Assignment 1: Port & Starboard

Revisions:
- Jan 7: Added note on how to clean project for submission.

- Submit a ZIP file of all the deliverables to the CourSys: https://courses.cs.sfu.ca/
  - All submissions must be done electronically via this web site.
  - Late penalty is 10% per calendar day (each 0 to 24 hour period past due).
  - Maximum submission 2 days late.
- This assignment is to be done individually. Do not share your code or solution, do not copy code found online, and do not post questions about the assignment online. Please direct all questions to the instructor or TA (cmpt-276-d2-help@sfu.ca).
  - You may use code provided by the instructor, or in help guides/videos provided by the instructor (such as my YouTube videos).
  - You may follow any guides you like online as long as it is providing you information on how to solve the problem vs a solution to the assignment.

1. Installation

Install and configure an Android development environment on a computer.

- Download the latest Java SE JDK (it includes the JRE) http://www.oracle.com/technetwork/java/javase/downloads/index.html
- You are encouraged to install this software on your own computer; however, if you wish, you can complete this assignment without doing an installation using the Linux CSIL labs at SFU Surrey.

1.1 Resources

- Read chapters 1 to 4 of Android Programming: The Big Nerd Ranch book and complete this assignment.
- You may also want watch some of the provided videos on YouTube. Suggested videos:
  - Creating a button
  - UI Layouts
  - Java Objects in Android Activities
- The value (to you!) of this assignment is not what you turn in, but what you learn for future assignments.
2. Port & Starboard Application

Create an Android application to the following specification:
- Name your application “Port And Starboard”
- Minimum SDK version API 15 or lower (Android 4.0.3, IceCreamSandwich)

2.1 Part 1: Basics

2.1.1 App Features
- Have the main screen look similar to Figure 1.
  You must have:
  - Descriptive title text (such as “How to navigate a ship”).
  - Two buttons just below the title text. Each labeled “Show Left Name” (and Right).
- Buttons:
  - When Show Left Name button is clicked:
    1) Display Toast message: “Port (left) is red”).
    2) Log the same message (into LogCat).
  - Show Right Name does the same thing but message is “Starboard (right) is green.”.
- Hints:
  - Look at the Creating a button tutorial on YouTube.
  - LogCat is accessed using the Log class.
- Constraints:
  - All strings that appear in a TextView or as labels on Buttons must be in strings.xml. OK if toast and log messages are in code vs read from strings.xml.
- Hint: Leave the Android emulator running while developing; just re-run your application via Android Studio each time you want to test a change. This should make development much faster.

2.1.2 GitLab
- Create a new GitLab project on SFU Computing Science’s GitLab server and push your project to that repository. See Appendix A at the end of this document for directions.

2.1.3 Debug
- Set a breakpoint in your onCreate() method. Debug the application and step through the onCreate() method while it is running on the emulator. Close and re-open your application within the emulator (use emulator's back button to close it, relaunch by finding it in Android’s list of all installed programs) and notice it triggers your breakpoint again.
- Set a breakpoint in the on-click listener for one of your buttons. Press the button on the UI to trigger the breakpoint.
Part 2: Quiz User

Extend the work you did in part 1 to include a quizzing feature where the app displays either “Port” or “Starboard” and the user answers if that means left or right.

- In the very centre of the screen, display a prompt to the user: “Which side of the ship is the following?” Below this (likely in a separate TextView) display either “Port” or “Starboard” depending on the current state of the game (see below).

- At the bottom of the screen, add two buttons: “Means Left” (on left), and “Means Right” (right)
  - The user clicks one to indicate they think the game’s currently chosen side is left or right.
  - If the user selects the correct answer then:
    - Display a toast message “Correct!”
    - Display a LogCat message: “User guess of Port was Correct!” (or Starboard)
    - Pick a new random side (port or starboard) for the next round of the game.
  - If the user selects the incorrect answer then:
    - Display a toast message “Incorrect. :(’
    - Display a LogCat message: “User guess of Port was Incorrect.” (or Starboard)
    - Pick a new random side (port or starboard) for the next round of the game.

- Use the provided Game class to have your app know about the “state” of the game (i.e., which side is the currently correct answer).
  - Note that Game is a plain-old-Java Object (POJO), meaning it has nothing special to do with Android. It is a very simple class that once instantiated automatically picks a random side to be the “chosen” side. It allows you to get a string describing its chosen side (“Port” or “Starboard”), and can check if a user’s answer is correct or not.
  - MainActivity class should have a field which is a reference to a Game object:
    private Game daGame;
  - DO NOT USE THE NAME “daGame”! Come up with a more descriptive name.
  - Have MainActivity’s onCreate() set daGame to a new Game object.
    daGame = new Game();
  - Each time the user clicks “Means Left” or “Means Right”, check with the game object if it’s the correct answer:
    if (daGame.checkIfCorrect(Game.Side.PORT)) {
        ...
    } else {
        ...
    }

Figure 2: Look of the main activity, part 2.
and then start the next round of the game by replacing the current Game object with a new one and updating the UI with the new chosen side’s name:

daGame = new Game();
// now refresh the UI using daGame.getChosenSideName()

**Constraints**
- All strings that appear in a TextView or as labels on Buttons must be in strings.xml. OK if toast and log messages are in code vs read from strings.xml.

**GitLab**
- Push your changes into GitLab. See Appendix A for directions.

**Hints**
- Any time you are picking a new random side for the game, this means that you are creating a new Game object and updating the UI to show the name of the new side.
- Note that the two “Show ... Name” buttons do very similar things. Consider making a function which extracts the shared code for each. Pass in any necessary arguments to have the function do the right thing each button.
- Likewise, note the two “Means ...” buttons do almost the same thing.

### 3. Deliverables

To CourSys ([https://courses.cs.sfu.ca/](https://courses.cs.sfu.ca/)) you must submit:

1. Zip file of your project folder, as shown in this tutorial video: [https://youtu.be/238A85m45-w](https://youtu.be/238A85m45-w)
   
   **NOTE:** You may need to clean from inside Android Studio; see the notes under the video for directions.

2. Screen shot of the commits page of GitLab repo (via gitlab.cs.sfu.ca, which redirects to a specific server). Open your repository and click “Commits”. It should now show at least 2 commits.

3. URL to your git repository

Please remember that all submissions will automatically be compared for unexplainable similar submissions. Everyone's submissions will be quite similar, given the nature of this assignment, but please make sure you do your own original work; we will still be checking.
Appendix A: Git Lab

Initial GitLab Checkin
1. Install Git on your computer.
2. Create a new project on SFU Computing Science GitLab server
   a. Via a web browser, log into gitlab.cs.sfu.ca and click “New Project” (top right).
   b. Name it something like cmpt276As1 and click “Create project”
   c. Generate an SSH key on your computer and upload it to GitLab. See video on course website for directions in Windows.
   d. Copy the Git URL for the project. It’s near the top middle of the page.
      In the drop-down to its left, select “SSH”, then the URL should be something like:
      git@csil-git1.cs.surrey.sfu.ca:bfraser/cmpt276As1.git
3. In Android Studio, enable Git and commit the project:
   a. From the menu select VCS --> Enable Version Control Integration
   b. Select Git and press OK.
   c. In the very bottom left corner of Android Studio, hover over the button to show the applications menu and select “Version Control”
      ▪ If it’s not there, then enabling VCS likely failed because the IDE could not find Git. Ensure you have Git installed and then go to:
         File --> Settings --> Version Control --> Git, and set the path to Git
         (on Windows likely “C:\Program Files\Git\bin\git.exe”)
   d. Expand the Unversioned Files and select them all.
   e. Right-click the selected files and say “Add to VCS”
   f. From the menu select VCS --> Commit Changes.
      Enter a description like “Initial commit”.
      Hover over the Commit button, from the drop-down select “Commit and Push”
      ▪ If asked about Code Analysis, for the time being you can just Commit instead of reviewing issues.
      ▪ In the Push Commits window, click “Define remote” in the top left. Enter that
         git@csil-git1.cs.surrey.sfu.ca:... URL you copied from above
         and click OK.
      ▪ Click Push
4. Ensure the files were pushed by viewing the project in GitLab via the web.

Checking in Changes
After making changes to your app, commit the changes to Git and push them to the GitLab server.
   1. Hover over the bottom-left corner of Android Studio to see the applications menu. Select Version Control.
   2. Right-click in the panel and select “Commit Changes”
   3. Enter a meaningful commit message.
   4. Hover over the Commit button and select “Commit and Push”
      ▪ If asked about any code analysis warnings, you should correct them and then repeat this process; however, you may just click Commit to push the code you currently have.