COE 212 – Engineering Programming

Welcome to Exam II
Tuesday November 28, 2018

Instructors: Dr. Dima El-khalil
Dr. Jawad Fahs
Dr. Joe Tekli
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 **130 minutes** to complete the **6** 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 (**20 minutes)** [11 points]

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

```
public class MethodCalls {
public static void main(String[] args) {
      method1();
      int output2 = method2();
      System.out.println(output2);
      int output3 = method3();
      System.out.print(output3);
}
private static void method1() {
      int i;
      for(i=0; i<=4; i+=2)
            System.out.print(i + " ");
      System.out.println(i);
}
private static int method2() {
      int index = 0, output = 0;
      while(index < 7) {
            if(index == 3) {
                  index++;
                  continue;}
            output+=index;
            index++;}
      return output;
}
private static int method3() {
      int output = 1;
      int counter = 1;
      while(true) {
            int index = 1;
            output *= index;
            index++;
            counter++;
            if(counter == 4)
                  break;}
      return output;
}
}
```

1) How many `static` single-parameter methods does the `MethodCalls` class contain?
   a. 3
   b. 2
   c. **1**
   d. None of the above
2) Which of the following is true about the `continue` statement included inside `method2`?
   a. `continue` is used to exit the body of the `while` loop for good
   b. **Whenever `continue` is executed, the Boolean expression used by the `while` loop is re-evaluated immediately**
   c. `continue` cannot be included inside the body of a conditional statement
   d. All of the above statements are false
3) How many constructor methods does `MethodCalls` have?
   a. **1**
   b. 0
   c. 2

   d. None of the above

4) What output does `method1()` print to the screen?
   a. `0 2 4`
   b. **`0 2 4 6`**
   c. `0 2`
   d. None of the above

5) What value does `method2()` store in the `output2` variable?
   a. `15`
   b. **`18`**
   c. `25`
   d. None of the above

6) What value does `method3()`store in the `output3` variable?
   a. **`1`**
   b. 6
   c. `24`
   d. None of the above

7) If a non-static instance variable called `var1` is added to `MethodCalls`, which of the following methods of the `MethodCalls` class can reference `var1` by name?
   a. All of the methods defined in `MethodCalls` can reference `var1` by name
   b. **None of the methods defined in `MethodCalls` can reference `var1` by name**
   c. Only the main method can reference `var1` by name
   d. Only `method1`, `method2`, and `method3` can reference `var1` by name

8) The removal of the `static` reserved word from the header of `method3`
   a. is acceptable
   b. results in a run-time error
   c. **causes the compiler to issue an error message**
   d. is considered as a logical error

9) Consider another Java file that contains a `public class` called `MethodCallsUser`. Assume that `MethodCallsUser.java` is part of the same folder that includes `MethodCalls.java`. Suppose that the `MethodCallsUser` class encloses a method called `user1` that wishes to call `method1` as defined in the class `MethodCalls`, which of the following statements can be used by `user1` to invoke `method1`?
   a. `MethodCalls.method1();`
   b.
   ```
   MethodCalls obj = new MethodCalls();
   obj.method1();
   ```
   c. Either of the above
   d. **None of the above**

10) Consider `MethodCallsUser` from the previous question, if it manages to correctly call `method1` of the `MethodCalls` class, then what type of relationship is said to exist between `MethodCallsUser` and `MethodCalls`?
   a. Self-dependency
   b. Aggregation
   c. Inheritance
   d. **None of the above**

11) To avoid using the `continue` reserved word inside the body of the `while` loop of `method2`, one can replace the existing body of that `while` loop with :
   a.
   ```
   if(index!=3)
           output+=index;
   index++;
   ```
   b.
   ```
   if(index == 3) {
           index++;
   } else {
           output+=index;
           index++;
   }
   ```
   c. **Both of the above**
   d. None of the above

Problem 2: True or false questions (**20 minutes**) [14 points]

1. Assuming that `x`, `y` and `z` are `int` variables, the following code fragment:
```
if(x<1)
      if(y < 2)
            if(z < 3)
                  System.out.print(“Bravo”);
```
can be rewritten as:
```
if(x<1 && y < 2 && z < 3)
      System.out.print(“Bravo”);
```
Answer: **True** False

2. The following code fragment correctly finds the minimum between `x`, `y`, and `z`, which are assumed to be `int` variables:
```
int min = x;
if(min < y)
      min = y;
if(min < z)
      min = z;
System.out.print(“Min: ” + min);
```
Answer: **True** False

3. The following code fragment correctly finds the maximum between `x` and `y`, which are assumed to be `int` variables:
```
int max;
int z = (x>y)?0:1;
switch(z) {
case 0 : max = x ;
case 1 : max = y;
}
System.out.print(“Max: ” + max);
```
Answer: True **False**

4. The following method correctly prints `10` multiples of `2` over `5` lines of output with `2` multiple values per line.
```
public void method1() {
      int counter = 0 ;
      for(int mult = 2; mult<=20; mult +=2) {
            System.out.print(mult + “ ”);
            counter++;
            if(counter % 2 == 0)
                  System.out.println();
      }
}
```
Answer: **True** False

5. The body of the following repetition statement executes 4 times:
```
int count = 0;
do {
      count++;
      System.out.println(count);
} while(count < 5);
```
Answer: True **False**

6. The body of the following repetition statement executes 10 times:
```
for(int i=-3; i<=6; i++)
      System.out.print(i);
```
Answer: **True** False

7. The `String` class implements the Java predefined `Comparable` interface.
Answer: **True** False

8. The following method correctly returns a reversed version of the input `String` parameter `str`.

```
public String method2(String str) {
      String output = “” ;
      for(int i=str.length()-1; i>0; i--)
            output += str.charAt(i);
}
```

Answer: True **False**

9. The following method correctly returns the number of prime digits found in the `int` parameter `val`. For example, if a value of `2358` is submitted to that method, it returns a value of `3`. Note that the prime numbers that are less than `10` are as follows: `2`, `3`, `5`, and `7`.

```
public int method3( int val) {
      int lastDigit, counter = 0;
      while(val!=0) {
            lastDigit = val % 10;
            if(lastDigit==2 || lastDigit==3 || lastDigit==5
              || lastDigit == 7)
                  counter++;
            val = val/10;
      }
      return counter;
}
```

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**

11. The following code uses two overloaded versions of the `println` method:

```
System.out.println(“First message”);
System.out.println(“Second message”);
```

Answer: True **False**

12. The following code prints: `2`

```
int x = 0;
for(int i=0; i<2; i++)
for(int j=i+1; j<3; j++)
x += i*j;
System.out.print(x);
```

Answer: **True** False

13. The header of a method that uses a `Scanner` to read data from an input `File` must include the following clause: `throws IOException`

Answer: **True** False

14. The `Scanner` class is an iterator that implements the Java predefined `Iterator` interface. The latter interface includes two methods, namely `nextInt()` and `hasNext()`.

Answer: True **False**

Problem 3: Code analysis (**15 minutes**) [10 points]

1) Consider the class given below. What output is produced when it is executed?

```
public class ClassA{
      public static void main(String[] args) {
            int value = method1("Radar")?method2():method3();
            System.out.print("Value: " + value);
      }
      private static boolean method1(String str) {
            String temp="";
            for(int i=str.length()-1;i>=0;i--) temp+=str.charAt(i);
            return (temp.compareTo(str)==0);}
      private static int method2() {
            int output=0;
            for(int i=1; i<=5; i++) output += i;
            return output;}
      private static int method3() {
            int output = 1;
            for(int i=1; i<=4; i++) output *= i;
            return output;}
}
```

a. `Value: 15`
b. `Value: 24`
c. `Value: 0`
d. It doesn't compile correctly
e. None of the above

2) Consider the class given below, along with a driver class for it.

```
public class ClassB {
      private static int x=64;
      public static int method1() {
            int d, counter = 0;
            while(x > 0) {
                  d = x%10;
                  if(d%2 == 0)
                        counter++;
                  x = x/10;
            }
            return counter;
      }
      public static int method2() {
            return x;}
}
```

```
public class ClassBDriver {
public static void main(String[] args){
        int val;
        val =
method3(ClassB.method1(),ClassB.method2());
        System.out.print("Value: " + val);
}

private static int method3(int x, int y){
      return x * y;
}
}
```

When running `ClassBDriver` class, what output is produced?

a. `Value: 0`
b. `Value: 128`
c. It produces a run-time error
d. It doesn't compile correctly
e. None of the above

<u>Problem 4:</u> Method design (**15 minutes**) [10 points]

1. Write a method called *Words* that accepts a sentence as a String parameter and returns the number of words in that string (assume that the sentence does not have any punctuation).

```
public static int Words(String S)
{
       Scanner stringScan = new Scanner(S);
       int count = 0;
       while(stringScan.hasnext())
       {
          String dummy = stringScan.next();
          count++;
       }
       return count;
}
```

2. In number theory, a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself. For example, 6 = 1 + 2 + 3 and 28 = 1 + 2 + 4 + 7 + 14 are perfect. Write a method called *Perfect* that accepts a positive integer as input parameter, and prints on-screen whether the number is: "Perfect" or "Not Perfect".

```
public static void Perfect(int x)
{
    int sum = 1;
    for(int i=2; i<=x/2; i++)
    {
        if(x%i == 0)
        sum += i;
     }
    if (sum == x) System.out.print("Prefect");
    else System.out.print("Not Prefect");

}
```

Problem 5: Evaluating Java Expressions (**20 minutes**) [15 points]

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

(1)
```
int x=3;
switch(x) {
      case 0:
          x--;
      case 1:
          x *=  2;
      case 2:
          x += 3;
      case 3:
          x *= 2;
      case 4:
          x = x % 5;
      default:
          x = x * 25;}
```
Answer: x= 25

(2)
```
int c=6, x=1;
 if(!(c<2) && !(c>5))
   x*=2;
 x*=5;
```
Answer: x= 5

(3)
```
String S1 = "COE";
String S2 = "Exam";
String x;
int count=1;
  if(count == 2)
      if(count > 0)
            x=S1;
      else
            x=S2;
  else
      x=S1.concat(S2);
```
Answer: x= “COEExam”

(4)
```
String x="";
for(int i= 1;i <= 4; i++) {
  if(i % 2 == 0 && i>2) continue;
      x += i;
  }
```
Answer: x= 123

(5)
```
String S1 = "Apple";
String S2 = "applesauce";
boolean x=(S1.compareTo(S2) > 0);
```
Answer: x= false

(6)
```
String S = "You're never wrong to do the right thing ", x = "";
for(int i=0; i < S.length(); i+=5)
x += S.charAt(i);
```
Answer: x= “Yeeootit ”

(7)
```
String S = "If you judge, you will not have time to love";
int y = 0; char c; String x = "";
Scanner scan = new Scanner(S);
while (scan.hasNext())
   x += scan.next().substring(scan.next().length()/2);
```
Answer: x= "fudge,illve"

(8)
```
DecimalFormat fmt1 = new DecimalFormat("0.000");
DecimalFormat fmt2 = new DecimalFormat("0.###");
double val = Double.parseDouble(fmt2.format(123.45));
char x = fmt1.format(val).charAt(4);
```
Answer: x= 4

(9)
```
int x = 0;
for(int i=1; i < 10; i = i+3)
for(int j=1; j<i; j++) x++;
```
Answer: x= 9

(10)
```
int y = 24;
boolean x =(y / 10 != 0 && y%3 == 0 || y / 4 == 0);
```
Answer: x= true

(11)
```
String S = new String("Opportunities are like Sunrises: easily
missed");
int y = 0; char c; String x = "";
do {c = S.charAt(y); if(c== 'e') x += c; y+=2;
} while(y < S.length());
```
Answer: x= "ee"

(12)
```
double val1 = 20.304050;
double val2 = Math.floor(val1*100);
double x = val2 - (int)val1*10;
```
Answer: x= 1830.0

Problem 6: Coding Problems (**40 minutes**) [40 points]

1. Write a program called `StringExplosion`, which reads from the user a string `S`, and an integer `n`, and then produces a new string `S2` made of the same characters as `S` where every character is repeated `n` times as shown in the sample run. The program needs to perform the necessary user input validation in order to make sure that the number `n` provided as input is positive.

Sample run:
Enter string S: Java
Enter n: 3
String S2: JJJaaavvvaaa

```
import java.util.Scanner;

class StringExplosion{

public static  void main (String [] args)
{

     Scanner scan = new Scanner(System.in);

     System.out.print("Enter string S: ");
     String S = scan.nextLine();

     String S2 = "";
     int n;

     do{
           System.out.print("Enter number n: ");
           n = scan.nextInt();
     } while (n<0);

     for(int i =0; i< S.length(); i++)
     {
           for (int j = 0; j< n; j++)
           {
                 S2 += S.charAt(i);
           }
     }

     System.out.print("String S2: " + S2);
  }
}
```

2. Write a program called `SentenceCount`, which reads a sentence from the user, then counts the number of vowels (a, e, i, o, u, A, E, I, O, U), digits (0-9), spaces, and consonants contained in it, and prints the respective count as shown in the sample run. We assume that the input sentence is made of the above characters only, and does not contain any other punctuation or special characters.

Sample run:

Enter a Sentence: COE 212 Exam 2
The Sentence is made up of: 4 vowels, 4 numbers, 3 spaces and 3 consonants

```
import java.util.Scanner;

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

      Scanner scan= new Scanner(System.in);

      System.out.print("Enter a Sentence: ");

      String S1=scan.nextLine();
      String S2=new String("AOEIUaoeiu");

      int countv=0; int counti=0; int counts=0;

      for (int i=0; i<S1.length(); i++)
            {if (S2.indexOf(S1.charAt(i)) != -1)
                  countv++;
            else if (S1.charAt(i)>='1' && S1.charAt(i) <='9')
                  counti++;
            else if (S1.charAt(i) ==' ')
                  counts++;}

      int countc=S1.length()-(counti+countv+counts);

      System.out.println("The Sentence is made up of “ +
            countv + " vowels, " + counti + " numbers, " +
            counts+" spaces and "+ countc + " consonants");

      }
}
```

3. Write a Java program called `ComputeAvg` that reads the student grades from a textual file called `“GradesFile.txt”`, and then computes and prints on-screen the average grade rounded to two decimal digits.

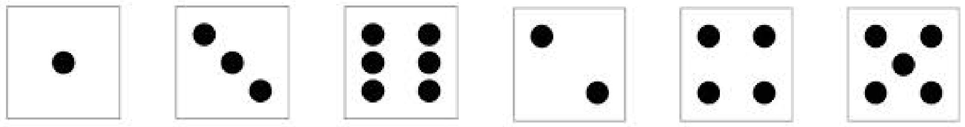
   Given the sample content of file `GradesFile.txt:`

```
Grade of student #1: 65
Grade of student #2: 75
Grade of student #3: 89
Grade of student #4: 90
```

   Sample output:
   Average grade = 75.79

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

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

           File F = new File("GradesFile.txt");
           Scanner scanF = new Scanner(F);

           int count = 0;
           int sum = 0;

           while(scanF.hasNextLine())
           {
               count++;

               String S = scanF.nextLine();
               Scanner scanS = new Scanner(S);

               while(!scanS.hasNextInt())
               {
                  scanS.next();
               }

               int grade = scanS.nextInt();
               sum += grade;
            }

            double avg = (double) sum / count;

            System.out.print("Average grade = " + avg);
      }
}
```

4. Write a Java program titled `DiceGame` that simulates the rolling of a pair of dice in a dice game. Each individual die consists of 6 faces, numbered from 1 to 6.



The program will display the result of rolling a pair of dice, and will repeat a maximum of 5 times or until either: the user wins, or decides to quit. The user wins the game if the pair of dice comes up *snake eyes* (NOTE: "*Snake eyes*" means that both dice show a value of 1). See different sample outputs below:

Sample output 1:
```
Roll #1:  2 3  - Continue? Y/N:  Y
Roll #2:  1 1
You WIN after 2 rounds!
```

Sample output 2:
```
Roll #1:  2 6  - Continue? Y/N:  Y
Roll #2:  4 5  - Continue? Y/N:  Y
Roll #3:  3 3  - Continue? Y/N:  Y
Roll #4:  2 1  - Continue? Y/N:  Y
Roll #5:  5 5
You LOOSE!
```

Sample output 3:
```
Roll #1:  1 6  - Continue? Y/N:  Y
Roll #2:  5 5  - Continue? Y/N:  Y
Roll #3:  5 4  - Continue? Y/N:  N
You quit after 3 rolls!
```

The user decides to continue or quit the game by providing as answer: `'Y'` or `'N'`. The program needs to perform the necessary user input validation in order to make sure that no other character is accepted.

```
import java.util.Scanner;
import java.util.Random;

public class DiceGame{

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

      int count = 0;
      char answer = 'Y';
      boolean won = false;
      int die1, die2;

      while(answer == 'Y' && !won && count <5)
      {
         count++;
         die1 = rnd.nextInt(6) + 1;
         die2 = rnd.nextInt(6) + 1;
         System.out.print("Roll # " + count + " : " + die1 +
                              " " + die2);
         if(die1 == 1 && die2 == 1) won = true;
         else
```

```
          {

            do{
               System.out.print(" - Coninue? Y/N: ");
               answer = scan.nextLine().toUpperCase().charAt(0);
            } while (answer != 'Y' && answer != 'N');
          }
       }


    if(won) System.out.print("\n You WIN after " + count
                              + " rounds!");
    else if(count <5) System.out.print("You quit after " +
                                        count + " rounds!");
    else System.out.print("You LOOSE");
   }
  }
```