# COE 212 – Engineering Programming

Welcome to Exam II
Monday December 16, 2013

Instructors: Dr. Randa Zakhour
Dr. Joe Tekli
Dr. George Sakr
Dr. Wissam F. Fawaz

**Name:** ____________________

**Student ID:** ________________

**<u>Instructions:</u>**

1. This exam is **Closed Book**. Please do not forget to write your name and ID on the first page.
2. You have exactly **120 minutes** to complete the 5 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!!**

## **<u>Problem 1:</u>** Multiple choice questions (**15 minutes)** [10 points]

For each question, choose the **single** correct answer.

1) Which of the following statements is equivalent to the following code fragment:
```
if (value > MAX)
sum += MAX;
else
sum += value;
```
   a. `sum += (value > MAX) value ? MAX;`
   b. `sum = (value > MAX) MAX ? value;`
   c. `sum += (value <= MAX) value ? MAX;`
   **d. None of the above**

2) The output of the below piece of code is:
```
for(int i=1; i<=5; i++) {
     i++; System.out.print(“X”);
}
System.out.println(“X”);
```
   a. XXXXX
   **b. XXXX**
   c. XXX
   d. None of the above

3) Consider the program segment given below. Its output is:
```
String theWord = “test string”, output = “”;
for(int i=1; i < theWord.length(); i+=2)
     output += theWord.charAt(i);
System.out.print(output);
```
   a. ts tig
   b. s tig
   **c. etsrn**
   d. None of the above

4) What is the output of the following code fragment?
```
int count=20; boolean done=false; double sum=123.5;
if((done && count>0) || !done || sum < 0)
if(sum > 0)
System.out.println(“Sum is positive”);
else
System.out.println(“Sum is not positive”);
else
done = !done;
System.out.print((done==false)?“Done: ”+sum:“Not Done: ”+sum);
```
   **a. Sum is positive**
      **Done: 123.5**
   b. Sum is negative
      Not Done: 123.5
   c. Sum is positive
   d. None of the above

5) After the execution of the following piece of code the value of `count` will be:

```
int count = 0;
for(int i=0; i < 3; i++)
     for(int j=0; j<i; j++) count++;
```

   a. 1
   **b. 3**
   c. 6
   d. None of the above

6) Given: `String word1="FOREVER", word2="4EVER", word3="For";` which of the following Boolean expressions evaluates to true?
   a. `(word1.compareTo(word2)>0) && (word1.compareTo(word3)<0)`
   b. `(word2.compareTo(word3)< 0) && (word3.compareTo(word1)>0)`
   c. **All of the above**
   d. None of the above

7) Which of the following expressions checks to see whether the first 4 characters in the `String` variable called `str` form the word "here"?
   a. `str.substring(0, 3).equals("here")`
   b. **`str.substring(0, 4).compareTo("here") == 0`**
   c. `str.substring(0, 5).equalsIgnoreCase("Here")`
   d. Both (b) and (c)

8) Consider the following `switch` statement. What is the value of the `sum` variable after it is executed?

```
char choice = 'C'; int sum = 5;
switch(choice) {
case 'B':
     sum += 20;
case 'D':
     sum += 30;
case 'C':
     sum += 40;
default:
     sum += 25;
}
```

   a. 35
   b. 100
   c. 75
   **d. None of the above**

9) Which of the following represents the header of a constructor of the `FileWriter` class?
   a. `FileWriter(BufferedWriter)`
   b. `FileWriter(PrintWriter)`
   c. **`FileWriter(String)`**
   d. `FileWriter(Scanner)`

10) Which of the following methods can be used to change the delimiter of the `Scanner` class?
   a. `setDelimiter`
   b. `changeDelimiter`
   **c. `useDelimiter`**
   d. `modifyDelimiter`

## **Problem 2:** True or false questions (**15 minutes)** [12 points]

1. The following statement:
   ```
   boolean result = (x % 4 == 0 && x % 100 != 0) || (x % 400 == 0);
   ```
   will be `false` if the `int` variable x is equal to 2000.

Answer: True **False**

2. The output of the for loop given below is:
   ```
   *
   **
   ***
   ****
   *****
   ******
   for (int i = 0; i <= 5; i++) {
         for (int j = 0; j<= i; j++)
         System.out.print("*");
         System.out.println();
   }
   ```

Answer: **True** False

3. `continue` cannot be used in a `switch` selection statement

Answer: **True** False

4. The boolean expression `(a && b) || !(c && !d)` is `false` if and only if the `boolean` variables `a`, `b`, `c` and `d` are equal to `false`, `false`, `true` and `false`, respectively.

Answer: True **False**

5. Consider the code fragment given in the box below. Its output is: `tiiit`

```
String s = new String("this is it");
int y = 0; char letter;
do {
     letter = s.charAt(y);
     if(letter == 'i' && letter == 't')
          System.out.print(letter);
     y++;
} while(y < s.length());
```

Answer: True **False**

6. The cases of a `switch` statement are not mutually exclusive by default, in the sense that without a `break` statement more than one `switch` case can be executed at a time.

Answer: **True** False

7. The following code in Java finds the largest integer value in the `String` variable called `str` and stores it in the variable called `x`.

```
String str = “3 4 9 2 8”;
int x = -1, num;
Scanner strScan = new Scanner(str);
while(strScan.hasNext()) {
     num=Integer.parseInt(strScan.next());
     if(num > x)
          x = num;
}
```

Answer: **True** False

8. Not including a “`throws IOException`” as part of the header of a `main` method that reads data from a file results in a run-time error.

Answer: True **False**

9. The output of the below piece of code is: `multiple three two`

```
int num = 4;
while(num < 12) {
     switch(num % 4) {
     case 1:
          System.out.print(“one ”); break;
     case 2:
          System.out.print(“two ”); break;
     case 3:
          System.out.print(“three ”); break;
     case 0:
          System.out.print(“multiple ”);
     }
     num += 3;
}
```

Answer: **True** False

10. The output of the program segment given below is: `true`

```
String str = “hi56”;
char digit = str.charAt(str.length()-1);
if(digit >= ‘0’ || digit <= ‘9’)
     System.out.print(digit == ‘5’);
```

Answer: True **False**

11. The `default` case is optional in a `switch` selection statement.

Answer: **True** False

12. Methods with private visibility, also known as support (service) methods, can be directly accessed by client programs.

Answer: True **False**

## <u>**Problem 3:**</u> Code analysis (**15 minutes**) [8 points]

1) Consider the following method.

```
 public void stringManipulations(String s1, String s2) {
      int v = Integer.parseInt(s2);
      for (int i = 0; i < s1.length(); i++){
            String subs1 = s1.substring(0, i + 1);
            String subs2 = s1.substring(i +1);
            if (i == s1.length() - 1)
                  subs2 = "0";
            int v1 = Integer.parseInt(subs1);
            int v2 = Integer.parseInt(subs2);
            if (v1 + v2 == v){
                 System.out.println(subs1 + " + " + subs2 + " = " + s2);
            }
      }
}
```

Which of the following invocations of this method **prints an output message to the screen**?

a. `stringManipulations("56111","582");`
**b.** **`stringManipulations("21561","582");`**
**c.** **`stringManipulations("56121","582");`**
**d.** **`stringManipulations("56220","582");`**

2) Consider the class given below, along with the driver class following it. Assume the driver class is located in the same folder.

```
public class ClassA {
      private String val;
      private boolean status;
      public ClassA(String v) {val = v; status = false;}
      public void setStatus(){
             String str = "";
             for(int i=1; i<=str.length(); i++)
                   str+=val.charAt(str.length() - i);
           if(val.equals(str))status = true;
      }
      public boolean getStatus() { return status;}
}
```

```
public class ClassADriver {
      public static void main(String[] args) {
        ClassA a=new ClassA("Kayak");
        a.setStatus();
        System.out.println("Status: " + a.getStatus());
      }
}
```

When running the `ClassADriver` class, what output is produced?

a. `Status: true`
**b.** **`Status: false`**
c. It doesn't compile correctly
d. None of the above

## **Problem 4:** Method definition (**15 minutes)** [9 points]

You are given below information about 3 methods called **`sumOfMultiples`**, **`findOccurrences`** and **`isMoreFrequent`**. Your job is to complete the definition of each one of these methods as per the provided guidelines.

1. **`sumOfMultiples()`**is a method that accepts two `int` parameters, `a` and `c`, and returns the sum of the multiples of `c` up to `a`.

```
      int sumOfMultipe (int a, int c)
       {
             int sum = 0;
             while (sum < a)
             {
                sum += c;
             }
             return c;
       }
```

2. **`findOccurrences()`** is a method that accepts a `String` object and a `char` as parameters and returns the number of times the `char` occurs in the `String`.

```
int findOccurrences(String S, char c)
{
       int count = 0;
       for(int i=0; i< S.length(); i++)
       {
              if (S.charAt(i) == c)  count++;
       }
       return(count);
}
```

3. **`isMoreFrequent()`** is a method that accepts a `String` parameter along with two `char` parameters and returns a `boolean` indicating whether or not the first `char` parameter occurs strictly more often than the second `char` parameter in the String. You should use the **`findOccurrences()`** method defined earlier. For example, if the String is “The rain in Spain falls mainly on the plain”, and the first char is ‘a’ and the second is ‘r’, the method will return true since ‘a’ occurs 5 times and ‘r’ only once in that String.

```
 boolean isMoreFrequent(String S, char c1, char c2)
 {
    boolean result = false;
    if (findOccurrences(S,c1) > findOccurrences(S,c2))
          result= true;
    return result;
}
```

## **Problem 5:** Debugging Problem (**10 minutes)** [9 points]

Correct each of the following code segments to match the associated description. Make the fewest changes possible (e.g. do not change the type of loop used).

a) Output the odd integers from 55 down to 1:

```
for (i = 55; i >= 1; i++)
      System.out.println(i);
```

```
      for (int i = 55; i >= 1; i=i-2)
         System.out.println(i);
```

b) Output the even integers from 2 to 1000 printing 5 values per line:

```
counter  = 2;
i = 0;
do {
      System.out.print (counter + " ");
      counter += 2;
      if (i %= 5)
            System.out.println();

} while (counter < 1000);
```

```
int counter  = 2;
int i = 0;
do {
       System.out.print (counter + " ");
       counter += 2;
       i++;
       if (i%5 == 0)
       {
           System.out.println();
           i=0;
       }
 } while (counter <= 1000);
```

c) Output whether the integer variable `value` is odd or even:

```
switch(value % 2) {
      case 0:
            System.out.println(" Even integer ");
      case 1:
            System.out.println(" Odd integer ");
}
```

```
     switch(value % 2) {
                  case 0:
                        System.out.println(" Even integer ");
                        break;
                  case 1:
                        System.out.println(" Odd integer ");
      }
```

## **Problem 5:** Coding (**50 minutes)** [52 points]

1. Write a program named `Divisors` that reads a positive integer `val` from the user and prints out all of its divisors including 1 but excluding `val`.

**Sample output**

**Enter a number: 8**

**Divisors: 1  2 4**

```
import java.util.*;

public class Divisors
{

    public    static    void    main (String[] args)
      {
            Scanner scan = new Scanner(System.in);
            int val;

            do{
               System.out.println("Enter a number");
                   val = scan.nextInt();
            } while (val<0);

            System.out.print(“Divisors: ”);
            for (int i=1; i<val; i++)
            {
                if(val%i == 0)  System.out.print(i + " ");
            }
    }
}
```

2. Design and implement a Java program called `GradeReport` that reads from the user a sequence of grades, as integer numbers comprised between [0, 100] each, until the user types -1 (sentinel value). After reading -1, the program will print to the screen some grade-related statistics as follows:
   - The number of input grades strictly below 60, the number of input grades between 60 (inclusive) and 80 (inclusive), and the number of input grades strictly above 80.
   - The maximum and minimum grades entered by the user
   - The average of all of the obtained grades

**<u>Sample output</u>**

```
Enter a grade between [0, 100]: 80
Enter a grade between [0, 100]: 60
Enter a grade between [0, 100]: -1
Grade stats:
- Number of grades below 60: 0
- Number of grades between [60, 80]: 2
- Number of grades above 80: 0
- Minimum grade: 60
- Maximum grade 80
- Average of all grades: 70.0
```

```
import java.util.Scanner;

public class GradeReport {
     public static void main (String[] args)    {
       int grade, Min=100, Max=0, count=0, clower=0, cupper=0, Sum = 0;
       final int Lower = 60;
       final int upper = 80;

       Scanner scan = new Scanner(System.in);

       do{

           do {
               System.out.println("Enter a grade between [0, 100]: ");
               grade = scan.nextInt();
              } while (grade < -1 || grade >100);

           if(grade != -1)
           {
             count++;
             Sum += grade;

             if(grade < Lower) clower++;
             else if (grade > upper) cupper++;

             if (grade < Min) Min = grade;
             else if (grade > Max) Max = grade;
           }
      } while (grade != -1);
```

```
  System.out.println("Grade stats:");
  System.out.println(" - Number of grades below 60: " + clower);
  System.out.println(" - Number of grades between [60, 80]: " +
                    (count – cupper - clower));
  System.out.println(" - Number of grades above 80: " + cupper);
  System.out.println(" - Minimum grade: " + Min);
  System.out.println(" - Maximum grade: " + Max);
  System.out.println(" - Average of all grades: " +
                       (double)Sum/count);

  }
}
```

3. Write a program named `OddCount` that reads an integer value from the user and prints out the number of odd digits that it contains.

**Sample output**

**Enter a number: 1234**

**Number of odd digits in 1234: 2 digits**

```
import java.util.*;

public class OddCount
{
    public    static    void    main (String[] args)
   {

      int oddCount = 0;
      int value, digit;

      Scanner scan = new Scanner(System.in);

      System.out.print ("Enter a number: ");
      value = scan.nextInt();

      value = Math.abs (value);

      while (value > 0)
      {
         digit = value % 10;
         if (digit != 0 && digit%2 !=0) oddCount++;
         value = value / 10;
      }
      System.out.println ("Number of odd digits in " + value +”: ” +
oddCount + “ digits”);
    }
}
```

4. A store owner keeps a text file called "store.txt", containing information about the different products in his shop. Each line in the file corresponds to one product and consists of the name of the product, its unit price (in dollars) and the quantity sold, *separated by commas*. For example, the following line:

   **Product2, 1.4, 567**

   indicates that each item of Product2 costs $1.4 and that 567 Product2 items were sold. Write a Java program called `TotalSales` that goes through the file to determine the total number of items sold and then print out the total sales figure (i.e., amount of money cashed), formatted to 2 decimal places.

```
import java.util.*;
import java.io.*;
import java.text.*;

public class TotalSales{
    public static void main (String[] args) throws IOException {

      String product;
      Scanner fileScan, productScan;
      int TotalNbItems = 0;
      double TotalSales = 0;

      fileScan = new Scanner (new File("store.txt"));

      // Read and process each line of the file
      while (fileScan.hasNext())
      {
         product = fileScan.nextLine();
         productScan = new Scanner (product);
         productScan.useDelimiter(",");

         //  Process each product line
         int i=0;
         double price = 0;
         int NbUnits = 0;

         while (productScan.hasNext())
         {
            switch(i)
            {
               case 0: productScan.next();  i++;
                      break;
               case 1: price = productScan.nextDouble(); i++;
                       break;
               case 2: NbUnits = productScan.nextInt(); i++;
                       break;
               default: break;
             }
          }
       TotalSales += price * NbUnits;
       TotalNbItems += NbUnits;

      }
```

```
       DecimalFormat fmt = new DecimalFormat("0.00");
       System.out.println("Total nb of items sold = " +
                            TotalNbItems);
         System.out.println("Total sales figure = " +
                             fmt.format(TotalSales));
     }
}
```

5. Write a Java program called `Powers` that reads two positive integer values from the user, say `a` and `b`. The program should then print out the powers of all the non-zero positive integers that are less than or equal to `a`, from the `0`th to the `b`th power (see the sample run below for the formatting required). For instance, if `a=2` and `b=2`, then the program should print two sets of powers for the values of `1` and `2` with the exponents varying between `0` and `2` for each set. Note that the first positive integer cannot be greater than 50, and that the second one cannot be greater than 10.

**Sample output:**

```
Enter a: 4
Enter b: 3
1       1       1       1
1       2       4       8
1       3       9       27
1       4       16      64
```

```
import java.util.*;

public class Powers {

   public static void main (String[] args){

         Scanner scan = new Scanner (System.in);
         int a, b;

         do{
              System.out.println("Enter a: ");
              a = scan.nextInt();
         } while (a > 50);

         do{
             System.out.println("Enter b: ");
             b = scan.nextInt();
         } while (b > 10);

         for (int i=1; i <= a; i++)
         {
            for (int j=0; j <= b; j++)
            {
               System.out.print((int)Math.pow(i, j) + " ");
            }
            System.out.println();
         }
    }
}
```