Instituto Artek
Ingeniería en Inteligencia Artificial
Programación Orientada a Objetos
Profesor: Irving Rocha Reséndiz

Práctica 2 – Concepto de Instancia

INA0722
Polo Lucy Luis Alejandro

12 de febrero de 2023
Ciudad de México

Introducción

La programación orientada a objetos (OOP) es un paradigma de programación que permite modelar sistemas y problemas en términos de objetos. Los objetos son instancias de una clase, lo que significa que tienen una estructura y comportamiento definidos por dicha clase. En el lenguaje de programación Java, una instancia de una clase se crea cuando se ejecuta un constructor de la clase. Una instancia de una clase contiene los datos y parámetros que se definieron en la clase. Estos datos y parámetros se pueden acceder y modificar a través de los métodos de instancia. En Java, los métodos de instancia son aquellos que se definen dentro de la clase y no se declaran estáticos. El concepto de instancia es fundamental para entender la programación orientada a objetos en Java y es la base para crear programas robustos, reutilizables y extensibles.

El uso de las instancias en la programación orientada a objetos usando Java ofrece una serie de beneficios, incluyendo un mejor diseño, una mayor facilidad de mantenimiento, un mayor control de los datos y una mayor reutilización de código. El concepto de instancia también se puede usar para crear sistemas dinámicos y flexibles, ya que se pueden crear y destruir instancias en cualquier momento durante la ejecución de un programa. Esto permite al programador crear aplicaciones extensibles que pueden cambiar y evolucionar según las necesidades del usuario. Además, el uso de instancias permite la encapsulación de datos, lo que permite al programador ocultar partes del código para evitar que los usuarios los modifiquen o los vean. Esto permite un mayor control sobre la seguridad y el acceso a los datos.

Objetivos

Los objetivos de aprender a usar el concepto de instancia dentro de la programación orientada a objetos usando java incluyen:

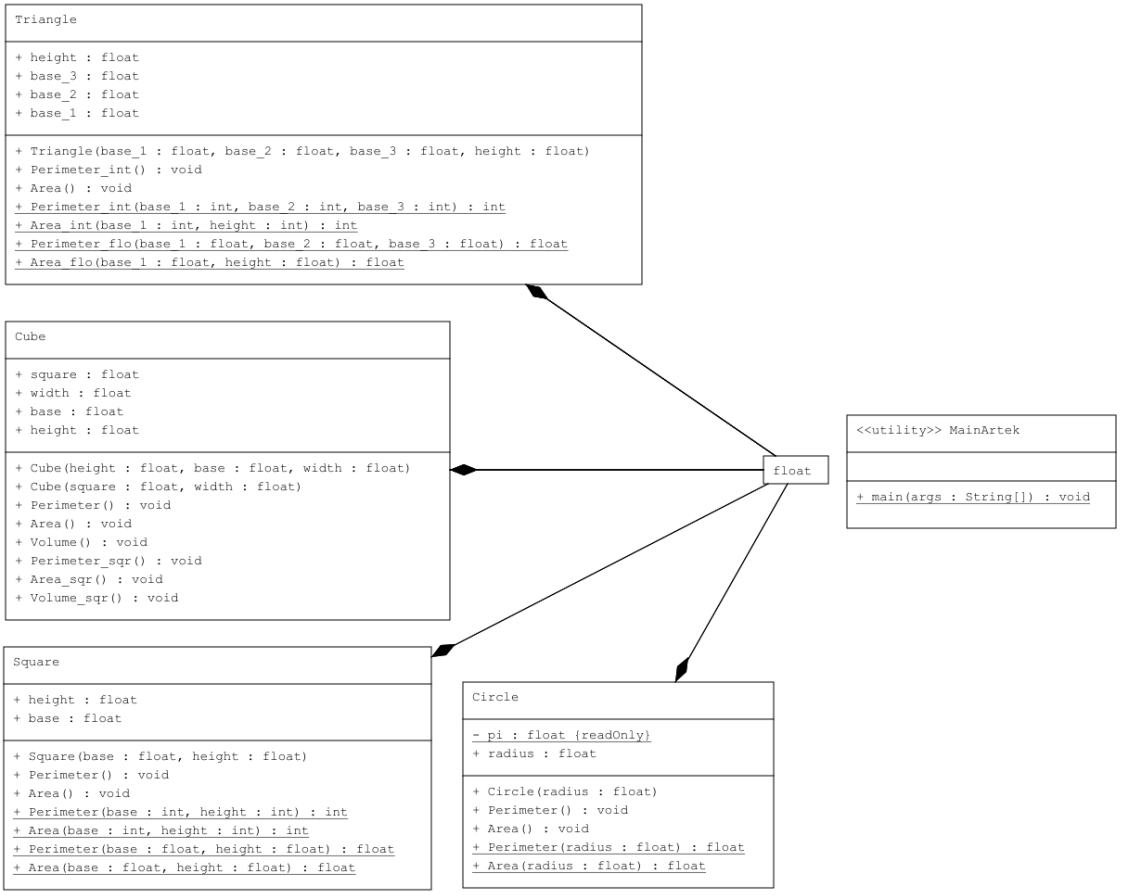
- Comprender cómo se crean instancias de una clase y cómo estas instancias pueden acceder y modificar los datos y parámetros de una clase.

- Aprender a utilizar los métodos de instancia para manipular los datos de una instancia.

- Entender cómo el concepto de instancia se puede utilizar para crear programas reutilizables, robustos y extensibles.

Desarrollo

Diagrama UML



Código

Clase MainArtek.java

```
public class MainArtek {
    public static void main(String[] args)
    {
    //Clase Cuadrado
        //Constructor Square
        Square square = new Square(2.5f,7.13f);
        //Métodos no estáticos
        square.Perimeter();
        square.Area();
        //Métodos estáticos
        int per_int_squ = Square.Perimeter(4,4);
        int are_int_squ = Square.Area(7,10);
        float per_flo_squ = Square.Perimeter(2.5f,7.13f);
        float are_flo_squ = Square.Area(7.8f,9.1f);

    //Clase Círculo
        //Constructor Circle
        Circle circle = new Circle(3f);
        //Métodos no estáticos
        circle.Perimeter();
        circle.Area();
        //Métodos estáticos
        float per_cir = Circle.Perimeter(3f);
        float are_cir = Circle.Area(3f);

    //Clase Triángulo
        //Constructor Triangle
        Triangle triangle = new Triangle(2.5f,3.5f,1.5f,5.1f);
        //Métodos no estáticos
```

```
        triangle.Area();
        triangle.Perimeter_int();
        //Métodas estáticos
        int per_int_tri = Triangle.Perimeter_int(1,2,3);
        int are_int_tri = Triangle.Area_int(3,4);
        float per_flo_tri = Triangle.Perimeter_flo(1.5f,2.3f,3.2f);
        float are_flo_tri = Triangle.Area_flo(1.5f,2.3f);

    //Clase Cubo
        //Constructor (Cube - height, base, width)
        Cube cube = new Cube(3f,2f,3.1f);
        //Métodos de instancia
        cube.Perimeter();
        cube.Area();
        cube.Volume();
        //Constructor (Cube - square y width)
        Cube cube_sqr = new Cube(3f,2f);
        //Métodos de instancia
        cube_sqr.Perimeter_sqr();
        cube_sqr.Area_sqr();
        cube_sqr.Volume_sqr();
    }
}
```

Clase Square.java

```
public class Square
{
    //Variables
    public
    float base,height;
    //Constructor
    public Square(float base, float height)
    {
        this.base=base;
        this.height=height;
    }
    //Método que devuelve el perímetro del cuadrado
    public void Perimeter()
    {
        float result = base+base+height+height;
        System.out.println("Square Perimeter");
        System.out.println("    P: " + result + " cm");
    }
    //Método que devuelve el área del cuadrado
    public void Area()
    {
        float result = base*height;
        System.out.println("Square Area");
        System.out.println("    A: " + result + " cm^2");

    }
    //Método estático con parámetros de tipo enteros
    // que devuelve el perímetro del cuadrado
    public static int Perimeter(int base, int height)
    {
```

```
    int result = base+base+height+height;
    System.out.println("\nSquare Perimeter (int)");
    System.out.println("    P: " + result + " cm");
    return result;
}
//Método estático con parámetros de tipo enteros
// que devuelve el área del cuadrado
public static int Area(int base, int height)
{
    int result = base*height;
    System.out.println("\nSquare Area (int)");
    System.out.println("    A: " + result + " cm^2");
    return result;
}
//Método estático con parámetros de tipo flotantes
// que devuelve el perímetro del cuadrado
public static float Perimeter(float base, float height)
{
    float result = base+base+height+height;
    System.out.println("\nSquare Perimeter (float)");
    System.out.println("    P: " + result + " cm");
    return result;
}
//Método estático con parámetros de tipo flotantes
// que devuelve el área del cuadrado
public static float Area(float base, float height)
{
    float result = base*height;
    System.out.println("\nSquare Area (float)");
    System.out.println("    A: " + result + " cm^2");
    return result;
```

```
    }

}
```

Clase Circle.java

```
public class Circle
{
    public
    float radius;
    private
    static final float pi = 3.14159f;
    //Constructor
    public Circle(float radius)
    {
        this.radius = radius;
    }
    //Método que devuelve el perímetro del círculo
    public void Perimeter()
    {
        float result = pi*(2*radius);
        System.out.println("Circle Perimeter");
        System.out.println("    P: " + result + " cm");
    }
    //Método que devuelve el área del círculo
    public void Area()
    {
        float result = pi*(radius*radius);
        System.out.println("Circle Area");
        System.out.println("    A: " + result + " cm^2");
    }
    //Método estático con parámetros de tipo flotantes
    // que devuelve el perímetro del círculo
```

```
    public static float Perimeter(float radius)
    {
        float result = pi*(2*radius);
        System.out.println("Circle Perimeter");
        System.out.println("    P: " + result + " cm");
        return result;
    }
    //Método estático con parámetros de tipo flotantes
    // que devuelve el área del círculo
    public static float Area (float radius)
    {
        float result = pi*(radius*radius);
        System.out.println("Cricle Area");
        System.out.println("    A: " + result + " cm^2");
        return result;
    }

}
```

Clase Triangle.java

```
public class Triangle
{
    //Variables
    public
    float base_1, base_2, base_3, height;
    //Constructor
    public Triangle(float base_1, float base_2, float base_3, float height)
    {
        this.base_1 = base_1;
        this.base_2 = base_2;
        this.base_3 = base_3;
        this.height = height;
```

```
}
//Método que devuelve el perímetro del triángulo
public void Perimeter_int()
{
    float tri_per = base_1+base_2+base_3;
    System.out.println("Triangle Perimeter");
    System.out.println("    P: " + tri_per + " cm");

}
//Método que devuelve el área del triángulo
public void Area()
{
    float tri_are;

    tri_are=(base_1*height)/2;

    System.out.println("Triangle Area");
    System.out.println("    A: " + tri_are + " cm^2");

}

//Método estático con parámetros de tipo enteros
// que devuelve el perímetro del triángulo
public static int Perimeter_int(int base_1, int base_2, int base_3)
{
    int result = (base_1+base_2+base_3);
    System.out.println("Triangle Perimeter (int)");
    System.out.println("    P: " + result + " cm");
    return result;
}
//Método estático con parámetros de tipo enteros
```

```
// que devuelve el área del triángulo
public static int Area_int(int base_1,int height)
{
    int tri_are_int;

    tri_are_int=(base_1*height)/2;

    System.out.println("Triangle Area (int)");
    System.out.println("    A: " + tri_are_int + " cm^2");
    return tri_are_int;
}
//Método estático con parámetros de tipo flotantes
// que devuelve el perímetro del triángulo
public static float Perimeter_flo(float base_1, float base_2, float base_3)
{
    float result = base_1+base_2+base_3;
    System.out.println("Triangle Perimeter (float)");
    System.out.println("    P: " + result + " cm");
    return result;
}
//Método estático con parámetros de tipo flotantes
// que devuelve el área del triángulo
public static float Area_flo(float base_1,float height)
{
    float tri_are_flo;

    tri_are_flo=(base_1*height)/2;
    System.out.println("Triangle Area (float)");
    System.out.println("    A: " + tri_are_flo + " cm^2");
    return tri_are_flo;
}
```

```
}
```

Clase Cube.java

```
public class Cube
{
    public
    float height, base, width, square;
    //Constructor usando los parámetros: height, base y width.
    public Cube(float height, float base, float width)
    {
        this.height=height;
        this.base=base;
        this.width=width;
    }
    //Constructor usando los parámetros: square y width.
    public Cube(float square, float width)
    {
        this.square = square;
        this.width = width;
    }
    //Método que devuelve el perimetro
    public void Perimeter()
    {
        float perimeter;
        perimeter = (height*4)+(base*4)+(width*4);
        System.out.println("Cube Perimeter: "+perimeter+" cm");
    }
    //Método que devuelve el área
    public void Area()
    {
        float area;
```

```
    area = (base*height)*6;
    System.out.println("Cube Area: "+area+" cm^2");
}
//Método que devuelve el volumen
public void Volume()
{
    float volume;
    volume = (base*height*width);
    System.out.println("Cube Volume: "+volume+" cm^3");
}
//Método que devuelve el perimetro, usando la variable 'square'
public void Perimeter_sqr()
{
    float perimeter;
    perimeter = width*12;
    System.out.println("Cube Perimeter: "+perimeter+" cm");
}
//Método que devuelve el área, usando la variable 'square'
public void Area_sqr()
{
    float area;
    area = square*6;
    System.out.println("Cube Area: "+area+" cm^2");
}
//Método que devuelve el volumen, usando la variable 'square'
public void Volume_sqr()
{
    float volume;
    volume = square*width;
    System.out.println("Cube Volume: "+volume+" cm^3");
}
```

}

Resultados:

```
Square Perimeter: 19.26 cm
Square Area: 17.825 cm^2
Square Perimeter (int): 16 cm
Square Area (int): 70 cm^2
Square Perimeter (float): 19.26 cm
Square Area (float): 70.98 cm^2

Circle Perimeter: 18.84954 cm
Circle Area: 28.274311 cm^2
Circle Perimeter: 18.84954 cm
Circle Area: 28.274311 cm^2

Triangle Perimeter: 7.5 cm
Triangle Area: 6.375 cm^2
Triangle Perimeter (int): 6 cm
Triangle Area (int): 6 cm^2
Triangle Perimeter (float): 7.0 cm
Triangle Area (float): 1.7249999 cm^2

Cube Perimeter: 32.4 cm
Cube Area: 36.0 cm^2
Cube Volume: 18.599998 cm^3
Cube Perimeter: 24.0 cm
Cube Area: 18.0 cm^2
Cube Volume: 6.0 cm^3

Process finished with exit code 0
```

Conclusiones:

Los métodos estáticos son una parte importante de la programación orientada a objetos usando Java. Estos métodos se definen fuera de la clase y se declaran como estáticos. Esto significa que se pueden llamar directamente sin crear una instancia de una clase. Esto es útil cuando un método necesita ser compartido entre varias instancias de una clase. Los métodos estáticos también son útiles para crear un comportamiento estándar para una clase, ya que se pueden definir en la clase en lugar de tener que definirlo en cada instancia. Por último, los métodos estáticos son una forma útil de encapsulamiento, ya que se pueden utilizar para ocultar información y comportamiento que no se quiere mostrar a los usuarios.

Lo que más me llamó la atención de este trabajo es la diferencia entre los métodos estáticos y no estáticos. Los métodos estáticos no están asociados a una instancia de clase específica, lo que significa que se pueden invocar directamente sin tener que crear una instancia de la clase. Esto tiene la ventaja de ahorrar tiempo y esfuerzo al momento de ejecutar el código, ya que no es necesario crear una nueva instancia de la clase. Por otro lado, los métodos no estáticos están asociados a una instancia específica de la clase, lo que significa que es necesario crear una instancia para poder acceder a ellos. Esto puede ser una desventaja en algunos casos, especialmente cuando se trata de grandes cantidades de datos, ya que la creación de instancias puede ser un proceso muy costoso.

Los métodos estáticos son los más populares en entornos estables, ya que ofrecen simplicidad y estabilidad. Sin embargo, los métodos no estáticos son más adecuados para entornos cambiantes, ya que son capaces de adaptarse a los cambios en el entorno.

Bibliografía:

Chaves, G. (2018). Características de los métodos estáticos y no estáticos en programación. Digital Learning, 5(1), 51-63.

Pérez, J. (2009). Aplicación de los métodos estáticos y no estáticos en programación orientada a objetos. Revista de Ingeniería y Ciencias Aplicadas, 9(2), 73-91.

García, L. (2011). Ventajas de los métodos estáticos y no estáticos en el desarrollo de software. Informática y actualidad, 11(4), 88-99.