TALLER DE PARADIGMAS

Clase N°4

Magni Gastón
Massietti Alexander
Fermani Julián

INTRODUCCION

INTRODUCCION
INTRODUCCION
INTRODUCCION
INTRODUCCION
INTRODUCCION
INTRODUCCION

¿QUE ES LA HERENCIA?

Cuando se diseña un programa con herencia, se coloca código común en una clase y, luego, se determina a otras clases más específicas que la clase común es su superclase (Schildt, 2018).

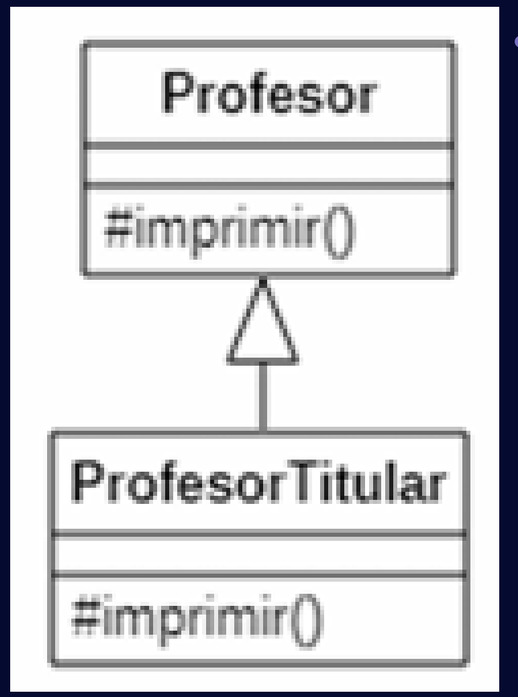
Los atributos y métodos heredados se pueden utilizar tal como están, reemplazarlos o complementarlos con nuevos atributos y métodos (Reyes y Stepp, 2016).

SUPERCLASE

Una superclase o clase padre es la clase que se hereda a otras clases.

SUBCLASE

Una subclase o clase hija es la clase que hereda de otras clases.



SUBCLASE

```java
public class InmuebleVivienda extends Inmueble {  no usages
    protected int numeroHabitaciones;  2 usages
    protected int numeroBanios;  2 usages
    public InmuebleVivienda(int identificadorInmobiliario, int area,
                            String direccion, int numeroHabitaciones,
                            int numeroBanios) {
        super(identificadorInmobiliario, area, direccion);
        this.numeroHabitaciones = numeroHabitaciones;
        this.numeroBanios = numeroBanios;
    }

    @Override
    public String toString() {
        System.out.println(super.toString());
        return "Número de habitaciones = " + this.numeroHabitaciones +
                "\nNúmero de banios = " + this.numeroBanios;
    }
}
```

SUPERCLASE

```java
public class Inmueble {  1 usage  1 inheritor
    protected int identificadorInmobiliario;  2 usages
    protected int area;  3 usages
    protected String direccion;  2 usages
    protected double precioVenta;  3 usages

    Inmueble(int identificadorInmobiliario, int area, String direccion) {  1 usage
        this.identificadorInmobiliario = identificadorInmobiliario;
        this.area = area;
        this.direccion = direccion;
    }

    public double calcularPrecioVenta(double valorArea) {  no usages
        this.precioVenta = area * valorArea;
        return this.precioVenta;
    }

    @Override  1 override
    public String toString() {
        return "Identificador inmobiliario = " + this.identificadorInmobiliario +
                "\nArea = " + this.area +
                "\nDirección = " + this.direccion +
                "\nPrecio de venta = $ " + this.precioVenta;
    }
}
```

¿QUÉ ES EL POLIMORFISMO?

Es la capacidad de un objeto para adoptar muchas formas. El uso más común del polimorfismo ocurre cuando se utiliza una referencia a una clase padre para referirse a un objeto de una clase hija (Vozmediano, 2017).
Alude al modo en que se pueden crear y utilizar dos o más métodos con el mismo nombre para ejecutar funciones diferentes.

CLASE PROFESOR

```java
public class Profesor {  2 usages  1 inheritor

    @Override  1 override
    public String toString() {
        return "Es un profesor.";
    }
}
```

CLASE PROFESORTITULAR

```java
public class ProfesorTitular extends Profesor {

    public String toString() {
        return "Es un profesor titular.";
    }
}
```

En el main se crea un Profesor pero instanciando la clase ProfesorTitular. ¿Qué se imprimirá en pantalla?

```java
public class Main {
    public static void main(String[] args) {
        Profesor profesor1 = new ProfesorTitular();
        System.out.println(profesor1.toString());
    }
}
```

Es un profesor titular.

Fuente:Programación orientada a objetos. Polimorfismo.

SOBRECARGA

Consiste en la capacidad que tiene una función o método de arrojar un resultado para diferentes tipos de datos que recibe como argumentos. Dicho de otra forma, una función o método polimórfico es capaz de despachar un resultado correcto para cada tipo que reciba.

SOBREESCRITURA

cuando un método en una subclase tiene el mismo nombre y la misma firma de tipo que un método en su superclase, se dice que el método en la subclase sobrescribe el método en la superclase. Cuando se llama a un método sobrescrito desde dentro de su subclase, siempre se referirá a la versión de ese método definida por la subclase. La versión del método definida por la superclase quedará oculta.

Clase MiembroDeLaComunidadUniversitaria

```java
public class MiembroDeLaComunidadUniversitaria{  1 usage  1 inheritor
    String nombre;  4 usages
    int dni;  3 usages

    public MiembroDeLaComunidadUniversitaria(String nombre, int dni){
        this.nombre = nombre;
        this.dni = dni;
    }

    public String inscribirse(String actividad){  no usages  1 override
        return "La inscripción de " + nombre + " para "
                + actividad + " se ha realizado satisfactoriamente.";
    }
}
```

Clase Estudiante

```java
import java.time.LocalDate;

public class Estudiante extends MiembroDeLaComunidadUniversitaria{  no usages
    String carrera;  2 usages
    int carnetNro;  1 usage
    String semestre_Ano;  1 usage

    public Estudiante(String nombre, int dni, String carrera,  no usages
                      int carnetNro, String semestre_Ano){
        super(nombre, dni);
        this.nombre = nombre;
        this.dni = dni;
        this.carrera = carrera;
        this.carnetNro = carnetNro;
        this.semestre_Ano = semestre_Ano;
    }

    public boolean verificar_cupo(Materia materia){  1 usage
        if (materia.getCupo()>= 1) return true;
        return false;
    }

    //método heredado que exhibe conducta propia de la clase hija
    //sobrescritura de método padre
    @Override  no usages
    public String inscribirse(String actividad){
        return "El alumno " + nombre + " DNI " + dni +
                " perteneciente a la carrera " + carrera +
                " ha completado su inscripción";
    }

    //sobrecarga de método padre
    public String inscribirse(String actividad, LocalDate fecha){  no usages
        return "Inscripción realizada para la actividad " + actividad
                + " a tener lugar el " + fecha;
    }

    public String inscribirse(Materia materia){  no usages
        if(verificar_cupo(materia)){
            materia.tomarCupo();
            return "Se completó la inscripción en " + materia.titulo;
        }else{
            return "No hay cupo disponible";
        }
    }
```

Clase Materia

```java
public class Materia {  2 usages
    String titulo;  3 usages
    String carrera;  1 usage
    int cupo;  6 usages

    public Materia(String titulo, String carrera){
        this.titulo=titulo;
        this.carrera=carrera;
        this.cupo=40;
    }

    public int getCupo(){  1 usage
        return cupo;
    }

    public int tomarCupo(){  1 usage
        if(cupo==0) return cupo;
        return this.cupo=cupo-1;
    }

    public String getTitulo(){  no usages
        return titulo;
    }
}
```

OBJETIVOS

- Detección de la relación de clases en un enunciado
- Representación de la relación de generalización
- Constructor de objetos donde heredan atributos y creando los propios
- Creación de métodos propios
- Uilización de métodos heredados

ACTIVIDAD FORMAS GEOMÉTRICAS

Hacer un programa que simule un sistema de cálculo de áreas para diferentes formas geométricas. Las formas que se considerarán son: Círculo y Rectángulo.

1. Crear una clase base llamada Forma que tenga un método para calcular el área.
2. Crea dos subclases: Circulo y Rectangulo, que sobrescriban el método para calcular el área.
3. En la clase Circulo, implementa la sobrecarga del método para calcular el área con diferentes tipos de parámetros, como el radio (tipo double) y el diámetro (tipo int).
4. En la clase Main, crea una instancia de cada forma y calcula el área usando polimorfismo.

ACTIVIDAD CTA CTE

Definir ahora una “Cuenta Joven”, para ello vamos a crear una nueva clase CuentaJoven que deriva de la anterior. Cuando se crea esta nueva clase, además del titular y la cantidad se debe guardar una bonificación que estará expresada en tanto por ciento. Construye los siguientes métodos para la clase:

- Un constructor.
- Los setters y getters para el nuevo atributo.
- En esta ocasión los titulares de este tipo de cuenta tienen que ser mayor de edad., por lo tanto hay que crear un método esTitularValido() que devuelve verdadero si el titular es mayor de edad pero menor de 25 años y falso en caso contrario.
- Además la retirada de dinero sólo se podrá hacer si el titular es válido.
- El método mostrar() debe devolver el mensaje de “Cuenta Joven” y la bonificación de la cuenta.

GRACIAS