Tarea 06: Herencia

Programación Avanzada

2023-1

Valeria Jimeno Villegas - vjimenov@ciencias.unam.mx

26 de septiembre de 2022

Ejercicio 6.1. Triángulos

- Construye las clases `Triangulo`, `Equilatero`, `Isosceles` y `Escaleno` de tal manera que podamos construir cada tipo de triangulo como se muestra en la clase `Test`.

  Hints

- `Triangulo` es super clase y contiene los métodos necesarios para construir cualquier tipo de triángulo (de los mencionados).
- Todos los triángulos tienen 3 lados, esto es una generalidad.
- Usa el constructor adecuado en cada clase derivada para construir el objeto.

```
import java.lang.Math;

public class Triangulo{
    // Definimos tres variables de tipo float protegidas para que solo
    // las clases que hereden de triangulo puedan acceder a ellas
    protected float lado1, lado2, lado3;

    // Se definen los tres constructores de acuerdo a las firmas que van
    // a necesitar las subclases
    public Triangulo(float lado1,float lado2,float lado3){
        this.lado1 = lado1;
        this.lado2 = lado2;
        this.lado3 = lado3;
    }
    //  definimos que cada uno de los lados va a estar dado por
```

```
    // el paramatro que se define como el mismo lado para
    // construir el triangulo equilatero
    public Triangulo(float lado1){
        this.lado1 = lado1;
        this.lado2 = lado1;
        this.lado3 = lado1;
    }

    public Triangulo(float lado1,float lado2){
        this.lado1 = lado1;
        this.lado2 = lado2;
        this.lado3 = lado2;
    }
    // método que regresa el area del traingulo usando la función
    // de heron
    protected double calcularArea(){
        // formula de heron
    float s = (lado1+lado2+lado3)/2;
    double area = Math.sqrt(s*(s-lado1)*(s-lado2)*(s-lado3));
    return area;
    }
    // metodo que regresa el perimetro del triangulo
    protected float calcularPerimetro(){
        return lado1 + lado2 + lado3;
    }
    // metodo que no regresa nada, sin embargo manda
    // imprimir en consola la dimension de cada uno
    // de los lados
    protected void imprimirLados(){
        System.out.printf("[%.2f, %.2f, %.2f]",lado1,lado2,lado3);
    }
}
```

```
// Clase Equilatero que extiende de Triangulo
public class Equilatero extends Triangulo{
    // Constructor que recibe un solo parametro y
    // manda llamar el constructor de super
    public Equilatero(float lado1){
        super(lado1);
```

```
    }
}
```

```
public class Escaleno extends Triangulo{

    public Escaleno(float lado1,float lado2,float lado3){
        super(lado1, lado2, lado3);
    }
}
```

```
public class Isosceles extends Triangulo{

    public Isosceles(float lado1,float lado2){
        super(lado1,lado2);
    }
}
```

Ejercicio 6.2. Matriz

```
public class Matriz{
    // Definimos la variable n para usarla en el constructor
    protected int n;
    // definimos a matrix como un arreglo 2D
    protected int[][] matrix;
    // la variable k va a funcionar para darle el valor
    // a las entradas de la matriz, se inicializa en 0
    protected int k = 0;

    // Método constructor el cual solo reciben como entrada
    public Matriz(int n){
        this.n = n;
        matrix = new int[n][n];

        // recorremos por i y j dándole el valor de k y
        // aumentando este valor una unidad cada iteración
        for(int i = 0; i < n ; i++) {
            for(int j = 0; j < n; j++){
```

```
                matrix[i][j] = k;
                k++;
            }
        }
    }
    // método que no regresa nada pero manda a imprimir en consola
    // el valor de la matriz
    public void imprimirMatriz(){

        for(int i = 0; i < n ; i++) {
            for(int j = 0; j < n; j++){
            System.out.print(matrix[i][j]+ "\t");
            }
            System.out.println();
        }
    }
}
```

```
// Para poder pedir al usuario que introduzca un input
import java.util.Scanner;

public class TestMatriz{
    public static void main(String[] args){

        // Creamos un Scanner para pedirle al usuario el
        // valor de n para introducirlo como parámetro
        // al momento de llamar al método constructor
        Scanner input = new Scanner(System.in);
        System.out.println("Introduce el valor de N:");
        Integer n = input.nextInt();

        Matriz mat = new Matriz(n);
        System.out.println("\nEsta es la matriz resultante: \n");
        // Se manda llamar al método imprimirMatriz de la
        // clase Matriz
        mat.imprimirMatriz();
    }

}
```

```java
// Esta clase hereda de Matriz, de modo que el método
// constructor manda llamar al método super
public class MatrizExtendida extends Matriz{

    public MatrizExtendida(int n){
        super(n);
    }
    // método que cambia el valor de las entradas i,j
    // como el indexado de la matriz comienza en 0, entonces
    // se toma en cuenta que si se quiere cambiar la entrada
    // (1,1) entonces esta se refiere a la entrada (0,0), po
    // lo que se le resta una unidad a cada entrada i, j.
    public void sustituirElemento(int i, int j, int valor){
        matrix[i-1][j-1] = valor;
    }

}
```

```java
import java.util.Scanner;

public class TestMatrizExtendida{

        public static void main(String[] args){

        // Se crea un objeto Scanner y se pide al usuario
        // introducir el valor de n, i,j y value(valor por
        // el que se quiere cambiar la entrada i,j)
        Scanner input = new Scanner(System.in);
        System.out.println("Introduce el valor de N:");
        Integer n = input.nextInt();

        System.out.println("Introduce el valor de i:");
        Integer i = input.nextInt();

        System.out.println("Introduce el valor de j:");
        Integer j = input.nextInt();

        System.out.println("Introduce el nuevo valor para [i][j]:");
        Integer value = input.nextInt();
```

```
        //Una vez teniendo los inputs anteriores creamos la matriz
        // y mandamos llamar al método  .sustituirElemento(i,j,value)
        MatrizExtendida mat = new MatrizExtendida(n);
        mat.sustituirElemento(i,j,value);

        // Se imprimir la matriz con el cambio de entrada (i,j)
        System.out.println("\nEsta es la matriz con el cambio:  \n");
        mat.imprimirMatriz();
    }
}
```