RHHS Java Programming Style Guide

The following guide specifies Java programming standards that are acceptable for students enrolled in computer science at RHHS. It is best for students to follow the correct etiquette as they learn to prevent bad programming habits from emerging.

1. File Naming and Class Layout

- Upper camel case must be used to name source files and classes
- Files should be named identical (case-sensitive) to the name of the top level class.
- All source files should have the extension .java *(example: HelloWorld.java)*
- No special characters should be used in the class/filename
- Only one class is permitted per file, unless an inner class is required
- Complex programs consisting of multiple files should have files placed in the appropriate subfolder *(example: src/resources/docs/bin)*
- Package definitions should be used *(gr 12 only)*

2. Source File Structure

- A source file should contain, in order, a header, package statements, import statements, a class
- Each of the above sections should be separated by a single blank line
- The header must contain class_name, version, author, date, and description
- The header should be contained in single a */* ... *** block
- Import statements should <u>not</u> include wildcards *(.*)*
- Main methods should be consistently placed either as the first or last method in the class

3. Source code Format

- Braces must always be used with *if, else, for, do , case, try-catch*, and *while* statements even if they consist of a single line
- Braces must be used according to the following example:

```
public static void main (String[] args) {          //braces immediately follow statement
    if (age > 12) {
        try {
            something();
        } catch (UnknownException e) {          //All braces spaced to line up (space used to indent)
            repairSomething();
        }
    } else {                                    //no space when a brace is part of a block statement
        while (1<0) {
            universeExplode();
        }
    }
}
```

4. Variable declarations

- One variable should be declared per line, unless they are closely related. *(example: int x, y;)*
- All variables should be named according to lower camel case and only consist of letters and numbers (no special characters) *(good example: double worldGravity, bad example: int X_loc)*
- Variables should always start with letters
- Variables should be declared at the start of their containing block of code *(gr 11)*
- Local variable should be declared close to the point they are used *(gr 12)*
- Array declarations should have square brackets immediately following the type. *(example: int[] numbers;)*
- All variables must be descriptive *(example: int playerXVelocity vs. int pX_Vel)*
- Constants are always capitalized *(example: static final double PI = 3.14;)*
- Multiple words in constants are separated with an underscore *(example: WORLD_GRAVITY)*
- Declarations should always be organized logically to allow for easy readability of code

5. Commenting

- Javadoc comments should always be used
- Logical sections of code should be organized in a group and can be identified with comments *(example: Mark where the inner classes section begin and end. Put all inner classes in this area)*
- All methods, classes, etc. are preceded by a block comment with method name, description, and Javadoc comments. See Example below.

```
/**
* universeExplode
* This method accepts a universe as a parameter and implodes it. The universe is removed and the
* method returns a boolean based on success of the algorithm.
* @param A 2D byte array that holds data representing a universe
* @return Boolean, true if the operation was a success, false otherwise.
*/
public boolean universeExplode (Byte[][] universe) {    //braces immediately follow statement
        …code here...
}
```

6. Method and Class Naming

- Upper camel case is always used for classes *(example: class HelloWorld)*
- Lower camel case is always used for methods *(int calculateProduct(int a, int b))*
- Returns, parameters, local variables are all done in lower camel case
- *@Override* is always used when a method has been overridden (gr 12)

7. Programming Practices

- Always use additional brackets to make complex conditions easy to read and avoid potential errors *(example: if ((x>3) && (x>0))*
- *Try-catch* statements never contain a blank catch block *(gr 12)*
- Blank lines should be used to separate logical sections of code
- Programming style should be consistent throughout the entire program
- Priority should be given to readability over conciseness