Programming Exercise 03

Problem Description

Hello, and welcome! Please make sure to read all parts carefully.

For programming exercise 03 you will be creating a class that calls 3 different methods a random number of times. Each method should print something. Throughout this assignment you will utilize loops, the `Random` class, and the `Math` class.

Solution Description

Your program must be named `RandomPrinter.java`. This class should contain all the static methods listed below (note: you may add additional helper methods). Please read all the steps and look at the Example Output carefully before you begin.

Your solution should contain the following static methods:

1. `powersOfTwo()`
    - Should print "`Running the powersOfTwo method`" before printing anything else.
        - Add a newline at the end
    - Should print every power of two that is less than 100 in ascending (i.e. smallest to largest) order.
        - Note: "power of two" refers to 2 raised to some power
        - Ex. 1, 2, 4, 8 are all "powers of two"
    - Each power of two should be printed on its own line using the following format
        - "`  2 raised to the [insert exponent] is [insert power of 2]`"
        - Make sure to add a newline at the end.
        - Note that the line begins with 4 spaces.
        - Note: do not include the brackets or the quotation marks in your printout
    - Implementation Notes:
        - You must use a **`while`** loop for this method
        - You may not hard code the values for the powers of 2. You must calculate them each time the method is called.
2. `parameterPrinter(int parameter)`
    - Should print "`Running the parameterPrinter method`" before printing anything else.
        - Add a newline at the end
    - Should simply print out the value of the parameter, `parameter`, in the following format:
        - "`  Method took the parameter [insert parameter value]`"
        - Make sure to add a newline at the end
        - Note that the line begins with 4 spaces
        - Note: do not include the brackets or the quotation marks in your printout
3. `randomNumber()`
    - Should print "`Running the randomNumber method`" before printing anything else.
        - Add a newline at the end
    - Should print a random number between 1 and 10 **(inclusive)** in the following format:

        - "   Your random number is [insert random number]"
        - Make sure to add a newline at the end
        - Note that the line begins with 4 spaces
        - Note: do not include the brackets or quotation marks in your printout
    - Implementation details:
        - You must use `Math.random` for this method
4. `randomController(Random rand)`
    - Should print `"Starting to call the methods!"` before doing anything else.
        - Add a newline at the end
    - This method should first call `powersOfTwo` a random number of times.
        - Use the passed in `Random` object to get the random number of times the method should be called – between 1 and 5 **(inclusive)**
    - It should then call `parameterPrinter` a random number of times
        - Use the passed in `Random` object to get the random number of times the method should be called – between 5 and 10 **(exclusive)**
        - You should pass the number of times you call this method as the parameter to this method
            - Ex. If your random number determined you would call the method 3 times; the method call would be `parameterPrinter(3)`
    - Finally, it should call `randomNumber a random number of times`
        - Use the passed in `Random` object to get the random number of times the method should be called – between 1 and 10 (**inclusive)**
    - Implementation details:
        - You must generate the random numbers for each call independently.
        - i.e. You will need to generate at least three different random numbers

The Main Method

- Make sure to add a main method to your program.
- Your main method should create a `Random` object and pass the object into the `randomController` method.
- Do nothing else

Example Outputs

Example partial outputs for the first three static methods are shown below.

powersOfTwo output:

```
Running the powersOfTwo method
    2 raised to the 0 is 1
```

parameterPrinter output (assuming 3 was passed in):

```
Running the parameterPrinter method
    Method took the parameter 3
```

randomNumber output:

```
Running the randomNumber method
    Your random number is 7
```

Rubric

[100] **RandomPrinter.java**

- [32] randomController calls each method a random number of times
    - [8] Uses Random object to determine how many times to call each method and each method's random number is generated independently
    - [8] powersOfTwo random number has the correct range
    - [8] parameterPrinter random number has the correct range
    - [8] randomNumber random number has the correct range
- [8] Main works as expected
- [20] powersOfTwo is correct
    - [12] Uses a while loop and is not hard coded
    - [4] format is correct
    - [4] calculations and bounds are correct
- [20] parameterPrinter is correct
    - [14] correctly prints the parameter
    - [6] format is correct
- [20] randomNumber is correct
    - [12] Uses Math.random
    - [4] format is correct
    - [4] bounds are correct

We reserve the right to adjust the rubric, but this is typically only done for correcting mistakes.

Allowed Imports

To prevent trivialization of the assignment, you may only import `java.util.Random`.

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`

Collaboration

Collaboration Statement

To ensure that you acknowledge a collaboration and give credit where credit is due, **we require that you place a collaboration statement as a comment at the top of at least one .java file that you submit**. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material

that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

Allowed Collaboration

When completing homeworks and programming exercises for CS1331 you may talk with other students about:

- What general strategies or algorithms you used to solve problems in the homeworks or programming exercises
- Parts of the homework you are unsure of and need more explanation
- Online resources that helped you find a solution
- Key course concepts and Java language features used in your solution

You may **not** discuss, show, or share by other means the specifics of your code, including screenshots, file sharing, or showing someone else the code on your computer, or use code shared by others.

Examples of approved/disapproved collaboration:

- **approved**: "Hey, I'm really confused on how we are supposed to implement this part of the homework. What strategies/resources did you use to solve it?"
- **disapproved**: "Hey, it's 10:40 on Thursday... Can I see your code? I won't copy it directly I promise"

In addition to the above rules, note that it is not allowed to upload your code to any sort of public repository. This could be considered an Honor Code violation, even if it is after the homework is due.

Turn-In Procedure

Submission

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

- RandomPrinter.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.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. 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. Each test typically corresponds to a rubric item, and the score returned represents the performance of your code on those rubric items only. 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:

- Prevent upload mistakes (e.g. non-compiling code)
- 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 or run your code; you can do that locally on your machine.

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.

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 Piazza for a note containing all official clarifications