CSCI 5308 Java Style Guide

Based on Nelson Fisler's CS 18 Java Style Guide (Brown University)

Naming

- All identifiers except constant names should begin with a letter and consist of only letters and numbers.
    - **Package** names are all lower case, with consecutive words concatenated together. For example: `mypackage` o **Class names and interface** names are written in *UpperCamelCase*, in which words are concatenated and every word starts with an uppercase letter. For example: `ArrayList` and `Comparable`
    - All other identifiers (variables, methods, fields, parameters, etc.) are written in *lowerCamelCase*, in which words are concatenated and every word except the first starts with an uppercase letter.
    - One-character names should not be used except as indexes in loops.
- Constants must begin with an upper-case letter consist of one or more words or numbers written in upper-case letters, with the words separated by underscores (_) For example: `MIN_COFFEE_AMOUNT`

Formatting

- Indent with the space character only, **not the tab character**.
- When a block or block-like construct starts, e.g., open brace "{", increase indent level by **two (2) spaces**. When the block ends, return to the previous indent level.

```
public void callMe(String name) {
if (name.equals("Bruce")) {
    System.out.println("Hi Mate!"); // Indented 2 spaces
  } else {
    System.out.println("Hi " + name);
  }
}
```

- The indent level applies to both code and comments throughout the block.
- A line of code should be **at most 80 characters** in length. Lines that are longer than 80 characters should be line wrapped. When line-wrapping, each continuation line must be indented by four (4) spaces from the original. For example:

```
  superDuperLongName = superDuperLongName +
yetAnotherSuperDuperLongName;
```

- Insert a single blank line (vertical white-space) in the following places, only:
  - After a class declaration
  - After each class member, e.g., fields, constructors, methods, etc.
  - After most closing braces. For example:

```
public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Hello world");
  }
 }
```

  There are exceptions:

- Do not skip a line until the very last curly brace in an if/else block. For example:

```
if (age < 16) {
  System.out.println("Please wait");
} else if (age < 18) {   System.out.println("Nearly
there");
} else {
  System.out.println("Missed it");
}
```

- If a brace is the last line in a method:

```
public static void main(String[] args) {
  if (args.length > 0) {
    System.out.println("Hello world: " + args[0]);
  }
}
```

  - Before a method's final return statement (usually). For example:

```
… counter++;
 return
counter;
```

  Note: if a method (or a block of code) consists of only a return statement, or is less than five (5) lines long, no blank lines are needed

  - Within method or block bodies as needed to create logical blocks of code. This informs the reader of the logical structure of your code.
  - Finally, DO NOT insert a blank line between a declaration and an opening brace.

```
// Bad style (this is more of a religious topic)
public class HorribleStyle
```

```
{
…
  public static void main(String[] args)
  {
  …
  }
}
```

- A (single) space appears in the following places:
  - o On both sides of any binary or ternary operator (operators that act on two or three elements) or operator-like symbols. Examples of binary operators include +, -, *, /
  - o Before any open curly brace. o After a semi-colon ';' or comma ',' if something is following it on a line o After a curly brace, if not followed by a ';' or an end-of-line.
  o Separating a keyword from an opening parenthesis. For example:

```
for (int i = 1; i < 10; i++) {
  …
}
if (age < 10) {
  …
} else {
  …
}
```

Variable Declarations

- Variable declarations should declare only one variable per line
- Variables should be declared as close as possible where they are first used. • Variables should be initialized either at, or right after, declaration For example:

```
int xCoord = INITIAL_X_COORD; int
yCoord = computeYCoord(xCoord);
int distance = getDistance(xCoord, yCoord);
…
```

Statements

- Each line should contain at most one statement.
- Compound statements such as `if` statements, loop (`for, while, do while`) statements, `try-catch` statements, etc, are all statements of the form

```
… { // body of  compoundstatement
  statement1;
statement2;
  …
} …
```

Always use braces to enclose the body of the compound statement, even if there is only one statement within it.

- `return` statements with a value should not use parentheses (unless they are required for clarity)
- A switch statement should have the following form:

```
switch (condition) {    case CASE1:     statements;     // falls
through   case CASE2:     statements;      break;   case CASE3:
statements;      break;   default:      statements;      break;
}
```

If a case statement doesn't have a break, add a comment so there is no doubt that the missing break is intended.

Comments

- **Block Comments** are used to provide descriptions of files, methods, data structures, and algorithms.
  - Block comments **must** be used at the beginning of each file and before each method. ○ Block comments can also be used in other places, such as within methods. In such cases, the comment should be indented to the same level as the code.
  - A block comment should be preceded by a blank line to set it apart from the rest of the code. Code can immediately follow a comment, without a new line.
- **Single-Line Comments** are a special form of block comments that appear on a single line.
  - If a comment can't be written in a single line, use the block comment format.
  - It is ok to use the // style comments for single line comments.
- **End-Of-Line Comments** use the // comment delimiter and are used to
  - Describe what a single line of code is doing ○ Single-Line Comments
  - Comment out a line of code (or a partial line of code). **Never leave commented out code in production code**. • Example of all coding styles

```
/* This is a comment about how to comment
*    It is common (but not necessary) to have a
*    line of * on the left-hand side  * The answer to
life, the universe, and everything
 */
int answer = 42;            // This is the best answer so far

…  if (answer != 42) {          // Do we know the
answer?
  /* Time to hitchhike across the galaxy. */
…
}
```

Miscellaneous

- Avoid using a wildcard to import entire packages if multiple classes from one package are imported. Example of all coding styles

```
import java.util.List;
import java.util.Map;
…
```

- The structure of Java File should comprise the following components:
  - Comment block
    - Name of file
    - Author
    - Purpose of class or interface
    - Description of algorithms or data structures implemented by the class o Import statements
  - Class declaration, with the same name as the file
    - If this class is where the program starts running, the main method should be near the top of the file.