ADRIAN MAXIMILIANO MUNOZ MARTINEZ

Laboratorio POO  Sabado 7 am – 9am

PRactica 9: Java Generics

**Introducción**

Se desarrollará una aplicación en Java que en la que se implemente colecciones genéricas.

**Objetivo**

El objetivo de la practica es familiarizarnos con el concepto de colecciones genéricas y objetos genéricos, así como practicar su implementación y conocer sus ventajas y desventajas.

**Practica**

Practica9.java

package practica9;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Scanner;

public class Practica9 {

public static void main(String[] args) {

// Inicializar scanner

Scanner sc = new Scanner(System.in);

// Crear array list generico con elementos de tipo auto

ArrayList<Automovil> autos = new ArrayList<Automovil>();

// Ciclo for en el que se pedira la informacion de los 10 autos que seran

// agregados a la lista

for (int i = 0; i < 10; i++) {

// Darle formato al String para incluir, el numero del auto que sera anadido

System.out.print(String.format("\nAutomovil numero %d", i + 1));

// Llamar a constructor del objeto Automovil

// Agregar nuevo objeto a la lista

autos.add(i, new Automovil(10));

// Llenar los atributos del objeto, con los vaolres indicados por el usuario

System.out.print("\nIngresa la marca del automvil: ");

autos.get(i).setMarca(sc.nextLine());

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

autos.get(i).setModelo(sc.nextLine());

System.out.print("Ingresa el precio del automovil: ");

autos.get(i).setPrecio(sc.nextLine());

}

// Imprimir los atributos y llamar las funciones de cada objeto Automovil en el

// array list

for (Automovil auto : autos) {

// Imprimir el numero del automovil en el orden en que se ingreso

System.out.println(String.format("\nAutomovil %d", (autos.indexOf(auto) + 1)));

// Imprimir los atributos y llamar a los metodos de la clase Automovil

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

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

System.out.println("Precio: " + auto.getPrecio());

// Preguntar al usario por la direccion en que desea moverse

System.out.println("Ingresa la direccion en la que el auto tiene que moverse: ");

System.out.println(auto.movimiento(sc.nextLine()));

System.out.println(auto.alto());

}

}

}

Automovil.java

package practica9;

public class Automovil {

private String marca;

private String modelo;

private String precio;

private int gas;

// Constructor

public Automovil(int gas) {

this.encender(gas);

}

// Encender funciona como un setter de gas

private void encender(int gas) {

this.gas = gas;

}

public String alto() {

return "El automovil se ha detenido";

}

public String movimiento(String mov) {

if (this.gas > 1) {

return "Moviendose a la " + mov;

} else {

return "No hay suficiente gasolina";

}

}

public void setMarca(String marca) {

this.marca = marca;

}

public void setModelo(String modelo) {

this.modelo = modelo;

}

public void setPrecio(String precio) {

this.precio = precio;

}

public String getMarca() {

return this.marca;

}

public String getModelo() {

return this.modelo;

}

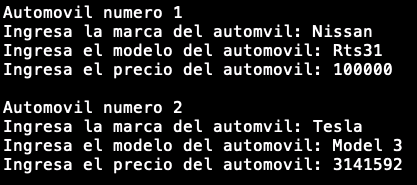
public String getPrecio() {

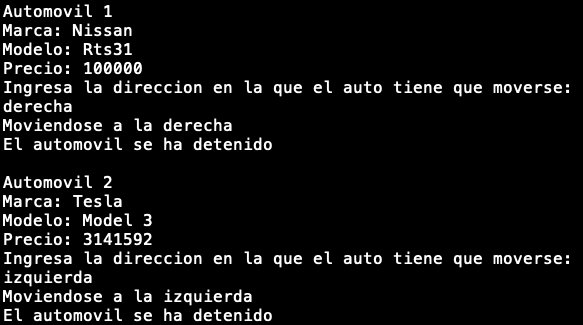
return this.precio;

}

}

**Screenshots**



****

**Explicación**

Se creo una aplicación que implemento un objeto Automóvil. Dentro de la clase principal (Practica8) se creo un ArrayList cuya función era la de almacenar los objetos de tipo Automovil que fueron creados y configurados con la entrada del usuario.

A diferencia de la practica anterior en esta practica no fue necesario castear el objeto en el array ni almacenar su referencia en una variable aparte, ya que desde un inicio se le especifico a Java el tipo de objeto que almacenaría el ArrayList, esta fue una de las ventajas de usar colecciones genéricas, pero al mismo tiempo eso nos robo de la posibilidad de almacenar algún otro tipo de objeto dentro del ArrayList.

Aunque no fue necesario agregar otro tipo de objeto a la colección, no hubiéramos podido hacerlo si lo necesitáramos sin hacer modificaciones drásticas a la estructura del programa.

**Conclusión**

Las colecciones genéricas agilizan y simplifican mucho la implementación del código cuando sabemos que la colecciones solo va a almacenar un tipo de dato en especifico, pero al mismo tiempo nos roban un poco de la flexibilidad que una colección no genérica ofrece.