CS 163 - Exam 1 Study Guide and Practice Exam

September 15, 2016

Summary

1 Disclaimer

This is a review of the material so far, but there may be material on the exam not covered in this study guide.

2 Variables

2.1 Primitive Types

```
//Declaration
int i;
double d;
char c0, c1, c2;

//Assigning
i = 3;
d = 4.0;
c0 = 'a';
c1 = 'b';
c2 = 'c';

//Initialization
int myInt = 0;
double myDouble = 1.0;
char myChar = 'd';
float myFloat = 2.2F;
long myLong = 1.23L;

//Type casting
double d;
int i = 3;
char c = 'a';
d = (double) i;
System.out.println((int)c);
```

2.2 Printing

```
//Printing with a new line
System.out.println("hi");
System.out.print("hi\n");
System.out.printf("%s\n", "hi");

//Printf formats
int i = 0;
double d = 3.22311090;
System.out.printf("%d\n", i); //prints 0
System.out.printf("%4d\n", i); //prints with 3 spaces then 0
//default 6 decimal points and it rounds! (3.223111)
System.out.printf("%f\n", d);
System.out.printf("%.3f\n", d); //prints 3.223
```

2.3 Reference Types

```
public class Review {
    public static void main (String [] args)}
         //Declaration
         String s;
         //here is an example of creating an object of the class
         Review r = new Review();

         //Assigning
         s = "Hi there";

         //Declaring & Assigning
         String myString = "Okay then";
         String myString1 = new String ("Okay fine then");

         //Common Methods
         String s = "SmIlEy mOnKEy";
         char c0 = s.charAt(0); //assigns S to variable c
         int i0 = s.indexOf('z'); //assigns -1 to variable i0
                                  //because there is no 'z' in s
         String sub = s.substring(0); //assigns s to sub
                                      // returns
                                      // (stringName.substring(#)
                                      //the character from #
                                      //to the end of the string
         String s1 = s.substring(0,1); //only returns S because
             //stringName.substring(inclusive index, exclusive index)
         System.out.println(s.toUpperCase());
         String newString = s.toLowerCase();
         System.out.println(s.equals(newString)); //prints false
         boolean b = s.equals(sub); //assigns true to b
         int length = sub.length(); //assigns 13 to length
                                    //(1 based not 0 based
                                    //(like indexes)
    }
}
```

3 Conditional Statements

3.1 if-else if-else Statements

```
public class Conditionals {
  public static void main (String [] args){
    int i = 0;
    // if statement
    if (i == 0) {
      System.out.println("i == 0");
    }
```

```
    boolean b = true;
    // I don't need brackets because I only have one line of code
    // under each statement. However, you can always have brackts
    // like the example above.
    if (b == false)
      System.out.println("b is FALSE");
    else
      System.out.println("b is TRUE");

    int i1 = 32;
    // here I do need brackets because there is more than one
    // line of code nested in my if statement.
    if (i1 >= 15){
       System.out.println("i1 is greater than or equal to 15");
       i1++;
    }
    else if (i < 3)
       System.out.println("i is less than 3");
    // elses are like catch-alls there are a section of numbers
    // that aren't covered by my if and else if so my else will c
    // handle those values
    else
       System.out.println("i must be between 3 and 14");

    // sometimes you don't need an else
    String name = "Waldo";
    // if I have found Waldo I want to say I found him but...
    if (name.equals("Waldo"))
       System.out.println("Found him!");
    // if I didn't find him I don't want to do anything
  }
}
/* Output:
i == 0
b is TRUE
i1 is greater than or equal to 15
Found him!
*/
```

3.2 Difference between if-if-if and if-else if-else

If you use an if-else if-else statement and multiple conditionals are true, the program executes the first true conditional and then exits.

If you use an if-if-if statement and multiple conditionals are true, the program will execute all of the true conditionals.

```
public class Conditionals {
  public static void main (String [] args){
    int i = 128;
    System.out.println("if-else if-else statment");
    if (i == 128)
      System.out.println("i == 128");
    else if (i > 100)
      System.out.println("i > 100");
    else if (i < 130)
      System.out.println("i < 130");
```

```
    System.out.println("if-if-if statement");

    if (i == 128)
       System.out.println("i == 128");
    if (i > 100)
       System.out.println("i > 100");
    if (i < 130)
       System.out.println("I < 130");
    }
}
/* Output:
if-else if-else statment
i == 128
if-if-if statement
i == 128
i > 100
I < 130
*/
```

3.3 Switch Statements

```
public class Conditionals {
  public static void main (String [] args) {
    int i = 32;
    switch (i) {
    case 10: System.out.println("i = 10"); break;
    case 32: System.out.println("i = 32"); break;
    case 101: System.out.println("i = 101"); break;
    default: System.out.println("i is not 10, 32, or 101");
    }

    // beware of your breaks!
    char c = 'b';
    switch(c) {
    case 'a': System.out.println("c = a");
    case 'b': System.out.println("c = b");
    case 'c': System.out.println("c = c");
    default: System.out.println("c is not a, b, or c");
    }

    // times you might want to not have breaks
    int dogs = 0, cats = 0;
    String breed = "German Shepard";

    // spacing doesn't matter
    // you aren't required to have a default
    //    like you aren't required to have an else
    //    just be aware that there are possible cases
    //    that aren't being handled.
    switch (breed) {
    case "Boxer":
    case "Great Dane":
    case "German Shepard":
    case "Lab":
```

```
            dogs++; break;

        case "Calico": case "British Shorthair": case "Siamese":
          cats++; break;
        }
        System.out.printf("Dogs: %d, Cats %d\n", dogs, cats);
    }
}
/* Output:
i = 32
c = b
c = c
c is not a, b, or c
Dogs: 1, Cats 0
*/
```

3.4 Hints and Warnings

- Warning: Don't add a semicolon after your conditional. For example `if (32 == 32);`.
- Warning: Be aware of when you should and shouldn't have breaks in your switch statement.

4 Loops

4.1 For loops

For loops are used when you know how many iterations the loop needs to complete.
General syntax:

```
for (int i = 0; i < someLength; i++){
    //some code
}
```

You can also think of for loops as such:

```
for (initialization; range; update) {
    //some code
}
```

"Real life example": A loop used print every second letter backwards (z, x, v, t...).

```
for (char c = 'z'; c >= 'a'; c-=2){
    System.out.print(c);
}
```

4.2 While loops

While loops are generally used when you don't know how many iterations you need the loop to complete (i.e. if you are reading entered words from the keyboard until the user types "stop").
General syntax:

```
int i = 0;
while (i < someLength){
    //some code
    i++;
}
```

You can also think of for loops as such:

```
initialization; //outside of loop
while (range){
    //some code
    update
}
```

Same "Real life example" as above.

```
char c = 'z';
while (c >= 'a'){
    System.out.print(c);
    c-=2;
}
```

4.3 Do-While Loops

Do-While loops are used when you want the loop to run at least once, regardless of whether the value is in the range. General syntax:

```
int i = 0;
do {
    //code
    i++;
} while (range);
```

Same "Real life example" as from before.

```
char c = 'z';
do {
    System.out.print(c);
    c-=2;
} while (c >= 'a');
```

4.4 Hints and Warnings

- Warning: Watch the ranges! When a String's length is of size 8, i $<$ 9 is the same as i $\leq$ 8. However, i $\leq$ 9 will give you an error.
- Hint: You can always change your update by using the "+=" or "-=" method, especially if you are updating by more than 1.
- Hint: You can always change your starting point, you don't have to start at 0, it is just the most common. If you were asked to reverse a string, you can always make the starting point at stringName.length-1! (Remember the -1 though, lengths are 1 based while loops are 0 based usually).
- Summary: Basically you can solve a problem with loops in many different ways whether it's using a different loop, starting at a different place, changing your ending value, or changing how you update.

5 Practice Written Exam

5.1 Short Answer

1. Declare an integer name `i0`.
2. Initialize a float named `f` to 4.0.
3. Initialize a Scanner to read from the keyboard, called `read`.
4. Assign `i0` to an integer entered from the user.
5. Close the Scanner.
6. Change the pre-assigned String variable called `s` to be all upper case.
7. Name the eight primitive types.
8. Given that `String s = new String ("Bob Marley"); String m = "mummies";)` find the following (write "error" if necessary, assume these are printed using println):

    1. m.substring(0,3)
    2. s.substring(4)
    3. s.substring(0,11)
    4. m.charAt(1)
    5. s.charAt(1)
    6. s.charAt(11)
    7. s.indexOf('q')
    8. m.indexOf(3)
    9. s.length()
    10. s + m
    11. m.indexOf('m')
    12. s.length() + m.length()

9. What is 18 % 3?
10. Write a for loop, while loop, and do-while loop that prints odd numbers from 1 to 101 all on one line.
11. Write a switch statement based off of the character variable c. If c is 'a' print "hi", if c is 'b' print "okay", if c is 'c' print "then", if c is 'd' print "bye", if it is not 'a', 'b', 'c', or 'd' print" fine then".
12. Write an if statement that does the same thing as question 12.
13. Declare a Scanner called `scan`. Read in one word and store it into the predefined String variable called `word`. Print out the second through fourth character of `word`. Close your scanner.
14. Check to see if the predefined String variables `s0` and `s1` are equal. Print the result.
15. Check to see if the predefined int variables `i0` and `i1` are equal. Print the result.

5.2 Tracing Code

```
/* Instructions:
 * By each commented number write down what the console would print.
 * There are some questions that return errors. Either write "error" or you
    can write the exception name.
 * If you find an infinite loop. Continue through the questions(even though
    the rest of the program would never print.
 * Write "infinite loop" for those cases for the questions.
 * */
import java.util.Scanner;
```

```
public class Trace {
    public static void main (String [] args){
        char c = 'b';
        String s0 = "Hello World";
        //Question 1: What does this print?
        switch (c){
        case 'a':
            System.out.println("c = a");
        case 'b':
            System.out.println("c = b");
        case 'c':
            System.out.println("c = c");
        case 'd':
            System.out.println("c = d");
        default:
            System.out.println("c is not a, b, c, or d");
        }
        //Question 2: What is printed?
        s0.toLowerCase();
        if (s0.equals("hello world"))
            System.out.println("hiya Mars");
        else
            System.out.println("s0 doesn't say hello world...");
        //Question 3: What does the console print EXACTLY
        for (int i = 0; i < 30; i++){
            if (i % 3 == 0){
                System.out.print(i + ", ");
            }
        }
        System.out.println(); // used for spacing

        int userNum = 3;
        //Question 4:
        do {
            System.out.println("The userNum value is: " + userNum);
        } while (userNum > 5);
        //Question 5:
        for (int i = 0; i < s0.length(); i++);
            System.out.print(s0.charAt(i) + ":");
        System.out.println(); //used for spacing
        //Question 6:
        System.out.println(s0.indexOf('u'));


        //Question 7:
        for (int i = 1; i > 0; i++)
                if (i % 2 == 1)
                        System.out.print(i + ";");
        System.out.println(); // used for spacing
    }
}
```

6 Programming Quiz Practice Exam

Re-do past recitations and assignments without using the internet, past code, friends/family, etc. Treat them like a quiz.

7 Suggestions for Studying and Test Taking

7.1 Written

There are times when $>$ and $\geq$ cause the same result (for example: int age = 8; if (age $>$ 7) and if (age $\geq$8)).

When reading through code and writing the output: Write your variables on the side and as your variables change, you change your list on the side.

Write down for, while, do-while loops/templates. Know what each piece does, that way when you come across a problem that needs a loop then you can quickly change a piece (initialization/starting point, range, or the update) of the loop to solve your problem.

Practice writing code in Eclipse and before you run your program guess what the output would be. This is good practice for testing your own programs and also for the code tracing part of the exam.

If you need more tracing examples (or more coding examples in general), there is a "Programs" tab on the CS163/164 homepage. There are also examples on the Progress page.

7.2 Programming Quiz

Go back to past recitations and assignments and redo them without using the internet, friends, or past recitations/assignments.

Practice writing code in Eclipse. Make up projects and problems or ask a TA and they can give you some challenges.

Look at code, the more exposure you get to code (whether it's your own code or not) the easier it is to understand. Some sample code is under the "Programs" tab and the Progress Page.

7.3 Common Errors

- Loop ranges either go too far or too little
- Scanner token processing(.next, .nextInt(), .nextDouble()) to line processing (.nextLine()). Need an extra .nextLine() when going from token processing to line processing.
- Use .equals when comparing Strings, NOT == !
- Incorrect brackets around loops and conditional statements

7.4 Challenges

CodingBat.com and Hackerrank.com offer good extra coding practice.