# COE 212 – Engineering Programming

Welcome to Exam II
Thursday April 21, 2016

Instructors: Dr. Salim Haddad
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 **120 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)** [16 points]

Consider an input text file called "**in.txt**" that contains the following data:

```
2+4
3*5
6-5
```

Assume that a `Scanner` object called `inFromFile` was instantiated properly to read data from **in.txt**

1) Which of the following statements creates a `String` called `line1` that stores the first line of **in.txt**?
    a. `String line1 = inFromFile.next();`
    b. `String line1 = inFromFile.nextLine();`
    c. **Both of the above**
    d. None of the above
2) Given the `line1` object created earlier, which of the following code evaluates the summation in `line1` correctly? Assume that `sum` is an `int` that was properly declared and initialized to `0`
    a.
    ```
    for(int i=0; i<line1.length; i++)
            if(line1.charAt(i)!='+')
                    sum+=Integer.parseInt(line1.charAt(i)+"");
    ```
    b.
    ```
    for(int i=0; i<line1.length; i++)
            if(line1.substring(i, i+1).equals("+"))
                    sum+=Integer.parseInt(line1.substring(i, i+1));
    ```
    c. Both of the above
    d. **None of the above**
3) Assume that `line2` is a `String` variable holding the second line of **in.txt**. Which of the following creates an object that can be used to obtain the numbers on either side of `*` in `line2`?
    a.
    ```
    Scanner line2Scan = new Scanner(line2);
    lineScan.setDelimiter("*");
    ```
    b.
    ```
    Scanner line2Scan = new Scanner(line2);
    lineScan.setDelimiter('*');
    ```
    c. `Scanner line2Scan = new Scanner(line2, "*");`
    d. **None of the above**
4) Given the `line2Scan` object created earlier. Which of the following evaluates the product in `line2` correctly? Assume that `prod` is an `int` that was properly declared and initialized to `0`
    a. `for(int i=1;i<=2;i++) prod*=Integer.parseInt(line2Scan.next());`
    b. `for(int i=0;i<=1;i++) prod*=Integer.parseInt(line2Scan.next());`
    c. Both of the above
    d. **None of the above**
5) `line3` is a `String` variable holding the third line of **in.txt**. Which of the following creates 2 variables called `num1` and `num2` holding the numbers on either side of the – operator in `line3`?
    a.
    ```
    int num1=Integer.parseInt(line3.charAt(line3.indexOf('-')-1));
    int num2=Integer.parseInt(line3.charAt(line3.indexOf('-')+1));
    ```
    b. **`int num1=Integer.parseInt(line3.substring(0,1));`**
       **`int num2=Integer.parseInt(line3.substring(2,line3.length());`**
    c. Both of the above
    d. None of the above
6) Given `the num1` and `num2` variables created earlier. Which of the following correctly finds their greatest common divisor?
    a.
    ```
    while(num1 != num2) {
            if(num1<num2) num1 = num1 – num2;
            else num2 = num2 – num1;}
    ```
    b.
    ```
    while(num1 != num2) {
            if(num1<num2) num1-=num2;
            if(num2<num1) num2-=num1;}
    ```
    c. **Both of the above**
    d. None of the above

7) Which of the following finds the larger value between `num1` and `num2` and stores it in an `int` variable called `larger`?
   a. `larger = (num1>num2) : num1 ? num2;`
   b. `larger = (num1>num2) : num2 ? num1;`
   c. **`if(num1 – num2 > 0) larger = num1; else larger = num2;`**
   d. None of the above
8) When writing data to a file, which of the following `import` declaration statements is not needed?
   a. `import java.io.IOException;`
   b. `import java.io.File;`
   c. **`import java.io.FileReader;`**
   d. None of the above
9) Which of the following methods have been overloaded multiple times by the Java language?
   a. `print`
   b. `nextInt`
   c. **Both of the above**
   d. None of the above
10) Which of the following results in a compile-time error?
   a. **Not placing "`throws IOException`" at the end of the header of a main method that writes data to an output file**
   b. Not closing the `PrintWriter` object inside a main method that writes data to an output file
   c. Both of the above
   d. None of the above
11) What type of relationship exists between `String` and `Comparable`?
   a. "uses" relationship with `String` implementing the methods of `Comparable`
   b. "has a" relationship with `String` containing references to `Comparable` objects
   c. "aggregation" relationship with `String` implementing the methods of `Comparable`
   d. **None of the above**
12) Which of the following cannot be placed inside the body of a conditional statement?
   a. `break`
   b. `continue`
   c. Both of the above
   d. **None of the above**
13) Which of the following is not part of any Java built-in interface?
   a. `compareTo`
   b. **`nextLine`**
   c. `hasNext`
   d. None of the above
14) Which of the following is false about static variables?
   a. **Each object has its own copy of a static variable with its own memory space**
   b. A static variable cannot be referenced by name by a non-static method
   c. A static variable can be referenced by name by a static method
   d. All of the above
15) Which of the following is false about `switch`?
   a. The `default` case is optional
   b. The expression evaluated at the beginning of a `switch` statement can be of type `byte`
   c. **Not including a `break` statement inside the first case results always in the execution of the statements corresponding to all the cases**
   d. None of the above
16) In Java, which of the following runs forever?
   a. `int val = 12; while(val < 13) val--;`
   b. **`while(true) continue;`**
   c. Both of the above
   d. None of the above

## **<u>Problem 2:</u>** True or false questions (**20 minutes)** [14 points]

1. Assuming that `x`, `y` and `z` are `int` variables, the following code fragment:
```
if(!(x<y) || (y>z)) System.out.print("Met");
else System.out.print("Not met");
```
can be rewritten as:
```
if(x>=y && y<=z) System.out.print("Not met");
else System.out.print("Met");
```
Answer: True **False**

2. The output of the program segment below is: `1`
```
int x=2, y=2, z=3, w=1, s;
if(x < y)
if(x < z) s = x ;
else s = z;
else if(y < z) s = y;
else s = z;
if(w < s) s = w;
System.out.print(s);
```
Answer: **True** False

3. The following method returns `true` if `x` is prime and `false` otherwise. Assume that `x` is strictly greater than `1`.
```
public boolean method1(int x) {
      int y=0;
      for(int i=2; i<x/2; i++)
            if(x%i==0)
                  y++;
      return (y==0);}
```
Answer: True **False**

4. Assume that the `method1` given earlier can be used to determine whether or not its parameter is prime. The following method prints the largest prime number that is less than or equal to `x`. Assume that `x` is strictly greater than `1`.
```
public void method2(int x) {
      for(int y=x; y>=2; y--)
            if(method1(y)) {System.out.print(y); break;}
}
```
Answer: **True** False

5. The body of the following repetition statement executes `5` times:
```
int count = 1;
while(true) {
      count++;
      if(count == 6) break;}
```
Answer: **True** False

6. Consider the following nested `while` loops. The body of the inner `while` loop executes `100` times.
```
int count1=1, count2;
while(count1<=10) {
      count2 = 0;
      while(count2 <= 10) {
            System.out.println(count2);
            count2++;}
      count1++;}
```
Answer: True **False**

7. In Java, overloading makes it possible for two different classes like `Scanner` and `Random` to both have a method called `nextInt()`.

Answer: True **False**

8. The following method returns the middle digit of the input parameter `x`. Assume that x has an odd number of digits.

```
public int method2(int x) {
      int y;
      String z = x + “”;
      y = Integer.parseInt(
      z.substring(x.length()/2, x.length()/2+1));
      return y;}
```

Answer: True **False**

9. The following code prints the even numbers between `1` (inclusive) and `10` (inclusive).

```
int count=2;
do {
      if(count % 2 != 0) {
            count++;
            continue;}
      System.out.println(count);
      count++;} while(count <= 10);
```

Answer: **True** False

10. The following code prints Tails when `x` is `0` and Heads when `x` is `1`.

```
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 prints: `2nd threshold reached`

```
int val=2;
if(val == 3) System.out.println(“1st threshold reached”);
else if(val == 2) System.out.println(“2nd threshold reached”);
System.out.println(“Neither thresholds reached”);
```

Answer: True **False**

12. The following code prints: `4`

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

Answer: **True** False

13. A local variable is created when a method is called and then destroyed when the method is exited. An instance variable is created when the class containing it is instantiated and then destroyed when the program using that instance of the class terminates its execution.

Answer: True **False**

14. The following code uses two overloaded versions of the `nextInt` method.

```
Random rnd = new Random();
System.out.println(rnd.nextInt());
System.out.println(rnd.nextInt(3));
```

Answer: **True** False

## **Problem 3:** Code analysis (**10 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) {
            String name1="Salim", name2="Joe";
            method1(name1, name2);
            System.out.print("Name1: "+name1+", Name2: "+name2);
      }
      private static void method1(String name1, String name2) {
            String temp;
            temp = name1;
            name1 = name2;
            name2 = temp;
      }
}
```

- **a. `Name1: Salim, Name2: Joe`**
- b. `Name1: Joe, Name2: Salim`
- c. `Name1: Joe, Name2: Joe`
- 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 int x ;
      public ClassB(int x) {
            this.x=x;
      }
      public int getX() {
            return x;
      }
      public void setX(int x) {
            this.x=x;
      }
      public String toString() {
            return Integer.toString(x);
      }
}
```

```
public class ClassBDriver {
public static void main(String[] args){
        ClassB b1 = new ClassB(2);
        ClassB b2 = new ClassB(3);
        method2(b1, b2);
        System.out.print("b1:"+b1+
            ", b2:"+b2);}

private void method2(ClassB b1, ClassB b2){
      int temp;
      temp = b1.getX();
      b1.setX(b2.getX());
      b2.setX(temp);
}
}
```

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

- a. `b1:2, b2:3`
- b. `b1:3, b1:2`
- c. It produces a run-time error
- **d. It doesn't compile correctly**
- e. None of the above

## **Problem 4:** Code analysis (**15 minutes**) [10 points]

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

(1)
```
char S= ‘+’; int x = 20;
 switch(S) {
   case ‘-’:
        x -= 3;
        break;
   case ‘+’:
        x += 2;
   case ‘*’:
        x /= 4;
        break;
   case ‘/’:
        x *= 4;
   default:
        x+=10;
   }
```

**Answer: x = 5**

(2)
```
int n=10, x=0;
 while(x<n) x += 5; for(int j=0; j<x; j+=3)
 n= n+2;
 x=n;
```

**Answer: x= 18**

(3)
```
int x=2, i=7;
do {
i++; x -= i;
i*=3;
} while(i < 20);
x -= i;
```

**Answer: x= -30**

(4)
```
boolean x=true;
 for(int i=0; i<=3; i++)
 if(i/3 == 0)
 x = !((i > 0) && x);
```

**Answer: x= true**

(5)
```
String S = "5 a 4 b 3 c 2 d 1";
int x = 1, n;
Scanner scan = new Scanner(S);
while(scan.hasNextInt()) {
   n = scan.nextInt();
   if(n > x+1)x = n;}
```

**Answer x = 5**

(6)
```
String S = new String("Jim ate the fig");
int y = 0; char c; String x = "";
Scanner scan = new Scanner(S);
while (scan.hasNext())
    x += scan.next().substring(scan.next().length()-1);
```

**Answer: x = “me”**

(7)
```
int x=8, s;
for(s=--x; s>0; s -=2)
x -= s; s--;
```

**Answer: x = -9**

(8)
```
String x = ""; int price = 80;
if(price >= 90 || price <=50)
if(price <= 50) if (price >= 40) x = "average price";
else x = "cheap"; else x = "expensive";
```

**Answer: x = “”**

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

**Answer: x = 9**

(10)
```
String x=""; int y = 1;
for(;y<= 5; y+=1) {
if(y % 2 == 0) continue;
x += y;
y++;}
```

**Answer: x = 135**

## **Problem 5:** Method definition (**15 minutes)** [10 points]

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

1. **`sumEvenRange()`** is a method that accepts two integers `a` and `b` as input parameters, and returns the sum of the even integers in that range.

```
public static int sumEvenRange(int a, int b)
{
      int sum = 0;
      for(int i = a; i<=b; i++)
      {
          if(i%2 == 0)   sum += i;
      }
      return sum;
}
```

2. **`minThree()`** is a method that accepts three double numbers `x`, `y`, and `z` as input parameters, and returns the minimum of the three.

```
public static double minThree(double x, double y, double z)
{
     double min;
     if (x < y)
         if (x < z) min = x;
         else min = z;
     else
         if (y < z) min = y;
         else min = z;

     return min;
}
```

3. **`countX()`** is a method that accepts a `String` input parameter `S`, and returns the number of times character `'x'` appears in `S`, considering both its uppercase and lowercase forms (counting the occurrences of both: 'x' and `'X').`

```
public static int countX(String S)
{
   int count = 0;
   S = S.toLowerCase();
   for(int i=0; i< S.length(); i++)
      if(S.charAt(i) == 'x') count++;

   return count;
}
```

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

1. Write a program called `ReverseSentence` which accepts as input a sentence consisting of a string `S1` of words, and then produces another string `S2` consisting of the words of `S1` in reverse order.

**Sample output:**
**Enter sentence: Those who dare fail achieve**
**Reverse sentence: achieve fail dare who those**

```
import java.util.Scanner;

public class ReverseSentence
{
    public static void main(String [] args)
   {
      Scanner scan = new Scanner(System.in);
      System.out.print("Enter sentence: ");
      String S1 = scan.nextLine();

      String S2 = "";

      Scanner scanS1 = new Scanner(S1);

      while(scanS1.hasNext())
      {
           S2 = scanS1.next() + " " + S2;
      }

      System.out.print("Reverse sentence: " + S2);

   }

}
```

2. Write a program called `PowerOf2` that reads a positive integer number `n` as input, and then determines whether or not `n` is a power of 2 (for instance, numbers 1, 2, 4, and 8 are powers of 2). 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 output:**

```
Enter a positive number: -3
Wrong input! Try again: 16
Number 16 is a power of 2
```

```
import java.util.Scanner;

public class PowerOf2
{

    public static void main(String [] args)
   {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");

        int n = scan.nextInt();
   int a = n;

   while (n <0)
   {
        System.out.print("Wrong input! Try again: ");
        n = scan.nextInt();
   }

   while(n/2 > 1 && n%2 == 0)
   {
       n = n/2;
   }

   if( n == 2) System.out.print("Number " + a + " is a power of 2");
   else System.out.print("Number " + a + " is NOT a power of 2");

   }

}
```

3. Write a program called `PrintingStars` that reads a positive integer number `n` as input, and then prints the following star (asterisk) configuration on-screen.

**<u>Sample output</u>:**
**Enter a positive number: 4**
**Output configuration:**

```
*
**
***
****
***
**
*
```

```
import java.util.Scanner;

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

       System.out.print("Enter a positive number: ");
       int n = scan.nextInt();

       while(n <=0)
       {
           System.out.print("Wrong input! Try again: ");
           n = scan.nextInt();
       }

      System.out.println(“Output configuration:”);

       for(int i=0; i< 2*n; i++)
       {
            if(i<=n)

                  for(int j=0; (j<i); j++)
                  {
                      System.out.print("*");
                  }

            else
                  for(int j= 2*n-i; j>0; j--)
                  {
                      System.out.print("*");
                  }

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

4. Write a program called `MathExpress` that evaluates basic mathematical expressions such as `2+15.1` and `3.1416*4.5` provided as input by the user. The program accepts as input a string expression consisting of a number, followed by an arithmetic operator, followed by another number, such as `“2+15.1”`. (operators that are allowed are: +, -, *, and /). Then the program scans the expression in order to compute and print its value on-screen.

**<u>Sample output</u>:**
**Enter math expression: 2+15.1**
**Result: 17.1**

```
import java.util.Scanner;

public class MathExpress
{

   public static void main(String [] args)
   {

       Scanner scan = new Scanner(System.in);

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

          double n1, n2, result;
          char op;


          String operators = new String("+-*/");

          int i = 0;

          while (operators.indexOf(S.charAt(i))<0 && i<S.length()) i++;


          n1 = Double.parseDouble(S.substring(0, i));
          op = S.charAt(i);
          n2 = Double.parseDouble(S.substring(i+1));


          if(op == '+')  result = n1 + n2;
          else if(op == '-') result = n1 - n2;
          else if(op == '*') result = n1 * n2;
          else result = n1 / n2;

          System.out.print("Result: " + result);
     }

 }
```