**COE 212 – Engineering Programming**

Welcome to the Final Exam

Thursday May 30, 2013

Instructors: Dr. Randa Zakhour

Dr. Maurice Khabbaz

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 **130** **minutes** to complete the **seven** 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 **false**?
   1. A superclass is a more specific version of its subclass
   2. The class following the extends keyword in a class header is the direct child class of the class
   3. **All of the above are false**
   4. None of the above is false
2. Using the protected keyword gives a member
   1. Public access
   2. Private access
   3. **Package access**
   4. None of the above
3. Which of the following classes of methods can be called by name by subclasses?
   1. Private methods
   2. Constructors
   3. All of the above
   4. **None of the above**
4. Every class in Java extends an existing class. Which one below doesn’t extend any class?
   1. Integer
   2. object
   3. Class
   4. **None of the above**
5. Overriding a method differs from overloading a method in that
   1. Overloaded versions and original version of a method do not have the same signature
   2. Original version and overridden versions of a method have the same signature
   3. **Both of the above**
   4. None of the above
6. Which of the following statements can be used to call the default constructor of the base class?
   1. base();
   2. Super();
   3. this();
   4. **None of the above**
7. protected fields of a superclass can be accessed inside a subclass
   1. directly
   2. by calling public and protected methods declared in the superclass
   3. **Both of the above**
   4. None of the above
8. Which of the following is **true**?
   1. A superclass specifies commonalities
   2. A subclass can be modified without having to modify its superclass
   3. **All of the above are true**
   4. None of the above is true
9. The default equals implementation of class Object determines
   1. **whether two references refer to the same object in memory**
   2. whether two references have the same type
   3. whether two objects have the same instance variables
   4. None of the above
10. An abstract class differs from an interface in that
    1. The abstract class forces a derived class to implement all of its abstract methods
    2. An interface can be instantiated while an abstract class cannot
    3. Inheritance can be applied to abstract classes but not to interfaces
    4. **None of the above**

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

1. If a program attempts to access an element outside the range of the array indexes, a syntax error will occur.

Answer: True **False**

1. It is possible for a method to accept two variable-length parameter lists.

Answer: True  **False**

1. Linear search does not work properly with data that is not sorted.

Answer: True **False**

1. Consider the code shown below, where method swapValues is called. The output of this segment of code is: Values are: 3 7

Answer: True **False**

int[] numbers={3,7};

swapValues(numbers);

Sytem.out.println(“Values are: ” + numbers[0] + “ ” + numbers[1]);

//…other code

public void swapValues(int[] numbers) {

int temp = numbers[0];

numbers[1] = numbers[0];

numbers[0] = temp;

}

1. The following code in Java stores in the variable called sum the sum of the prime numbers that are less than or equal to 7.

int sum = 2;

int x;

for(x=-1; x <= 10; x+=2) {

if(x > 7)

break;

if(x >= 1)

sum += x;

}

Answer:  **True** False

1. Two different methods in Java can have the same name and the same number of parameters.

Answer: **True** False

1. The following Java code would compile and run correctly.

int[] values={1, 2, 3, 4, 5};

int x = values[values[4]];

System.out.println(x) ;

Answer: True  **False**

1. If a parameter has the same name as an instance variable, using that variable will by default refer to the instance variable not to the parameter.

Answer: True **False**

1. An abstract method cannot be defined to be final or static.

Answer: **True** False

1. A static variable’s value cannot be changed.

Answer: True **False**

**Problem 3:** Multiple Choice Questions (**20** **minutes**) [20 points]

1. What is the output of the code given below when method useArray() is called?

public void useArray() { public void processArray(char[] w)

char[] theArray = {

{‘E’,’n’,‘t’,‘r’,‘o’,‘p’,‘y’}; int x = w.length;

char c;

processArray(theArray); for(int i=0; i < x/2; i++) {

c = w[i];

for(int i=0; i<theArray.length;i++) w[i] = w[x-i-1];

System.out.print(theArray[i]); w[x-i-1]=c;

}

} }

1. **The contents of the original array in reverse order**
2. The contents of the original array in the original order
3. The contents of the original array with half of the characters reversed
4. The original array with characters rearranged so they are neither in the original nor reversed order
5. None of the above
6. Consider the program segment given below. Its output is:

int x = 1357;

int z = 0;

for(int i=x; i > 0; i=i/10)

z = z + (i%10);

System.out.print(z);

1. 7531
2. 1357
3. 156
4. **16**
5. None of the above
6. Consider the program segment given below. Its output is:

int i=1;

while(i<=50) {

System.out.print(i + “ ”);

if(i%5 == 0)

System.out.println();

i++;

}

1. All the numbers from 1 to 50, with a line break after the number 5
2. **All the numbers from 1 to 50 in a grid of 10 rows and 5 columns**
3. All the numbers from 1 to 51, 4 per row
4. All the numbers from 1 to 51, with a line break after every set of 6 numbers
5. None of the above
6. What is the output of the code below when the foo method is called as follows:

int[] values = {1, 5, 7, 8, 9, 13, 15};

foo(values);

void foo(int[] arr) {

int counter = 0;

int pass, current, temp;

for(pass=1; pass < arr.length; pass++) {

for(current=0; current<arr.length-pass;current++) {

if(arr[current]<arr[current+1]) {

temp = arr[current];

arr[current] = arr[current+1];

arr[current+1] = temp;

counter++;}

} // end for (current…

} // end for(pass…

System.out.print(counter);

} // end method foo

1. 7
2. 19
3. 20
4. **21**
5. None of the above
6. Consider the method called methodB. If its output is:

3 6 9

4 8 12

5 10 15

6 12 18

How was the method called?

void methodB(int r, int k, int n) {

for(int j=k; j<=n; j++) {

for(int i=1; i<=r; i++) {

System.out.print(i\*j + “\t”);

}

System.out.println();

}

}

1. methodB(3, 3, 1);
2. **methodB(3, 3, 6);**
3. methodB(3, 6, 3);
4. methodB(4, 6, 3);
5. None of the above

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

Fill in the values for the array called cars after the corresponding Java code has been executed. You must enter your final answers into the boxes provided.

int[] occupancy ={6, 4, 4, 5, 5};

int[] cars = new int[occupancy.length];

int people = 20;

int count = 0, index = 0;

while (count < people) {

if (occupancy[index] < (people-count)) {

cars[index] = occupancy[index];

count += occupancy[index];

index++;

}

else {

cars[index] = people-count;

count = people;

}

}

*6 4 4 5 1*

**cars**

**Answer:**

int[][] occupancy = {{2, 3}, {1, 4}, {2, 7}, {3,3}};

int[][] cars=new int[occupancy[1].length][occupancy.length];

int target = 16, temp;

boolean done = false

for(int i=0; i< occupancy.length; i++)

for(int j=0; j<occupancy[0].length && !done; j++) {

if (target > occupancy[i][j]) {

temp = occupancy[i][j];

cars[j][i] = temp;

target -= temp;

} else {

cars[j][i] = target;

done = true;

}

}

*3 4 4*

*2 1 2*

*0*

*0*

**cars**

**Answer:**

**Problem 5:** Method definition (**20** **minutes**) [9 points]

You are given below the headers of 3 methods called inArray, arraysEqual, and sortArray. Your job is to complete the definition of each one of these methods as per the provided guidelines.

1. **inArray** is a method that accepts as parameters an int[] array, a start index, an end index and a target int value. The method should return a boolean indicating whether the target value can be found in the array in the range defined by the two indices (inclusive). So for example, if the array’s content is {1, 3, 5, 6, 7, 2} and the start and end indices are 2 and 4 respectively, then if the target is 2, the method should return false.

public boolean inArray(int[] arr, int start, int end, int target) {

boolean found = false;

int index = start;

while(!found && index <= end) {

if(arr[index] == target)

found = true;

index++;

}

return found;

}

1. **arraysEqual** is a method that accepts two int[] array parameters and returns true if they contain exactly the same elements in the same order and false otherwise. For instance, your method should conclude that the arrays {1, 2, 3} and {1, 2} are not equal while the arrays of int {1, 2} and {1, 2} are.

public boolean arraysEqual(int[] arr1, int[] arr2) {

boolean equal = false;

if(arr1.length == arr2.length) {

int i=0;

for(; i<arr1.length; i++)

if(arr1[i] != arr2[i])

break;

if(i==arr1.length)

equal = true;

}

return equal;

}

1. **sortArray** is a method that accepts an int[] array parameter and sorts its elements in a **descending** order.

public void sortArray(int[] arr) {

int temp, max;

for(int index=0; index < arr.length; index++) {

max = index;

for(int scan=index+1; scan < arr.length; scan++)

if(arr[max] < arr[scan])

scan = max;

temp = list[max];

list[max] = list[index];

list[index] = temp;

}

}

**Problem 6:** Completing code fragments (**20** **minutes)** [10 points]

The abstract class GeometricObject is defined as follows:

public abstract class GeometricObject {

public abstract double area();

public abstract double perim();

}

Complete **on the next page** the definition of its child class Circle. This class represents a circle that has a radius r and a center c. Both r along with the x and y coordinates of c are stored in a 1-D array called data that serves as an instance variable for Circle. **Your job again is to complete the implementation of the code skeleton that is given on the next page.**

**//Complete the class header**

public class Circle **extends GeometricObject** {

**// Put the instance variable** data **here**

**double[] data;**

**// Constructor: instantiate the array called** data**, then insert**

**//** radius**,** x**, and** y **as its 1st, 2nd and 3rd elements**

public Circle(double radius, double x, double y) {

**data = new double[3];**

**data[0] = radius;**

**data[1] = x;**

**data[2] = y;**

}

**// Use the extra space below to override the implementation of**

**// all of the methods inherited from** GeometricObject

**public double area() {**

**return Math.PI \* data[0] \* data[0];**

}

**public double perimeter() {**

**return 2 \* Math.PI \* data[0];**

}

**// Provide getter methods for** radius, x-coordinate, and

// y-coordinate **of the center**

**public double getRadius() {**

**return data[0];**

**}**

**public double getX() {**

**return data[1];**

**}**

**public double getY() {**

**return data[2];**

**}**

public boolean equals(Object o) {

**// Two circles are equal if they have the same radius**

**// and the same center coordinates**

**Circle c = (Circle) o;**

**return (data[0] == c.getRadius() &&**

**data[1] == c.getX() &&**

**data[2] == c.getY())**

}}

**Problem 7:** Coding (**35** **minutes**) [30 points]

1. Write a JAVA program called GradingScheme that reads in **from the user** a set of student scores (each between 0 and 100). First, the program has to find the best score and store in a variable called bestScore. Second, the program will assign letter grades to the remaining scores based on the following scheme:

**Grade is A if score is >= best – 10;**

**Grade is B if score is >= best – 20;**

**Grade is C if score is >= best – 30;**

**Grade is D if score is >= best – 40;**

**Grade is F otherwise.**

**Note:** The program should first ask for the total number of students, then prompt the user to enter all of the scores **sequentially**, and finally terminate by displaying the letter grades.

**Sample output:**

Enter the number of students: 5

Enter 5 scores: 70 75 85 91 87

Student 0 scored 70; his letter grade is C

Student 1 scored 75; his letter grade is B

Student 2 scored 85; his letter grade is A

Student 3 scored 91; his letter grade is A

Student 4 scored 87; his letter grade is A

import java.util.Scanner;

public class GradingScheme {

public static void main(String[] args) {

int nbStudents;

Scanner scan = new Scanner(System.in);

System.out.println(“Enter nb of students”);

nbStudents = scan.nextInt();

double[]scores = new double[nbStudents];

double bestScore=0.0;

System.out.println(“Enter ” + nbStudents + “ scores:”);

for(int i=0; i < nbStudents; i++) {

scores[i] = scan.nextDouble();

if(scores[i] > bestScore)

bestScore = scores[i];

}

for(int i=0; i<scores.length; i++) {

if(scores[i] >= (bestScores-10))

System.out.println(“Student ” + i + “ scored ”

+ scores[i] + “; his letter grade is A”);

else if(scores[i] >= (bestScores-20))

System.out.println(“Student ” + i + “ scored ”

+ scores[i] + “; his letter grade is B”);

else if(scores[i] >= (bestScores-30))

System.out.println(“Student ” + i + “ scored ”

+ scores[i] + “; his letter grade is C”);

else if(scores[i] >= (bestScores-40))

System.out.println(“Student ” + i + “ scored ”

+ scores[i] + “; his letter grade is D”);

else

System.out.println(“Student ” + i + “ scored ”

+ scores[i] + “; his letter grade is F”);

}

}

}

1. Write a JAVA program called String2Dthat reads from the user a String of length 16. Then the program should store every character of this String in a 2-D array (4 by 4). Finally the program must print the elements of the 2-D array in a table format consisting of **4 rows and 4 columns**.

**Sample output:**

Enter a 16 character String: abeautifulexam:)

The 2D String is:

a b e a

u t i f

u l e x

a m : )

import java.util.Scanner;

public class String2D {

public static void main(String[] args) {

String line;

Scanner scan = new Scanner(System.in);

char[][] arr = new char[4][4];

do {

System.out.println(“Enter a 16 char long line:”);

line = scan.nextLine();

} while(line.length() != 16);

for(int i=0; i<4; i++)

for(int j=0; j<4; j++) {

arr[i][j] = line[i\*4+j];

System.out.print(arr[i][j] + “\t”);

}

System.out.println();

}

}

}

1. Write a JAVA program that reads an integer N from the user and creates an **N-by-N** boolean array a[][] such that a[i][j] is true if (i+1) and (j+1) have a greatest common divisor (gcd) of 1 and false otherwise. Then, the program should print the elements stored in the array a in a table format with **N rows and N columns**.

**Sample output:**

Enter the value of N: 3

The boolean values in a:

true true true

true false true

true true false

import java.util.Scanner;

public class NByNMatrix {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int N;

System.out.println(“Enter value of N:”);

N = scan.nextInt();

boolean[][] a = new boolean[N][N];

for(int i=0; i<N; i++)

for(int j=0; j<N; j++)

if(gcd(i+1,j+1) == 1)

a[i][j] = true;

else

a[i][j] = false;

for(int i=0; i<N; i++) {

for(int j=0; j<N; j++)

System.out.print(a[i][j] + “ ”);

System.out.println();

}

}

private static int gcd(int x, int y) {

while(x!=y)

if(x>y)

x = x-y;

else

y = y-x;

return x;

}

}