# COE 212 – Engineering Programming

Welcome to Exam II
Friday April 20, 2018

Instructors: Dr. Dima El-khalil
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!!**

## **Problem 1:** Multiple choice questions (**25 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.text.DecimalFormat;
public class ProblemI {
      public static void main(String[] args) {
            int v1=0, c1=0, v2, v3=0, d1, d2=1, d3;
            String s1 = "", s2 = "";
            char L1, L2;
            DecimalFormat fmt = new DecimalFormat("00000");
            while(c1 <= 14) {
                  if(c1 == 13) {
                        c1++;
                        continue;
                  }
                  if(c1%2 == 0)
                        v1 += c1;
                  else
                        v1 -= c1;
                  c1++;
            }
            System.out.println("V1: " + v1);
            v2 = Integer.parseInt("" + c1 + v1);
            System.out.println("V2: " + fmt.format(v2));
            do {
                  d1 = v2%10;
                  v3=v3*10 + d1;
                  v2 /= 10;
                  d3 = d1;
                  if(d3 != 0 && d2 != 0) {
                        while(d3 != d2) {
                              if(d3 > d2)
                                    d3 = d3 - d2;
                              else
                                    d2 = d2 - d3;
                        }
                  }
                  s1 = s1 + d3;
                  d2 = d1;
            } while(v2 != 0);
            System.out.println("S1: " + s1);
            System.out.println("V3: " + v3);
            s1 = "12345";
            int i;
            for(i=1; i<s1.length(); i++)
                  s2 += s1.charAt(s1.length()-i);
            System.out.println("S2: " + s2);
            L1 = s2.charAt(s2.length()-1);
            System.out.println("L1: " + L1);
            switch(L1) {
            case '1': L2 = 'a'; break;
            case '2': L2 = 'b'; break;
            case '3': L2 = 'c'; break;
            default: L2 = 'z';
            }
            System.out.println("L2: " + L2);
      }
}
```

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. `6`
    b. **`5`**
    c. `4`
    d. None of the above
3) Excluding the `args` parameter, how many local variables does the `main` method have?
    a. `11`
    b. `12`
    c. **`13`**
    d. None of the above
4) How many `static` methods are called inside the body of the `main` method?
    a. `0`
    b. **`1`**
    c. `2`
    d. None of the above
5) How many iterations does the first repetition statement perform, namely the one whose termination condition is based on the value of the `c1` variable?
    a. `12`
    b. `13`
    c. `14`
    d. **None of the above**
6) How many iterations does the `do..while` repetition statement, present inside the `main` method, perform?
    a. `3`
    b. **`4`**
    c. `5`
    d. None of the above
7) How many iterations does the `for` repetition statement, present inside the `main` method, perform?
    a. `3`
    b. **`4`**
    c. `5`
    d. None of the above
8) What output is produced by the first `println` statement of the `main` method?
    a. `V1: 5`
    b. `V1: 7`
    c. **`V1: 20`**
    d. None of the above
9) What output is produced by the second `println` statement of the `main` method?
    a. `1420`
    b. `01420`
    c. `001420`
    d. **None of the above**
10) What output is produced by the third `println` statement of the `main` method?
    a. **`S1: 0211`**
    b. `S1: 11`
    c. `S1: 011`
    d. None of the above
11) What output is produced by the fourth `println` statement of the `main` method?
    a. **`V3: 251`**
    b. `V3: 241`
    c. `V3: 240`
    d. None of the above
12) What output is produced by the fifth `println` statement of the `main` method?
    a. `S2: 4321`

b. `S2: 54321`
c. **`V3: 5432`**
d. None of the above

13) What output is produced by the sixth `println` statement of the `main` method?
a. `L1: 1`
b. **`L1: 2`**
c. `L1: 3`
d. None of the above

14) What output is produced by the seventh `println` statement of the `main` method?
a. `L1: a`
b. `L1: b`
c. `L1: c`
d. **None of the above**

15) 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: 4`
b. **`i: 5`**
c. `i: 6`
d. None of the above

16) Consider the following Boolean expression employed inside the main method: `d3!=0 && d2!=0`
Which of the following Boolean expressions is equivalent to that expression?
a. `!(d3=0 || d2=0)`
b. `!(d3=0) && !(d2=0)`
c. Both of the above
d. **None of the above**

17) Consider the following statement employed inside the main method: `s1 = s1 + d3;`
Which of the following Boolean expressions is equivalent to that expression?
a. `s1.concat(d3);`
b. `s1.concat(""+d3);`
c. **`s1+=d3;`**
d. Both (b) and (c)

18) Assuming that the `break` statement is removed from within the second `case` of the `switch` statement used inside the `main` method, what output does the last `println` statement produce in that case?
a. `L2: b`
b. **`L2: c`**
c. `L2: z`
d. None of the above

19) Consider the `s1` variable used inside the `main` method. Which of the following can be used to extract its middle character if it has an odd number of characters?
a. `s1.charAt(s1.length()/2);`
b. `s1.substring(s1.length()/2, s1.length()/2+1);`
c. **Both of the above**
d. None of the above

20) Consider the `if..else` statement employed inside the first `while` repetition statement of the `main` method. Which of the following statements can be used to replace it?
a. `v1 = v1 + ((c1%2==0)?c1:-c1);`
b. `v1 = v1 + ((c1%2==0)?-c1:c1);`
c. `v1 += ((c1%2==0)?c1:-c1);`
d. **Both (a) and (c)**

## **Problem 2:** True or false questions (**15 minutes)** [10 points]

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

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

can be rewritten as:

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

Answer: True **False**

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

```
int diff = str1.length()-str2.length();
int flag;
if(diff > 0)
       flag = 1;
else
       flag = -1;
switch(flag) {
case 1 :
       System.out.print(str1) ;
       break ;
case -1:
       System.out.print(str2);
}
```

Answer: True **False**

3. In what follows, the method called `sum` has been properly overloaded:

```
double sum(int val1, int val2) {
       return val1 + val2 ;
}
int sum(int val1, int val2) {
       return val1+val2 ;
}
```

Answer: True **False**

4. The following method correctly prints the sum of all the multiples of `5` or `7` that are less than or equal to `1000`.

```
public void method1() {
       int sum = 12;
       for(int val = 10; val<=1000; val++) {
              if(val%5==0||val%7==0) sum+=val;
       }
       System.out.print(sum);
}
```

Answer: **True** False

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

```
int count = 0;
do {
       count--;
       System.out.println(count);
} while(count <= 0);
```

Answer: True **False**

6. Consider the following nested loops. The body of the inner loop executes a total of `99` times:

```
int counter=1;
for(int i=0; i<11; i++) {
       while(counter<=9) {
              System.out.println("Bravo");
              counter++;
       }
}
```

Answer: True **False**

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

```
String str = "Exam is fun";
int i=0, j=str.length()-1;
for(; str.length()>2; ) {
       str = str.substring(i, j);
       i=0;
       j=str.length()-1;
}
System.out.print("\"" + str + "\"");
```

Answer: **True** False

8. The following method correctly modifies the `String` called `str` by exchanging its first half with its second half. That is, if `str` is equal to "`Equal`", then the return value would be "`aluEq`" and if `str` is equal to "`Door`", then the return value would be "`orDo`".

```
public String method2(String str) {
       String output = "" ;
       for(int i=str.length()/2+1; i<str.length();i++)
              output+=str.charAt(i);
       output += str.charAt(str.length()/2);
       for(int i=0; i<str.length()/2; i++)
              output+=str.charAt(i);
       return output;
}
```

Answer: True **False**

9. The following method correctly returns the number of non-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 `1`. 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) {
              } else
                     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"); break;
case 1: System.out.print("Heads");}
```

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) {
		String str1 = "I Feel Good";
		String str2 = method1(str1);
		System.out.print("Phrase: " + str2);}
	private static String method1(String str) {
		char current, letter;
		for(int i=0; i<str.length()-1; i+=2) {
			current = str.charAt(i);
			if(current != ' ') {
				letter = (char) method2(i, current);
				str = method3(str, i, letter);}
		}
		return str;}
	private static int method2(int i, char current) {
		if(current >= 'a' && current <= 'z')
			return i + 'a';
		else if(current >= 'A' && current <= 'Z')
			return i + 'A';
		else
			return ' ';}
	private static String method3(String str, int i, char letter) {
		str = str.substring(0, i) + letter + str.substring(i+1);
		return str;}}
```

a. `Phrase: a ceel giod`
b. `Phrase: B Deel Ghod`
c. `Phrase: I Deel Giod`
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 = 4;
	public static void method1(int y) {
		for(int i=0; i<y; i++)
			x++;
	}
	public static int method2(int z) {
		int w=0;
		do {
			x = x/2;
			w++;
		} while(w < z);
		return x;
	}
}
```

```
public class ClassBDriver {
	public static void main(String[] args) {
		ClassB b1 = new ClassB();
		ClassB b2 = new ClassB();
		b1.method1(4);
		b1.method2(2);
		b2.method1(3);
		int val = b2.method2(3);
		System.out.println("Val: " + val);
	}
}
```

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

a. `Value: 1`
**b. `Value: 0`**
c. It produces a run-time error
d. It doesn't compile correctly
e. None of the above

## **Problem 4:** Method design (**15 minutes**) [10 points]

1. Write a method called *removeNonAlphabetic* that accepts a sentence as a `String` parameter, modifies that input sentence by removing all of its non-alphabetic characters, and finally returns the resulting modified version of the input sentence.

```
public static String removeNonAlphabetic(String S){
char current;
String output = "";
for(int i=0; i<S.length(); i++) {
current = S.charAt(i);
if((current>='a' && current <= 'z') ||
(current>='A' && current <= 'Z'))
output += current;
}

return output;

}
```

2. Write a method called *eventDigitAvg* that takes an `int` parameter called `x` and returns the average of all of the even digits that are present in `x`.

```
public static double eventDigitAvg(int x){
double avg = 0.0;
int count = 0;
int digit;
do {
digit = x%10;
if(digit % 2 == 0) {
count++;
avg += digit;
}
x=x/10;
} while(x!=0);

if(count != 0)
avg/=count;

return avg;
}
```

## **Problem 5:** Evaluating Java Expressions (**15 minutes**) [10 points]

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

(1)
```
boolean p = true, q = false, r=false;
int y = 4;
int x = ((p&&!q)||(q&&!r))?y++:++y;
```
**Answer: x= 4**

(2)
```
int x = 2, y = 11, z = 3;
switch(x) {
case 2:
      if(y%z == 2)
            x*=2;
      else
            x/=2;
case 3:
      if(y/z == 1)
            x+=2;
      else
            x%=2;
}
```
**Answer: x= 0**

(3)
```
String S1 = "COE";
String S2 = "INE";
int x = 0;
do {
  if(S1.charAt(x) == S2.charAt(x))
      break;
  x++;
} while(S1.compareTo(S2) != 0);
```
**Answer: x= 2**

(4)
```
String x="";
for(int i= 1;i <= 3; i++) {
  if(i%3==0)
  x+=2*i;
  if(i%2!=0)
  x+=3/i;
  else
  x+=i;}
```
**Answer: x= “3261”**

(5)
```
 String S1 = "Apple";
 String S2 = "applesauce";
 boolean x=
(S1.toLowerCase().compareTo(S2.substring(0, S2.indexOf(‘e’))) >
 0);
```
**Answer: x= true**

(6)
```
String S = "It is not over until the fat lady sings";
int y=0;
String x;
for(int i=S.length()/2; i < S.length(); i++)
if(S.charAt(i) == ‘t’) y = i;
x = S.substring(y);
```
**Answer: x= “t lady sings”**

(7)
```
String S = "If you judge a lot-you won’t care";
int y = 0; String x = "";
Scanner scan = new Scanner(S);
```

```
    scan.useDelimiter("-");
    while (scan.hasNext()) {
           x += scan.next().substring(scan.next().length()/2);
    }
```

**Answer: x= “judge a lot”**

(8)
```
    int x = 0;
    for(int i=1; i < 10; i = i+4)
    for(int j=1; j<i; j++)
    x++;
```

**Answer: x= 12**

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

**Answer: x= 14**

(10)
```
    DecimalFormat fmt= new DecimalFormat(“00.##”);
    double y = 1.4567;
    String x = fmt.format(y);
```

**Answer: x= “01.46”**

## **Problem 6:** Coding Problems (**45 minutes**) [40 points]

1. Write a program called `PrimeDigits`, which reads from the user an integer `n`, and then prints out to the screen the number of prime digits that are present in `n` as well as the average of these digits.

**Sample run:**

**Enter n: 12379**
**Nb. of prime digits found in 12379 is: 3**
**Average of prime digits found in 12379: 12.0**

**Solution:**

```
import java.util.Scanner;
public class PrimeDigits {
	public static void main(String[] args) {
		int input, n;
		Scanner scan = new Scanner(System.in);
		int primesCount = 0;
		double primesAvg = 0.0;
		System.out.println("Enter n: ");
		input = scan.nextInt();
		n = Math.abs(input);
		int lastDigit;
		do {
			lastDigit = n % 10;
			if(isPrime(lastDigit)) {
				primesCount++;
				primesAvg += lastDigit;
			}
			n /= 10;
		} while(n != 0);

System.out.println("Nb of prime digits found in " + input + "is: " +
primesCount);
if(primesCount != 0)
		System.out.println("Average of prime digits found in " +
		input + ": " +
		(primesAvg/primesCount));
	}
	private static boolean isPrime(int digit) {
		if(digit==0 || digit == 1)
			return false;
		else {
			int count = 0;
			for(int divisor = 2; divisor <= digit/2; divisor++)
				if(digit % divisor == 0) {
					count++;
					break;
				}
			if(count != 0)
				return false;
			else
				return true;

		}
	}
}
```

2. 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 characters that are randomly selected from `S1` and `S2`. The length of `S` should be equal to the sum of the lengths of `S1` and `S2`. Given that the resulting `S` may contain duplicate characters, your program should then produce a modified version of `S` called `SPrime`, in which the duplicate characters of `S` are removed, as illustrated in the sample run below.

**Sample run:**
**Enter S1: COE**
**Enter S2: 212**
**S: COO112**
**S': CO12**

**Solution:**

```
import java.util.Scanner;
import java.util.Random;
public class StringCreation {
	public static void main(String[] args) {
		String S1, S2, S="", SPrime="";
		Scanner scan = new Scanner(System.in);
		Random rnd = new Random();
		int length;
		char current;
		System.out.println("Enter S1: ");
		S1 = scan.nextLine();
		System.out.println("Enter S2: ");
		S2 = scan.nextLine();
		length = S1.length();
		for(int i=0; i<length; i++)
			S+=S1.charAt(rnd.nextInt(length));
		length = S2.length();
		for(int i=0; i<length; i++)
			S+=S2.charAt(rnd.nextInt(length));
		System.out.println("S: " + S);
		for(int i=0; i<S.length(); i++) {
			current = S.charAt(i);
			if(SPrime.indexOf(current) == -1)
				SPrime += current;
		}
		System.out.println("S': " + SPrime);
	}
}
```

3. Write a Java program that extracts email addresses 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 email addresses , numbers them, and stores them in an output file called `Emails.txt`, according to their order of appearance in the input file.
   For instance given the following content for `source.txt`:
   ```
   Hello, my name is John, my email: john@lau.edu.lb I live in Byblos.
   I use jj@hotmail.com instead of john@lau.edu.lb to make my purchases online. My friend David uses david@gmail.com for identification.
   ```
   The resulting output file `Emails.txt` would contain:
   ```
   0 - john@lau.edu.lb
   1 - jj@hotmail.com
   2 - john@lau.edu.lb
   3 - david@gmail.com
   ```

**Solution:**

```
import java.io.*;
import java.util.Scanner;
public class EmailAddressSelector{
       public static void main (String[] args) throws IOException {
            int i=0;
            String S;
            Scanner scan  = new Scanner (new File("Source.txt"));
            String file = "Emails.txt";
            FileWriter fw = new FileWriter(file);
            PrintWriter outFile = new PrintWriter(fw);
             while (scan.hasNext()) {
                   S = scan.next();
                   System.out.println(S);
                   if(S.indexOf('@') != -1 && S.indexOf('.') != -1) {
                      outFile.print(i + " - " + S);
                      outFile.println();
                      i++;
                   }
             }
                outFile.close();
         }
}
```

4. Encryption is used to secure data messages in computer networks. One possible way of encrypting text consists of rewriting the text by including first the characters found at even indexes (starting from 0) followed by the characters stored at odd indexes, while ignoring whitespace characters. For instance, the `String` "`Exam is fun`" can be encoded using the previously explained technique as "`Easfnxmiu`". To be able to recover the original message based on the encrypted version of the message, some additional information should be added to the encrypted message, namely, the number of characters found at even indexes, the number of characters found at odd indexes, and the indexes at which the white space characters existed in the original message. In this way, reconsidering our previous example, the `String` "`Exam is fun`" can be encoded using the previously explained technique as "`5 4 4 7 Easfnxmiu`". This is particularly true since the original message had `5` characters at even indexes (i.e., `Easfn`), `4` characters at odd indexes (i.e., `xmiu`) and two spaces at indexes `4` and `7`, respectively.

   When reconstructing the original message based on "`5 4 4 7 Easfnxmiu`" in this example, a new `String` is created, the portion containing the characters is extracted, and then characters are taken from both sides of that portion (the even-index characters and odd-index characters) until the index of a white space character is reached. The process is then repeated to reconstruct all of the remaining words of the original message.

   Write a Java program called `EncryptDecrypt` that reads a `String` from the user called `S` and then prints out to the screen a new `String` called `encryptedS` that represents the encrypted version of `S` as per the encryption technique described above. Then your program should use `encryptedS` to recover the original message using the decryption technique explained above and store the decrypted message in a variable called `decryptedS` that should be printed out to the screen.

**Sample run:**
**Enter S: Exam is fun**
**Encrypted message: 5 4 4 7 Easfnxmiu**
**Decrypted message: Exam is fun**

**Solution:**

```
import java.util.Scanner;
import java.util.ArrayList;
public class EncryptDecrypt {
	public static void main(String[] args) {
		String S, encryptedS="";
		int evenCharsCount = 0, oddCharsCount = 0, spaceIndex;
		String spaceIndexes = "";
		char current;
		Scanner scan = new Scanner(System.in);
		System.out.println("Enter S: ");
		S = scan.nextLine();

		// Encryption
		for(int i=0; i<S.length(); i++)
			if(S.charAt(i) == ' ')
				spaceIndexes += i + " ";
		for(int i=0; i<S.length(); i+=2) {
			current = S.charAt(i);
			if(current != ' ') {
				encryptedS += current;
				evenCharsCount++;
			}

		}
		for(int i=1; i<S.length(); i+=2) {
```

```
                current = S.charAt(i);
                if(current != ' ') {
                      encryptedS += current;
                      oddCharsCount++;
                }
          }
          encryptedS=evenCharsCount + " " + oddCharsCount + " " +
                    spaceIndexes + " " + encryptedS;
          System.out.println("Encrypted message: " + encryptedS);

          // Decryption
          Scanner encryptedSScanner = new Scanner(encryptedS);
          String decryptedS = "";
          String charsPortion = "";
          String evenCharsPortion, oddCharsPortion;
ArrayList<Integer> spaceIndexesPortion = new ArrayList<Integer>();
int nbOfEvenChars = 0, nbOfOddChars = 0, nbOfSpaces, totalNbOfChars;
          int insertionIndex = 0, evenCharIndex = 0, oddCharIndex = 0;

          if(encryptedSScanner.hasNextInt())
                nbOfEvenChars = encryptedSScanner.nextInt();
          if(encryptedSScanner.hasNextInt())
                nbOfOddChars = encryptedSScanner.nextInt();
          while(encryptedSScanner.hasNextInt())
                spaceIndexesPortion.add(encryptedSScanner.nextInt());
          nbOfSpaces = spaceIndexesPortion.size();
          if(encryptedSScanner.hasNext())
                charsPortion = encryptedSScanner.next();


          totalNbOfChars = nbOfEvenChars + nbOfOddChars + nbOfSpaces;
          evenCharsPortion = charsPortion.substring(0, nbOfEvenChars);
          oddCharsPortion = charsPortion.substring(nbOfEvenChars);
          while(insertionIndex < totalNbOfChars) {
                if(spaceIndexesPortion.indexOf(insertionIndex) != -1)
{
                      decryptedS += " ";
                } else if(insertionIndex % 2 == 0) {
                      if(evenCharIndex < evenCharsPortion.length()) {
                            decryptedS +=
                            evenCharsPortion.charAt(evenCharIndex);
                            evenCharIndex++;
                      }
                } else {
                      if(oddCharIndex < oddCharsPortion.length()) {
                            decryptedS +=
                            oddCharsPortion.charAt(oddCharIndex);
                            oddCharIndex++;
                      }
                }
                insertionIndex++;
          }

          System.out.println("Decrypted message: " + decryptedS);

    }}
```