

# CSCE 145, Sections 4, 5, 6

## Study Guide

### Second Midterm

## Chapter 5

### Miscellaneous

This chapter covers the basics of fence post errors, which you should be able to explain.

This chapter also covers the issue of the scope of temporary variables, as in the question, “what’s wrong with the following code?”

```
public class ThisClass extends Object
{
    public ThisClass()
    {
    }

    // either add or subtract, depending on the last parameter
    public int myIncrement(int inValue, int theAddin, boolean addItInQ)
    {
        int thisValue;
        if(addItInQ)
        {
            thisValue += theAddin;
        }
        else
        {
            thisValue -= theAddin;
        }
        return thisValue;
    } // public int MyMethod1(int inValue)

    // always add in the increment value
    public int MyIncrement(int inValue, int theAddin)
    {
        thisValue += theAddin;
        return thisValue;
    } // public void MyMethod2(int inValue2)

} // public class ThisClass extends Object
```

The `switch` statement has not been covered and essentially will not be covered. (You are free to use it, or free to ignore it and just use `if-else if` statements.)

You should know how to nest and cascade `if` statements, and you should know what happens if you dangle the `else` because you didn't use braces (page 250). (I will not, repeat not, do a trick question on this. I do not expect you to be able to read the intricacies of bad code, but I do expect you to be able to write *correct* nested/cascaded `if` statements.)

## **Patterns, pp. 257ff.**

### **Loop and a half**

You should be able to write a loop with an initial segment before the steady state of the loop iteration.

You should be able to write a loop with a final segment after the steady state of the loop iteration.

### **Temporary variable**

You should be able to write a `for` or `while` loop with a counter that is incremented or referenced only when an `if` statement happens to be true.

### **Counting pattern**

You should be able to write a `while` loop with a counter that is always incremented inside each iteration.

### **for loops versus while loops**

You should know when a `for` loop is more appropriate than a `while` loop, and vice versa.

### **Boolean expressions**

You should be able to write a correct boolean expression or series of expressions. Again, I will not play games with asking you for evaluating an expression that has no parentheses. You should be writing your expressions with adequate parentheses to make the order of evaluation crystal clear, and you should recognize the absence of parentheses to be inherently evil.

## **General Comments**

You should be able to write simple methods that do not return a value, such as

```
public void myMethod()  
{  
}  
} // public void myMethod()
```

and you should be able to write simple methods that do return a value, such as

```

public String myOtherMethod()
{
    String returnValue;
    returnValue = "pneumonoultramicroscopicsilicovolcanoconiosis";
    return returnValue;
} // public String myOtherMethod()

```

## Chapter 6

You should be able to write multiple constructors for a class, to know why and when you can do this, and why you would want to do this.

You should know the proper (proper in this case being interpreted under Buell's strict rules) use of **this** and of accessor and mutator methods.

You should be able to explain the use of **final** and you should be able to explain what happens when one uses **static final** as in the two routines in the `Utilities.java` modules in the examples I gave you.

**I have decided that I will give you one page that contains the code for the `ScannerOpen` and the `PrintWriterOpen` methods. That way, I can expect you to be able to use those methods and understand what they do without requiring you to be able to reproduce those methods.**

You should read and thoroughly understand the bottom of page 309 on the use of temporary versus instance variables.

## Arithmetic, Input, and Output

We covered some material on arithmetic and on input and output that is not in Chapters 5 through 8. This material is in fact on the exam. You should be able to write correct code for output using

```
System.out.println(stuff);
```

You should be able to tell me the effect of a statement such as

```
System.out.printf("%d ZZZZ %f YYYY %s\n", i, x, theString);
```

(An acceptable answer is: "This prints the value of `i`, assuming that `i` is an `int` variable, then the string `ZZZZ` with a blank space on each side, then the value of `x` as a floating point number, assuming that `x` is a `float` or `double` variable, then the string `YYYY` with a blank space on either side, and then the string `theString`, followed by a newline so that the next line of output would start on the next line.")

Note: I will not, repeat not, expect you to parrot back the intricacies of output of floating point numbers using something like

```
System.out.printf("%11.4f\n", -1234567.890123);
```

## Chapter 7

As mentioned above, you should know about different numeric data types and about character, `boolean`, and `String` variables. You should know the syntax of the methods on page 350, but you do not need to know the syntax of the million and one other methods for a `String` variable.

Accessors and mutators, yes.

## Chapter 8

If given a piece of code similar to the `Person` class in the text, you should be able to create a simple class like the `DateTime` class so that the original code could be simplified.

You should be able to deal with `try - catch` for exception handling. (And remember, you'll have an example of this in the sample code for the `Utilities.java` class.)

## General Comment

There are a great many things about programming that require remembering specific and detailed rules. This include:

- Exactly what the precedence order is for evaluating boolean or arithmetic expressions without any parentheses.
- Exactly what gets printed in something like a `%12.7f` format if, for some reason, this format isn't going to provide complete accuracy for the numbers being printed.
- Exactly what substring is extracted from a string using one of the particular methods in the `String` class. (This is included here because the method

```
public String substring(int beginIndex, int endIndex)
```

returns a new string that is a substring of this string, beginning at the specified `beginIndex` and extending to the character at index `endIndex - 1`, so the trick is to remember the minus one.

In these and similar contexts, what I expect you to be able to do is the following.

- You should recognize any and all instances of this sort of thing as places where arcane and specific rules are in effect and you need to be especially careful to get it right.
- You should be able to find the reference information that says exactly how to do it right.

- You need not be able to tell me exactly what happens in an instance in which there might be a problem, but you should be able to recognize that a problem might exist.

If you follow this approach, then you will be able to write correct code yourself, and you will be able to recognize instances in other people's code where there might be a problem.

Remember that debugging a program is an empirical process and not a metaphysical process. That is, when a program is not producing the correct output it's because it is doing something different from what you intended. The first and most important debugging step is to find out what in fact the program is doing, not to sit back and contemplate the philosophical or moral nature of the program as written.