CS 1331 Programming Exercise 03 – Math and Random

Authors: Ananay, Ishuma, Will, Lucas, Jeff, Rachna, Jack

Problem Description

This assignment will test your basic knowledge of the Math library and java.util.Random.
A board game manufacturer is trying to see how much the plastic used in different parts of their board game will cost. The board game is called "Shapes of Dunshire" and has 3 different types of pieces in the game: Spheres, Cylinders and Cones. There are a random number of random sized shapes in the game, so the board game manufacturer needs your help in determining the total plastic that go into making the pieces for a game.

Solution Description

Final Cost of Plastic in the game

Important Notes:

For all random objects, use a useful class from the java.util package.
For all rounding, use java.lang.Math. (Hint: you will also have to use *, /, and watch out for integer division!).

1. Create a class called `ShapesOfDunshire`
2. Inside `ShapesOfDunshire`, create the main method. All code for this assignment will be written in the main method.
3. Within the main method:
    i. Create a double variable called `finalVolume`, that will store the final volume of all the pieces.
    ii. Create an integer variable called `numPieces`. This variable will contain a random value in the range of 10 inclusive to 20 exclusive. (An alternate representation is [10, 20), where a parenthesis means exclusive, and a square bracket means inclusive.)
    iii. Print the `numPieces` using the following format:
    "Creating Shapes of Dunshire game with <`numPieces`> pieces."
    Don't forget to leave a new line after you print this!
4. Create a loop that iterates `numPieces` times. Within the loop, generate a random value of type int and store it in a variable called `shapeType` for each shape. `shapeType` is in the range [1, 3] where 1 represents a Sphere, 2 represents a Cylinder and 3 represents a Cone.
5. Use a switch statement (within the for loop) on the `shapeType` to carry out the rest of the operations:
    i. For a Sphere
        A. Pick a random double radius in the range [5, 10) cm and round it to 2 decimal places.
        B. Calculate `volume` as a variable of type double, with the formula $(4 \div 3)\pi r^3$. Round the result to 2 decimal places. Use Math library functions and constants to calculate `volume`.
        C. Add the volume to the `finalVolume`.
    ii. For a Cylinder
        A. Pick a random double radius in the range [5, 10) cm and round it to 2 decimal places.
        B. Pick a random double height in the range [7, 13) cm and round it to 2 decimal places.

C. Calculate `volume` as a variable of type double, using the formula $\pi r^2 h$. Round the result to 2 decimal places. Use Math Library functions and constants to calculate volume.
D. Add this volume to the `finalVolume`.

iii. For a Cone

A. Pick a random double radius in the range [5, 10) cm and round it to 2 decimal places.
B. Pick a random double height in the range [7, 13) cm and round it to 2 decimal places.
C. Calculate `volume` as a variable of type double, using the formula $(1 \div 3) \pi r^2 h$. Round the result to 2 decimal places. Use Math Library functions and constants to calculate `volume`.
D. Add this volume to the `finalVolume`.

6. After you have calculated the volume of each shape, print its type and volume in the following formats:

   For a Sphere:

   "Manufacturing shape of type: <`shapeType`> and volume: <`volume`> cm3. Dimensions: radius = <`radius`> cm."

   Don't forget to leave a new line after you print this! Make sure you print the integer value of `shapeType`.

   For Cylinders and Cones:

   "Manufacturing shape of type: <`shapeType`> and volume: <`volume`> cm3. Dimensions: radius = <`radius`> cm and height = <`height`> cm."

   Don't forget to leave a new line after you print this! Make sure you print the integer value of `shapeType`.

7. Close the for loop.
8. After the for loop, sum the volumes of all the shapes [`finalVolume`]. Round this to 2 decimal places.
   a. Multiply `finalVolume` by the rate, which is \$0.55 per cm^3 of plastic. Store the result in a variable called `price`, which represents the price of the plastic used to manufacture all the parts. Round this variable to 2 decimal places.
9. Print the final volume and price in the following format:

   "Total cost of manufacturing <`numPieces`> pieces, worth <`finalVolume`> cm3 plastic, is \$<`price`>."

   Don't forget to leave a new line after you print this!

Reminder:

For all random objects, use a useful class from the java.util package.
For all rounding, use java.lang.Math. (Hint: you will also have to use *, /, and watch out for integer division!).

Example Output

This will be posted on the PE03 Clarification Thread on Ed Discussion.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- `var` (the reserved keyword)
- `System.exit`

Import Restrictions

You may only import the following:

- `java.util.Random`

Collaboration

Only discussion of the PE at a conceptual high level is allowed. You can discuss course concepts and HW assignments broadly, that is, at a conceptual level to increase your understanding. If you find yourself dropping to a level where specific Java code is being discussed, that is going too far. Those discussions should be reserved for the instructor and TAs. To be clear, you should never exchange code related to an assignment with anyone other than the instructor and TAs.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files.
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- Check on Ed Discussion for all official clarifications

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- `ShapesOfDunshire.java`

Make sure you see the message stating "PE03 submitted successfully". From this point, Gradescope will run a basic autograder on your submission as discussed in the next section. **Any autograder test are provided as a courtesy to help "sanity test" your work and you may not see all of the test cases used to grade your work.** You are responsible for thoroughly testing your submission on your own to ensure you have fulfilled the requirements of this assignment. If you have questions about the requirements given, reach out to a TA or professor via Piazza for clarification.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the assignment (submit early and often). We will only grade your last submission: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

For each submission, you will be able to see the results of a few basic test cases on your code. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.

The Gradescope tests serve two main purposes:

1. Prevent upload mistakes (e.g. forgetting checkstyle, non-compiling code)
2. Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or checkstyle your code; you can do that locally on your machine. (We are not using checkstyle for this programming exercise, but in future HWs we will).

Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.