Style Guidelines

The code for all assignments in cs601 starting Lab 3 Part 2 should follow "standard" code conventions for Java. Please refer to the guidelines below. Note that if your code does not satisfy these requirements, you may receive a deduction for your assignment.

Java Naming Conventions[1]

Class names - should start with an uppercase letter and be a noun (Examples: Student, PetFinder, MovieDatabase). Use camelCase when the name consists of multiple words.

Variable names: should start with a lowercase letter. Use camelCase. (Examples: book, lastName, menuItem, movieTitle).

Method names: should start with a lowercase, should be verbs (Examples: search(), computeArea(), printInfo(), checkoutBook()).

Constants: should contain only uppercase letters, parts of the name should be separated by an underscore (Example: MAX_WIDTH).

White spaces

Surround each binary operator with one space on either side. Use one space on each side of the = operator. Example:
int a = length + width**;**
Here we have spaces around a + and around a =. Same for any binary operator.

No white space inside of the parentheses (after the opening parenthesis and before the closing one). Example: int a = (6 + x) / 3; Here there are no spaces before 6 and after x.

Place one space after comma. Example: public int sum(int i, int j, int k)
Here we have a space after each comma.

Place an empty line after each method.

Indentation

Indent the body of the class, the body of the loop, and the code inside conditionals. Example:

```
int sum = 0;
for (int i = 0; i < 5; i++) {
        sum += i*i;
}
```

[1] Reference: http://www.javatpoint.com/java-naming-conventions

Encapsulation

All instance variables should be **private**, unless there is a very good reason to make them protected (in all assignments in cs601, all instance variables should be private, except for constants). Accessor ("get") methods that return a reference to private data of the class, break encapsulation - you code should not have them.

Documentation

Write a javadoc comment on top of each class, and on top of each method of the class. Example:

```
/** This class contains methods to print greetings in two languages */
public class Hello {
	/** greetInEnglish prints a greeting in English */
	public void greetInEnglish() {
		System.out.println("Hello!");
	}

	/** greetInSpanish prints a greeting in Spanish */
	public void greetInSpanish() {
		System.out.println("Hola!");
	}
}
```

Use @param, @return and other javadoc tags we discussed. Use inline comments too if the logic of the method is complicated.

Exceptions

Handle checked exceptions. Use *try with resources* whenever possible. Catch the most specific exception first before catching a more general one. Print a meaningful message inside each catch block and print the exception.

Github

Your commit messages should be meaningful; they should describe what works in this version of the assignment ("Commit1", "Commit2" etc are *not* meaningful commit messages). You should commit and push your code frequently as you work on the assignment. Most assignments have a *minimum number of commits* requirement.