Basic Classes + Arrays Timed Lab

1 Introduction

This timed lab will test your ability to design classes and perform basic array manipulations and operations. You will have 45 minutes to complete the assignment. If you have time left over, you are recommended to make your own `main` method to test out your code! This will allow you to ensure that your code works in the way you intend before submitting it for grading.

2 Problem Description

Part 1: reverseProduct

Create a file named `ArrayUtils.java` which contains a public static method named `reverseProduct`. The `reverseProduct` method takes in two int arrays `a` and `b`, each of length `n`, and returns the **sum** of the **products** of each array's opposite element. In more rigorous mathematical notation, it returns:

$$\sum_{i=1}^{n} a_i b_{n-i}$$

For example, the reverseProduct of `[3, 7, 10, 2]` and `[1, 4, 9, 5]` is

`(3 * 5) + (7 * 9) + (10 * 4) + (2 * 1) = 15 + 63 + 40 + 2 = 120.`

Part 2: House class

Create a file named `House.java`, then write a `House` class that keeps track of the following properties:

- Name of the homeowner. *(String)*
- Number of bedrooms. *(int)*
- Number of bathrooms. *(double)*
- Presence of a garage. *(boolean)*
- The total number of `Houses` instantiated.
- It should maintain the *class invariant* that the name of the owner is neither the empty string nor null. If ever given an invalid name, it should have a default value of "HOMEOWNER."

It should have the following methods:

- A constructor which initializes the instance variables of the House properly, taking in the name of the homeowner, number of bedrooms, number of bathrooms, and presence of garage IN THAT ORDER.

- `public String getOwner()`
  - This returns the name of the home's owner.
- `public void setOwner(String owner)`
  - This method sets the owner to a given value.
- `public void buildBedroom()`
  - This method increases the number of bedrooms in the house by one.
- `public String toString()`
  - This method returns a String representation of the `House` in the example format "House owned by Samantha, 3 bed/1.5 bath, garage: true, total houses: 4". Make sure your output matches the example as closely as possible for full credit.

3 Checkstyle

Review the Style Guide and download the Checkstyle jar and associated XML file. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. You can easily count Checkstyle errors by piping the output of Checkstyle through `wc -l` and subtracting 2 for the two non-error lines printed above (which is how we will deduct points). For example:

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | wc -l
      2
```

> Food for thought: is there a one-liner like above that shows you only the number of errors? Hint: `man grep`.

Alternatively, if you are on Windows, you can use the following instead:

```
C:\> java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | findstr /v "Starting audit..." |
     findstr /v "Audit done" | find /c /v "hascode()"
0
```

The Java source files we provide contain no Checkstyle errors. For this assignment, there will be a maximum of **10** points lost due to Checkstyle errors (1 point per error). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

4 Turn-in Procedure

Submit all of the Java source files you created or modified to T-Square. Do not submit any compiled bytecode (`.class` files), the Checkstyle jar file, or the `cs1331-checkstyle.xml` file. When you're ready, double-check that you have submitted and not just saved a draft.

5 Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your timed lab files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your work and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your TL submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this timed lab. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
   (a) It helps insure that you turn in the correct files.
   (b) It helps you realize if you omit a file or files.[1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
   (c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling solutions will receive few to zero points. Also recall that late work will not be accepted regardless of excuse. Treat the due date with respect. The real due date is posted on T-Square. Do not wait until the last minute!