COE 212 – Engineering Programming

Welcome to the Final Exam
Monday January 27, 2014

Instructors: Dr. Randa Zakhour
Dr. Joe Tekli
Dr. George Sakr
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!!

Problem 1: Inheritance (**10 minutes)** [10 points]

1) Which of the following statements is **true**?
   a. Inheritance is also known as the "has-a" relationship
   b. An abstract method can be `final`
   c. The `static` modifier can be applied to an abstract method
   d. None of the above
2) Which of the following allows a subclass to access a private superclass method?
   a. `base`
   b. `this`
   c. `super`
   d. None of the above
3) A class called `Employee` is the base class of class `Volunteer`, what is a valid way to call its default constructor from inside the child's constructor?
   a. `super(); (a)`
   b. `Employee();`
   c. `base();`
   d. None of the above
4) The default `equals` implementation of class `Object` determines:
   a. whether two object reference variables are storing the same addresses
   b. whether two object reference variables refer to the same object in memory
   c. whether two object reference variables are aliases of each other
   d. All of the above
5) Which of the following is **false** about method overriding?
   a. The child's version and the parent's version of the method must have the same header
   b. The subclass cannot access the parent's version of an overridden method
   c. Both of the above
   d. None of the above
6) If a class `X` `implements` an interface `Y` that serves as a parent for an interface `Z`, then?
   a. `X` must implement each of the abstract methods defined in interfaces `Y` and `Z`
   b. `X` must implement each of the abstract methods defined in interface `Z`
   c. **`X` must implement each of the abstract methods defined in interfaces** `Y`
   d. None of the above
7) Which of the following statements is **true**?
   a. All the methods of an `abstract` class are considered to be `abstract`
   b. All the methods of an interface are considered to be `abstract`
   c. Both of the above
   d. None of the above
8) Which of the following is **false**?
   a. A subclass does not inherit the constructors of its superclass
   b. All modifications made to a subclass are automatically reflected in its superclass
   c. A final method of a superclass cannot be overridden by any of its descendants
   d. All of the above
9) Which of the following has overridden the `toString` method?
   a. `ArrayList`
   b. `String`
   c. Both of the above
   d. None of the above
10) Which of the following represents the header of the `equals` method of class `Object`?
   a. `public Boolean equals(Object o)`
   b. `public Boolean equals(String s)`
   c. `public boolean equals(String o)`
   d. None of the above

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

1. The following code in Java stores in variable `sum` the sum of the positive integers that are strictly less than 10.

```
int sum=-1;
for(int x=++sum; x<10; ++x)
      sum+=x:
```

Answer: **True** False

2. The `length()` method is used to find the length of an array, but the `length` property is used to find the number of characters in a `String`.

Answer: True **False**

3. The following code in Java can be used to print each element of the array called "`theArray`" on a different line.

```
int[] theArray={1, 3, 5, 7};
System.out.println(theArray);
```

Answer: True **False**

4. The following code in Java stores in the variable called `sum` the sum of the even numbers less than or equal to 10.

```
int sum = 4;
int[] arr = {2, 6, 8, 10};
for(int val : arr) {
      sum+=val:}
```

Answer: **True** False

5. In a Java program using arrays, the maximum size of an array must be known before we begin running the program.

Answer: True **False**

6. The following statements compile correctly but give a run-time error in execution.

```
int[] values={1, 2, 3};
int x = values[values[2]];
System.out.println(x) ;
```

Answer: **True** False

7. `protected` instance variables can only be directly accessed through code in the class where they are declared.

Answer: True **False**

8. The following code prints the words: `Exam is Fun`

```
int x = 92 ;
if(92 >= x >= 92)
      System.out.print("Exam is Fun") ;
```

Answer: True **False**

9. Two different methods in the same class cannot have the same name and the same number of parameters.

Answer: True **False**

10. A Java class can have more than one constructor.

Answer: **True** False

Problem 3: Completing code fragments (**20 minutes**) [10 points]

Below is given a class named `Circle`. A circle has a radius and a color.

```
public class Circle {
     private double radius;
     protected String color;
     public Circle()
     {
          radius =1;
          color = “red”;
     }
     public Circle(double r, String c) {
     radius=r;
     color=c;}
     public double getRadius() {return radius;}
     public double getArea()
     {return Math.PI*radius*radius;}
     public String toString() {
          return color + “ ” + radius+” “+getArea();
     }}
```

Complete the partial definition given below for a subclass called `Cylinder` derived from the `Circle` class. Note that a cylinder object also has a height (a `double` value). Complete the implementation of the constructors, `equals`, getter, `getVolume` and `toString` methods of the `Cylinder` class as noted by the comments highlighted in bold.

```
public class Cylinder extends Circle{
// instance variable representing height goes here.
private double height;

//no-parameters constructor must invoke the no
//parameters constructor of parent class and
//initialize the height to 1.
public Cylinder() {
     super();
     height = 1.0;}
//2nd constructor must invoke the 2nd constructor of
//parent class and also initialize the height variable.
public Cylinder(double r, String c, double h){
     super(r, c);
     height = h;}
// A method that returns the height.
public double getHeight(){
     return height;
}
```

```
    public boolean equals(Object other) {
    // two Cylinders are equal if they have the same
    // height, the same color, and the same radius.
         Cylinder other_cyl = (Cylinder) other;
         return
              (height == other_cyl.getHeight() &&
              getRadius() == other_cyl.getRadius() &&
              color.equals(other_cyl.color));
    }
    public double getVolume(){
    // returns the volume of the cylinder defined as
    // V=Area*height
         double  volume = getArea() * height;
         return volume;
    }
    public String toString ( ) {
    //Invokes the Circle's version of toString and then
    //concatenate the height to the returned String.

         String output = super.toString();
         return output + "\n"+ "Height: "+height;

    }
}
```

Problem 4: Evaluating Java Expressions (**10 minutes**) [10 points]

For each of the following 2 code fragments, what output is produced after the statements are executed?

(1)
```
int [] arr = new int[5];
for (int k = arr.length - 1; k >= 0; k-- )
      arr[ 4 - k ] = k + 10;
System.out.println( arr[ 0 ]+arr[2]+arr[3] );
```

Output: **37**

(2)
```
int x = 0;
int [] arr = { 5, -5, 7, 1 };
for ( int n = 0; n < arr.length; n++)
      x = x + arr[ arr.length-1 ];
System.out.println(x);
```

Output: **4**

What will the following method return if the array called `a` stores {9, 5, 2, 6, 5, 7}?

(3)
```
public 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: **2**

In what follows, fill in the values for the array called `data` after the corresponding Java code has run.

(4)
```
int[] data = new int[8];
data[0] = 3; data[7] = -18;
data[4] = 5; data[1] = data[0];
int x = data[4]; data[4] = 6;
data[x] = data[0] * data[1];
```

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Values	**3**	**3**	**0**	**0**	**6**	**9**	**0**	**-18**

(5)
```
int[] data = {2, 18, 6, -4, 5, 1};
for (int i = 0; i < data.length; i++) {
    data[i] = data[i] + (data[i] / data[0]);
}
```

Index	**0**	**1**	**2**	**3**	**4**	**5**
Values	**3**	**27**	**9**	**-6**	**7**	**1**

Problem 5: Code Analysis (**15 minutes**) [10 points]

Consider the classes *Grade* and *ManipulatingGrades* given below:

```
class Grade {
   private int numGrade;
   public Grade(int numGrade){ this.numGrade = numGrade;}
   public String toString(){ return "" + numGrade; }
   public static void Avg(int g1, int g2) {
      int avg = (g1 + g2)/2;
      System.out.println("Average of " +g1+ " and " +g2+ " = " +avg);
      g1 = avg++;
      g2 = ++avg;
    }
   public static void Avg(Grade g1, Grade g2){
      int avg = (g1.numGrade + g2.numGrade)/2;
      System.out.println("Average of " +g1+ " and " +g2+ " = " +avg);
      g1.numGrade = avg++;
      g2.numGrade = ++avg;
   }
   public static void Avg(Grade g1, int g2) {
      int avg = (g1.numGrade + g2)/2;
      System.out.println("Average of " +g1+ " and " +g2+ " = " +avg);
      g1.numGrade = avg++;
      g2 = ++avg;
   } }
```

```
class ManipulatingGrades{
   public static void main(String[] args){
   Grade g1 = new Grade(60);
   Grade g2 = new Grade(80);
   int g3 = 100, g4 = 50;
   System.out.println("g1=" +g1+ " g2=" +g2+ " g3=" +g3+ " g4=" +g4);
   Grade.Avg(g3, g4);
   System.out.println("g1=" +g1+ " g2=" +g2+ " g3=" +g3+ " g4=" +g4);
   Grade.Avg(g1, g2);
   System.out.println("g1=" +g1+ " g2=" +g2+ " g3=" +g3+ " g4=" +g4);
   Grade.Avg(g1, g3);
   System.out.println("g1=" +g1+ " g2=" +g2+ " g3=" +g3+ " g4=" +g4);
   }}
```

Provide in the box below the output that would be displayed on-screen when the class `ManipulatingGrades` is executed:

```
g1=60 g2=80 g3=100 g4=50
Average of 100 and 50 = 75
g1=60 g2=80 g3=100 g4=50
Average of 60 and 80 = 70
g1=70 g2=72 g3=100 g4=50
Average of 70 and 100 = 85
g1=85 g2=72 g3=100 g4=50
```

Problem 6: Coding (**55 minutes**) [50 points]

1. Write a JAVA program called `PerfectNumbers` that reads in **from the user** an integer `n` and then prints out all the perfect numbers between 1 (inclusive) and n (inclusive). Note that a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its divisors excluding the number itself. For example, the number 6 is perfect because 1, 2, and 3 are its proper positive divisors, and 1+2+3=6. However, the number 10 is not perfect because its proper divisors 1, 2, and 5 do not add up to 10.

   Sample output:

```
Enter an int: 6
Perfect numbers between 1 and 6: 1 6
```

```
import java.util.Scanner;
public class PerfectNumbers {
     public static void main(String[] args) {
          Scanner scan = new Scanner(System.in);
          int X, sum;
          System.out.println("Enter an int:");
          X = scan.nextInt();

          System.out.println("Perfect numbers between 1"
          + " and " + X);
          for(int i=1; i<=X; i++) {
               sum = 0;
               for(int j=1; j<i; j++)
                    if(i%j==0)
                         sum+=i;
               if(sum==i)
                    System.out.print(i + " ");
          }
     }
}
```

2. Write a JAVA program called `Bits2HT` that reads from the user a `String` of length 9 consisting only of 0s and 1s, representing the states of 9 coins (0 corresponds to Heads and 1 to Tails). Then the program should store every character of this `String` in a 1-D array. Finally the program must print the elements of the 1-D array in a table format consisting of **3 rows and 3 columns**, where a 0 is printed as H (for Heads) and a 1 is displayed as T (for Tails).

Sample output:

```
Enter a 9 character String of 0s and 1s: 100111000
The 2D String is:
T  H H
T  T T
H  H H
```

```
import java.util.Scanner;
public class Bits2HT{
     public static void main(String[] args) {
          Scanner scan = new Scanner(System.in);
          String bits;
          char[] arr = new char[9];
          System.out.println("Enter a 9 chars String of " +
               "0s and 1s: ");
          bits = scan.nextLine();

          for(int i=0; i<9; i++)
               arr[i] = bits.charAt(i);

          int counter = 1;
          for(int i=0; i<9; i++) {
               if(arr[i] == '0')
                    System.out.print("T ");
               if(arr[i] == '1')
                    System.out.print("H ");
               if(counter % 3 == 0)
                    System.out.println();
               counter++;
          }
     }
}
```

3. Consider the following mathematical series:

$$S = 1 - \left(\frac{1}{2}\right)^3 + \left(\frac{1}{3}\right)^3 - \left(\frac{1}{4}\right)^3 + \left(\frac{1}{5}\right)^3 - \cdots + (-1)^{n+1}\left(\frac{1}{n}\right)^3$$

Write a JAVA program that reads an integer `n` from the user then computes the value of `S` using the formula given above and finally outputs that value to the screen.

Sample output:

```
Enter the value of n: 2
Value of S: 0.875
```

```
import java.util.Scanner;
public class Series {
     public static void main(String[] args) {
          int n;
          Scanner scan = new Scanner(System.in);
          double S = 0.0;
          System.out.println("Enter value of n:");
          n = scan.nextInt();

          for(int i=1;i<=n;i++) {
               S+=Math.pow(-1,i+1)/Math.pow(i,3);
          }

          System.out.println("S: "+S);
     }
}
```

4. Write a program called `Interleave` that asks the user to enter two sets of 10 `int` values into two separate 1D arrays, then creates and outputs the elements of an array of 20 values that interleaves the entries of the two input arrays. See the sample output below.

Sample output:

```
Enter the first 10 values: 5  10 23 15 6 12 8 100 -1 6
Enter the second 10 values: 1 25 59 456 -30 6 10 1 2 5
The interleaved values are:
5 1 10 25 23 59 15 456 6 -30 12 6 8 10 100 1 -1 2 6 5
```

```
import java.util.Scanner;
public class Interleave {
   public static void main(String[] args) {
      int[] arr1 = new int[10];
      int[] arr2 = new int[10];
      int[] arr = new int[20];

      Scanner scan = new Scanner(System.in);
      System.out.println("Enter first 10 values:");
      for(int i=0; i<arr1.length; i++)
           arr1[i] = scan.nextInt();

      System.out.println("Enter second 10 values:");
      for(int i=0; i<arr2.length; i++)
           arr2[i] = scan.nextInt();

      int counter1 = 0, counter2 = 0;

      for(int i=0; i<arr.length; i++) {
           if(i%2 ==0) {
                arr[i] = arr1[counter1];
                counter1++;
           } else {
                arr[i] = arr2[counter2];
                counter2++;
           }
      }

      for(int i=0; i<arr.length; i++)
           System.out.print(arr[i] + " ");

   }
}
```

5. Write a Java program called `Shuffle` that creates an integer array of size 52 and loads it with successive increments of 1, i.e., the first array cell would hold a value of 1, the second cell a value of 2, and so on and so forth. Next, the program should shuffle the array. Note that shuffling is the process of randomly re-ordering the elements of an array. This can be achieved by iterating through the array one element at a time and swapping each visited element with another randomly chosen element of the array (make sure to select elements at random from higher numbered positions, i.e., positions whose indexes are greater than or equal to the current index). Finally, print the elements of the shuffled version of the array to the screen.

```
import java.util.Random;
public class Shuffle {
   public static void main(String[] args) {
        int[] arr = new int[52];
        for(int i=0; i<arr.length; i++){
             arr[i] =  i+1;
        }

        Random rnd = new Random();
        int[] arr_shuffled = new int[arr.length];
        int temp, index;

        for(int i=0; i<arr.length; i++) {
             index = rnd.nextInt(52-i) + i;
             temp = arr[i];
             arr[i] = arr[index];
             arr[index] = temp;
        }

        arr_shuffled = arr;
        for(int a : arr_shuffled)
             System.out.print(a + " ");
   }
}
```