In this Homework, we will work with the local CrapsSimulation Eclipse project, "gitifying" and pushing to GitHub your repo **craps-simulation**, as installed in [H2-1] below.

I will grade this assignment by checking that each student has performed the following requested changes to **CrapsSimulation** and then pushed it to their remote H-1 repository **craps-simulation**. Specify the URI of this remote GitHub repo by posting it as a text submission to the Canvas H-2 Assignment.

You can work together on this assignment, but each student must push their final **CrapsSimulation** project to their own GitHub repository.

We will start the following steps in class. If we don't finish, do so outside of class, pushing your final version to your GitHub repo.

**[H2.1]** Start Eclipse on your local PC (try to use the latest version 2019-06, but likely earlier versions of Eclipse 2019 will work OK). Specify some local folder as the workspace to use.

First, download the **CrapsSimulation.zip** file that's posted on Canvas next to this handout.

Next, import the Eclipse (non-Git) project **CrapsSimulation.zip**: **File->Import->General-> Existing Projects into Workspace**, select/click on **Select Archive File**, browse and select the then **Finish**.

You should now see the Eclipse Java project **CrapsSimulation** within Eclipse's **Project Explorer** view, containing multiple classes and other resources.

This project contains multiple classes that together implement an OO "framework" for simulating multiple games of the casino game of Craps. It simulates playing the game using two fair dice (instances of **Die**), as well as using inheritance to allow easy modification enabling other combinations of fair and "crooked" dice (instances of **CrookedDie1** and **CrookedDie2**).

Together in class, we'll run **TestCraps** and some of the scenarios within **PlayWithInheritance** and discuss how they illustrate Java's OO results.

**[H2-2]** Put your Eclipse project under control by Git ("gitify" the project) via **Project->Team->Share Projec**t.

Eclipse's EGit provides the ability to do (almost) all of the Git command-line commands ("shell commands"). We will sample some of these from Eclipse's Help on **EGit Documentation->Git for Eclipse Users,** practicing with both shell commands and EGit. You should follow along, learning (or reviewing) the basics of working with Git both in the command-line and in EGit.

Along the way, we will change the code; you should save these changes and commit them locally, along the way.

**[H2-3] *Unit testing using JUnit*** will be an ongoing part of class homework and projects. We will review how to write such tests using Eclipse and practice creating tests that test parts of the **Die/CrookedDie1/CrookedDie2** classes, verifying the correct behavior of method overriding and dynamic method dispatch.

Specifically, we'll create several tests that automatically verify the different inheritance examples given in **PlayWithInheritance**. You should add similar JUnit tests to your project. Best practice: put all your tests in an Eclipse source folder named **test**, with code in a folder named **src**.

**[H2-5]** Can you override **static** methods in subclasses? Add a **static** method **public static String testStatic()** to **Die** that returns **"Die.testStatic()"**, then another **static** method with the same signature in subclass **CrookedDie1** that returns **"CrookedDie1.testStatic()"**.

Finally add code to **PlayWithInheritance**'s **RUN\_EXAMPLE == OVERRIDING\_STATIC**code area that creates a **Die** instance variable named **die**, assigns to **die** an instance of the subclass **CrookedDie1**, then calls **die.testStatic()**. The returned value indicates the answer to the question.

Then add one or more JUnit tests that verifies your answer to the posed question.

**[H2-6]** **Each student should push a final version of the Eclipse CrapsSimulation** project **to their own empty remote GitHub repo craps-simulation** that you created for H-1. We'll explore how to do this in class, using both Eclipse's EGit features and shell commands.

**[H2-7]** If we have time, we'll explore some additional Eclipse tools (Coverage, Reversing-Engineering Classes from Code).