Lab 2: Variables and Types

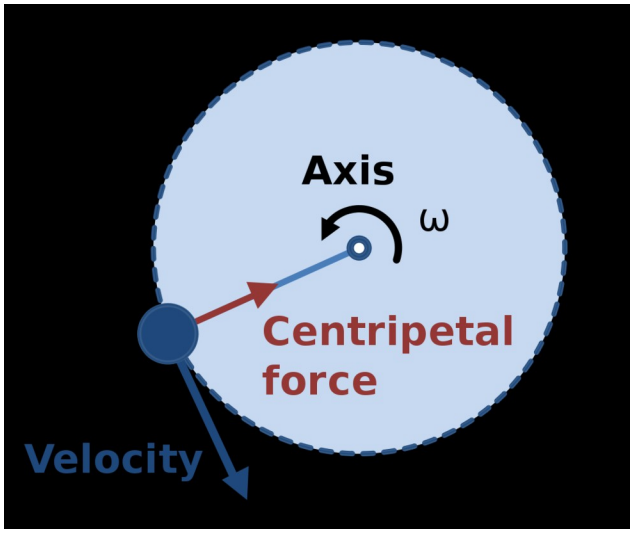
Centripetal Force

Purpose

- Practice using variables
- Practice using mathematical operators
- Practice collecting input from stdin

Description of Task

For this lab, we will be writing a program to calculate the centripetal force of an object. Centripetal force is the force of an object traveling in a circular path.



Centripetal force is calculated with the following formula:

$$F_c = mr\left(\frac{2\pi}{T}\right)^2$$

Where:

- m is the mass of the object
- r is the radius of the circle
- T is the time it takes for the object to complete one orbit, in seconds.

Here is an outline of how to structure your program:

- Start off with the boilerplate code of the class definition and main method.
- Create a constant double variable, PI, set to 3.14159.
- Create a double variable to store the mass.

- Create a double variable to store the radius.
- Create an int variable to store T, the time in seconds for the object to complete one orbit.
- Create a new Scanner object variable. Print the prompts, then use the Scanner to enter the two doubles and one int described above. Use nextDouble() and nextInt().
- Create a double variable to store the result of the calculation for the centripetal force.
- When performing the calculation, use your variables, constants, and the Math.pow() method to raise to the power of 2.
- Print the results to the user, using string concatenation.

Example Script Output:

```
PS C:\workspace\2023_spring\CSC260\Lab2> javac CentripetalForce.java
PS C:\workspace\2023_spring\CSC260\Lab2> java CentripetalForce
Enter the mass:
10.123
Enter the radius:
4.5
Enter the orbital period:
3
The centripetal force is: 199.81967314311254
PS C:\workspace\2023_spring\CSC260\Lab2>
```

Other Test Cases:

Mass (m)	Radius (r)	Orbital Period (t)	Centripetal Force (F)
0	3.4	5	0.0
32.5	0	10	0.0
1.23	4.56	7	4.518901522397329

* Note that there may be some inaccuracy due to floating point precision, but the first 2 decimal places should match.

Helpful Tips

Only One Scanner is Needed
You only need one Scanner initialized, and you can call nextDouble() or nextInt() on it multiple times. So, the correct way would be:

```
Scanner input = new Scanner(System.in);
double x = input.nextDouble();
doubly y = input.nextDouble();
double z = input.nextDouble();
```

The incorrect way would be:

```
Scanner input = new Scanner(System.in);
double x = input.nextDouble();
Scanner input2 = new Scanner(System.in);
double y = input2.nextDouble();
Scanner input3 = new Scanner(System.in);
double z = input3.nextDouble();
```

Order of Operations
Don't forget about the order of operations. The textbook has a good demonstration of how operators work in Java. For calculating the force, you may need to group certain parts of the equation in parenthesis. You can also try separating the equation into multiple smaller equations to get each "part" and then perform a final calculation.

Constants and Math.pow()
Make sure you use a constant for PI, and Math.pow(). Review the slides for details on how to use both.

Watch the Demo Videos
Everything you need, in terms of Java features, to complete this lab is in the demo videos. However, you will need to figure out exactly how to structure your code and solution. Watch the videos and try to apply the skills learned there to this problem! If you have questions, send me an email!

Deliverables

- A .java file for your program. Remember, the filename must match the classname (example: If the class is called "CentripetalForce", the file will be named "CentripetalForce.java")
- A .class file with your compiled script for your program.
- A .rtf or .txt file with a reflection. The reflection should contain a few paragraphs covering the following:
  - What went well in this lab? What was easy and made sense?
  - What didn't go well? What was difficult or your struggled with?
  - What did you learn from this lab? What was the most valuable piece?

Rubric (100 Points)

Code that does not compile will be given an automatic grade of 25, provided that sufficient attempt was made to solve the problem.

The .java Source Code File (80 Points)

25 Points (Source Code Compiles)
The source code compiles when run with the `javac` program. It creates a classfile that can then be run by the `java` program.

15 Points (Calculation of centripetal force is correct)
The code calculates centripetal force correctly. It correctly declares and initializes variables. It correctly handles the order of operations.

5 Points (Uses Constants)
The code uses constants in the calculations, and does not hard code literal values.

10 Points (Scanner used correctly)
The code successfully gets input from the user on stdin using a Scanner. Only one Scanner object is created.

10 Points (User Friendly Input and Output)
The program uses good prompts to tell the user what to enter. Prompts are well formatted and easy to read. Output is well formatted and easy to read.

5 Points (Math.pow() Used)
The code uses Math.pow to raise a number to a power.

5 Points (String Concatenation)
The code prints a well formatted string to the user, stating the results of the calculations.

5 Points (Style)
The source code features good indentation and spacing. The class name is descriptive for the task it performs.

The .class ByteCode File (10 Points)

5 Points (.class file runs)
The submitted .class file runs with the `java` program.

5 Points (Matches Functionality of Source File)
The submitted .class file appears to match the functionality described in the source file.

Reflection (10 Points)

5 Points (Completeness)

The reflection covers the required topics; what went well, what didn’t go well, what was learned.

5 Points (Quality)
The reflection was written in complete sentences. It features multiple sentences, is coherent, and thoughtful.