COE 212 – Engineering Programming

Welcome to the Final Exam
Saturday January 10, 2014

Instructors: Dr. Bachir Habib
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 **130 minutes** to complete the **six** 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> Inheritance (**10 minutes)** [10 points]

1) Which of the following is not a superclass/subclass relationship?
    a. Book/Dictionary
    b. **Sailboat/Raceboat**
    c. Vehicle/Car
    d. None of the above
2) Which of the following keywords allows a subclass to access a superclass's methods/variables even when the superclass has declared them with private visibility?
    a. `super`
    b. `this`
    c. `base`
    d. **None of the above**
3) Which of the following visibility modifiers can be used to enable a subclass to access the data members of the superclass without violating the encapsulation principle?
    a. **protected**
    b. public
    c. All of the above
    d. None of the above
4) A method declared as final
    a. cannot be overloaded
    b. **cannot be overridden**
    c. Both of the above
    d. None of the above
5) If the constructor of a child class does not make an explicit call to a superclass's constructor, then
    a. a run-time error will occur
    b. **the default constructor of the parent class is called anyway**
    c. the child class becomes abstract
    d. None of the above
6) Which of the following methods are included in every Java class?
    a. **`public String toString()`**
    b. `public boolean equals(String s)`
    c. Both of the above
    d. None of the above
7) Of the classes below, which one is the most likely to be higher in the Inheritance hierarchy?
    a. Bat
    b. Parrot
    c. **Animal**
    d. None of the above
8) Which of the following is true?
    a. In Java, a parent class cannot have multiple children
    b. **In Java, a child class cannot be derived from multiple parent classes**
    c. Both of the above
    d. None of the above
9) Which of the following reserved words must be included in the header of a child class?
    a. extend
    b. inherit
    c. implements
    d. **None of the above**
10) Sibling classes are
    a. classes that have exactly the same set of properties
    b. classes that have the same set of children
    c. classes are related to each other via inheritance
    d. **classes that are derived from the same parent class**

<u>Problem 2:</u> True or false questions (**15 minutes)** [15 points]

1. To swap the first and last elements in an array of integers called `arr`, you would do:

```
int temp = arr[0];
arr[0] = arr[arr.length()-1];
arr[arr.length()-1]=temp;
```

Answer: True **False**

2. To check to see if the character stored in a char variable called `letter` is an alphabetic character, you could use the following:

```
if('a' <= letter <= 'z' || 'A'<= letter <= 'Z')
      System.out.println("It is an alphabetic char");
```

Answer: True **False**

3. The following Java code would produce a run time error:

```
int[] arr = {3, 4, 5, 6};
int x = arr[arr[1]];
System.out.println(x);
```

Answer: **True** False

4. In a Java program using arrays, the maximum size of an array must be known before we begin running the program.
   Answer: True **False**

5. The following code in Java prints: `0 1 1 2 3`

```
int current, prev = 0, prevprev=1;
System.out.print(prev + " " + prevprev);
for(int i=1; i<4; i++) {
      current = prev + prevprev;
      System.out.print(current + " ");
      prev = prevprev;
      prevprev=current;
}
```

Answer: True **False**

6. The following method returns the smallest value in a two dimensional array `A`.

```
public void max(int[][] A) {
      int x = A[0][0];
      for(int i=0; i<A.length; i++)
            for(int j=0;j<A[i].length;j++)
                  if(A[i][j] < x)
                        x = A[i][j];
}
```

Answer: True **False**

7. An array of double values cannot receive integer values since all array elements must have the same data type.

Answer: True **False**

8. The size of an array cannot be changed once it has been instantiated.

Answer: **True** False

9. The value of `k` returned by the following method will be equal to `3` when the array called `A` stores `{2, 2, 0, 5}`.

```
public static int foo(int[] a) {
   int k = 0;
   for (int n = 1; n < A.length; n++ ){
   if ( A[n] > A[k] )
   k = n;}
         return k;}
```

Answer: True **False**

10. Any code that can be written with multiple `if-else` statements can also be rewritten with a `switch-case` statement.

Answer: True **False**

11. A child class is allowed to define a method with the same name and the same parameter list as a method in the parent class. This technique is called method overloading.

Answer: True **False**

12. The following code gives as output a value of `3`

```
String[][] arr={{“Exam”, “is”},
                {“extremely”, “fun”}};
System.out.print(arr[1][1].length());
```

Answer: **True** False

13. The following code can be used to print the elements of an array called `arr` in reverse order, each value on a different line.

```
for(int i=0; i < arr.length; i++)
      System.out.println(arr[arr.length-1-i]);
```

Answer: **True** False

14. An array called `arr` cannot hold more than `arr.length-1` elements.

Answer: True **False**

15. The following code can be used to print the product of all the elements stored in an array called `arr`.

```
int product=1;
for(int i=1; i < arr.length; i++)
      product*=i;
System.out.print(product);
```

Answer: True **False**

Problem 3: Multiple Choice Questions (**30 minutes**) [18 points]

1) Consider the method given below. It can be best described as a method that:

```
boolean method1(int number){
      if(number == 0 || number == 0)
            return true;
      for(int i=2; i<=number/2; i++)
            if(number%i==0)
                  return true;
      return false;}
```

a. checks to see if a number is even
b. checks to see if a number is prime
c. does not compile correctly
d. checks to see if a number is not prime
e. None of the above

2) Consider the method given below. It can be best described as a method that:

```
void method2(int[] numbers) {
      int temp, index;
      int upper = numbers.length;
      for(int i=0; i < numbers.length; i++) {
            index = i;
            for(int j=i+1;j<upper;j++)
                  if(numbers[j]>numbers[index])
                        index = j;
            temp = list[index];
            list[index] = list[i];
            list[i] = temp;
      }
}
```

a. sorts the elements of an array in a descending order
b. sorts the elements of an array in an ascending order
c. does not compile correctly
d. produces a run-time error
e. None of the above

3) Consider the method given below. It can be best described as a method that:

```
boolean method3(int value, int[] numbers) {
      int k, i=0, j=numbers.length-1;
      boolean flag = false;
      while(!flag && i <= j) {
            k = (i+j)/2;
            if(numbers[k]==value)
                  flag = true;
            else if(numbers[k] < value)
                  i = k+1;
            else
                  j = k-1;}
      return flag;}
```

a. performs binary search correctly if the input array is sorted in an ascending order
b. performs binary search correctly if the input array is sorted in a descending order
c. does not compile correctly
d. produces a run-time error
e. None of the above

4) Consider the method given below. It can be best described as a method that:

```
Void method4( String aWord) {
      aWord = aWord.toLowerCase();
      for( int i=0; i< aWord.length(); i++) {
            char c = aWord.charAt( i);
            int x = c - 'a';
            if( x==0 || x==4 || x==8 || x==14 || x==20) {
                  System.out.print( (char)(x + ‘a’) + “ ”);
            }
      }
}
```

a. outputs only the consonants from the input string
b. outputs only the vowels from the input string
c. does not compile correctly
d. produces a run-time error
e. None of the above

5) Consider the method given below. If its output is:

6 9 12
8 12 16
10 15 20

What parameters were used in calling this method?

```
public void method5(int k, int n, int p, int r) {
      int i, j;
      for( j=k; j<=n; j++) {
            for( i=p; i<=r; i++) {
                  System.out.print(i*j + “  ”);
            } System.out.println();
      }
}
```

a. method5(3, 5, 1, 4);
b. method5(4, 3, 5, 4);
c. method5(3, 5, 2, 4);
d. method5(5, 3, 4, 2);
e. None of the above

6) Consider the method given below. If `aWord` has an even number of letters, the method returns a string made of:

```
String method6( String aWord) {
      char c,d;
      String word1 = "", word2 = "";
      for( int i=0; i<aWord.length()/2; i++) {
            c = aWord.charAt( i);
            d = aWord.charAt( aWord.length() - i-1);
            System.out.println(c + " " + d);
            word1 = c + word1;
            word2 = word2 + d;}
      return word1 + word2;}
```

a. The letters from `aWord` in reverse order
b. The first half of `aWord` in reverse order followed by its second half in reverse order
c. The first half of `aWord` followed by its second half in reverse order
d. The first half of `aWord` in reverse order followed by its second half
e. None of the above

Problem 4: Code analysis (**20 minutes**) [11 points]

For each of the following code fragments, what is the value of **x** after the statements are executed? (5 pts)

(1)
```
int x = 0;
int[] A = {0, -2, 7, -5, -8};
for(int i=0; i< A.length; i++)
{
   if(i%2 == 0)  {x = A[i]; A[i] = ++x;}
   else { x = ++A[i];}
 }
```
Answer: x= **-7**

(2)
```
int x = 0;
int [] T = { 1, 2, 3, 4, 5, 6 };
for ( int n = 0; n < T.length; n++)
x = x + T[n%T.length];
```
Answer: x= **21**

(3)
```
int [] A = new int[8];
for (int i = 0; i < A.length; i+=2)
A[i] = i * 2;
int x=0;
for (int i = 0; i < A.length/2; i++)
x += ++A[i];
```
Answer: x= **8**

(4)
```
int [] T = new int[10];
int x = 0;
for (int i = T.length - 1; i >= 0; i-=2 )
T[i] = i + 3; x++;
x = T[0]+T[3];
```
Answer: x= **6**

(5)
```
String x = "Green Zone";
int S = x.length();
for(int i=0; i < S/2; i++) {
      x = x.concat(x.substring(i, i+2));
}
```
Answer: x= **"Green ZoneGrreeeenn"**

For each of the following code fragments, what are the values of the array elements after the statements are executed? (6 pts)

(6)
```
int[] T = {2, 18, 8, 2, 10, 21};
for (int i = 1; i < T.length; i++) {
    T[i] = (T[i-1] % T[i])+1;
}
```

Indexes	0	1	2	3	4	5
Values	**2**	**3**	**4**	**1**	**2**	**3**

(7)
```
int[][] A = {{1, 2, 1}, {2, 3, 2}};
   for (int i = 0; i < A.length; i++)
       for(int j=0; j<A[i].length; j++)
            A[i][j] = A[1][j] + A[i][1];
```

Indexes	0	1	2
0	**4**	**5**	**7**
1	**5**	**6**	**8**

(8)
```
int[][] A = {{1, -1, 11, 111}, {2, -2, 22, 222}};
for (int i = 0; i < A.length; i++)
{     for(int j=0; j<A[i].length; j++)
      System.out.print(A[0][j]++ + " ");
                  System.out.println();}
```

Indexes	0	1	2	3
0	**3**	**1**	**13**	**113**
1	**2**	**-2**	**22**	**222**

<u>Problem 5:</u> Method definition (**10 minutes**) [6 points]

You are given below the headers of 2 methods called `count` and `shuffle`. Your job is to complete the definition of each one of these methods as per the provided guidelines.

1. **`count`** is a method that accepts 2 arrays of integers `a` and `b` of the same size, compares them element by element (element 1 from `a` with element 1 from `b`, element 2 with element 2 etc…) and returns the number of elements in `a` that are greater than the elements in `b`.

```
public int countA(int[] a, int[]b) {

        int cnt=0;
        for (int i=0;i<a.length;i++)
                if(a[i]>b[i])
                        cnt++;
        return cnt;


      }
```

2. **shuffle** is a method that accepts an array of integers as parameter, loops 52 times and in every iteration picks 2 elements at random from the array and swaps them.

```
public void shuffle(int[] a) {
        for (int i=0;i<52;i++)
        {
                int ind1=(int)(a.length*Math.random());
                int ind2=(int)(a.length*Math.random());
                int tmp=a[ind1];
                a[ind1]=a[ind2];
                a[ind2]=tmp;
        }

}
```

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

1. On your phone keypad, the alphabets are mapped to digits as follows: `ABC(2)`, `DEF(3)`, `GHI(4)`, `JKL(5)`, `MNO(6)`, `PQRS(7)`, `TUV(8)`, `WXYZ(9)`. Write a program called `PhoneKeyPad` which asks the user to enter an input `string`, then converts the string into a sequence of Keypad digits and prints the resulting digits out.
   Sample output:
   ```
   Enter a string: John
   The sequence of equivalent digits: 5646
   ```

```
import java.util.Scanner;
public class PhoneKeyPad {
   public static void main(String[] args) {

       Scanner scan = new Scanner (System.in);
       System.out.println("Enter a string");

       String S = scan.nextLine().toUpperCase();
       String S2 = "";

       for(int i=0; i< S.length(); i++)
       {
            char c = S.charAt(i);

            if(c >= 'A' && c <= 'C')  S2 += 2;
            else if (c >= 'D' && c <= 'F')  S2 += 3;
            else if (c >= 'G' && c <= 'I')  S2 += 4;
            else if (c >= 'J' && c <= 'L')  S2 += 5;
            else if (c >= 'M' && c <= 'O')  S2 += 6;
            else if (c >= 'P' && c <= 'S')  S2 += 7;
            else if (c >= 'T' && c <= 'V')  S2 += 8;
            else S2 += 9;
       }

      System.out.println("The sequence of equivalent digits: " + S2);
   }
}
```

2. Write a program called `AvgPerColumn` which reads the elements of a 10×10 two-dimensional array of integers from the user, and then produces a one-dimensional array storing the averages of the integers stored in each column of the two-dimensional array (i.e., column averages). Print out the column averages.

```
import java.util.Scanner;
public class AvgPerColumn {
   public static void main(String[] args) {

      Scanner scan = new Scanner(System.in);

      final int NbRows = 10;
      final int NbCols = 10;

      int [][] A = new int[NbRows][NbCols];

      for (int i=0; i< NbRows; i++)
      {
         System.out.println("Enter " + NbCols + " values of row n# "
                            + (i+1) + " seperated by spaces:");
         for (int j=0; j< NbCols; j++)
         {
            A[i][j] = scan.nextInt();
         }
      }

      double [] avg = new double[NbCols];
      int sum;

      for(int j=0; j< NbCols; j++)
      {
          sum = 0;
          for (int i=0; i < NbRows; i++)
          {
             sum += A[i][j];
          }

          avg[j] = (double)sum /(NbRows);
          sum = 0;
      }

      // Printing out column averages
      System.out.print("The column averages: ");
      for (int i=0; i< avg.length; i++)
      {
          System.out.print(avg[i] + "  ");
      }
   }
}
```

3. Write a program called `rowSwitching` which reads the elements of a 3 ×3 two-dimensional array of integers from the user, as well as two integer values `i1` and `i2`, and then switches (exchanges) the values in the row of index `i1` with those in the row of index `i2`. Verify that `i1` and `i2` are within the boundaries of the array to prevent run-time errors during execution.
   Sample output:
   ```
   Enter elements of first row: 1 2 3
   Enter elements of second row: 4 5 6
   Enter elements of third row: 7 8 9
   Enter i1: 0
   Enter i2: 2
   Output array:
   7 8 9
   4 5 6
   1 2 3
   ```

```
import java.util.Scanner;
public class rowSwitching {
   public static void main(String[] args) {

      Scanner scan = new Scanner(System.in);
      final int NbRows = 3;
      final int NbCols = 3;
      int [][] A = new int[NbRows][NbCols];

      for (int i=0; i< NbRows; i++)
      {
         if(i==0) System.out.print("Enter elements of first row: ");
         else if (i==2) System.out.print("Enter elements of second row :");
         else System.out.print("Enter elements of third row: ");

         for (int j=0; j< NbCols; j++)
         {
            A[i][j] = scan.nextInt();
         }
      }

      int i1, i2;
      do {
           System.out.print("Enter i1: ");
           i1 = scan.nextInt();  } while (i1 <0 || i1> NbRows-1);

      do{
           System.out.print("Enter i2: ");
           i2 = scan.nextInt();  } while (i2 <0 || i2> NbCols-1);

      int aux;
      for(int j=0; j< NbCols; j++)      // Switching rows
      {
          aux = A[i1][j];
          A[i1][j] = A[i2][j];
          A[i2][j] = aux;
      }
```

```
       System.out.println("Output array ");     // Printing array
       for (int i=0; i< NbRows; i++)
       {
           for (int j=0; j < NbCols; j++)
           {
                 System.out.print(A[i][j] + "  ");
           }
           System.out.println();
       }
    }
}
```

4. Write a program called `SumOutsideFlags` that keeps on reading integers from the user until the user enters -1. Then, your program must print the sum of all the input values excluding both -1 and all the values that are entered between an input value of 6 and an input value of 7, as illustrated in the two sample output runs given below. You can assume that every 6 will be followed by at least one 7.

Sample output1:

```
                   Ignored
Enter int values:  1 6 9 7 4 -1
Sum=5
```

Sample output2:

```
Enter int values:  1 6 7 4 10 11 6 8 5 4 7 7 -1
                   Ignored       Ignored

Sum=33
```

```
import java.util.Scanner;
public class SumOutsideFlags {
      public static void main(String[] args) {

            Scanner scan = new Scanner(System.in);

            System.out.print("Enter int values: ");

            int x=0;
            int Sum = 0;
            boolean flag = true;

            while(x != -1){

                  if (flag) Sum += x;

                  if (x==7) flag = true;

                  x = scan.nextInt();

                  if(x == 6)  flag = false;

             }

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