DEVS101

CODING STANDARDS DOCUMENT

SKILL TREE MAKER

1 Naming Conventions

1.1 Class Names

- eg *ClassName* -Class names should begin with a capital letter and each new word within the name should begin with a capital letter.

1.2 Exception Classes

- *ClassNameException* - Exception classess should follow the same rules as normal classes but in addition to that, the name should always end with the word Exception.

1.3 Constants

- These public static variable should always be written using upper-case letters and sometimes an underscore to make the name more clearer and readable.

1.4 Methods or Functions

- Methods should be given a descriptive name of what exactly they do and the name should begin with a lower-case letter and any new name within the name should begin with a Capital letter.

1.5 Variables and Attributes

- variableName - variables and attributes should always start with a lowercase letter and each new word within the same name should begin with a Capital letter, meaning to say they have to be exactly the same as the methods.

1.6 Accessors

- *eg getAttribute()* - they should follow the same rules as the methods and should always be named after the variable which they intent to access.

1.7 Mutators

- eg setVariable(.....) - they should follow the same naming conventions as methods i.e they start with a lowercase and any new name within the name should begin with a capital letter.

1.8 Packages

- eg display.package - names of the packages should always be in lower-case letters and no upper-case.

1.9 File names

- Area.java - file names should be descriptive enough to tell what they contain inside and name should always begin with a capital letter.

2 Exceptions

- Exceptions should be applied and used wherever necessary and these should always be checked.
- Exception classes should be avoided at all costs and more meaningful classes with meaningful names should be used.No more functionality should be in these classes except only catch statements.
- Appropriate catch statements should be used which allows the program to continue if need be rather than crashing.

3 Bracketing

- Curly brackets should always be used for while, do-while, if, for statements even if they are not really needed sometimes for example when a body contains only one statement.
- An opening curly bracket that follows a class header should be indented indented by one space but those that follows after should not be indented by any space.

- Closing curly brackets should always be on their own line i.e the line after the last statement and on the same indentetion as the header on which the block begins.

4 Identifiers

- Identifiers should be descriptive and of a reasonable length. For example a variable name should have a single use and its name should be related to its use or what it does.Strictly the variable name should not be used for multiple purposes as this would create confusion amongst the variables.
- The convention is that variable names, methods,attributes they should begin with a small letter and then a capital letter wherever there is a new name within that name.
- Class names should begin with a capital letter and then everytime a new word is encountered it becomes a capital letter.

  eg of Variable is *myVariable*, of class attribute is aClassAttribute, class name is ClassName.

- accessors and mutators should have names related to attributes they access or provide access to. They should begin with the prefix *get* for those getting values and *set* for those setting values.

  for example *setItem() and getItem()*.

- Variable names with a single character should be avoided for example int x.They can only be declared with a single character if its in a loop otherwise, its strictly forbidden.

5 Final Variables

- Magic numbers should be avoided within the code.Incase the program needs to be altered it becomes very difficult to know what to change and know what not to change.

- Instead final variables should be defined within the code.
  for example
  private final int A = 2. private final int b = 20.

- The only time when they can be used is when used as variable initialiser only.
- In the case that final variables are needed by more than one class then it means they should be declared with the word public so they can be accessed anywhere in any class.
- Final variables should be defined with uppercase letters hence easily distinguishable from other class attributes.
- the word final clearly indicates that they cannot be changed.

6 Indentation and Layout

- Lines should be kept of very small length so that they can be readable and easier to understand.
- Once a line reaches a certain length, it has to be broken and indented further than the previous line.
- Indentation should be 4 spaces for each level, blank lines to seperate methods and also to emphasize or make it clearer where each method ends.
- Methods should be defined firstly in class, in order of visibility – public methods first, private ones after
- attributes should be defined mostly after the methods in a class.
- public final variables being the exception which should be defined first.

7 Imports

- The use if the * form of*import* should be avoided. Each class should be specifically imported as required, and any unused imports should be removed since they are not needed.

8 General Commenting

- Its a mandatory to add comments so that the code becomes very easy to read and understand
- Its advisable to add comments in blocks and not in lines as it might cause some confusion and the code wont be clean.
- For line comments use *// and multiline use /* ..*/*

  for example int index = 1; // index of starting point.

9 Github File Structure

- All files and documents are sorted in their respective folders.
- Documentation will be seperate in it's own folder "Documents".
- All source code and packages will be sorted in the folder "src".
- All build files will be located in the "build" folder.