COE 212 – Engineering Programming

Welcome to Exam II
Tuesday July 10, 2018

Instructor: Dr. Wissam F. Fawaz

Name: _____

Student ID: _____

Instructions:

1. This exam is **Closed Book**. Please do not forget to write your name and ID on the first page.
2. You have exactly **85 minutes** to complete the **4** required problems.
3. Read each problem carefully. If something appears ambiguous, please write your assumptions.
4. Do not get bogged-down on any one problem, you will have to work fast to complete this exam.
5. Put your answers in the space provided only. No other spaces will be graded or even looked at.

Good Luck!!

Problem 1: Multiple choice questions (**20 minutes)** [20 points]

For the questions given below, consider the following Java class, which is stored in a Java file called `ProblemI.java`:

```
import java.util.*;
public class ProblemI {
      public static void main(String[] args) {
            String str1 = "www.wissamfawaz.com/index.htm";
            String str2 =
      "www.wissamfawaz.com/engineer_programming/index.htm";
            String str="";
            int val1=0, val2=0, val3=0;
            char current= ' ';
            Scanner scan = new Scanner(str1);
            scan.useDelimiter("/");
            StringTokenizer st=new StringTokenizer(str2,".");
            while(scan.hasNext()) {
                  scan.next();
                  val1++;
            }
            System.out.println(val1);
            while(st.hasMoreTokens()) {
                  if(val2 == 2)
                        str = st.nextToken();
                  else
                        st.nextToken();
                  val2++;
            }
            System.out.println(val2);
            System.out.println(str.length());
            str1 = str1.substring(0, str1.indexOf("/"));
            int i;
            for(i=0; i<str1.length(); i++) {
                  current = str1.charAt(i);
                  if(current == 'a')
                        val3++;
            }
            System.out.println(val3);
      }
}
```

1) How many repetition statements does the `main` method contain?
   a. 4
   b. **3**
   c. 2
   d. None of the above
2) How many conditional statements does the `main` method contain?
   a. 4
   b. **3**
   c. **2**
   d. None of the above
3) Excluding the `args` parameter, how many local variables does the `main` method have?
   a. **10**
   b. 11
   c. 12

d. None of the above

4) How many iterations does the first repetition statement perform, namely, the one whose termination condition is based on `scan.hasNext()`?
   a. `1`
   b. **`2`**
   c. `3`
   d. None of the above
5) How many iterations does the `for` repetition statement, present inside the `main` method, perform?
   a. `17`
   b. 18
   c. **`19`**
   d. None of the above
6) What output is produced by the first `println` statement of the `main` method?
   a. **`2`**
   b. `3`
   c. `4`
   d. None of the above
7) What output is produced by the second `println` statement of the `main` method?
   a. `2`
   b. `3`
   c. **`4`**
   d. None of the above
8) What output is produced by the third `println` statement of the `main` method?
   a. `20`
   b. `21`
   c. `22`
   d. **None of the above**
9) What output is produced by the fourth `println` statement of the `main` method?
   a. `2`
   b. **`3`**
   c. `4`
   d. None of the above
10) If the following `println` statement is placed immediately after the body of the for repetition statement (outside the body of the loop): `System.out.println("i: " + i);`
    What output does that statement produce?
    a. **`i: 19`**
    b. `i: 20`
    c. `i: 21`
    d. None of the above
11) If the following `println` statement is placed immediately after the body of the for repetition statement (outside the body of the loop): `System.out.println(current);`
    What output does that statement produce?
    a. **`m`**
    b. `z`
    c. `a`
    d. None of the above
12) Which of the following classes used inside the `main` method is not an iterator?
    a. `Scanner`
    b. **`StringTokenizer`**
    c. Both (a) and (b) are iterators
    d. Neither (a) nor (b) is an iterator
13) Which of the following must be added at the end of the header of a `main` method that reads data from a file?
    a. `throw IOException`
    b. **`throws IOException`**
    c. `throw OIException`
    d. None of the above

Problem 2: True or false questions (**10 minutes)** [20 points]

1. Assuming that `x` and `y` are `boolean` variables, the following code fragment:

```
if(x || y)
        System.out.print("Yes");
else
        System.out.print("No");
```

can be rewritten as:

```
if(!x && !y)
        System.out.print("No");
else
        System.out.print("Yes");
```

Answer: **True** False

2. The following code correctly prints the `String` having the maximum length between the two `String` variables called `str1` and `str2`. Assume that `str1` and `str2` were declared and instantiated correctly.

```
int flag = ((str1.length()-str2.length())<0)?1:-1;
switch(flag) {
case 1 :
        System.out.print(str1) ;
        break;
case -1:
        System.out.print(str2);
}
```

Answer: True **False**

3. In the following code fragment, the `do…while` repetition statement loops until the user enters a strictly negative value (i.e., strictly less than zero) for the `val` variable.

```
Scanner scan = new Scanner(System.in);
int val;
do { val = scan.nextInt(); } while(val >= 0);
```

Answer: **True** False

4. The following code fragment correctly prints the product of all the multiples of `4` that are less than or equal to `28`.

```
int product = 1;
for(int val = 4; val<28; val+=4) {
        product*=val;
}
System.out.print(product);
```

Answer: True **False**

5. The body of the following repetition statement executes forever:

```
boolean a = true, b = false, c = true;
boolean flag= a && b || !c;
do {
        System.out.println("hi");
} while(flag);
```

Answer: True **False**

6. Consider the following nested loops. The body of the inner loop executes a total of `88` times:

```
int count1=1, count2=1;
while(count1<=11) {
        while(count2<=8) {
                System.out.println("Bravo");
                count2++;
        }
        count1++;
}
```

Answer: True **False**

7. The following code fragment prints out to the screen: `un`

```
String str = "Exam is fun";
while(str.length()>=2) {
       str = str.substring(1);
}
System.out.print(str);
```

Answer: True **False**

8. The following code fragment correctly prints the number of non-zero even digits found in the `int` variable called `val`. For example, if `val` is equal to `2467`, then the code fragment prints a value of `3` since there are `3` non-zero even digits in `val`, namely `2`, `4`, and `6`. Assume that `val` is an `int` variable that was declared and initialized properly.

```
int counter = 0, lastDigit;
while(val != 0) {
       lastDigit = val % 10;
       if(lastDigit==2 || lastDigit==4 || lastDigit==6 || lastDigit==8)
              counter++;
       val/=10;
}
System.out.print(counter);
```

Answer: **True** False

9. The following code fragment prints `true` if the `int` variable called `val` is prime and prints `false` otherwise. Recall that a prime number is a number that has only two divisors, namely `1` and itself.

```
Scanner scan = new Scanner(System.in);
int val;
do { val = scan.nextInt(); } while(val >= 9);
if(val == 2 || val == 3 || val == 5 || val == 7)
       System.out.print(true);
else
       System.out.print(false);
```

Answer: **True** False

10. The following code always prints: `Tails`

```
int x = (int) Math.random() * 2;
switch(x) {
case 0: System.out.print("Tails");
case 1: System.out.print("Heads");}
```

Answer: True **False**

Problem 3: Evaluating Java Expressions (**15 minutes**) [20 points]

For each of the following code fragments, what is the value of **x** after the statements are executed?

(1)
```
String str = "C2!O1e2*";
String x = "";
char c;
for(int i=0; i<str.length(); i++) {
      c = str.charAt(i);
      if(c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z')
            x += c;
}
```
Answer: x= Coe

(2)
```
int x = 0, y = 2377, z=0, w=0;
do {
  w = y%10;
  if(w%2 != 0) {
      x+=w;
      z++;
  }
  y=y/10;
} while(y != 0);
if(z != 0)
  x = x/z;
```
Answer: x= 5

(3)
```
String S1 = "CIE";
String S2 = "CIS";
int x = 0;
do {
  if(S1.charAt(x) != S2.charAt(x))
      break;
  x++;
} while(x < S1.length());
```
Answer: x= 2

(4)
```
String x="";
for(int i= 1;i <= 3; i++) {
  if(i%3==0)
  x+=2*i;
  else if(i%2!=0)
  x+=3/i;
  else
  x+=i;}
```
Answer: x= "326"

(5)
```
String S1 = "Banana";
String S2 = "bananaSplit";
boolean x=(S1.compareTo(S2.substring(0, S2.indexOf('S')))>0);
```
Answer: x= false

(6)
```
String S = "It is not over until the fat lady sings";
int y=0;
String x;
for(int i=0; i < S.indexOf('f'); i++)
if(S.charAt(i) == 'h') y = i;
x = S.substring(y);
```
Answer: x= "he fat lady sings"

(7)
```
String S = "2 - 0 better than 1 - 0";
String x = "";
Scanner scan = new Scanner(S);
x+=scan.nextInt();
scan.next();
x+=scan.nextInt();
scan.next();
scan.next();
x+=scan.nextInt();
```
Answer: x= "201"

(8)
```
int x = 0;
for(int i=1; i < 10; i+=3)
for(int j=1; j<i; j+=2)
x++;
```
Answer: x= 5

(9)
```
final int UPPER=2;
int x = 0, y=2, z=0;
while(z<=UPPER) {
      x+=y;
      z+=2;
      y*=2;
}
```
Answer: x= 6

(10)
```
DecimalFormat fmt= new DecimalFormat("000.##");
double y = 1.4567;
String x = fmt.format(y).substring(2);
```
Answer: x= "1.46"

Problem 4: Coding Problems (**40 minutes**) [40 points]

1. Write a program called `OddDigits`, which reads from the user an integer `n`, and then prints out to the screen the number of odd digits that are present in `n` as well as the average of these digits. For example, in the sample run given below, the user entered a value of `4378` for `n`. This value contains `2` odd digits, namely the digits `3` and `7`. Therefore, the average of the odd digits in `n` is found to be (7+3)/2 = 5.0.

Sample run:
Enter n: 4378
Nb. of prime digits found in 4378 is: 2
Average of the prime digits found in 4378: 5.0

```
import java.util.Scanner;
public class OddDigits {
      public static void main(String[] args) {
            int n, lastDigit;
            int count=0;
            double avg=0.0;
            Scanner scan = new Scanner(System.in);
            System.out.println("Enter n:");
            n = scan.nextInt();
            do {
                  lastDigit = n%10;
                  if(lastDigit%2!=0) {
                        count++;
                        avg+=lastDigit;
                  }
                  n = n/10;
            } while(n!=0);
            if(count != 0) {
                  avg = avg/count;
                  System.out.println("Nb of odd digits:"+count);
                  System.out.println("Avg of odd digits:"+avg);
            } else {
                  System.out.println("No prime digits were found");
            }
      }
}
```

2. Write a program called `StringModification` that reads a sentence from the user called `str` and then creates a modified `String` called `strModified` based on `str` as follows. `strModified` should be identical to `str` except that every occurrence of the white space character and the dash character '`-`' in `str` must be removed in `strModified`, as illustrated in the sample output below.

Sample run:

Enter a sentence: The exam is fun-and-long
Modified sentence: Theexamisfunandlong

```
import java.util.Scanner;
public class StringModification {
      public static void main(String[] args) {
            String str, strModified= “”;
            char current;
            System.out.println(“Enter a sentence:”);
            str = scan.nextLine();
            for(int i=0; i<str.length(); i++) {
                  if(current != ‘ ’ && current != ‘-’)
                        strModified += current;
            }
            System.out.println(“Modified sentence:”+strModified);
      }
}
```

3. Write a program called `StringCreation` that reads from the user 2 `String` values called `S1` and `S2`. Your program should then form and print out a new `String` called `S` that is composed of the first half of `S2`, followed by the second half of `S1`, then the second half of `S2`, which in turn is followed by the first half of `S1`. Your program should force the user to enter `S1` and `S2` strings that have an even number of characters.

Sample run:
Enter S1: rice
Enter S2: salt
S: saceltri

```
import java.util.Scanner;
public class StringCreation {
      public static void main(String[] args) {
            String S1, S2;
            String S;
            Scanner scan = new Scanner(System.in);
            do {
                  System.out.println("Enter S1:");
                  S1 = scan.nextLine();
            } while(S1.length()%2!=0);
            do {
                  System.out.println("Enter S2:");
                  S2 = scan.nextLine();
            } while(S2.length()%2!=0);
            S =
            S2.substring(0,S2.length()/2)
            +S1.substring(S1.length()/2)
            +S2.substring(S2.length()/2)
            + S1.substring(0,S1.length()/2);
            System.out.println("S:" + S);
      }
}
```

4. Write a Java program that extracts twitter user names from a text file. The program reads the text tokens (which we suppose are separated by single spaces) from a source file called `source.txt`, identifies the twitter user names found in the input file, numbers them, and then stores them in an output file called `users.txt`, according to their order of appearance in the input file. Note that a twitter user name is a token that begins with the "@" symbol. Note also that the "@" symbol must be removed from the twitter user name before being stored in the output file.
   For instance given the following content for `source.txt`:
   ```
   Hello, my name is John, my twitter user name is: @John I live in Byblos.
   I use @William instead of @John to make my purchases online. My friend David uses @David for identification.
   ```
   The resulting output file `users.txt` would contain:
   ```
   0 - John
   1 - William
   2 - John
   3 - David
   ```

```
import java.util.Scanner;
import java.io.*;
public class TwitterUserNames {
      public static void main(String[] args) throws IOException {
            Scanner inFromFile =
            new Scanner(new File("source.txt"));
            FileWriter fw = new FileWriter("users.txt");
            BufferedWriter bw = new BufferedWriter(fw);
            PrintWriter outToFile = new PrintWriter(bw);
            int count = 0;
            String word;
            while(inFromFile.hasNext()) {
                  word = inFromFile.next();
                  if(word.charAt(0) == '@')
                        outToFile.println(counter+":"+word);
                  counter++;
            }
            outToFile.close();
      }
}
```