CS161 Style Guide

1 Usage of Variables

- Declare and initialize variables on the same line if possible (the only exception being instance variables)
- Declare a variable as close as possible to where you need to use it
- Delete any variables that are not used
- Any variable used in only one method should be declared locally in that method

2 Commenting

Be specific with your comments without over commenting. Break up long comments onto multiple lines. (A person should not need to scroll right to read the end of your comment.)

Classes

At the top of each class, you should have a Javadoc block comment that includes a brief description of what the class is for, your name, and the date. For example:

```
/**
 * Brief description of the class
 * @author <your name here>
 * @version <date>
 */
```

Methods

Above each method there should be a brief description of *what* the method does but not *how* the method does it. That is, you should describe the purpose of the method without discussing any implementation details at all. For each input argument, there should be a corresponding `@param` line. If the method has a return value, there should be a corresponding `@return` line. For example:

```
/**
 * Tell what the method does (but not how)
 *
 * @param <paramName1>  <what the parameter represents>
 * @param <paramName2>  <what the parameter represents>
 * ...
 * @return <what is returned>
 */
```

Variables

In general, the names you pick for your variables should be informative enough that you do not need to add a comment. There are, however, instances where comments might be needed for variables:

1. You need to communicate additional important information

   ```
   private double foodEaten; // units are lbs
   ```

```
private int[][] wordCounts;  // A document-by-word matrix of word frequencies
```

2. You want to describe the purpose of a whole group of variables

```
// These variables keep track of the state of the game
private float time;
private boolean isAlive;
private int numLivesRemaining;
```

Inline Comments

Add inline comments to code inside a method if the code is complicated and it is not obvious what's happening. Don't comment every line – instead, try to comment entire segments of code.

3 Naming Conventions

Choose names that suggest the meanings of the things being named. For example, if there is a variable of type `Circle` that is used as the body of a robot, then a good name would be `robotBody`.

- Names that have nothing to do with the program are very bad! For example, `Circle frodo`, just makes your program harder to read.
- Names that are not specific or descriptive enough are generally bad names, for example `Circle circle` or `Square i`.
- Try *not* to name objects sequentially. You should only do this when the objects are related in some way that is not otherwise expressible. Ex., `Circle circle1`, `Circle circle2`, `Circle circle3` are not as good as `robotHead`, `leftHand`, and `rightHand` if the circles are being used to draw a robot body.

Following Java convention,

- All class names are capitalized
- All method and non-final variable names use `camelCase`
- All final variable names use `CAPITAL_LETTERS_WITH_UNDERSCORES`

4 Magic Numbers

A magic number is a number that appears in your code with no explanation, e.g.

```
h = 31*h + ch;
```

The number 31 is a magic number – any person reading your code would have no understanding of what this number represents.

Your code should never contain magic numbers. *All numbers should be replaced by appropriately-named constants*, e.g.

```
public class SomeClass{
    private static final int NUM_DAYS_IN_MONTH = 31;
    ...
    public void someMethod(){
        h = NUM_DAYS_IN_MONTH * h + ch;
    }
}
```

The only exceptions to this rule are the numbers 0, 1, and sometimes 2 if the context is extremely clear.

5 Format and Overall Organization

The arrangement of code inside your class should be as follows:

- All constants should be declared at the top of the class
- After constants you should place the declaration of all your instance variables
- After instance variables should be the class constructors
- After the class constructors should come the instance methods

Blank lines are used to delineate different areas of the code. There should always be a few blank lines between the different sections of your code: separating instance variables from constructors, and separating methods from each other. Get rid of large sections of blank lines.

Delete any extra code that is not used. You would not hand in an English paper with crossed out lines. Similarly, you should not hand in an assignment with commented out code unless there is a good reason you want me to look at it.

Always start a new line after a semicolon. Always leave a blank line before a comment.

In BlueJ, you can highlight your code and select `Edit > AutoLayout` to automatically adjust the indentation of your code.