CS1073 - Assignment #3 - Fall 2023

Submission Deadline: Thursday, February 15th before 12:00 NOON (Atlantic) in the Assignment 3 submission folder in Desire2Learn. (Read the submission instructions at the end of this document carefully).

The purpose of this assignment is to:

- introduce the boolean data type and decision statements
- illustrate the "has-a" relationship between classes
- review javadoc.

This assignment is to be done individually. What you hand in must be your own work. Incidents of plagiarism will be reported.

If you have questions about the assignment, you should first go to a scheduled help session. (Locations and times for all help sessions can be found on D2L). If you have attended a help session and the issue is unresolved, you may contact your course instructor. You are NOT to discuss this assignment with anyone else (including your classmates).

As always, begin by creating a new folder to hold your work for this assignment.

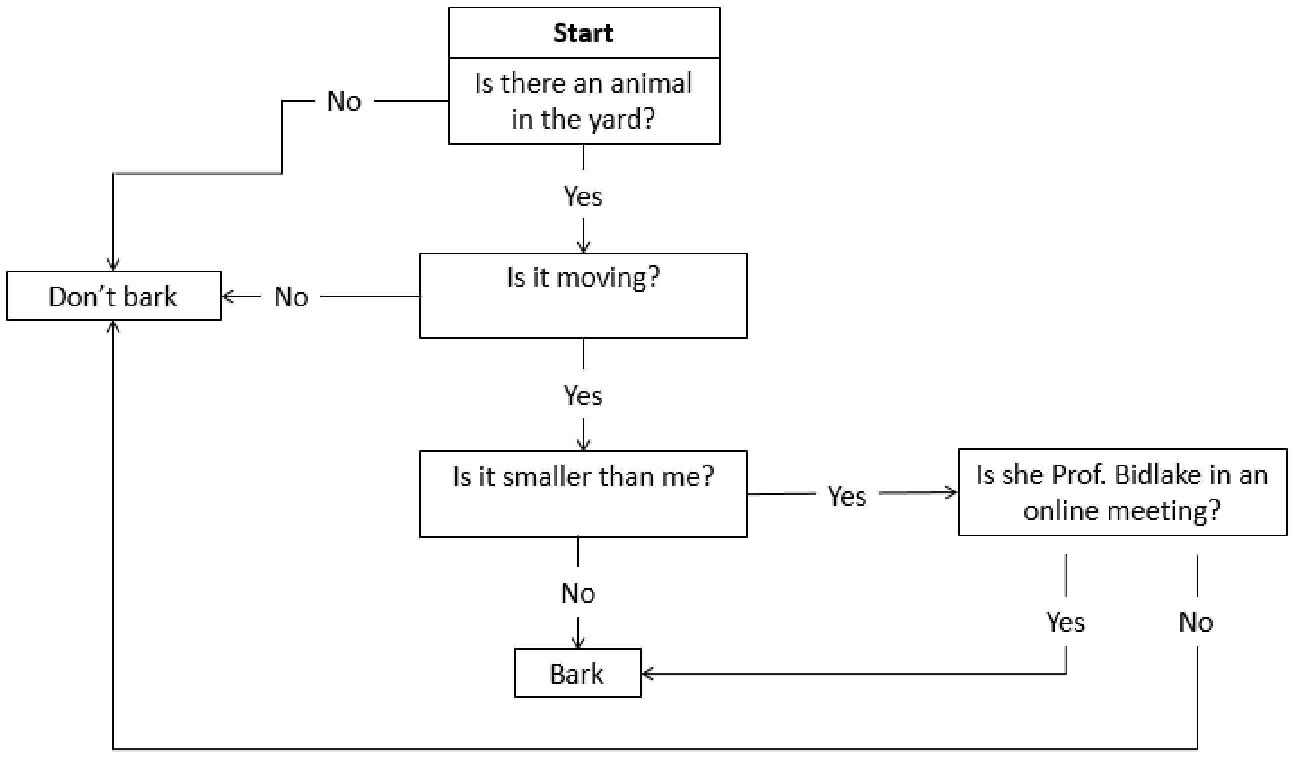
I. Programming Exercise: How does Luna the dog think?

Write a program to help someone understand how Prof. Bidlake's dog, Luna, thinks. Luna is generally a quiet dog, but she likes to bark at certain things. To help Prof. Bidlake understand why Luna barks when she does, write a program to walk her through Luna's "thought" process.

Write an interactive program based on the flow chart below. The program will help the user understand what action Luna would take if she is trying to decide whether or not she should bark at something.

The boxes with questions are what you will ask the user. They can answer with either "yes" or "no" to each question. Eventually they will reach one of the two final boxes, where you will be able to determine if Luna will bark or not.

HINT: Start by doing the first question only, assuming "no" means Luna would not bark, and "yes" means Luna would bark. If you can get that working, how can you add another question? (and so on)



This program can be written completely in a main method. You should include a javadoc comment block for your class (with the @author tag and your name). Other non-javadoc comments (using /* ... */ and/or //) could also be included within your main method to explain different sections of your code, but that is entirely optional.

Once this first question is complete...

After you have tested your application and you're sure that it works properly, save sample output showing at least 3 runs of your program. For this question, take a snapshot of the terminal (or Command Prompt) window and save that as your output. That way, the markers will be able to see the input values as well.

Note: Always adjust the size of your terminal window before taking a screenshot. Make the window taller if some of the sample output/input text is cut off. If there is a lot of extra whitespace beside your text, you can make the terminal window narrower. (That way, the text in the image will be larger and more legible when you copy it into your report.) It is important that the marker can easily read all of your sample output!

II. Enhancing a Triangle Class

Part A

In Desire2Learn you will find a .zip archive containing two Java classes: `CartesianPoint.java` and `Triangle.java`. Download and unzip that file. Compile both classes.

Part B

Add the following methods to the `Triangle` class:

- `public double getPerimeter()` - returns the perimeter of the triangle.
- `public boolean isEquilateral()` - returns true if the triangle is an equilateral triangle, false otherwise.
- `public boolean isRight()` - returns true if the triangle is a right-angle triangle, false otherwise. (*Hint*: Can use the Pythagorean Formula).

Note: make use of methods in the `CartesianPoint.java`

Consult math texts or online sources if you forget the definitions of "equilateral" or "right-angle", or if you can't recall a formula.

IMPORTANT NOTE regarding equality comparisons[1]: When working with floating-point numbers in Java (or any programming language), calculations don't work out exactly as we might expect. For instance, we may perform some calculations and end up with an answer like: 3.999999... instead of the expected answer of 4. This happens because of the limited precision used to store/represent floating–point numbers in a computer.

Therefore, rather than comparing two double values using:

```
  if (num1 == num2) { ...
```

you should instead calculate the difference between the two numbers and then compare that to some really, really small value. If the difference is less than that really small value, then the two numbers are, essentially, equal.

For example, use:

```
  if (Math.abs(num1-num2) < TOLERANCE) { ...
```

where double `TOLERANCE = 1E-14` (or some other really small number).

Aside: `Math.abs(num1-num2)` simply gives the absolute value of the difference between num1 and num2.

Continued on the next page...

[1] See also: Section 5.3 of the course textbook.

Part C

In a separate file, create a test driver that exercises the two new methods in your `Triangle` class; name this class `TestTriangle`. In your test driver, create at least 2 `Triangle` objects (one that is an equilateral triangle, and another that is a right triangle). Call both methods on each of these objects and print out the results in a meaningful way, such as "The triangle t2 is not an equilateral triangle" or “The triangle t2 is a right-angle triange.” Compile and run this program. Examine the output and return to part b if you notice any problems. Once it is running correctly, please save sample output.

Hint (if you're having trouble coming up with test data): For an equilateral triangle you can use the points (-0.5,0.0),(0.5,0.0),(0.0,Math.sqrt(3)/2), and for a right triangle you can use (0.0,0.0),(1.0,0.0),(0.0,1.0).

Part D

Add a second @author tag to the javadoc comment at the top of the `Triangle` class and add your name (since you have also now written parts of this class). In addition, add a javadoc comment for each of the two new methods that you added; be sure to include an @return tag for each. You do not need to write javadoc comments for the other methods and instance variables; you only need to document the two methods that you added.

Add a javadoc comment to the top of your `TestTriangle` class; include the @author tag (with your name).

You are not required to generate and hand in the html documentation. (However, running javadoc on your files is a good way to check to see if you have typed in your javadoc comments correctly.)

Submission instructions are on the next 2 pages...

Your electronic assignment submission (submitted via Desire2Learn) will consist of <u>two files</u>:

i. a written report. This should begin with a title page; your title page should include: the course (CS 1073), your section (FR01A, FR02A, FR03A, FR04A, FR05A or FR06A), the assignment number (Assignment #3 in this case), your full name, and your UNB student number. That should be followed by six sections, with each part clearly identified with a section heading. Include:

   a. the source code for your solution to Question I.
   b. the sample output that you captured by running your solution to Question I (at least 3 times)
   c. the source code for the updated `Triangle` class (Question II)
   d. the source code for your `TestTriangle` class (Question II)
   e. the sample output for `TestTriangle` (Question II)

   (Aside: Your source code should contain all of the javadoc comments mentioned above. However, you do not need to include the .html files in your report.)

   This written report should be prepared using a word processor. (Options were outlined in Lab #1. If you choose to use the online version of MS Word, please see the note from Lab #1 about the issue when copying and pasting tabs into a document.)

   Copy & paste the items listed above into the report document. (These should appear in the document in the order that they are listed above). Add appropriate headings for each part. Fix up the formatting where necessary, adjusting line breaks & page breaks to ensure that your document is easy to read. Use a monospaced font for your code and question II output to maintain proper indentation. (Examples of monospaced fonts were mentioned in Lab #1.)

Continued on the next page…

Once the report is complete and you've checked it all over, save the final document file for your own records. Then **save a second copy in PDF format for submission**. (Note: Be sure to open the second file in a PDF viewer to verify that the PDF was generated correctly.) The **single .pdf file** containing your report will be submitted to the appropriate lab submission folder in Desire2Learn. (It is important that you submit a PDF file and NOT the original Word or LibreOffice document. This PDF will allow the marker to write comments directly on your work to give you better feedback.)

Note: Please name this report as follows: **YourName_As3_Report.pdf**

i. an archive file (.zip) that contains your Java source code and output for this assignment. Make sure that your archive includes **all the .java files** (in case the marker wishes to compile & run your code to test it), the output from testing your code. You should not include the report document or the .class files in your archive. This archive should be submitted as a single .zip file to the appropriate submission folder on Desire2Learn.

   Side note: While we encourage you to run the javadoc tool on your .java files (to check to make sure that you wrote the Javadoc comments correctly), you do not need to add to your As3 archive all of the files & folders that the javadoc utility creates.

   This archive should be submitted as a **single .zip file** to the appropriate submission folder in Desire2Learn.

   Note: Please name this archive file as follows: **YourName_As3_Archive.zip**

Reminder: Your submission in Desire2Learn should consist of TWO files (a .pdf and a .zip). Do NOT put your report inside your archive.

End of Assignment 3