

## CS 101: Practice Questions for Chapter 6 and 7

### SOLUTIONS

1. Modify the class `Student` below to add a member `num` which counts the number of `Student` objects which have been instantiated. (Hint: should `num` be `static` or not?)

**Solution:**

```
public class Student{

    private String name;

    private char grade;

    private static int num = 0;

    public Student(String n, char gr){

        name = n;

        grade = gr;

        num++;

    }

    public char getGrade(){return grade;}

    public String getName(){return name;}

    public String toString(){

        return name + "\t" + grade;

    }

    public static int getNum(){return num;}
```

```
}
```

2. Write a line of code which creates a `Student` object `st` with name "Jonah Epstein" and grade C.

**Solution:**

```
Student st = new Student("Jonah Epstein", 'C');
```

3. Write a driver class using the `Student` class above in which the user enters 10 students and their grades. Then your program should print out all the students and their grades, with the A's printed out first, and then the B's, and then the C's, then the D's and finally the F's.

**Solution:**

```
import java.util.Scanner;
public class StudentDriver{
    public static void main (String[] args){
        Scanner scan = new Scanner (System.in);
        Student ar[] = new Student[10];
        for (int i=0; i<ar.length; i++){
            System.out.print("Enter a student's name: ");
            String tempName = scan.nextLine();
            System.out.print("Enter the grade: ");
            char g = scan.next().charAt(0);
            ar[i] = new Student (tempName,g);
        }
        printGrade (ar,'A');
        printGrade (ar,'B');
        printGrade (ar,'C');
        printGrade (ar,'D');
        printGrade (ar,'F');
    }
    private void printGrade(st Student[], char gr){
        for (int i=0; i<st.length; i++)
            if (st[i].getGrade() == gr)
                System.out.println(st[i]);
    }
}
```

4. Write a method

```
public static double trace (double[] [] matrix)
```

which returns the sum

$matrix[0][0] + matrix[1][1] + \dots + matrix[n-1][n-1]$ ,  
where  $n$  is the quantity `matrix.length`.

**Solution:**

```
public static double trace (double[] [] matrix){
    double sum = 0;
    for (int i=0; i<matrix.length; i++)
        sum += matrix[i][i];
    return sum;
}
```

5. Write a method

```
public static int maxVal(int[] ar)
```

which returns the maximum value of all the integers stored in the array parameter `ar`. (You may find it useful to use `Integer.MIN_VALUE`, which is the smallest integer value in Java.)

**Solution:**

```
public static int maxVal(int[] ar){
    int max = Integer.MIN_VALUE;
    for (int val : ar)
        if (val>max)
            max = val;
    return max;
}
```

6. Write a method

```
public static void switchVals(double[] ar)
```

which switches the initial and last elements of the array `ar`. So for example, if `ar` initially contains the values

```
7.0 4.2 7.9 13.4 15.9 10.3
```

then after the call to the `switchVals` method, the array will contain

```
10.3 4.2 7.9 13.4 15.9 7.0
```

**Solution:**

```
public static void switchVals(double[] ar){
    double temp = ar[0];
    ar[0] = ar[ar.length - 1];
    ar[ar.length - 1] = temp;
}
```

7. Write a class `NVec` which implements the mathematics of  $n$ -dimensional vectors. An  $n$ -dimensional vector is given by an array of  $n$  double numbers  $(x_0, \dots, x_{n-1})$ . The class should implement the following public methods:

```
NVec(double[] ar)
double dot (NVec x) // returns the dot product of this and x
double norm ()      // returns the norm of this
double angle (NVec x) // returns the angle between this and x
String toString()
```

The dot product

$$(x_0, x_1, \dots, x_{n-1}) \cdot (y_0, y_1, \dots, y_{n-1}) = x_0y_0 + x_1y_1 + \dots + x_{n-1}y_{n-1}$$

as long as the two vectors have the same dimension  $n$ . (If their dimensions are different, print an error message to the screen.)

The norm of a vector  $x$  is defined by

$$\text{norm}(x) = \|x\| = \sqrt{x \cdot x}.$$

The angle between two vectors  $x$  and  $y$  is given by

$$\text{angle}(x, y) = \arccos\left(\frac{x \cdot y}{\|x\| \|y\|}\right).$$

(Hint: the method `Math.acos` computes the arccos function).

**Solution:**

```
public class NVec{
    private double[] coordinates;
    public NVec(double[] ar){
        coordinates = new double[ar.length];
        for (int i=0; i<ar.length; i++)
            coordinates[i] = ar[i];
    }
    public double dot (NVec x){
        if (this.coordinates.length != x.coordinates.length){
            System.out.println("The two vectors must have the same"
                + " dimension to take their dot"
                + " product.");
            return 0;
        }
        else{
            double sum = 0;
            for (int i=0; i<this.coordinates.length; i++)
                sum += this.coordinates[i] * x.coordinates[i];
            return sum;
        }
    }
    public double norm(){
        return Math.sqrt(this.dot(this));
    }
    public double angle(NVec x){
        double d = this.dot(x);
        double n1 = this.norm();
        double n2 = x.norm();
        return Math.acos(d/(n1*n2));
    }
}
```

8. Consider the class

```
public class PlanePoint{
    private double x;
    private double y;
    public PlanePoint(double first, double second){
        x = first;
        y = second;
    }
    public void setX(double n){
        x = n;
    }
    public void setY(double n){
        y = n;
    }
    public double getX(){
        return x;
    }
    public double getY(){
        return y;
    }
    public String toString(){
        return "(" + x + "," + y + ")";
    }
}
```

What is the output of the the following driver class?

```
public class PlaneTester{
    public static void main(String[] args){
        PlanePoint a = new PlanePoint(8.3,2.4);
        PlanePoint b = new PlanePoint(1.2,3.6);
        PlanePoint c = new PlanePoint(1.7,13);
        System.out.println(a+" "+b+" "+c);
        change(a,b,c);
        System.out.println(a+" "+b+" "+c);
        change(b,c,a);
        System.out.println(a+" "+b+" "+c);
    }
}
```

```

    }
    public static void change(PlanePoint u, PlanePoint v,
                              PlanePoint w){
        u = new PlanePoint(3.9,17);
        v.setX(1.9);
        w = u;
    }
}

```

**Solution:**

```

(8.3,2.4) (1.2,3.6) (1.7,13.0)
(8.3,2.4) (1.9,3.6) (1.7,13.0)
(8.3,2.4) (1.9,3.6) (1.9,13.0)

```

9. Consider a standard die which, upon being rolled, produces a random integer from 1 to 6. So if a pair of dice are rolled and the integers are summed, the value will be between 2 and 12. Write a program which rolls a pair of dice 100 times, and stores how many times each outcome (from 2 to 12) occurs. Then print a histogram listing all the outcomes from 2 to 12, followed by a number of asterisks representing how many times this outcome was rolled on the dice.

Here is a sample output:

```

2 **
3 *****
4 *****
5 *****
6 *****
7 *****
8 *****
9 *****
10 *****
11 *****
12 ****

```

**Solution:**

```

public class RollDice{
    public static void main(String[] args){
        int[] rollOutcomes = new int[13];
        for (int i=0; i<100; i++){
            rollOutcomes[rollPair()]++;
        }
        for (int j=2; j<=12; j++){
            System.out.print(j+" ");
            printAsterisks(rollOutcomes[j]);
            System.out.println();
        }
    }
    public static int rollPair(){
        int d1 = (int)(Math.random()*6) + 1;
        int d2 = (int)(Math.random()*6) + 1;
        return d1+d2;
    }
    public static void printAsterisks(int n){
        for (int i=0; i<n; i++){
            System.out.print('*');
        }
    }
}

```

10. What is the output?

```

public class X{
    public static void main (String[] args){
        int[] a = {2,5,7,3,10,15,13,12};
        for (int b : a)
            if (b%2 == 0)
                System.out.println(b);
    }
}

```

**Solution:**

```

2
10
12

```