CSE174 Fall 2019
Program#1: 25 Points – Due Sunday, September 1, by 11:59 p.m.

This first assignment is different from future assignments in two very important ways:

1. You will not normally be given Java code and told to copy it.
2. You are NOT expected to understand all of this code. You will probably figure out what some of this code does as you type it, but do not worry if you do not understand all of it.

Typical assignments in CSE174 will ask you to figure out code on your own, and will focus on programming concepts that have already been taught.

Outcomes:

- Use a contemporary programming language and programming environment.
- Write, compile, edit, and debug simple Java programs.
- Format and comment source code that adheres to a given set of formatting guidelines.
- Use the course website.

Scoring:

At a bare minimum, the program you submit must have the assigned source code, and your source code must compile and run without crashing.

- If you submit source code, but it does not compile, your score for this assignment will be zero.
- If you submit source code that roughly resembles the requirements and it compiles, but it crashes under normal operating conditions (nice input from the user), your score for this assignment will be 5 points.

	Full credit	No credit or Partial credit
Enter and run given code (10 points)	You entered the code as given, and it runs as expected (user input/output, game, triangle).	You entered the code, and it compiles and runs, but it contains multiple errors.
Format and comment source code (6 points)	You followed all of the given formatting requirements (indentation, comments, upper/lowercase, etc.).	You did not follow some or all of the formatting requirements as specified in the requirements.
Make program modifications (9 points)	You successfully made all of the required modifications to the assignment.	You did not make one or more of the required modifications to your program.

Part 1: Get your environment set up

- **If you are working on your own computer,** set up the Dr. Java IDE.
- **If you are working in one of the Benton computer labs,** Dr. Java is already installed.
- **Under Dr. Java's Edit menu, select "Preferences" and make some changes that will make Dr. Java easier to use.**
    - Under the "Display Options" section, *check* the box next to "Show All Line Numbers".
    - Under the “Display Options” section, change “Right Margin Position” to 70.
    - Under the "Miscellaneous" section, set the "Indent Level" to 3.
    - Click "OK" to apply these changes and return to the programming environment.

Part 2: Type your source code

The next several pages contain the source code for a Java program. Using Dr. Java, type it in exactly as shown, including indentation, comments, blank lines, upper/lowercase, etc.

Begin by typing this much (Don't type the line *numbers*. They will show up automatically.):

Modify this section with your name, instructor, section, and date.

```
1  /*
2   * Name:
3   * Instructor:
4   * CSE 174, Section A
5   * Date:
6   * Filename: Firstprogram.java
7   * Descirption: Practice with writing, saving and compiling code
8   */
9  import java.util.Scanner; // We need this for user input
10
11 public  class FirstProgram{
12
13    public static void main(String[] args) {
14
15    }
16
17    public static void greet(String name) {
18
19    }
20
21    public static void explainGame(String  name) {
22
23    }
24
25    public static void printBorder(char symbol, int size) {
26
27    }
28 }
29
```

Things to notice:

- **Indentation:** If you do not make any mistakes in your typing, then Dr. Java will automatically indent each line as shown above. Indentation is important because it helps humans (such as you and your peers and instructors) read your code.
- **Curly braces:** Lines 11, 13, 17, 21, and 25 contain opening curly braces, and lines 15, 19, 23, 27, and 29 contain closing curly braces. Curly braces are important because the Java compiler looks for them to know when a block of code begins and ends.
- **Comments:** Beginning a line with a double slash, //, indicates that the text is a comment for humans (the compiler ignores it). /* multiline comment */ : The compiler ignores everything from /* to */ . The comment can span over multiple lines.
- **Color coding:** Dr. Java will automatically color code certain words blue, comments will be green, and unrecognized words black. This helps humans read your code.
- **Upper/lowercase:** Note that only a few words begin with an uppercase letter (`Scanner`, `MyFirstProgram`, and `String`). Java is case sensitive, meaning that the Java compiler considers Scanner, scanner, SCANNER, and ScAnNeR to be different. (You will see later that choosing uppercase vs. lowercase helps humans as well.)

CSE174 Fall 2019

Program#1: 25 Points – Due Sunday, September 1, by 11:59 p.m.

Get to know the basic "structure" of a Java program:

```
11 public class FirstProgram {
12
13   public static void main(String[] args) {
14
15   }
16
17   public static void greet(String name) {
18
19   }
20
21   public static void explainGame(String name) {
22
23   }
24
25   public static void printBorder(char symbol, int size) {
26
27   }
28
29 }
```

Basic structure:

- **One class:** Represented by all the code within the yellow rectangle. Notice that the curly brace at the end of line 11 marks the beginning of that block of code, and the curly brace at the end of line 29 ends that block of code.
- **Four methods:** Our class contains four methods, highlighted in orange. The names of these methods are: `main`, `greet`, `explainGame`, and `printBorder`. Each method has its own set of curly braces.
- **One of those methods is named main:** When you tell the Java Virtual Machine to run a program, the JVM will look for the method named **main**. Without it, you would get an error message indicating that no main method was found.

In CSE 174, most of the programs we write will consist of one class with multiple methods.

Make it easier to spot your curly braces:

One technique that programmers often use to make their code easier to read is to put a comment after the closing brace of each class and method to indicate which code has just ended. Modify your code to include the comments shown below:

```
11 public  class FirstProgram{
12
13    public static void main(String[] args) {
14
15    }//end main method
16
17    public static void greet(String name) {
18
19    }//end greet method
20
21    public static void explainGame(String  name) {
22
23    }//end explainGame method
24
25    public static void printBorder(char symbol, int size) {
26
27    }//end printBorder method
28 }//end class
29
```

CSE174 Fall 2019

Program#1: 25 Points – Due Sunday, September 1, by 11:59 p.m.

Explain the purpose of your program:

Before writing the code for a class or any of its methods, write comments to summarize the purpose of that code:

- The comments before a class should summarize the purpose of the class as a whole.
- The comments before a method should summarize the purpose of that specific method.

Later, we may put comments within a method to explain what part of that method does. For now, add the following comments to your code.

```
10  /*
11   * Deomonstartes some basic programming concepts by getting information
12   * from the keyboard, displaying results to the screen, and playing an
13   * interactive game with the user.
14   */
15  public  class FirstProgram{
16     /*
17      * The starting point for the program. This method
18      * calls on the other three methods as needed
19      */
20     public static void main(String[] args) {
21
22     }//end main method
23
24     // Prints a personalized welcome message.
25     public static void greet(String name) {
26
27     }//end greet method
28
29     // Prints a personalized game introduction.
30     public static void explainGame(String  name) {
31
32     }//end explainGame method
33
34     // Prints a border by repeating the specified symbol
35     public static void printBorder(char symbol, int size) {
36
37     }//end printBorder method
38  }//end class
```

Use the Enter key!

Unlike when using a word processor, you need to press the "Enter" key at the end of each line.

As a bonus: when you press enter at the end of a line, Dr. Java should automatically indent your code correctly (Uses 3 white spaces for each level of indentation as we set it already in Part1 to be 3).

As a general rule we will look for a way to keep every line of our program at 70 characters or fewer (we set right margin position already in Part1 to be 70). Notice that Dr. Java indicates the current line number and column in the lower right-hand corner. For example, this is what would be displayed if the cursor were on line number 34, column 11:

CSE174 Fall 2019

Program#1: 25 Points – Due Sunday, September 1, by 11:59 p.m.

A key to writing programs is to save, compile, and test frequently.

- After writing a little, save it, compile it, and test it.
- Write a little more, save it, compile it, and test it.
- Write a little more, save it, compile it, and test it.

So, save your work and compile it. If you get an error message…

- Read the message.
- Note the line number where the error occurred (often, the actual mistake comes *before* that line number).
- Try to figure out how to fix the problem on your own. If you are stuck, copy the error message, go to the discussions at the course website and paste the exact error message along with any other pertinent information, and see if anyone has a suggestion.

Don't move on until the above compiles correctly. You may also run your compiled code at this point, but won't see anything happen yet (because the main method contains no code).

Write the code for the `greet()` method:

Put your cursor inside the `greet` method, and type the green highlighted code below. We call this the *body of the greet method*. Notice that the non-highlighted code is what you've already typed. Notice also that `println` stands for "print line". The character between the `t` and the `n` is a lowercase letter `l`.

```
24    // Prints a personalized welcome message.
25    public static void greet(String name) {
26        System.out.println("Hello " + name + ", ");
27        System.out.println("Welcome to my first CSE 174 program!");
28        System.out.println("Enjoy the show!");
29        System.out.println("  Sincerely,\n  John Smith");
30    }//end greet method
```

Modify the code so that your name is displayed, instead of John Smith's name.

Save and compile your code, fixing any errors.

Run your code, but you still won't see anything happen yet.

Why does nothing happen when you run your code?

The reason is that when you *run* a program, the Java Runtime Environment runs the code that you've written in the body of the `main()` method. Right now, you have a `main()` method, but no code in the body of that method. Once you start putting code in the body of the `main()` method, *then* you will find that your program actually does something when you click "Run".

Fixing indentation:

Dr. Java will automatically handle indentation for you as long as you press "enter" at the end of each line of text that you type. If at any time, you notice that your indentation looks incorrect, use the following Dr. Java indentation shortcut to fix it:

1. Select all your text. (Ctrl-A in Windows, Command-A on a Mac)

2. Press the "tab" key.

If Dr. Java is still not indenting correctly, there is a good chance that you've made an error in your code (such as missing parentheses, misspelling a Java command, or something similar).

Type the highlighted code for the body of the `explainGame()` method:

```
32    // Prints a personalized game introduction.
33    public static void explainGame(String  name) {
34       System.out.println("Lets play a game, " + name + "...");
35       System.out.println("I'm thinking of a number from 1 to 50.");
36       System.out.println("See if you can guess it in fewer than 5 tries.");
37    }//end explainGame method
```

Save and compile your code, fixing any errors.

Type the highlighted code for the body of the `printBorder()` method:

```
39    // Prints a border by repeating the specified symbol
40    public static void printBorder(char symbol, int size) {
41        for (int i = 0; i < size; i++) {
42          System.out.print(symbol);
43       }
44       System.out.println(); //moves to the next line
45    }//end printBorder method
```

Save and compile your code, fixing any errors. Note that one of the lines above contains a `print()` statement, and the other contains a `println()` statement. What's the difference? When you print with `println()`, the cursor moves to the next line after it is done printing. When you print with `print()`, the cursor stays on the same line when it is done printing.

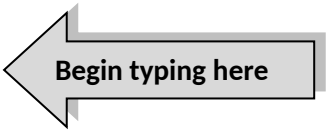
Looking back:

Recall that this class contains 4 methods. So far, you have written code for three of those methods. Your program still does nothing when it is "run", because it lacks any code in the method named `main()`. That's what you still need to do: type the body of the `main()` method. In doing so, you will be making use of all of your other methods. Pay attention and you should notice that you will eventually "call on" the `greet()`,`explainGame()` and `printBorder()` methods as you type the code for the `main()` method.

Next, write the *body* of the `main()` method.

Put your cursor inside the body of the `main()` method. Since this is a longer method, you should frequently compile and run your code. Throughout the code, you will see several stars ★ to indicate that it would be a good idea to save, compile, and run. Don't move on until the program compiles and runs correctly at each star. Note that the first three lines below were already typed in a previous step.

```
16    /*
17     * The starting point for the program. This method
18     * calls on the other three methods as needed
19     */
20    public static void main(String[] args) {
21       // Declaring local variables for later use
22       String firstName, lastName;
23       int targetNumber, userGuess, countGuesses;
24       int triangleHeight;
25       Scanner keyboardReader = new Scanner(System.in);
26
27       // Get user's name
28       System.out.print("Enter first and last name: ");
29       firstName = keyboardReader.next();
30       lastName = keyboardReader.next();
31
32       // Display a marquee with a personal greeting
33       printBorder('*', 30);
34       greet(firstName);
35       printBorder('*', 30);
36
37       // Explan how to play the game
38       explainGame(firstName);
39
40       // Set up the game
41       targetNumber = (int) (1 + 50 * Math.random());
42
```



(At this point, if you run your program, it should prompt the user for her first and last name, display a greeting, and introduce the game.)

Continued on the next page…

```
43        for(countGuesses = 1; countGuesses <= 5; countGuesses++){
44           printBorder('*', 30);
45           System.out.print("Enter guess #" + countGuesses + ": ");
46           userGuess = keyboardReader.nextInt();
47
48           //The user made the right guess.
49           if(userGuess == targetNumber){
50              System.out.print("Good job " + firstName + ", ");
51              System.out.println("you got it in " + countGuesses + " tries.");
52              break;
53           }
54            //Give advice to let the user know if his guess is higher or lower than the target number.
55           else if (userGuess <
56              System.out.print
57           }
58           else {
59              System.out.println("Too high. Guess lower.");
60           }
61        }
62        // The user could not guess the number in 5 trials
63        if(countGuesses > 5){
64           System.out.println("Hard luck! " + firstName + " ");
65        }
66        // Some artwork:
67        printBorder('#', 30);
68        System.out.println("And now, a triangle of money!");
69        triangleHeight = 8;
70        // Display a triangle
71        for (int row = 1; row <= triangleHeight; row++) {
72           printBorder('$', row);
73        }
74     }//end main method
```

Don't type this line. It's the end of the main() method, which you've typed

Here's what a sample run of the program should look like if you run it.

```
Enter first and last name: John Smith
******************************
Hello John,
Welcome to my first CSE 174 program!
Enjoy the show!
  Sincerely,
  John Smith
******************************
Lets play a game, John...
I'm thinking of a number from 1 to 50.
See if you can guess it in fewer than 5 tries.
******************************
Enter guess #1: 25
Too high. Guess lower.
******************************
Enter guess #2: 15
Too low. Guess higher.
******************************
Enter guess #3: 20
Too low. Guess higher.
******************************
Enter guess #4: 22
Good job John, you got it in 4 tries.
##############################
And now, a triangle of money!
$
$$
$$$
$$$$
$$$$$
$$$$$$
$$$$$$$
$$$$$$$$
```

Part 3: Submit the current version of your program

Is your program running as expected? If so, then now is a good time to submit your current working version, even though you will still be making changes. This is a very important step because soon you will be making several changes to your program. It is good to upload a version you know is working *before* you make those changes.

So, prior to making any of the modifications below, go to program1 on the course website and submit your source code for this assignment. When you do, the website automatically changes the name of the file and adds a number to the end of the file name. Don't worry about that. It's fine. **Do not change the name of your file on your computer. It should be FirstProgram.java at all times, even when you make changes below.**

Part 4: Modify the program

Now it is time to "play around" with your program. Note that you do not need to do any external "research" to solve the following. Instead, look carefully at the program you already typed, and learn some techniques from the code you already typed.

Once the program is working correctly, make all four of these modifications:

1. Even though the program prompts the user for her first and last name, it only displays her first name in the introduction. Don't change the part of the program that asks the user for her name, but fix the program so that the greeting displays the user's first and last name, separated with a space, rather than just the first name. For example, "Hello Mary Smith," rather than "Hello Mary,"). The *game introduction* should still only display the user's *first* name (so, it should still say, for example, "Let's play a game, Mary...".

2. The mystery number should be a random number from 1 to 100, rather than 1 to 50, and in order to "win" the game, the number must be guessed in fewer than 10 guesses (rather than 5). Modify any comments in your code to reflect this change.

3. Modify the part of the program that prints the triangle by choosing one of the following options. **Either:**
   - *Ask* the user how many rows should be in the triangle, and display that many rows. For example, if the user says 7, then the triangle would display 7 rows of dollar ($) signs, beginning with 1, then 2, then 3, then 4, then 5, then 6, then 7. **OR...**
   - Make the number of rows in the triangle be a random number between 1 and 30. Random does NOT mean that the programmer should pick her favorite number and put it in the program. Rather, "random" means that each time your program is run, the computer will generate a different, unpredictable number of rows in the triangle, with as few as one row, or as many as thirty rows.

4. Instead of displaying the dollar symbol ($) in the triangle of money, display the character associated with the ASCII code 64. You could search the Internet for how to convert a numerical value into an ASCII character.

What if something goes wrong with your file?

Whenever you upload your work to the course website, you are creating your own personal "backup copy" of your work. If something should go wrong as you make your modifications, remember that you can go back to the course website, locate the file you submitted, and download it to your computer.

Part 5: Submit your modified source code

Once you have made the required modifications to your code, and you find that your code works as expected; it's time to re-upload your source code. **On the course website, go to program1 and upload your updated source code file (.java file).**