Tarea 6
Programación avanzada

Emmanuel Peto Gutiérrez

24 de septiembre de 2022

1. Triángulos

1.1. Clase Triangulo

Se puede crear una instancia de la clase `Triangulo` donde, por defecto, todos sus lados son 0. Como sus lados son 0 también el perimetro y el área lo son. Observe que el método para calcular el perímetro devuelve la suma de los 3 lados.

Los lados del triángulo se guardan en un arreglo de tamaño 3.

```
public class Triangulo{
    protected double[] lados;

    public Triangulo(){
        lados = new double[3];
    }

    public double calcularPerimetro(){
        double suma = lados[0]+lados[1]+lados[2];
        return suma;
    }

    public double calcularArea(){
        return 0.0;
    }

    public void imprimirLados(){
        System.out.println("Lados "+Arrays.toString(lados));
    }
}
```

1.2. Clase Equilatero

El constructor de la clase recibe un double que representa la longitud de un lado del triángulo. Como es equilátero se guarda ese mismo valor para los 3 lados.

Para calcular el área de un triángulo equilátero se usó la siguiente fórmula:

$$area = \frac{b \times h}{2}$$

donde b es la base y h la altura. La altura se calcula de la siguiente manera:

$$h = \frac{\sqrt{3}}{2}\ell$$

donde ℓ es el lado del triángulo. El método `calcularArea` se tiene que redefinir para que devuelva el valor correcto, pues en la clase padre solo devuelve 0. Nótese que el método de perímetro no se tiene que redefinir, pues para cualquier triángulo es la suma de sus 3 lados.

```
public class Equilatero extends Triangulo {

    public Equilatero(double lado){
        lados[0] = lado;
        lados[1] = lado;
        lados[2] = lado;
    }

    @Override
    public double calcularArea(){
        double altura = (Math.sqrt(3)/2) * lados[0];
        return (lados[0]*altura)/2;
    }
}
```

1.3. Clase Isosceles

El constructor de la clase `Isosceles` recibe dos lados, donde el primero será la base y el segundo se repite para los otros dos lados. También se redefine el método `calcularArea`. El área también se calcula con $b * h/2$ y la altura se calcula con la siguiente fórmula:

$$h = \sqrt{\ell^2 - b^2/4}$$

donde ℓ es el lado que se repite y b es la base.

```
public class Isosceles extends Triangulo {
    public Isosceles(double lado1, double lado2){
        lados[0] = lado1;
```

```
        lados[1] = lado2;
        lados[2] = lado2;
    }

    @Override
    public double calcularArea(){
        double term1 = lados[1]*lados[1];
        double term2 = (lados[0]*lados[0])/4;
        double altura = Math.sqrt(term1-term2);
        return (lados[0]*altura)/2;
    }
}
```

1.4. Clase Escaleno

El constructor de la clase `Escaleno` recibe 3 lados y cada uno se guarda en una casilla diferente del arreglo de lados. En este caso para calcular el área se usa la fórmula de Herón. Si a, b y c son los lados del triángulo, sea sp el semiperímetro definido de la siguiente manera:

$$sp = \frac{a+b+c}{2}$$

entonces el área del triángulo es:

$$area = \sqrt{sp(sp-a)(sp-b)(sp-c)}$$

```
public class Escaleno extends Triangulo {

    public Escaleno(double lado1, double lado2, double lado3){
        lados[0] = lado1;
        lados[1] = lado2;
        lados[2] = lado3;
    }

    @Override
    public double calcularArea(){
        double sp = calcularPerimetro()/2;
        double producto = sp * (sp-lados[0]) * (sp-lados[1]) * (sp-lados[2]);
        return Math.sqrt(producto);
    }
}
```

1.5. TestTriangulo

En esta clase se prueba el funcionamiento de los tres tipos de triángulos.

```
public class TestTriangulo {
    public static void main(String[] args){
        Equilatero equi = new Equilatero(2);
        double area = equi.calcularArea();
        double peri = equi.calcularPerimetro();
        System.out.println("--- EQUILATERO ---");
        System.out.println("area = "+area);
        System.out.println("perimetro = "+peri);
        equi.imprimirLados();

        Isosceles iso = new Isosceles(2, 5);
        area = iso.calcularArea();
        peri = iso.calcularPerimetro();
        System.out.println("--- ISOSCELES ---");
        System.out.println("area = "+area);
        System.out.println("perimetro = "+peri);
        iso.imprimirLados();

        Escaleno esca = new Escaleno(3, 4, 5);
        area = esca.calcularArea();
        peri = esca.calcularPerimetro();
        System.out.println("--- ESCALENO ---");
        System.out.println("area = "+area);
        System.out.println("perimetro = "+peri);
        esca.imprimirLados();
    }
}
```

2. Matrices

2.1. Clase Matriz

```
public class Matriz{
    protected int[][] mat;

    public Matriz(int n){
        mat = new int[n][n];
        int k = 0;
        for(int i=0; i<n; i++){
            for(int j=0; j<n; j++){
                mat[i][j] = k;
                k++;
            }
        }
    }
```

```
    private int numDigs(int n){
        if(n == 0){
            return 1;
        }
        int d = 0;
        while(n > 0){
            n /= 10;
            d++;
        }
        return d;
    }

    public void imprimirMatriz(){
        int maxDig = 0;
        int n = mat.length;
        for(int i=0; i<n; i++){
            for(int j=0; j<n; j++){
                int nd = numDigs(mat[i][j]);
                if(nd > maxDig){
                    maxDig = nd;
                }
            }
        }

        for(int i=0; i<n; i++){
            System.out.print("|");
            for(int j=0; j<n; j++){
                String espacios = "␣";
                int nEsp = maxDig - numDigs(mat[i][j]);
                for(int k=0; k<nEsp; k++){
                    espacios += "␣";
                }
                System.out.print(espacios+mat[i][j]);
            }
            System.out.println("␣|");
        }
    }
}
```

2.2. Clase TestMatriz

```
public class TestMatriz {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
```

```
        System.out.println("Ingrese el lado de la matriz.");
        int n = sc.nextInt();
        sc.close();
        Matriz m1 = new Matriz(n);
        System.out.println("Matriz construida:");
        m1.imprimirMatriz();
    }
}
```

2.3. Clase MatrizExtendida

```
public class MatrizExtendida extends Matriz {

    public MatrizExtendida(int n){
        super(n);
    }

    public void sustituirElemento(int i, int j, int valor){
        mat[i][j] = valor;
    }
}
```

2.4. Clase TestMatrizExtendida

```
public class TestMatrizExtendida {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Ingrese el lado de la matriz.");
        int n = sc.nextInt();
        MatrizExtendida m1 = new MatrizExtendida(n);
        System.out.println("Ingrese la fila a modificar.");
        int f = sc.nextInt();
        System.out.println("Ingrese la columna a modificar.");
        int c = sc.nextInt();
        System.out.println("Ingrese el valor nuevo en esa posicion.");
        int val = sc.nextInt();
        m1.sustituirElemento(f, c, val);
        System.out.println("Matriz construida:");
        sc.close();
        m1.imprimirMatriz();
    }
}
```