “mAnswerTrue:boolean”
- Click “OK”

Maydm

Automatically Generate Getters and Setters

activity_main.xml | strings.xml | MainActivity.java | Question.java

```
package com.example.thequizzler;

public class Question {
    private int mTextResId;
    private boolean mAnswerTrue;

    public Question(int textResId, boolean answerTrue) {
        mTextResId = textResId;
        mAnswerTrue = answerTrue;
    }

    public int getTextResId() {
        return mTextResId;
    }

    public void setTextResId(int textResId) {
        mTextResId = textResId;
    }

    public boolean isAnswerTrue() {
        return mAnswerTrue;
    }

    public void setAnswerTrue(boolean answerTrue) {
        mAnswerTrue = answerTrue;
    }
}
```

Android Studio automatically populates the Get and Set methods for the Question Class.

Maydm

Model-View-Controller: Model Objects

We are creating an MVC Framework to better manage the different parts of The Quizzler App. In MVC Frameworks all objects must me one of three types of objects. A model object, view object, or a controller object.

Model Objects hold the data and business logic. Model objects no nothing about the app's UI, they only hold and manage data.

All of the model objects create what is known as a **model layer**.

Model classes are usually custom classes created by the programmer.

In The Quizzler the model layer consists of the **Question** class.

Model-View-Controller: View Objects

View Objects draw on the screen and respond to user input, such as touches and swipes. “If you see it on the screen, it’s a View Object.”

Android provides several customizable view classes. Programmers can create their own view classes, as well.

All of the view objects combine to make the **view layer**.

The Quizzler’s view layer consists of the widgets in *activity_quiz.xml*

Maydm

Model-View-Controller: Controller Objects

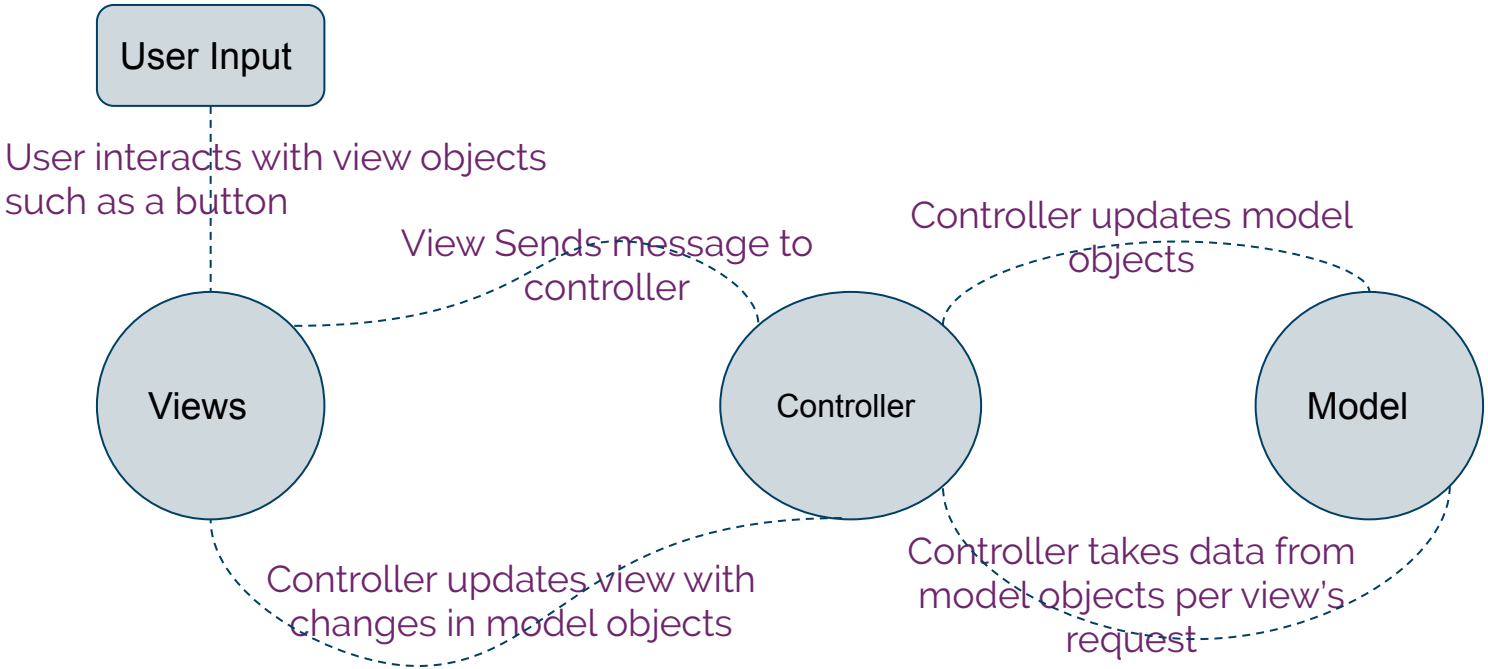
Controller Objects connect the view and model objects. The view and model never communicate directly with one another.

Controller Objects contain application logic. Controllers respond to events that are triggered by view objects, and they manage the flow of data to and from model objects and the view layer.

Often, controllers are subclasses of **Activities**, **Fragments**, and **Services**.

Currently, The Quizzlers controller layer is composed entirely of *QuizActivity*.

Model-View-Controller: Diagram



Maydm

Why Programmers Use MVC

Sometimes a code base can grow so large with features that it becomes unmanageable. When a bug is discovered in the code, an unstructured codebase can be very difficult to debug.

Breaking code into classes can help programmers think about the code holistically. Sometimes it can help to think of groups of layers rather than individual classes.

MVC makes classes easier to reuse, and can help us forget about individual classes when we consider MVC layers.