UNB EST. 1785 UNIVERSITY OF NEW BRUNSWICK

Assignment Guide

Module 3 Assignment

Worth 2% of your total course grade

CS1073 – Introduction to Computer Programming I (in Java)

Online Open Entry Version

Instructor: Andrew McAllister

Assignment Objectives

The purpose of this assignment is to give you practice:

- creating, compiling and running a basic Java program; and
- fixing errors in a Java program.

General Instructions for All Assignments

- For each .java file you create, include a javadoc comment (one that begins with /**) at the beginning of the file that describes that Java class. This comment block should include a line that begins with @author followed by your name and student number on the same line. Hello.java provides an example (minus the student number).
- Include comments throughout your programs to explain any non-obvious portions of your code.
- It is recommended that you create a separate folder on your computer for each assignment. You might even wish to create separate sub-folders within an assignment folder if the assignment has multiple parts. Keeping your work organized makes it easier to find things later when you want to review what you have done.
- Few things in life are more frustrating than losing your work while working on an assignment. Get in the habit of saving frequently when working on an assignment. Also, regularly make backup copies of any files you create as part of your work for this course.

- Feel free to email your instructor if you need help in completing an assignment. When you do so, please attach to the email a copy of *all* files required to compile and run the program you are asking about, even if you downloaded those files from D2L. (Otherwise the instructor will have to figure out what files are missing and then go looking for them. Make it convenient to help you.) Also describe the problem you are encountering and the specific help you would like to receive.
- Submitting your assignment involves creating a pdf file and uploading that single file to the assignment drop box on D2L. In general, that file will tend to include all Java code you wrote for the assignment, plus any output from running your programs that the assignment instructions specify you should capture. Specific submission instructions are included at the end of each assignment.
- To create the pdf file for each assignment, begin by opening a new document using the word processing program of your choice. Save the document using the name “Your Name CS1073 Module x Assignment Submission.docx” Replace x with the correct module number. “docx” may be different depending on which word processing software you choose to use.
- If you don’t already have a word processor, UNB students are able to use Microsoft Office 365 (includes Microsoft Word) for free, which can be accessed by logging in to MyUNB.
- At the beginning of your submission document enter your name, your student number, CS1073, the assignment name, and the date. It doesn’t matter if the date is when you started working on the assignment or when you submit it – either will do.
- You can add content to your submission document as you work on the various questions in the assignment. Clearly label each part of the assignment (“Part A” etc.) in your document. Be sure to save frequently as you work on this document.
- When your document is complete, save / export it as a pdf file. Always make sure your pdf file opens properly before uploading it.
- To include Java code in your submission document, copy all the text in your .java file and then paste that text into your submission document. Use a monospaced font (e.g.: Consolas , Courier) for your code to maintain proper indentation. Submitting code without proper indentation will result in marks being deducted. It’s not enough that your code is indented in your text editor or in your integrated programming environment – the indentation must show up that way in your submission document as well. This sort of thing is part of learning to be an IT professional.

- To include output from running your program in your submission document, the preferred method is to copy and paste the text from your command prompt window. Include the line with the “java” command you used to run your program.

  If the text shows up as a weird font or colour in your submission document, first paste the text into a blank text editor document, then copy and paste from there into your Microsoft Word submission document. This will remove all formatting from the text.

  Use a monospaced font (e.g.: Consolas , Courier) for output text in your Word document. This will maintain alignment of your output.

- If at all possible, each line of code and each line of output should appear on a single line in your submission document. Avoid allowing lines to “wrap” around onto the next line. Use the tips provided in the “Avoiding Wrapped Lines” section on the next page to accomplish this.

- To copy text from your command prompt window, try selecting the desired text and then pressing either command-c (Mac) or control-c (Windows or Linux). If you have issues, you can always use Google to see how to do this on your specific type of computer.

- If a program involves graphical output (such as a JavaFX GUI program), capture a screen shot of the output and include that as a picture / image in your submission document.

  Make sure the image includes **only** the relevant portion of the screen (such as a GUI window). Capturing an image of your entire computer screen often makes the relevant portion too small to see, with tiny text that is difficult to read. This makes your assignment submission difficult to grade.

- To capture a screen shot of a selected portion of your screen, try command-shift-4 (Mac), WindowsKey-shift-s (Windows), or shift-PrtScrn (Linux).

Avoiding Wrapped Lines

In the following example, the lines of code containing the comment and the println statement are both too long. The text is formatted so those lines can't fit all on one line in this document. This obscures the indentation and makes the code more difficult to read.

```
import java.util.Scanner;

public class WrapExample
{    public static void main(String[] args)
     {    double pay = hours * wage;
          int dollars = (int) pay;
          int pennies = (int) ((pay - dollars) * 100.0);
          // First all * and / operations are performed, left to
right, then all + and - operations, left to right
          System.out.println("\nThe pay for " + name + " is " +
dollars + " dollars and " + pennies + " cents.\n");

     } // end main method
} // end class
```

Below is the same code, but reformatted so none of the statements wrap around onto the next line. Several changes were made:

1. The font size (in the Word document) is changed to a smaller size,
2. The tab size (in the Word document) is reduced
3. The longer statements and long comments are broken up onto multiple lines (do this in your text editor, in the .java file), and
4. If need be, you can change the orientation of your Word document from Portrait to Landscape

Now the indentation of the code within the main method is easier to see.

```
import java.util.Scanner;

public class WrapExample
{   public static void main(String[] args)
    {   double pay = hours * wage;
        int dollars = (int) pay;
        int pennies = (int) ((pay - dollars) * 100.0);
        // First all * and / operations are performed, left to right
        // Then all + and - operations, left to right
        System.out.println("\nThe pay for " + name + " is " + dollars
                           + " dollars and " + pennies + " cents.\n");
    } // end main method

} // end class
```

Part A: A Circus Cannon

The circus is coming to town!

One of their acts involves shooting various objects out of a cannon. This can include basketballs, watermelons, or even a “human cannonball” act, like this:

For safety reasons, they need to know how far the objects will travel so they can prevent the objects from landing in the audience. Your task is to write a Java program to help them out.

Your program must prompt the user for the following information, in this order:

1. How fast the object is traveling when it leaves the cannon, in meters per second;
2. The launch angle in degrees; and
3. The type of object being launched (for example, a basketball or a human cannonball).

Your program is to calculate and display how many meters the object will travel. Include an appropriate label as part of this output so the meaning of the information is clear. Include the type of object and the word “meters”.

You can ignore air resistance when calculating the distance traveled. Use the following formula, where v is the initial velocity, a is the launch angle, g is the force of gravity. Use 9.81 meters per second^2 for g.

$$\frac{v^2 \sin(2a)}{g}$$

To make your code more readable, use Java’s `final` keyword to help declare a constant for the force of gravity. Use this constant in your calculation.

Do not use `v`, `a`, and `g` for your variable names. Get in the habit of using longer, more meaningful names for your variables, classes, and methods. This is another way you can make your code more readable.

Java's `Math` class includes a `sin` method you can use in your calculation. This method requires that the argument must be in radians, so you will have to convert from degrees to radians. (Don't know how to do that conversion? Google it! ☺ And depending on how you perform the conversion, you might wish to use the `Math.PI` constant.)

All numeric values can include fractional digits, so declare your variables with an appropriate data type. Any numeric constants should also be compatible with that data type; for example, use `2.0` instead of `2` in your calculation.

I've designed this assignment to purposely introduce you to an issue. A problem should crop up when you use nextDouble() to get the launch angle, and then attempt to use nextLine() to allow the user to enter the type of object. (The same issue would occur if another program used nextInt() followed by nextLine().)

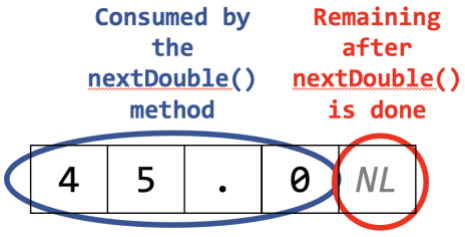
Do not change the order of the inputs, nor should you declare more than one Scanner object. Here's what's happening "under the covers" and how to deal with it.

It turns out that a Scanner object captures user input as a sequence of characters, and then figures out what to do with those characters depending on which methods are used in your program.

Let's suppose someone uses your program and enters 45.0 for the launch angle. This results in the following **five** (not four) characters being captured – the four characters in the numeric value *plus* the newline character is part of the input (shown as *NL*) due to the user hitting 'enter':

4	5	.	0	*NL*

Suppose your program includes a call to the nextDouble() method to enable this input. In this case, nextDouble() works its way through the characters '4', '5', '.', and '0'. After that, the next character (the newline) cannot be part of a double value, so nextDouble() ignores newline; the Scanner object automatically remembers that the newline character has not yet been consumed.



Next your program is likely to use nextLine() to enable entry of the type of object to be shot from the cannon.

Unfortunately, nextLine() consumes what it considers to be leftover user input – that single, unused newline character. Based on this, nextLine() produces a String of length zero and returns that result.

That's why your program may not pause and allow entry of the type of object – as far as nextLine() is concerned, it must chew through any unused input before pausing and allowing the user to enter more.

The simple solution is to add an extra call to nextLine() in between prompting for the launch angle and for the type of object. This new call to nextLine() will consume the newline character that remains after nextDouble() is done, which will then allow entry of the type of object.

Provide output from running your program twice, as follows:

1. Display how far a basketball will travel when shot from the cannon with a velocity of 10.0 meters per second at an angle of 45.0 degrees.
2. Make up another test case of your own choosing with three inputs that are all different from the first test case. Make sure the object travels more than 20.0 meters.

(Part B of the assignment is on the next page…)

Part B: Filling a Swimming Pool … the Hard Way

Imagine trying to fill an entire swimming pool using only a small bottle like this:

Wow … that would take a long time! But that's exactly the scenario on which your next Java program is based.

Assume there is a rectangular pool with a constant depth (there is no deep end or shallow end) and we want to fill the pool entirely with water. Your program should prompt the user to enter the length, width, and depth of the pool in meters. Also allow the user to enter the capacity of the bottle in liters.

Calculate and display (with informative labeling) the following three results:

1. The volume of the pool in cubic meters
2. The bottle capacity in cubic centimeters
3. The number of bottles of water required to fill the pool

The four input values plus the first two results can all include fractional digits, so declare your variables with an appropriate data type.

Calculate the number of bottles as an integer. For example, suppose you have a very tiny pool that only holds 5.0 liters of water, and your bottle holds 2.0 liters. 2.5 bottles are required to fill the pool, which means you have to fill the bottle 3 times. For that example, your program should display 3 as the number of bottles required. The `Math` class includes a `ceil` method, which you will likely find useful in calculating that result. (Google "Java api Math class" to see documentation on the Math class). You may also be required to "cast" a double value to int using the `(int)` operation.

Provide output from running your program twice, as follows:

1. Specify a pool that is 25.0 meters long, 11.0 meters wide, 2.2 meters deep, and is being filled with a 2.25 liter bottle.
2. Make up another test case of your own choosing with four values that are all different from the first test case. Make sure more than a million bottles are required to fill the pool.

Submission Instructions

Using the instructions provided on the first few pages of this document, include the following in your submission document:

Your name, student number, CS1073, Module 3 Assignment, and the date

Part A:

- Your complete Java program
- The outputs from running the two test cases specified in Part A of this assignment

Part B:

- Your complete Java program
- The outputs from running the two test cases specified in Part B of this assignment

D2L DROPBOX SUBMISSION INSTRUCTIONS

Upload only one pdf file to D2L. Do not upload separate files (such as your .java files) as they will be ignored. Upload your submission document as follows:

1. In the top-navigation bar on the course screen, select 'Assessments' and then 'Assignments'.

2. Select the assignment title and follow the instructions to upload your submission document.