ADRIAN MAXIMILIANO MUNOZ MARTINEZ

Laboratorio POO  Sabado 7 am – 9am

PRactica 6: POLImorfismo

**Introducción**

Se desarrollará una aplicación en Java que en la que se implemente la herencia entre clases para demostrar el concepto de polimorfismo.

**Objetivo**

El objetivo de la practica es familiarizarnos con el concepto de polimorfismo, así como conocer los usos que tiene y las diferentes maneras en que se puede implementar.

**Practica**

Practica6.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package practica6;

import java.util.Scanner;

/\*\*

\*

\* @author FCFM

\*/

public class Practica6 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Ingrese la marca del automovil: ");

String marca = sc.nextLine();

System.out.print("Ingrese el modelo del automovil: ");

String modelo = sc.nextLine();

System.out.print("Ingrese el color del automovil: ");

String color = sc.nextLine();

Automovil auto = new Automovil(marca, modelo);

auto.setColor(color);

System.out.println("\nInformacion del automovil: ");

System.out.println("Marca del automovil: " + auto.getMarca());

System.out.println("Modelo del automovil: " + auto.getModelo());

System.out.println("Color del automovil: " + auto.getColor());

System.out.println("\nClase automovil: ");

System.out.println(auto.encender(100));

System.out.println(auto.avanzar(10));

auto.frenar();

auto.apagar();

Transporte tr = auto;

System.out.println("\nClase Transporte:");

tr.encender(10);

tr.apagar();

IAutomovil iauto = new Automovil(marca, modelo, color);

System.out.println("\nInterfaz de automovil: ");

System.out.println(iauto.acelerar(0, true));

}

}

Transporte.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package practica6;

/\*\*

\*

\* @author FCFM

\*/

public abstract class Transporte {

private String marca;

private String modelo;

public Transporte(String marca, String modelo){

setMarca(marca);

setModelo(modelo);

}

public abstract String encender(int gasolina);

public void apagar(){

System.out.println("Automovil apagado");

}

/\*\*

\* @return the marca

\*/

public final String getMarca() {

return marca;

}

/\*\*

\* @param marca the marca to set

\*/

public final void setMarca(String marca) {

this.marca = marca;

}

/\*\*

\* @return the modelo

\*/

public final String getModelo() {

return modelo;

}

/\*\*

\* @param modelo the modelo to set

\*/

public final void setModelo(String modelo) {

this.modelo = modelo;

}

}

IAutomovil.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package practica6;

/\*\*

\*

\* @author FCFM

\*/

public interface IAutomovil {

public String acelerar(int gasolina, boolean estado);

}

Automovil.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package practica6;

/\*\*

\*

\* @author FCFM

\*/

public class Automovil extends Transporte implements IAutomovil {

private String color;

public Automovil(String marca, String modelo, String color){

super(marca, modelo);

setColor(color);

}

public Automovil(String marca, String modelo){

super(marca, modelo);

}

public String avanzar(int gasolina){

if(gasolina>0){

return "El auto color "+ getColor() + " esta avanzando";

}else{

return "El auto no tiene suficiente gasolina";

}

}

public void frenar(){

System.out.println("Automovil frenando");

}

@Override

public String encender(int gasolina){

if(gasolina>0){

return "Automovil encendido";

}else{

return "No hay suficiente gasolina";

}

}

@Override

public String acelerar(int gasolina, boolean estado){

if(gasolina>0 & estado){

return "Auto acelerando";

}else if(estado){

return "No hay suficiente gasolina";

}else{

return "El automovil esta apagado";

}

}

/\*\*

\* @return the color

\*/

public final String getColor() {

return color;

}

/\*\*

\* @param color the color to set

\*/

public final void setColor(String color) {

this.color = color;

}

}

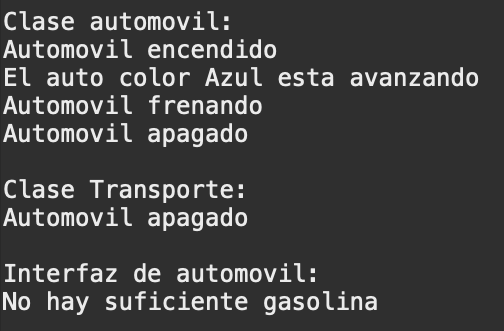
**Screenshots**

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Imagen que contiene texto

Descripción generada automáticamente



**Explicación**

Se creo una aplicación que implemento un objeto Automóvil haciendo uso de las características de polimorfismo que ofrece Java. Esto se hizo al crear diferentes variables de referencia las cuales eran del tipo de las interfaces o las clases que la clase automóvil heredo.

Aunque estas clases hayan sido abstractas, así como las interfaces, se pudo instanciar un objeto usando estas clases como referencia por medio de la clase hija Automóvil la cual es concreta. Esto nos permitió explorar las diferentes funciones que tiene el polimorfismo.

Por ejemplo, al crear la clase Automóvil con una referencia a la clase Automóvil, pudimos acceder a todas las funciones disponibles para automóvil incluyendo los métodos sobre escritos que fueron heredados de la clase padre. Al usar la clase Transporte como referencia para el objeto Automovil solo pudimos acceder a los métodos y atributos que fueron declarados en la clase Transporte, en el caso de los métodos abstractos se ejecutaron las rutinas definidas o sobre escritasen la clase Automovil. Lo mismo ocurrio al referenciar al objeto Automovil a través de la interfaz IAutomovil, en este caso solo pudimos acceder al método abstracto acelerar con las rutinas definidas en la clase hija.

**Conclusión**

El polimorfismo en Java nos permite crear código mas compacto y versátil, ya que al no ser necesario definir un objeto estrictamente como referencia a la clase que lo define finalmente, podemos utilizar una variable para referenciar diferentes tipos de objetos que compartan un ancestro y utilizar los mismos métodos en todos estos objetos mientras que cada uno realiza una tarea distinta dependiendo del objeto instanciado.