5TO ENSAYO DE LA CLASE

MÉTODOS

Los métodos en Java son los que indican el comportamiento de los objetos pertenecientes a estas clases, en otros lenguajes de programación son llamados funciones o procedimientos. Una de las ventajas es que nos permite dividir el programa en módulos lo cual hace más fácil su comprensión y mantenimiento, aparte es una forma de reutilización de software. Podemos crear métodos en nuestras clases pero además disponemos de otros métodos prediseñados que se encuentran en la API de Java. Ejemplos:

- Clase: Math Método: Math.pow(double a, doublé b).
- Clase: String Método: String.valueOf(int a).
- Clase: Object Método: toString()

Se sabe que hay miembros de clase y miembros de objeto. Cuando se va implementar un método de clase se usa la palabra reserva static (que quiere decir que es método que pertenece a la clase y no a un objeto en particular, lo cual para su llamada se usara el nombre de la clase).Sintaxis para la llamada (indica que se hará uso del método en ese momento) de métodos tanto de clase como de objetos:

Llamada a un método de clase:

- Nombre_de_la_Clase.Nombre_del_método(Argumentos);

Llamada a un método de objeto:

- Nombre_del_objeto.Nombre_del_método(Argumentos).

Un método especial es el método constructor, el cual se usa para inicializar los atributos de un objeto, ya que es de suma importancia inicializar los atributos debido a que podrían generarerrores más adelante es por eso que Java genera un constructor por defecto en caso que no se implemente uno asignándole a los atributos los valores por defecto dependiendo del tipo que sea.

SOBRECARGA DE MÉTODOS

Se da cuando se definen dos métodos con el mismo nombre, pero con diferentes parámetros (en base al número, tipo y orden de parámetros). Generalmente se usa para crear métodos querealicen tareas similares o iguales pero con distintos tipos o número de argumentos.

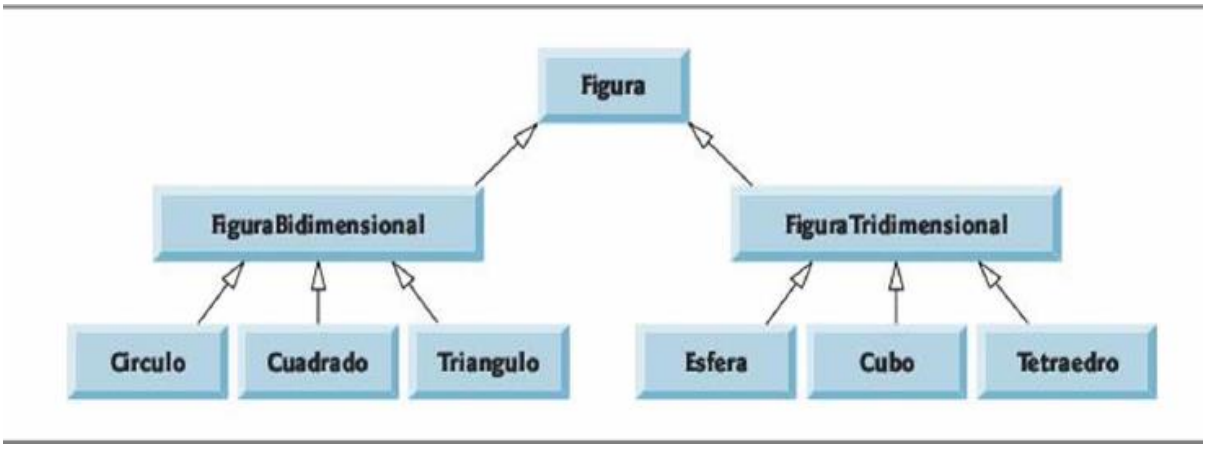
```java
public Vehiculo(){
    this.color="negro";
    this.cv=200;
    this.ruedas=2;
    this.velocidad=50;

}
public  Vehiculo(String color, Integer cv, Integer ruedas, Integer velocidad) {
    this.color = color;
    this.cv = cv;
    this.ruedas = ruedas;
    this.velocidad = velocidad;
}
```

En la imagen se ve que se ha sobrecargado el constructor en el primero no recibe ningún parámetro y en el segundo si los recibe los dos realizan funciones parecidas ya que ambos se encargan de inicializar los atributos.

Herencia

Herencia es una las propiedades del paradigma de la programación orientada a objetos. Una definición de herencia sería la siguiente: Herencia o relación es-un es la relación que existe entre dos clases, en que la clase denominada clase derivada (en Java se denomina subclase o clase hijo) se crea a partir de otra ya existente, denominada clase base (en Java se denomina superclase o clase padre). Al representar la herencia se forma una estructura tipo árbol en el cual se pue de apreciar notoriamente la relación es-un.



Para hacer que una clase herede de otra se usa la palabra reservada extends. Ejemplo:

```java
package pe.uni.fiis.herencia;

/**
 * Created by User on 04/06/2015.
 */
public class Moto extends Vehiculo {
    private Boolean reparar_cadena;

    public Moto(String color, Integer cv, Integer ruedas, Integer velocidad, Boolean reparar_cadena) {
        super(color, cv, ruedas, velocidad);
        this.reparar_cadena = reparar_cadena;
    }
    public void acelerar(){
        setVelocidad(getVelocidad()+5);
    }
}
```

En la imagen se puede ver que la clase Moto hereda de Vehiculo, lo cual cumple la relación es-un. Además se observa que la clase solo posee un atributo el cual es reparar_cadena, eso quiere decir que los demás atributos que se pueden observar como parámetros del constructor, son heredados por la clase Vehiculo.

```java
package pe.uni.fiis.herencia;

/**
 * Created by User on 04/06/2015.
 */
public abstract class Vehiculo {
    public  String color;
    private Integer cv;
    private final Integer ruedas;
    private Integer velocidad;

    protected Vehiculo(String color, Integer cv, Integer ruedas, Integer velocidad) {
        this.color = color;
        this.cv = cv;
        this.ruedas = ruedas;
        this.velocidad = velocidad;
    }
```

Pero los atributos de Vehículo son private eso quiere decir que solo pueden ser usados por la clase donde se encuentran declarados, entonces la subclase para acceder a ellos tienen que hacer uso de métodos para poder inicializar esos atributos en este caso hace uso del método constructor de Vehículo el cual es llamado en el método constructor de Moto, mediante las instrucción **super**(puede ser usado para llamar a cualquier método de la clase padre no solo al constructor), como primera instrucción del constructor de la clase hijo debe ser llamar al constructor de la superclase., esto se da de esta forma debido a que el constructor de la superclase no se hereda, además como se mencionó los atributos declarados como private tampoco pueden ser usados directamente.Si se quisiera hacer uso de los atributos de la superclase directamente podrían ser declarados como

public o protected(es un modificador acceso el cual restringe su accesibilidad a clases en el mismo paquete o clases heredadas) pero para respetar el principio de encapsulamiento es mejor declararlos como private y acceder a través de métodos.

Como se puede notar al hacer uso de la herencia se hace reutilización de software debido a que no implementamos un método adicional para iniciar todos los atributos de la clase moto si no que hacemos uso del constructor de la superclase además que evitamos el tener que declarar de nuevo esos atributos en la clase Moto.

POLIMORFISMO

El polimorfismo es una de las propiedades de la programación orientada a objetos. Una definición sería la siguiente: El polimorfismo es la propiedad que permite que una operación (método) pueda tener el mismo nombre en clases diferentes y que actúe de modo diferente en cada una de ellas.

El polimorfismo es importante tanto en el modelado de sistemas como en el desarrollo de software. En el modelado porque el uso de palabras iguales tiene comportamientos distintos, según el problema a resolver. En software por que el polimorfismo toma ventaja de la propiedad de la herencia.

Con el polimorfismo, podemos diseñar e implementar sistemas que puedan extenderse con facilidad. Las únicas partes de un programa que deben alterarse para dar cabida a las nuevas clases son las que requieren un conocimiento directo de las nuevas clases que el programador agregará a la jerarquía.

Como ejemplo de polimorfismo creamos las clases: Vehiculo(Superclase), Moto(Subclase que hereda de Vehiculo), Coche(Subclase que hereda de Vehiculo. Luego definimos un método llamado acelerar, el cual lo van poseer ambas subclases pero de una manera distinta para cada uno.

```
public void acelerar(){
    super.setVelocidad(getVelocidad()+10);//En la Clase Coche
}
```

```
public void acelerar(){
    setVelocidad(getVelocidad()+5);// En la clase Moto
}
```

Luego creamos una clase principal el cual va a tener nuestro método main. Procedemos a elaborar el siguiente código.

```java
public class Principal {
    public static void main(String[] args) {
        Vehiculo []medios=new Vehiculo[4];
        medios[0]=new Coche("rojo",100,4,20,4);
        medios[1]=new Coche("azul",150,4,100,2);
        medios[2]=new Moto("negro",200,2,30,false);
        medios[3]=new Moto("verde",100,4,20,true);
        for(Vehiculo e:medios){
            e.acelerar();
            System.out.println(e.getVelocidad());
        }
    }
}
```

En el cual se crea un arreglo de cuatro elementos de tipo Vehiculo pero al momento de instanciarlo lo hacemos con diferentes tipos de subclases en este caso Moto y Coche. Esto se puede hacer en Java debido a la propiedad de la herencia ya que un Coche es un Vehiculo y una Moto es un Vehiculo, luego elaboramos una sentencia for mejorada, en donde procedemos a llamar al método acelerar para cada uno de los elementos del arreglo. En ese momento al ejecutar el programa todas las llamadas al método acelerar se resuelven en tiempo de ejecución asignándose el método correspondiente en base al tipo de objeto que se hace referencia esto es conocido en Java como vinculación dinámica o enlazado dinámico, en base a esto Java decide a que método acelerar llama en tiempo de ejecución, esto es una expresión del polimorfismo.

Nota: Use la palabra Vehículo sin tilde por que así era el nombre de la clase creada.

BIBLIOGRAFÍA

- Fundamentos de Programación(Algoritmos, estructura de datos y objetos) Luis Joyanes Aguilar.
- Java Como Programar Paul Deitel, Harvey Deitel
- https://www.youtube.com/watch?v=1d35MF30xDo
- https://www.youtube.com/watch?v=Kr44pXrfzQQ

MENDOZA MEDRANO ADRIÁN ESTEBAN