



Autor: Profesores EDA

Estructura de Datos y Algoritmos

Tema 1 – Tipo Abstracto de Datos (TAD)

Problema 1 – Sea un polinomio:

$$Q(x)=a_0+a_1x+a_2x^2+a_3x^3+\dots+a_nx^n$$

La siguiente interfaz muestra la especificación formal del tipo de datos abstractos (TAD) de un polinomio

```
public interface iPolynomial {  
    public int getDegree();  
    public int getCoefficient(int n);  
    public void setCoefficient(int n, int newValue);  
    public int getValue(int x);  
    public iPolynomial suma(iPolynomial p);  
}
```

Crea una clase, Polynomial, que represente el TAD del polinomio. Se pueden seguir los siguientes pasos:

- Crea los atributos que consideres necesarios para representar el polinomio y escribe un método constructor que permita inicializar los coeficientes del polinomio (dichos valores se reciben como argumento del constructor)
- Implementa el método getDegree(), que devuelve el grado del polinomio

Por ejemplo, $Q(x)=5$ tiene grado 0. $Q(x)=x^2+5$ tiene grado 2.

- Implementa los métodos getCoefficient(int n) (devuelve el coeficiente del término de grado n) y el método setCoefficient(int n, int newValue) (modifica el coeficiente del término de grado n).

Por ejemplo, dado el polinomio $Q(x)=a_0+a_1x+a_2x^2+a_3x^3+\dots+a_nx^n$, getCoefficient(3) returns a_3 .

Por ejemplo, dado el polinomio $Q(x)=a_0+a_1x+a_2x^2+a_3x^3+\dots+a_nx^n$, this call `setCoefficient(3,b)` does that now $Q(x)$ is $a_0+a_1x+a_2x^2+bx^3+\dots+a_nx^n$

d) Implementa el método `CalculateValue(int x)`, que recibe un valor entero y calcular el valor del polinomio para dicho valor.

Por ejemplo, $Q(3)=a_0+a_13+a_29+a_327+\dots+a_n3^n$

e) Implementa el método `sum(IPolynomial p)` que devuelve la suma del polinomio objeto y el polinomio p.

Por ejemplo, $Q(x)=3x^2+4x+5$, $p(x)=x^3+4x^2-2x-3$
 $Q.sum(p) \rightarrow x^3+7x^2+2x+2$

Nota: Puedes utilizar el método `Math.power(a,b)` para calcular el valor de un número a elevado a una potencia b. También puedes usar la clase `ath.max(x,y)` para calcular el máximo de dos números x e y.

Solución:

```
package unit1;

public class Polynomial implements IPolynomial {

    int[] coef;

    public Polynomial(int[] coefficients) {
        coef=coefficients;
    }

    public int getDegree() {
        for (int i=coef.length-1;i>=0;i--) {
            if (coef[i]!=0) return i;
        }
        return 0;
    }

    public int getCoefficient(int n) {
        if (n>=coef.length || n<0) {
            System.out.println(n + "is out of range");
            return -1;
        }
        return coef[n];
    }

    public void setCoefficient(int n, int newValue) {
        if (n>=coef.length || n<0) {
            System.out.println(n + "is out of range");
            return;
        }
    }
}
```

```

        coef[n]=newValue;
    }

    public int getValue(int x) {
        int result=0;
        for (int i=0;i<coef.length;i++) {
            result+=coef[i]*(int)Math.pow(x, i);
        }
        return result;
    }

    public iPolynomial suma(iPolynomial p) {

        int maxDegree=Math.max(p.getDegree(), this.getDegree());
        int minDegree=Math.min(p.getDegree(), this.getDegree());

        int[] sumPoly=new int[maxDegree];

        if (p.getDegree()<this.getDegree()) {
            for (int i=minDegree; i<this.getDegree(); i++) {
                sumPoly[i]=this.getCoefficient(i);
            }
        } else if (p.getDegree()>this.getDegree()) {
            for (int i=minDegree; i<p.getDegree(); i++) {
                sumPoly[i]=p.getCoefficient(i);
            }
        }

        for (int i=0; i<minDegree; i++) {
            sumPoly[i]=p.getCoefficient(i)+this.getCoefficient(i);
        }

        iPolynomial newPol=new Polynomial(sumPoly);
        return newPol;
    }

    public void show() {
        for (int i=0; i<=this.getDegree(); i++) {
            System.out.println("Coefficient for grade " + i + ":"
+ this.coef[i]);
        }
    }
}

```

Problema 2 – Dada la siguiente especificación formal (interfaz) que define una muestra de números:

```
//Interface for a sample of numbers
public interface iSample {
    //return the maximum number in the sample
    public double getMax();
    //return the minimum number in the sample
    public double getMin();
    //return the sum of all the numbers in the sample
    double sum();
    //return the mean of the sample
    double mean();
    //return the mean of the sample
    double median();
}
```

Implementar una clase, Sample, que implemente este TAD.

Solución

```
package unit1;

import java.util.Arrays;

public class Sample implements iSample {
    private int size;
    private double m[];

    /**
     * creates a sample with random values
     * @param s size of the sample
     */
    public Sample(int s) {
        size = s;
        m = new double[size];
        for (int i = 0; i < size; i++) {
            m[i] = Math.random();
        }
    }

    public void show() {
        for (int i = 0; i < size; i++) {
            System.out.print(m[i] + " ");
        }
        System.out.println();
    }
}
```

```

/**
 * return the maximum number in the sample
 */
public double getMax() {
    double max = m[0];
    for (int i = 1; i < m.length; i++) {
        if (m[i] > max)
            max = m[i];
    }
    return max;
}

/**
 * return the minimum number in the sample
 */
public double getMin() {
    double min = m[0];
    for (int i = 1; i < m.length; i++) {
        if (m[i] < min)
            min = m[i];
    }
    return min;
}

/** return the sum of all the numbers of the sample*/
public double sum() {
    double total = 0;
    for (int i = 0; i < m.length; i++) {
        total = total + m[i];
    }
    return total;
}

/** return the mean of the sample */
@Override
public double mean() {
    return sum()/size;
}

//return the median of the sample
public double median() {
    //sort the array
    Arrays.sort(m);
    //return the central element of the sample
    return m[m.length / 2];
}
}

```