Java Programming Style Guide

CS 203 Computing and Algorithms III

Programming assignments should be written in a way that is easy to read, write, modify and maintain. Correctness of the program is important, however the style in which it is written is just as important. I will be looking for codes that are "excellent" in terms of design and style. The grade for style and design depends on how the appearance of the output is and on how easy it is to use.

You must check to ensure that the following items are met before you turn in your code.

1. Header for the file: Every file for the assignment can only one class. Every class that is part of an assignment must contain the following items as comments on the top the class. These include author, assignment number, name of the file and description of the class. Note that the comment must start with "/**" and must have a * on each line and end with */

```
/**
* Author: Saroja Kanchi
* Assignment Number #2
* Coin.java
* This class contains methods that represent a coin
*/
```

2. ***Variable Names:*** All variable names must be meaningful. Only exceptions are for loop indices and they can be i, j, k etc. Note that every variable name must begin with a small letter and if two or more words are joined together, then all except the first word must begin with capital letter. `Examples:`
`count, sum, grandTotal, lengthOfPassengerName etc.`

3. Method names: All method names must obey the above rule 2, however in addition, the method names are expected to indicate an action being performed, or a job that is being accomplished. `Examples:`
`initialize, countAll, checkIsleOrWindow, etc.`

3. Class names: All class names should be meaningful. All class names must begin with a capital letter and if two or more words are combined to form a class name, then the first letter of every name should be a capital letter. `Examples:Passenger, CheckingAccount, SavingsAccount etc`

4. Constants: User defined constants in Java are obtained by using the keyword "final". Use all capital letters in defining constants. Also use the "_" to combine words in constants. `Examples: PI, MAX_SIZE, MIN_SIZE etc.`

5. Header Comments: Every method within the class must contain a header as below. Note that the comment must start with "/**" and must have a * on each line and end with "*/". Note that "@param" represents input parameter and you should each one on a separate line. The "@return" indicates description of the return value.

```
/**
 *  This method adds two integers
 * @param  first  the first number to be added
 * @param  second the second number to be added
 * @return         the result of the addition.
 */
 public int add(int first, int second){
     int result;
     result = first+ second;
     return(result)
 }
```

No method should be more than a page of printout.

6. White Space: Separate the code into blocks by using white space. The code should look clean and should be readable, and it should easy to find the beginning and end of methods, classes, loops etc.

7. Indentation: Use proper indentation to separate code into blocks. This means that lines inside loops, branching statements and methods and classes should be properly indented. If the body of the loop, if statement of else statement is a single statement, (not a block) indent the statement three spaces on its own line. Put the left brace, {, starting each new block on the same line as the construct that defines it, such as a class, method, loop, if statement or else clause. The terminating right brace (}) should line up with the construct. Example,

```
while (total < 9) {
total += 5;
System.out.println("The total is " + total);
}
```

8. In-line Comments: Comments must be accurate. Keep comments updated when the code changes. Every related block of code (5 to 8 lines) must be preceded a line of comment. Do not comment every line. There cannot be pages and pages of code no single line of comment. Do not use java language while writing comments, instead use english sentences. Inline comments must begin with "//" for each line.

Note: Here is way to check if your comments are appropriate: Assume that a person with no knowledge of your assignment and no knowledge of Java reads only the comments of your program. The person should understand "what" task the program does and "how" it accomplishes that task.

9. Line Length: Restrict the length of lines in your source code and output to at most 80 characters.

10. Messages, Prompts : Do not condescend. Do not attempt to be humorous. Be informative, but succinct. Define specific input options in prompts, when appropriate. Specify default selections in prompts when appropriate. For every input you are requesting from the user, you must describe the format you want the imformation in.

11. Output: Label all output clearly. Present the information in a consistent manner. Ensure that output is readable.

Design Guidelines

A. Design Preparation:

1. The ultimate guideline is to develop a clean design.
2. Think before you start coding. List all the classes, and variables and methods you are going to have in each class.
3. A working program is not necessarily a good program.
4. Document your design with UML class diagrams.

B. Structured Programming:

1. Do not use "continue" statement.
2. Only use "break" statement to terminate cases in the "switch" statement".
3. Have only one "return" in each method unless it complicates the code too much.
4. Any piece of code (5 or more lines) that is repeated in the program must be made into a method.

C. Classes and Methods:

1. Do not have additional methods in a class that contains "main" method. Define the class that contains the "main" as the first class in the file.
2. You must have instance variables for each class that are meaningful. If multiple methods are using the same variable, make it an instance variable.
3. Every class should contain constructor, setters and getters, equals and toString methods. There can be exception to this rule for Driver classes and classes where these do not make sense.

D. ***Modifiers:***

1. You must not have any instance variable declared public. Only finals may be public if needed. All instance variables must be private or protected.

2. All methods in general should be public unless there is a reason to make them private

3. There can be ONLY ONE static method in your program. That method would be main().

4. There cannot ANY static variables in your program. You may have static finals however

E. Runtime behavior:

1. Check for errors as required by the assignment.

2. You must try-catch exceptions that occur with I/O or String/Array bounds.