***Notes on “Android Workshop” 2***

**Step 2: Data Encapsulating (compartmentalizing), MVC Paradigm**

* Add a TrueFalse class with fields for ID and isTrue, constructor, and setters and getters (auto-gen).
* Add appropriate string resource for the next button
* Add next button as field, initialize, and use anonymous class as handler.
* Create an array of TrueFalse objects
* Add code to make sure the “next” button cycles through all available questions.
* Add code to check an answer when true or false gets selected.
* Enable/disable *next* button so that an answer must be provided before you can continue.

package org.mathcs.geoquiz;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

public class QuizActivity extends Activity

{

 private final static TrueFalse[] QUESTIONS = new TrueFalse[]

 {

 new TrueFalse(R.string.question\_africa, false),

 new TrueFalse(R.string.question\_asia, true),

 new TrueFalse(R.string.question\_oceans, true)

 };

 private int currentQuestion = -1; // so that first call to "nextQuestion" works

 private Button buttonTrue = null;

 private Button buttonFalse = null;

 private Button buttonNext = null;

 private TextView question = null;

 private class ButtonListener implements View.OnClickListener

 {

 private Activity parent = null;

 private ButtonListener(Activity parent)

 {

 this.parent = parent;

 }

 @Override

 public void onClick(View v)

 {

 if (v.getId() == R.id.id\_button\_false)

 checkAnswer(false);

 else if (v.getId() == R.id.id\_button\_true)

 checkAnswer(true);

 }

 }

 private ButtonListener buttonHandler = null;

 @Override

 protected void onCreate(Bundle savedInstanceState)

 {

 super.onCreate(savedInstanceState);

 setContentView(R.layout.activity\_quiz);

 question = (TextView)this.findViewById(R.id.id\_question\_text);

 buttonFalse = (Button)this.findViewById(R.id.id\_button\_false);

 buttonTrue = (Button)this.findViewById(R.id.id\_button\_true);

 buttonNext = (Button)this.findViewById(R.id.id\_button\_next);

 buttonHandler = new ButtonListener(this);

 buttonFalse.setOnClickListener(buttonHandler);

 buttonTrue.setOnClickListener(buttonHandler);

 buttonNext.setOnClickListener(

 new View.OnClickListener()

 {

 @Override

 public void onClick(View v)

 {

 nextQuestion();

 }

 });

 nextQuestion();

 }

 @Override

 public boolean onCreateOptionsMenu(Menu menu)

 {

 // Inflate the menu; this adds items to the action bar if it is present.

 getMenuInflater().inflate(R.menu.quiz, menu);

 return true;

 }

 private void nextQuestion()

 {

 buttonNext.setEnabled(false);

 currentQuestion++;

 if (currentQuestion >= QUESTIONS.length)

 currentQuestion = 0;

 question.setText(QUESTIONS[currentQuestion].getQuestion());

 }

 private void checkAnswer(boolean answer)

 {

 buttonNext.setEnabled(true);

 if (QUESTIONS[currentQuestion].isTrue() == answer)

 Toast.makeText(this, R.string.toast\_correct, Toast.LENGTH\_SHORT).show();

 else

 Toast.makeText(this, R.string.toast\_incorrect,

 Toast.LENGTH\_SHORT).show();

 }

}

**package** org.mathcs.geoquiz;

**public** **class** TrueFalse

{

 **private** **int** question;

 **private** **boolean** isTrue;

 **public** TrueFalse(**int** question, **boolean** isTrue)

 {

 **this**.question = question;

 **this**.isTrue = isTrue;

 }

 **public** **int** getQuestion()

 {

 **return** question;

 }

 **public** **void** setQuestion(**int** question)

 {

 **this**.question = question;

 }

 **public** **boolean** isTrue()

 {

 **return** isTrue;

 }

 **public** **void** setTrue(**boolean** isTrue)

 {

 **this**.isTrue = isTrue;

 }

}

This is an example of the ***Model-View-Controller*** paradigm that Android uses quite a lot:

* The **model** holds the application’s data. It has no knowledge of the user interface. It only holds and manages the data. Here the model consists of the TrueFalse classes and the QUESTIONS array.
* **View** objects know how to draw themselves, how to represent data, and how to respond to user input. Here the view consists of the text view.
* The **controller** ties the view and the model together and manages the flow of data from the model to and from the view. Here the controller is embedded in the Activity via the “nextQuestion” and “checkAnswer” methods.

BUG: Start the app, go to the third question, then rotate device into landscape mode. Your app will resort to the first question again!!!!! To understand and fix, we need to understand the **app life cycle**.