Homework 1 due Jun 16, 2024 11:53 CDT

Problem Description

You've now been introduced to the basics of programming in Java. This means you can write simple, but useful programs! We're going to practice using those skills by creating and manipulating several variables, of different types.

For this homework, we'll give you the broad strokes of what you should do, and you'll implement the directions in Java code, inside two classes - one called `PrimitiveOperations`, and another called `StringOperations`. Refer to HW0 and Module 1 for a refresher on creating classes in Java. All of your code should execute inside the main method of each class - remember that a properly written `main` method in Java has a very specific method header!

Make sure you read the entire document! In fact, consider each assignment in this course series as both an evaluation of your ability to apply Java concepts **and follow problem specifications *to a T***. As a (future) professional software developer, you will find both of these skills highly important. You can have a program working fine on your computer's JDK, but when submitting it to the grading tool (Vocareum), errors might pop up. Heads up that the most common result of these grader errors has been due to a particular part of the assignment (possibly small and/or found at the very end of the assignment description) not being incorporated in the submitted program.

Solution Description

This homework will be split into two parts - the first will be focused on working with the various primitive types that Java provides us. The second will be focused on using methods that come with the `String` class. Additionally, this homework requires you to do some basic debugging on erroneous code that we have provided with this document and to submit some information to help us get to know you better.

Section 1: Primitive Operations

All of the instructions in this section should be carried out in the class `PrimitiveOperations`, in the main method, and must be in the respective order.

- First, declare and initialize two variables, an integer type (byte, short, int, or long) and a floating point value (float or double). The names and values can be whatever you like, for this step and all others. Make sure that the numbers you choose can be stored within the respective primitive type you choose. Print each of these values out on their own line using `System.out.println()`.
- Multiply these variables together, and assign the outcome to a `new` variable, ensuring that no data is lost. For example, if I multiply 5 and 3.5, the answer should be 17.5.Print out this new value.
- Use casting to convert the integer from the first step to a floating point value and store that in another `new` variable. Print out the value.
- Use casting to convert the floating point value from the first step to an integer type and store that in a `new` variable. Print out the value.
- Shifting focus, declare a `char` variable, and assign an uppercase letter to it. Print out this value.
- Using a *numerical operation*, change the letter to the same letter, but in lowercase. Use a numerical operation - do not reassign the variable. You may want to review a table of ASCII values as you're working on this section. Print out the new `char` value. **Hint:** you'll likely have to use casting to get this to work.

Section 2 - String Methods

All of the instructions in this section should be carried out in the class `StringOperations`, in the `main` method.

- Create a new `String` object and assign it your name. Print it out.

- Pick the first letter in your name, and replace it with 'A'. Then, replace the last letter in your name with 'Z'. Print out the result. Recall that, in Java, strings are *immutable*, meaning you cannot change a String in-place. Do NOT just hard-code a new String with the first and last letters changed.
- Lastly, let's work with some URLs. Declare a new `String` and give it the value of some web address, in the form `www.name.tld`, such as `www.gatech.edu` or `www.stackoverflow.com`. Print out this address.
- This last operation could be a little tricky. Create a substring of the variable that's just the "name" section, and concatenate the integer "1331" to the end. For example, `www.gatech.edu` would become gatech1331. Print out this final result. **Note:** the String class has a `.length()` method which you'll likely find useful here but is not necessary.

Section 3 - Learning to debug

With this document we have provided three Java programs, `Bad1.java`, `Bad2.java`, and `Bad3.java`, each of which contains a single error. Attempt to compile and run each program and observe the resulting error message. Note, you will have to compile and run them on your own local machine, outside of the Vocareum environment. Then, fix whatever is wrong with the programs so that you are able to compile and run them.

You will submit the fixed versions of `Bad1.java`, `Bad2.java`, and `Bad3.java`.

Allowed Imports

To prevent trivialization of the assignment, you are not allowed to import anything.

Feature Restrictions

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

- `var` (the reserved keyword)

Allowed Collaboration

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

- What general strategies or algorithms you used to solve problems in the homeworks
- 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.

The Vocareum (code editor) interface has six main components:

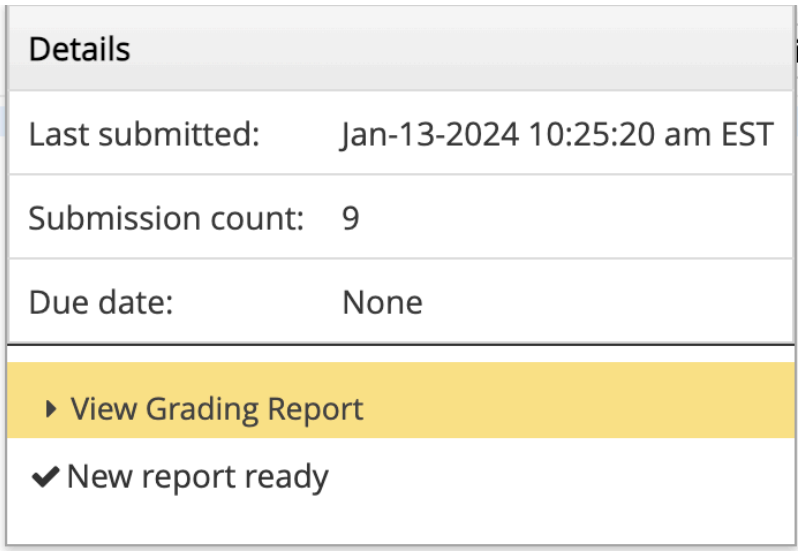
- The **Files Navigation** is in the top left. This lets you choose from multiple available files, in the "work" folder.
- The **Build / Run** button. For all assignments in this course, the build and run button will perform the same action: compile your code and run a file scan. Building and running your code will not count towards your total allowed submission attempts, therefore you are free to build / run as many times as needed.
- The **Submit** button. This will compile your code, grade your assignment, and produce a grading report. **We expect that you build or run your code on your local computer/JDK before submitting to ensure that there are no issues that will prevent your code from being graded and that every submission attempt will generate meaningful results**. Your local JDK's errors, in some cases, will be even more descriptive than what Vocareum can offer you.
- The **Reset** button. This will revert all your changes and reset your code to the default code template.

- The **Code Window**. This is where you will write your code. Again, We highly recommend copying the starter code and working in your preferred IDE.
- The **Output Window**. This window will appear whenever you build, run, or submit your code and will display the results for you to view.

For additional help, please visit the Vocareum information page located in the course information module!

Submitting

You will submit through Vocareum, which will be loaded in a new window. Edit the .java files, and then submit once you are ready to have your homework autograded. Once it finishes autograding (this may take several minutes - be patient), you can access your report in the details tab. You can resubmit as many times as you want.



For learners unable to access the Vocareum environment, the provided Bad#.java files are as shown:

Bad1.java

Bad2.java

Bad3.java