CS161: Introduction to Computer Science
Homework Assignment 1
Due: 1/23 by 11:59pm

Create a New Project

Open BlueJ. Create a new project called `hw1`. Make sure this new project is saved inside of the `cs161` folder you created during lab. (Refer back to lab 1 if you forgot how to create a new project in BlueJ). Again, if you navigate to your `cs161` folder on your system, you should see that BlueJ has created a new folder called `hw1` and inside of this new folder will be all of the Java code your write for this assignment.

For all of your labs and assignments, each class should have a Javadoc comment at the top of the file that provides a description of the class (in complete sentences), your name, and the date.

Compile-Time Errors

A large part of programming is finding and fixing errors in your code. This process is called *debugging*. This question focuses on a particular type of error known as a *compile-time error*.

When speaking a human language (like English), people can generally understand what you say even if you make grammatical mistakes. In a programming language, this is not the case. Your code must adhere 100% to the syntax of the programming language.

The compiler is a program that checks to make sure you are using the correct syntax. When you hit the "Compile" button in BlueJ, you are running the compiler on your code. If you have any statements that do not conform to the syntax rules of Java, the compiler will produce a *compile-time error*.

Create a new Java class named `Test` just like you did in lab 1. Copy the same text from lab into the class:

```
/**
    My first Java program
    @author Your Name
    @version The Date
*/
public class Test{
    public static void main(String[] args){
        System.out.println("This is a test of the emergency broadcast system!");
    }
}
```

Double click on the sheet of notebook paper icon in BlueJ. This will open a README file where you can type your answers to this next exercise. Introduce the following errors, one at a time, to your code. For each error, write down in the README file what you predict will happen. Then compile your code. If there is a compile-time error, write down the message that the compiler produces. If there is no compile-time error, try to explain why. Fix each error before you move on to the next error.

1. Change the class name from `Test` to `test`
2. Change `emergency` to `Emergency`
3. Change `main` to `man`
4. Change `println` to `bogus`

5. Remove the first quotation mark in the string

6. Remove the last quotation mark in the string

7. Remove the semicolon at the end of the `println` statement

8. Remove the last curly brace } in the program

Go to the next page to see an explanation of why each of these errors does or does not trigger a compile-time error. (*Don't change what you wrote for this question*).

1. In Java, the *convention* is that all class names should be capitalized. However this is not a syntax rule. This convention is for the benefit of humans reading your code. Any word you see that is capitalized, you can be assured is a class name.

2. Inside a string literal, capitalization of characters does not matter. You can capitalize whatever characters you want.

3. You can name a method whatever you would like. In this case, you changed the name of the method from `main` to `man`. This does not produce a compile-time error but it will produce a *runtime error* which we'll talk about in class.

4. Note that `System` is capitalized. This means that `System` is a class. In fact, this is a class that some other programmer wrote that we are using. Inside this class, the programmer wrote a method called `println`. When you changed `println` to `bogus`, the compiler tried to find the method named `bogus` inside the `System` class. When the compiler couldn't find the bogus method, it triggered a compile-time error.

5. A string literal must begin and end with double quotes. This is a syntax rule. This is how the computer identifies the boundaries of the string.

6. A string literal must begin and end with double quotes. This is a syntax rule. This is how the computer identifies the boundaries of the string.

7. All programming statements must end in a semicolon. This is a syntax rule.

8. All scopes that are opened with a { must be closed with a }. This is a syntax rule.

Programming Questions

1. Create another Java class called `Pictures` that has a `main` method.

   (Note: These programming questions assume that you have finished the assigned reading through Section 2.1. If you have not done the reading, then stop and do that first.)

2. Complete programming project 1.6. Use your imagination to come up with an interesting tree drawing. Your tree must be at least 20 lines long. You can refer to question 1.9 for an example of what it means to print something using asterisk characters.

3. The purpose of this next question is to practice using escape characters. In the same `main` method, draw a picture with the following restrictions:

   - You can only use the following characters: backslashes (\), forward slashes (/), pipes (|), dashes (-), quotation marks ("), the tab character (\t), and the newline character (\n). *In particular, you cannot use spaces!*
   - Your picture should use each of these characters at least once
   - Your picture should be at least 10 lines long
   - You should use `System.out.print()` only, i.e do not use `System.out.println()`

Submitting your assignment

Rename your `hw1` folder `hw1_firstName_lastName`. After you have renamed your folder, zip (or compress) the folder and submit it via Moodle.