Welcome to Java

Haverford CS 106 - Introduction to Data Structures

Lab 0 (one week)

1 Using Eclipse

Eclipse is the name of the IDE (integrated development environment) that we'll be using in this class to create and run Java programs. You should start to familiarize yourself with it and use it for this and future labs.

Downloading and Installation. Eclipse is already installed on the lab computers, but if you'd like to use it on your personal computer, you should download and install the latest version from https://www.eclipse.org/.

Hello World Tutorial. To get started with using Eclipse, follow the Hello World Tutorial. This can be found at: Help > Welcome > Create a Hello World Application.

2 A Basic Class

In order to make any Java program, you need to make a class. As you saw in the Hello World Tutorial, the most basic class you can make has only the single `public static void main` function. In this lab, we'll practice Java syntax by making a few small `public static` functions. You should test them by calling the functions from main. Be sure to write clear and complete comments for all the functions. Some of these should be familiar to you from CS 105.

You may *not* use recursion to do the exercises below; doing so will result in penalization.

1. Make a function `power(x,exp)` that takes a given number x and returns the number raised to a given power `exp`. You may assume that `exp`≥ 0 and that `exp` is a integer.

2. Make a function `GCD` that takes two numbers and returns the greatest common denominator using this algorithm from Euclid:

   > Given two positive integers, keep replacing whichever number is larger by the remainder of the larger divided by the smaller. When you get a remainder of zero, the other number is the GCD.

3. Make a function `isPrime` that takes a nonnegative integer and returns `true` if it's prime and `false` otherwise.

4. Make a function `round` that takes a double floating point number and returns the number rounded to the nearest integer. You may *not* use a library to do this for you: you should only use basic arithmetic operations and the usual program statements.

5. Follow these instructions to make a `builtBefore1950` function:

   (a) Declare an enum `FordDorms`.

      i. Populate the enum with the dorm names exactly (exclude the parenthesis and words between them) as shown on the residential life page.

      ii. You must not use abbreviations or an alternative name.

      iii. For ampersands, use the word "AND".

      iv. For spaces, use an underscore.

      v. Ignore any other punctuations. For encoding purposes, do not use accents (use letters without accents instead).

   (b) Make a function `builtBefore1950` that takes a dorm from the enum type as input and prints out the dorm name and year built only if the dorm was built before 1950.

      i. Again, for encoding purposes, do not use accents (use letters without accents instead).

      ii. For outputs, the dorm names must exactly match (excluding the parenthesis and the content between them) the name shown on residential life webpage. Do include punctuations.

      iii. The output format should be the dorm name followed by colon and a space and then the year built.

      iv. For example, `builtBefore1950(FordDorms.IRA_DE_A_REID_HOUSE)` should output

         Ira de A. Reid House: 1900