COE 212 – Engineering Programming

Welcome to Exam II
Wednesday November 30, 2016

Instructors: Dr. Bachir Habib
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 **115 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 I:</u> Multiple choice questions (**20 minutes)** [14 points]

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

```
radar:kayak
boat:car
```

Assume that two `Scanner` objects called `fileScan` and `strScan` were declared to process the data stored in **in.txt**. `fileScan` is used to read data from **in.txt** one line at a time and `strScan` is used to decompose each obtained line into words. Specifically, we are interested in the words separated by "**:**".

1) Which of the following statements correctly instantiates the `fileScan` object?
   a. `fileScan = new File(new Scanner("in.txt"));`
   b. `fileScan = new Scanner(new File(in.txt));`
   c. `fileScan = new Scanner(new File("in"));`
   d. **None of the above**
2) Given the `fileScan` object created earlier, which of the following correctly obtains the first line from **in.txt** and then prints its length to the screen?
   a. `fileScan.nextLine().length();`
   b. `fileScan.next().length();`
   c. Both of the above
   d. **None of the above**
3) Assume that `line1` is a `String` variable holding the first line of **in.txt**. Which of the following properly sets up `strScan` to obtain the desired tokens from `line1`?
   a.
   ```
   strScan = new Scanner(line1);
   String delimiter = strScan.substring(strScan.indexOf(":"), strScan.indexOf(":")+1);
   strScan.useDelimiter(delimiter);
   ```
   b.
   ```
   strScan = new Scanner(line1);
   char delimiter = strScan.charAt(strScan.indexOf(":"));
   strScan.useDelimiter(delimiter);
   ```
   c. Both of the above
   d. **None of the above**
4) Given the `strScan` object created earlier. Which of the following correctly extracts the first word from `line1`, stores it in a `String` variable called `word1`, and then prints its second character out?
   a.
   ```
   String word1 = strScan.nextLine();
   System.out.print(word1.charAt(2));
   ```
   b.
   ```
   String word1 = strScan.nextLine();
   System.out.print(word1.charAt(1));
   ```
   c. **`String word1 = strScan.next();`**
   `System.out.print(word1.charAt(1));`
   d. Both (b) and (c)
5) Given the `word1` object created earlier. Which of the following correctly reverses its characters and stores its reversed version in a variable called `word1Reversed`?
   a.
   ```
   String word1Reversed="";
   for(int i=word1.length(); i>=0; i--)
           word1Reversed += word1.charAt(i);
   ```
   b.
   ```
   String word1Reversed="";
   for(int i=1; i<=word1.length; i++)
           word1Reversed += word1.charAt(word1.length-i);
   ```
   c. Both of the above
   d. **None of the above**
6) Given the `word1` and `word1Reversed` variables created earlier. Which of the following correctly checks whether or not they are equal without regard to the case of their letters?
   a. **`if(word1.equalsIgnoreCase(word1Reversed))`**
   b. `if(word1.compareTo(word1Reversed)==0)`
   c. Both of the above
   d. None of the above

7) Assume that `strScan` is used to properly get from `line1` the second word, which is stored in a `String` variable called `word2`. Which of the following finds the longer string between `word1` and `word2` and stores it in a `String` variable called `longer`? Assume that `longer` was declared properly.
   a. `longer=(word1.length()>word2.length())?word1.length():word2.length();`
   b. `longer=(word1.length()>word2.length())?word2.length():word1.length();`
   c. **`if(word1.length()>word2.length()) longer = word1; else longer = word2;`**
   d. Both (a) and (c)

8) Given the `longer` variable created earlier, which of the following creates a new `String` called `firstLast` that consists of the first and last characters of `longer`? Assume that `firstLast` was declared properly.
   a. `firstLast = longer.charAt(0)+longer.charAt(longer.length()-1)+ "";`
   b. **`firstLast=longer.substring(0,1)+longer.substring(longer.length()-1);`**
   c. Both of the above
   d. None of the above

9) Assume that the second line is retrieved correctly from in.txt and then stored in a variable called `line2`. Which of the following correctly counts the number of tokens separated by "`:`" in `line2`?
   a.
   ```
   int count = 0; String token;
   while(line2.length()!=0) {
          token = line2.substring(0, line2.indexOf(":"));
          count++;
          line2 = line2.substring(line2.indexOf(":")+1);}
   ```
   b. **(bold)**
   ```
   int count = 0; String token; line2 = line2+":";
   while(line2.length()!=0) {
          token = line2.substring(0, line2.indexOf(":"));
          count++;
          line2 = line2.substring(line2.indexOf(":")+1);}
   ```
   c. Both of the above
   d. None of the above

10) Which of the following should be added to the header of the `main` method using `fileScan` to prevent the occurrence of a compile-time error?
   a. `throw IOException`
   b. `Throw IOException`
   c. `throw FileNotFoundException`
   d. **None of the above**

11) Which of the following interfaces provided `fileScan` with its `hasNext` and `next` methods?
   a. `java.lang.Iterator`
   b. `java.util.Comparable`
   c. **`java.util.Iterator`**
   d. `java.lang.Comparable`

12) Which of the following methods are offered by the `java.io.PrintWriter` class?
   a. `print`
   b. `close`
   c. **Both of the above**
   d. None of the above

13) Which of the following is not part of the `java.io` package?
   a. `FileWriter`
   b. `File`
   c. `IOException`
   d. **None of the above**

14) Which of the following is false about method parameters in Java?
   a. Method parameters are passed by value
   b. **Method parameters are passed by reference**
   c. Method parameters can be referenced by name inside the body of the method
   d. Both (a) and (c)

Problem II: True or false questions (**20 minutes)** [16 points]

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

2. Assuming that `x`, and `y` are `int` variables, the following code fragment:
```
switch(x) {
case 0: y = x+1;
case 1: y = 2*x; break;
default: y = x/2;
}
```
can be rewritten as:
```
if(x == 0) { y = x+1; y = 2*x;}
else if(x == 1) { y = 2*x; }
else y = x/2;
```
Answer: **True** False

3. The following method returns `true` if `x` is divisible by `5` but not divisible by `6` and returns false otherwise.
```
public boolean method1(int x) {return (x%5 == 0 || x % 6 != 0);}
```
Answer: True **False**

4. The following method correctly returns the sum of the integer values between `1` (inclusive) and `upper` (inclusive):
```
public void method2(int upper) {
        int sum=0;
        for(int i=1; i<=upper; i++) sum+=upper;
        return sum;}
```
Answer: True **False**

5. The following method correctly finds the greatest common divisor for `num1` and `num2`:
```
public int method3(int num1, int num2) {
        int min = (num1>num2) ? num2:num1, gcd = 1;
        for(int i=min; i>=2; i--)
                if(num1%i==0 && num2%i==0) {gcd = i; break;}
        return gcd;
}
```
Answer: **True** False

6. Consider the following nested `for` loops. The body of the inner `for` loop executes `100` times.
```
for(int i=1; i<= 10; i++)
        for(int j=1; j<=11; j++)
                if(i==1) break;
```
Answer: **True** False

7. The following code fragment outputs: `002.36`
```
DecimalFormat fmt = new DecimalFormat("000.##");
double x = 2.356;
String y = fmt.format(x);
System.out.print(y + 0);
```
Answer: True **False**

8. The following method correctly computes the product of the digits that the parameter `x` is composed of. Assume that x is positive.
```
public int method2(int x) {
        int product = 1;
        do {
                product *= (x%10);
                x = x/10;
        } while(x > 0);
}
```
Answer: **True** False

9. The following code fragment executes the body of the loop `20` times.
```
int count=1;
do {   count++;
       if(count == 20) break;} while(true);
```
Answer: True **False**

10. The following code prints the even numbers between `1` and `10`
```
int counter = 1;
while(counter<=9) {
       if(counter % 2 != 0) {
              counter++;
              continue;
       }
       System.out.println(counter);
       Counter++;}
System.out.println(counter);
```
Answer: **True** False

11. The following code fragment correctly swaps the values of `num1` and `num2`:
```
num1=num1+num2;
num2=num1-num2;
num1=num1-num2;
```
Answer: **True** False

12. Consider the following method definition. `method4(-5, -10)` returns `-5`
```
public int method4(int num1, int num2) {
       boolean flag = false;
       if(num1<0) {num1*=-1; flag=true;}
       if(num2<0) {num2*=-1; flag=true;}
       while(num1!=num2)
              if(num1>num2) num1=num1-num2;
              else num2 = num2-num1;
       if(flag) return –num1;
       else return num1;
}
```
Answer: **True** False

13. A class implementing a given interface is not restricted from having methods that are not included in that interface.

Answer: **True** False

14. Consider a static method called `foo` that is defined inside a class called `Foo`. `foo` is allowed to reference the non-static variables of `Foo` once a `Foo` object has been created.

Answer: True **False**

15. The following code fragment results in a run-time error.
```
int denom=3, val=0;
for(int counter=1; counter<=3; counter++) {
       val+=counter/denom;
       denom--;
}
System.out.println(“Val: ” + val + “, counter: ” + counter);
```
Answer: True **False**

16. Consider the following code fragment. The body of the inner loop is executed `55` times.
```
for(int i=1; i<=10; i++)
       for(int j=1; j<=i; j++)
              System.out.println(j);
```
Answer: **True** False

<u>Poblem III:</u> Code analysis (**15 minutes**) [10 points]

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

```
public class ClassA {
      private int val;
      public ClassA(int a) {
            val = a;
      }
      public int getVal() {
            return val;
      }
      public boolean compare(int b) {
            return (val<b);
      }
      public static void setVal(int c) {
            val = c;}
      public String toString() {
            return val+ “”;}
}
```

```
import java.util.Scanner;
public class ClassADriver {
public static void main(String[] args){
      Scanner scan = new Scanner(System.in);
      int num;
      System.out.print(“Enter num:”);
      num = scan.nextInt();
      ClassA obj=new ClassA(num);
      System.out.print(“Enter num:”);
      num = scan.nextInt();
      if(obj.compare(num))
            ClassA.setVal(num);
      System.out.print(“Enter num:”);
      num = scan.nextInt();
      if(obj.compare(num))
            ClassA.setVal(num);
      System.out.print(obj);}}
```

Assume that if `ClassADriver` executes correctly, the user enters the following values consecutively: `10` followed by `9` and then `5`. What output is produced based on these three input values?

a. `10`
b. `9`
c. `5`
d. A run-time error occurs
e. None of the above

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

```
public class ClassB {
      int num1, num2;
      public ClassB(int a, int b) {
            num1=a; num2=b;
      }
      public int method1() {
            return num1*2;}
      public int method1(int a) {
            num1=method1();
            return num1*a;}
      public int method1(int a,int b){
            num1=method1(a);
            num2=num1*b;
            return num1+num2;}
      public String toString(){
            return num1+num2+“”;}}
```

```
public class ClassBDriver {

public static void main(String[] args){
      ClassB b1 = new ClassB(1, 2);
      b1.method1(1, 2);

      System.out.print(b1);

      ClassB b2 = new ClassB(2, 3);
      b2.method1(2, 3);

      System.out.print(b2);
}

}
```

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

a. `846`
b. `632`
c. It produces a run-time error
d. It doesn’t compile correctly
e. None of the above

<u>Problem IV:</u> Code analysis (**20 minutes**) [20 pts]

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

(1)
```
String x = "Universe";
int S = x.length();
for(int i=0; i < S%3; i++) {
      x = x.concat(x.substring(i, i+3));
}
```
Answer: x= “UniverseUniniv ”

(2)
```
String answer = "Mul"; int x = 10;
switch(answer) {
case "Add":
      x += 2;
      System.out.println(x);
      break;
case "Sub":
      x -= 2;
case "Mul":
      x *= 2;
      break;
case "Div":
      x /= 2;
default:
      x+=2;
}
```
Answer: x =20

(3)
```
int y = 20;
boolean x =(y%2 == 0 && y / 10 != 0) || (y / 5 == 0);
```
Answer: x = true

(4)
```
String str = "Event Horizon";
char x = str.charAt(str.length() -
str.substring(10).length());
```
Answer: x= ‘z’

(5)
```
double val1 = 11.2233;
double val2 = Math.floor(val1*10);
double x = val2 - (int)val1*10;
```
Answer: x= 2.0

(6)
```
String S = new String("If you judge you won’t have time to
love");
int y = 0; char c; String x = "";
do {c = S.charAt(y);
if(c== 'r' || c == 'e’)
x += c; y++;
} while(y < S.length());
```
Answer: x = “eeee”

(7)
```
int x = 2;
for(int i=1; i < 4; i = i+2)
for(int j=1; j<i; j++) x++;
```
Answer: x = 4

(8)
```
DecimalFormat fmt = new DecimalFormat("000.##");
double a = 22.111;
double b = 10;
b = a++;
String x = fmt.format(b);
x += 1;
```
Answer: x= “002.111”

(9)
```
int number=4343; int x=number, count=0;
while(x>0) {
     x=x/10; count++;}
for(int i=0; i<count/2; i++)
     number = number/10;
```
Answer: x =0

(10)
```
DecimalFormat fmt1 = new DecimalFormat("0.##");
DecimalFormat fmt2 = new DecimalFormat("0.#");
double val = Double.parseDouble(fmt1.format(11.22));
char x = fmt2.format(val).charAt(2);
```
Answer: x= ‘.’

(11)
```
int N, P; boolean x;
N=2; P=3; x = N++ > P || P++ != 3;
```
Answer: x =

(12)
```
double u = 20.21;
double v = Math.floor(u*10);
int x = (int)(Math.ceil(v) - u/100);
```
Answer: x= 201

(13)
```
String S = new String("She was sky net");
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 = “ey”

(14)
```
String S = "Help me save one more!", x = "";
for(int i=0; i < S.length(); i+=3)
x += S.charAt(i);
```
Answer: x = “Hpea eo!”

(15)
```
int n=400; int x=n, i=1;
while(x>2) {
     x=x/2; i+=2;}
for(int j=1; j<=i; j +=2)
     n= n/2;
x=n;
```
Answer: x = 0

<u>Problem V:</u> Coding (**40 minutes**) [40 points]

1. Write a program called `WordIdentification` which reads as input a sentence consisting of a string `S`, and then breaks it up into words and prints each word on a separate line, numbered following their order of appearance in the sentence.
   <u>Sample output</u>:
   Enter sentence: We were outnumbered
   Words in the sentence:
   1. We
   2. were
   3. outnumbered

```
import java.util.Scanner;

public class WordIdentification{

   public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);

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

        Scanner scanS = new Scanner(S);

        System.out.println("Words in the sentence: ");
        int count = 1;
        while(scanS.hasNext())
        {
                  System.out.println(count + ". " + scanS.next());
                  count++;
        }


   }
}
```

2. Write a program called `StringRep` which reads a string `S1` and a positive integer `n` from the user, and generates a string `S2` made of: `S1` followed by the last `n` characters of `S1` repeated `n-1` times. Note that the program has to make sure that `n` is positive and greater than 0.
   <u>Sample output</u>:
   Enter string S1: Melody
   Enter n: -3
   Wrong input! n should be greater than 0! Try again: 3
   Output string S2: Melodyodyody

```
import java.util.Scanner;

public class StringRep{

   public static void main(String[] args) {

       Scanner scan = new Scanner(System.in);

       System.out.print("Enter string: ");
       String S1 = scan.nextLine();

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

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

       String S2 = S1;

       for (int i=0; i< n-1; i++)
       {
                S2 = S2.concat(S1.substring(S1.length()-n));
       }

       System.out.print("Output string S2: " + S2);

   }
}
```

3. Write a Java program called `ClosestInt` that reads a sequence of positive integer values from the user, and then prints on-screen the number which is closest to a *target* integer number between 1 and 1000 randomly generated by the program. The program will keep on reading numbers provided by the user until it reads `-1` (sentinel value) indicating the end of user input. Note that the program should only consider the positive numbers provided by the user and disregard negative ones**.**
Sample output: (*given that the randomly generated target* = 6)
Enter sequence of numbers: 10 11 200 -3 5 50 350 -1
Number closest to randomly chosen target: 5

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

public class ClosestInt{

   public static void main(String[] args) {

       Scanner scan = new Scanner(System.in);
       Random rand = new Random();

       int target = rand.nextInt(999) + 1;

       System.out.print("Enter sequence of numbers: ");

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

       while(n != -1)
       {
          if(n>0)
                if (Math.abs(n-target) < Math.abs(nClose - target))
                     nClose = n;

           n = scan.nextInt();
       }


       System.out.print("Number closest to randomly chosen target: "
                          + nClose);

   }
}
```

4. Write a program called `SalesStats` which reads and processes a company's sales data from a file called `sales.txt`. The file contains information about the company's sales figures in various cities. Each line of the file contains a city name, followed by a colon (:), followed by the total sales revenue for that city represented as a number of type double. The program will compute and print on-screen the total sales revenue from all the cities, as well as the total number of cities.

Sample `sales.txt` file content:

```
Beirut: 195000.32
Tripoli: 280000.12
```

<u>Sample output</u>:
Total sales revenue: 475000.44
Number of cities: 2

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

public class SalesStats{

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

      File f = new File("sales.txt");
      Scanner scanF = new Scanner(f);

      double total = 0;
      int nbCities = 0;

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

               String line = scanF.nextLine();
               Scanner scanLine = new Scanner(line);
               scanLine.next();   // reading first token: the city
                          // name followed by ":" and disregarding it

               total += scanLine.nextDouble(); // reading second
                   // token: the sales value, and adding it to total

      }

      DecimalFormat fmt = new DecimalFormat("#.##");

      System.out.println("Total sales revenue:" + fmt.format(total));
      System.out.print("Number of cities: " + nbCities);

   }
}
```