CS 210 Programming Style Guide

Overall Principle

Whenever you write a piece of computer program code, keep in mind that it's not only intended to be run by the computer, but also (or more importantly) intended to be read and understood by people (e.g. your instructor, grader, TA or tutor, even yourself). It is truly "write once, read multiple times". So the overall principle is to **make your code readable**. One good question to ask yourself is: "if some other random person reads my code, can he/she understand it easily?"

Formatting

Keep your formatting **consistent** throughout your program. This includes:

- Indentation: indent one tab after "{" and un-indent one tab after "}".
- Blank lines: have a blank line between methods and between groups of related code.
- Spaces: put a space between operators.

Some of the above aspects can be easily achieved by a simple command in any modern IDE. For example, in NetBeans or Eclipse, use "Source -> Format".

```
int total = 0;
for (int i = 1; i <= 100; i++) {
	total += i;
	System.out.print(total + ", ");

	// Add a line break every 10th line
	if (i % 10 == 0) {
		System.out.print("\n");
	}
}
```

Header

Always include a header at the top of your assignment. The header should include your first name, last name, the assignment number and name, and a short description of your code.

Names (variable, method, class...)

All names should:

- Follow naming conventions. Use camel case, such as "arrivalTime" for variable names and "changeFlightNumber" for method names.
- Concisely describe what a variable means or what a method does. Never use mysteriously abbreviated names, such as "x" or "tktAry1". Don't use unnecessarily long names, such as "theNewArrayThatContainsTicketInformation".

In some situations, such as a for loop, a single-letter variable is appropriate, such as "i".

Commenting

Comments are ignored by the Java compiler, so they are not necessary for actually making code work properly. But good comments are useful for anyone who read your code, including yourself -- if you don't believe me, try reading your code six months from now :)
When adding comments to your code, keep in mind the following:

- Describe the **behavior**, not the **implementation**. In other words, commenting on "what it does", not "how it does it". It's helpful to give your reader a high level overview before the code block (for example, the header before a method); if the reader is interested in exactly how the code works, he/she can read through the code block.
- For a method, describe its parameters and return value.

```
// Searches a list of integers for the maximum between the start index
// (inclusive) and the end index (exclusive).
// Takes a list of integers, a start index, and an end index.
// Returns the maximum found.
public static int findMax(int[] list, int start, int end) { ...
```

- Don't comment the obvious. Let your code speak for itself by using descriptive variable and method names. For example, the comment below should not be there, since the code is self-explaining:

```
// print out the total
System.out.print(total + ", ");
```

- But do point out your intention for any special or subtle cases.

When you are debugging your program, you might have added some temporary code and commented out some other code. Don't forget to clean them up before submit your assignment, since code commented out is not considered good comments.

Decomposition

The main() method should be a concise summary of the program. It should not be a long, "do everything by itself" method full of low-level details.

For other methods, make sure that:

- One method should only do one simple task for the overall program.
- All parameters are necessary and are used within the method.

You should use variables to avoid repeated calculations. Variables should live in the smallest scope possible (declaration close to its usage).