



Instituto Politécnico Nacional
Unidad Profesional Interdisciplinaria en
Ingeniería y Tecnologías Avanzadas



Ingeniería Mecatrónica
Programación Avanzada

REPORTE DE LA PRÁCTICA No. 2

Objetos

Nombre del Alumnos: Zarazua Aguilar Luis Fernando

Grupo: 2MV4

Fecha:

29/Agosto/2017

Objetivo: Identificar las características de los objetos, así como su sintaxis y el paradigma que representan.

Resumen

En el presente documento se menciona el hecho de estar trabajando con la programación orientada a objetos y como se debe de manejar de una manera apropiada, comprendiendo el paradigma que este tipo de programación representa. Posteriormente se abordan algunas de las características que posee un objeto, con las cuales se puede dar una definición básica de lo que es un objeto en programación y como este tipo de programación se puede representar gráficamente.

Ya habiendo definido varios conceptos de objetos se menciona con un ejemplo como fueron implementados los objetos en un ejemplo práctico de un estacionamiento, viendo los distintos métodos y atributos que se usaron para lograr la aplicación.

Introducción

Actualmente vemos una variedad de programas implementados alrededor del mundo, entre los cuales destacan, las aplicaciones móviles, controladores en la industria, software para el desarrollo de ingeniería, videojuegos de variados géneros, programas para la administración y en si infinitudes de programas para casi cada propósito en el que se piense, sin embargo, poco se menciona de los distintos tipos de paradigmas en la programación que se usan actualmente como lo son el orientado a procedimientos, a objetos, a la lógica, a las reglas, al constreñimiento y el orientado a eventos. En el caso del paradigma orientado a objetos el programador que estará trabajando con objetos y clases, debe de preocuparse en quien efectuará un procedimiento y ya no en como implantar una función, haciéndolo no estrictamente orientado a datos [1].

Planteamiento del problema.

Cuando se maneja un nuevo paradigma lo más difícil es dejar las costumbres pasadas a las que el programador estaba acostumbrado, tratando de tener siempre presente la filosofía del paradigma POO, para que no se entrelacen distintos conceptos los cuales llevan al mal funcionamiento de la aplicación a programar, además el programador al momento de trabajar con objetos deberá de ser consciente que debe tener un nuevo punto de vista así como un orden al momento de programar. Con este orden es como se debe diseñar el objeto deseado, el cual contendrá por naturaleza un método y un atributo, el método será el encargado de realizar las tareas que se le encomiendan en conjunto con los atributos, que es la información que necesitan los métodos para poder operar, teniendo así una posible definición de objeto la cual es:

"Es una entidad que posee una función la cual realiza tareas específicas, con cierta información que le permite realizar dichas tareas, además de poseer características específicas".

Entre las características específicas se encuentran el estado, la conducta y la identidad. La conducta hace referencia a su capacidad de responder ante distintos efectos o situaciones que le son propuestos, la identidad maneja que a pesar que 2 objetos tengan las mismas características, se trata de elementos distintos que son independientes. Otra característica es su capacidad de poder agrupar las funciones e información dentro de un solo conjunto, esta característica es conocida como encapsulamiento y se compone de información pública y privada, la pública debe de ser usada cuando el objeto por sus métodos no sea capaz de realizar la tarea que le pide completamente por lo cual se debe de comunicar con otros objetos exteriores, cuando el objeto puede realizar su tarea sin necesidad del exterior entonces se deberá declarar como privado, además de estos 2 tipos de tener la información se puede contar el tipo protegido que permite más flexibilidad entre objetos con elementos en común. Para representar todas estas características que posee el objeto se tiene una representación gráfica la cual indica los métodos y atributos que posee una clase y por lo tanto que poseerá el objeto cuando sea creado, este diagrama se llama UML [2].

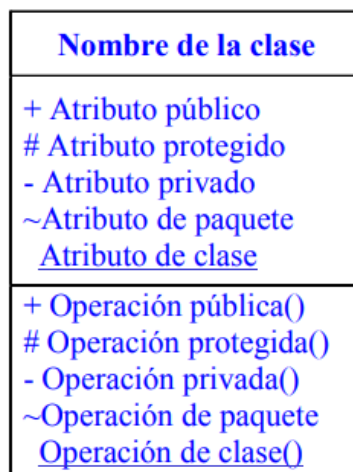


Figura 1. Diagrama UML de clases

Desarrollo

Para poder crear un objeto es necesario tener la clase que describe cómo será dicho objeto. En el siguiente código se incluyen las clases Parking y Carro, además de otras incluidas por los paquetes.

Aplicacion01.java

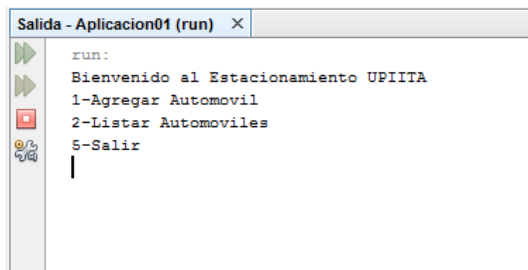
```
package aplicacion01;
import java.util.Scanner;
public class Aplicacion01 {
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner Opcion;
        Parking UPIITA;
        UPIITA=new Parking();
        int opc;
        Opcion=new Scanner(System.in);
        do
        {
            System.out.println("Bienvenido al
Estacionamiento UPIITA");
            System.out.println("1-Agregar Automovil");
            System.out.println("2-Listar Automoviles");
            System.out.println("5-Salir");
            opc=Opcion.nextInt();
            //IngresaCoche(UPIITA);
            switch(opc)
            {
                case 1: IngresaCoche(UPIITA);
                    break;
                case 2: ListarCoches(UPIITA);
                    break;
            }
        }while(opc!=5);
    }
    public static boolean IngresaCoche(Parking P)
    {
        if (P==null)
        {System.out.println("Error al crear objeto");
            return false;}
        else
        {
            String Propietario;
            String Marca;
            int Modelo;
            Scanner Texto;
            Texto=new Scanner(System.in);
            System.out.println("Ingresa el nombre del
propietario:");
            Propietario=Texto.nextLine();
            System.out.println("Ingresa la marca del
automovil:");
            Marca=Texto.nextLine();

            System.out.println("Ingresa el modelo");
            Modelo=Texto.nextInt();
            Carro c;
            c=new Carro(Propietario, Marca, Modelo);
            P.addCar(c);
            return true;
        }
    }
    public static void ListarCoches(Parking p)
    {
        Carro Aux;
        int i;
        int n=p.getNumCar();
        for (i=0;i<n;i++)
        {
            Aux=p.getCar(i);
            System.out.println(i+"-"+Aux.getModelo());
            System.out.println("Propietario:
"+Aux.getPropietario());
            System.out.println("Marca:
"+Aux.getMarca());
        }
    }
}
```

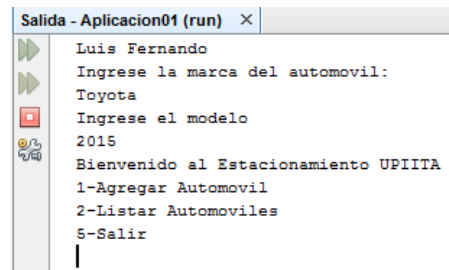
Observaciones: En el código se nota el uso de los objetos como entidades que incluyen los métodos necesarios para la ejecución del programa como lo es el objeto Opcion de la clase Scanner el cual nos sirve para la obtención de datos desde el teclado, este objeto necesita un constructor en este caso con el parámetro System.in, además el constructor necesita la instrucción new con lo cual el compilador reserva un espacio de memoria para el objeto. Este objeto Opción usa el método nextInt para poder capturar un dato de tipo entero, también en este objeto puede usar el método nextLine para recopilar datos de tipo string como es usado en el objeto Texto.

En la clase principal se tienen los métodos IngresaCoche y ListarCoches con los cuales se controla el ingreso y salida de la información, el método de Ingresar coche usa el objeto P de tipo Parking para guardar el objeto c tipo Carro ingresado, mediante el uso de los métodos mencionados anteriormente de Texto de Tipo Scanner para la recopilación de datos y mediante el método addCar que posee el objeto P con lo cual guarda un elemento Carro más en el objeto P, dicho objeto se ingresa al inicio de la función como UPIITA de tipo Parking. El siguiente método "ListarCoches" mediante la creación de un objeto Aux de tipo Carro obtiene cada elemento Carro de P tipo Parking esto gracias al método getNumCar con el cual se sabe cuántos elementos tipo Carro hay en P, posteriormente va desplegando en pantalla la información de cada objeto tipo Carro guardándolo en Aux también de tipo Carro el cual lee, se despliega en pantalla sus atributos y posteriormente se sobrescribe por el siguiente elemento.

Además cabe mencionar que para el llamado de un método usa la sintaxis "Objeto.metodo()", donde según el método puede llevar atributos o no dentro del paréntesis, para la escritura o lectura de un atributo se usa "Objeto.Atributo" y para la creación del objeto se usa "Objeto=new Clase()" que también es conocido como constructor.

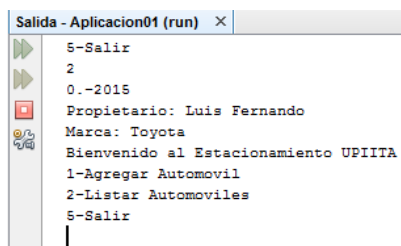


```
Salida - Aplicacion01 (run) X
run:
Bienvenido al Estacionamiento UPIITA
1-Agregar Automovil
2-Listar Automoviles
5-Salir
|
```

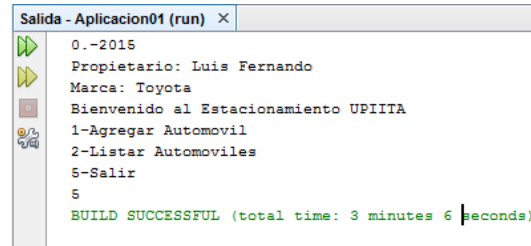


```
Salida - Aplicacion01 (run) X
Luis Fernando
Ingrese la marca del automovil:
Toyota
Ingrese el modelo
2015
Bienvenido al Estacionamiento UPIITA
1-Agregar Automovil
2-Listar Automoviles
5-Salir
|
```

Figuras 2 y 3. Ejecución del programa con ingreso de datos.



```
Salida - Aplicacion01 (run) X
5-Salir
2
0.-2015
Propietario: Luis Fernando
Marca: Toyota
Bienvenido al Estacionamiento UPIITA
1-Agregar Automovil
2-Listar Automoviles
5-Salir
|
```



```
Salida - Aplicacion01 (run) X
0.-2015
Propietario: Luis Fernando
Marca: Toyota
Bienvenido al Estacionamiento UPIITA
1-Agregar Automovil
2-Listar Automoviles
5-Salir
5
BUILD SUCCESSFUL (total time: 3 minutes 6 seconds)
```

Figuras 4 y 5. Lectura de los datos y finalización del programa.

Resultados y conclusiones

En la práctica con el uso de los objetos se pueden realizar las acciones requeridas para un conjunto de datos sin necesidad de alterar otro de mismo tipo.

Para la utilización de un método de una clase es necesario primeramente crear el objeto.

Cuando se tiene un objeto de que almacena varios objetos del mismo tipo como es caso de carro, es útil usar un objeto auxiliar para guardar la información en vez de crear varios tipos de datos más simples para estar guardando la información.

Los métodos también son capaces de tener objetos como atributos, siendo esto de mucha utilidad para no tener que crear variables globales.

Con el uso de la palabra reservada null se puede ver si un objeto no fue creado bien, con lo cual se pueden evitar errores en la ejecución del programa.

En la clase Parking se tiene un vector que contiene la cantidad de objetos Carro que el usuario necesite, gestionando automáticamente la memoria que se vaya necesitando, a diferencia de un lenguaje como C que se debe de contemplar la memoria asignada desde un inicio.

En conclusión los objetos entre otras aplicaciones nos permiten guardar la información en orden adecuado, con el cual sabemos qué resultados esperamos y que entradas de información tenemos pero sin la necesidad de conocer todo el proceso de tratamiento de la información.

Referencias

[1] H. M. Deitel y P. J. Deitel, Cómo programar en C/C++ y Java, Pearson Educación, 2004.

[2] E. Peñaloza Romero, Fundamentos de Programación, México: Alfaomega, 2004.