Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

upiita-ipn

Ingeniería Mecatrónica
Programación Avanzada

**REPORTE DE LA PRÁCTICA No. 1**
**Clases**

**Nombre del Alumnos:** Zarazua Aguilar Luis Fernando
**Grupo:** 2MV4 **Fecha:** 22/Agosto/2017

**Objetivo:** Conocer e implementar el concepto de clases.

**Resumen**

Este documento tiene como finalidad mostrar la necesidad del uso de las clases como elementos de programación orientado a objetos, siendo este tipo de programación una ayuda al programador donde ya no se enfoca en las acciones dejando después a los datos como en C, sino que con el paradigma de objetos tiene los datos necesarios presentes y la funciones que los controlan, además se dan a conocer las propiedades que una clase tiene para ocultar la información sin afectar al producto final de lo que se desea realizar, ya que a pesar de no conocer a fondo todos los datos que se usaron, los miembros de la clase podrán entregar correctamente el resultado que se les pide [1].

Por último se muestra como llevar a cabo la implementación de clases en el ambiente de Java mediante la IDE Netbeans, en la cual se compilan y se verifican los resultados de las clases propuestas.

**Introducción**

En el uso de los sistemas electrónicos que se usan actualmente, frecuentemente nos encontramos con dispositivos que nos ayudan a poder resolver de una manera ordenada, clara y rápida problemas involucrados con el procesamiento de la información teniendo dos tareas a realizar con esos datos que básicamente son poder almacenarlos y manejarlos. Esta es una las razones por las que surgió la necesidad de tener una entidad que involucrara de una forma sencilla, pero a la vez estructurada y entendible de almacenar y manejar los datos, con ello tiene origen el concepto de clase que basado en la idea de estructuras nos permite no solo definir nuevos tipos de datos, sino que además se incluye la posibilidad de tener métodos (funciones) que nos ayuda a procesar estructuradamente esta información. Siendo así la clase una entidad que nos permite crear una

especie de plantilla de cómo podemos almacenar y procesar nuestra información, otorgándonos también la facilidad como programadores de poder ir resolviendo los problemas por partes y no todo en un solo conjunto cómo se maneja en c [2].

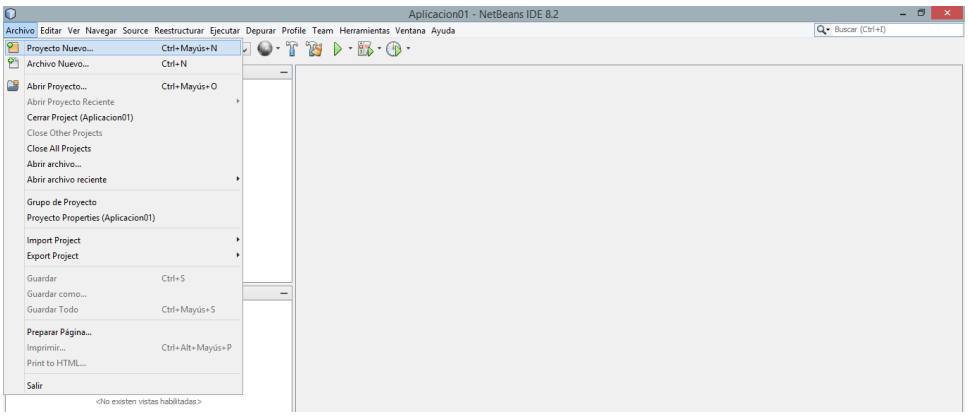
**Planteamiento del problema.**

La necesidad de poder organizar y procesar los datos de una manera más fácil y adecuada, no solamente para quien realiza el programa base, sino para los que posteriormente lo utilizaran, ya sea que se trate de un equipo, o simplemente de la ejecución de un algoritmo para recabar datos por medio de un circuito, es necesario que tanto los datos necesarios como sus procesos estén ahí, con los permisos adecuados para que quien use el programa base no obtenga resultados erróneos o fuera de lo establecido.

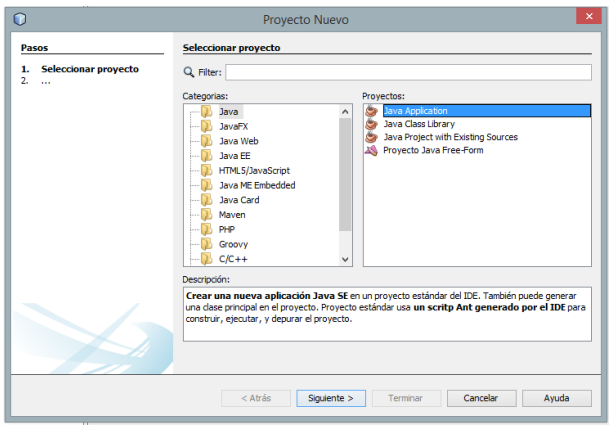
Es por eso que las clases cuentan con métodos y atributos que están restringidos por un tipo de acceso hacia los distintos miembros de la clase, estos pueden dividirse en public, private o protected. A su vez con la idea que las clases puedan ser usadas fácilmente por un distinto programador se incluye la posibilidad de declarar los atributos y métodos como una cabecera donde solo se especifican los tipos de datos que tratan y que devuelven, así como una explicación de lo que realiza, todo esto para que el programador tenga una idea general de toda la clase.

**Desarrollo**

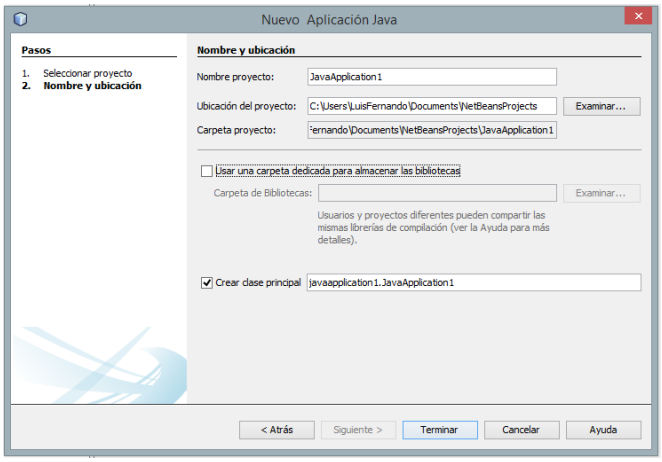
Para la elaboración de un programa en la plataforma de NetBeans se crea un nuevo proyecto:



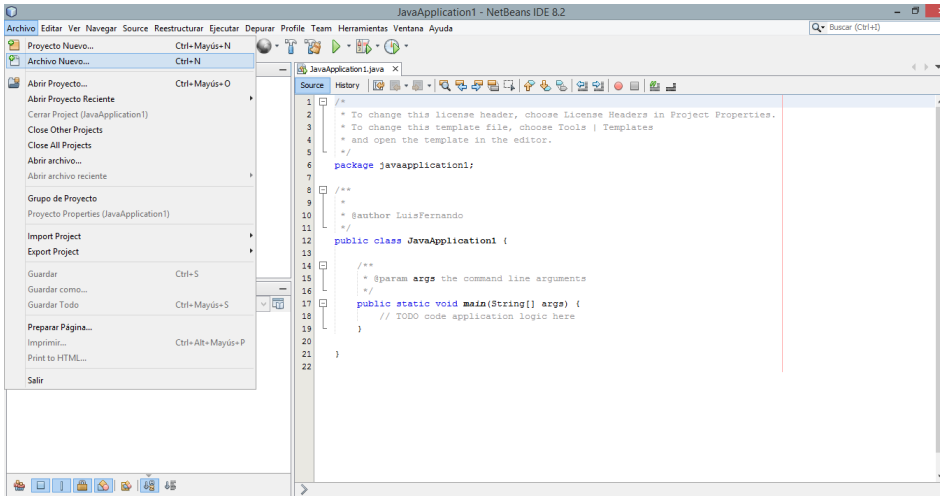

Posteriormente se selecciona Java Application



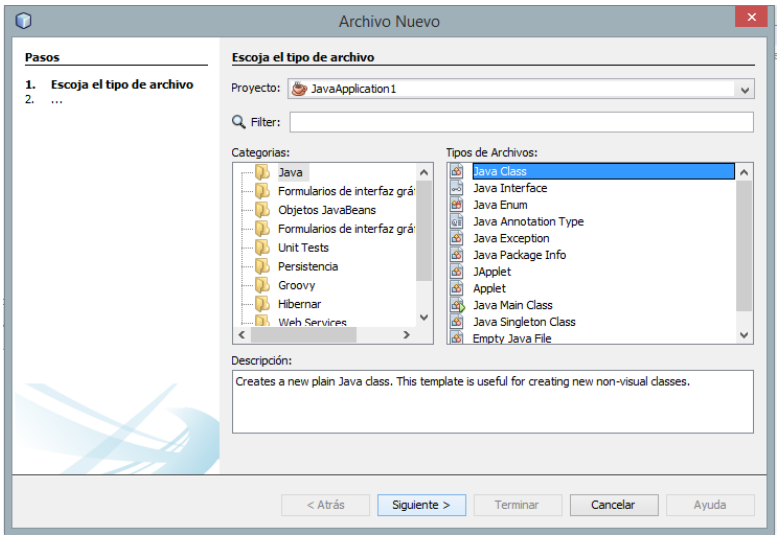
Y se selecciona el lugar donde se guardará el proyecto



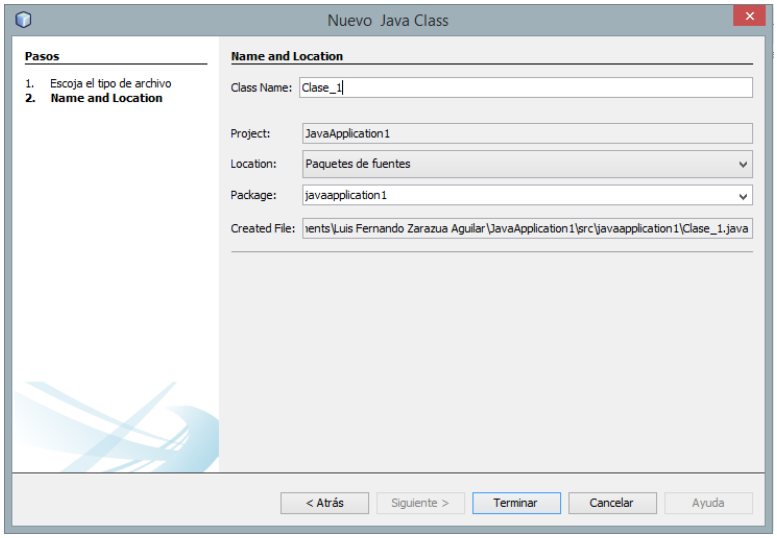

Para agregar una clase se selecciona nuevo archivo




Se selecciona Java Class



Y se nombra a la clase




Una vez ejecutados estos pasos las clases aun sido creadas y agregadas al proyecto.

El uso de las clases se ejemplifica en el siguiente programa donde por medio de clases se puede ejecutar correctamente una aplicación.

Aplicacion01.java

```
package aplicacion01;
import java.util.Scanner;
public class Aplicacion01 {
   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
    Scanner Opcion;
    Parking UPIITA;
    int opc;
    Opcion=new Scanner(System.in);
    do
    {
      System.out.println("Bienvenido al Estacionamiento UPIITA");
      System.out.println("1-Agregar Automovil");
      System.out.println("5-Salir");
      opc=Opcion.nextInt();
    }while(opc!=5);
   }
}
```

Observaciones: Este código representa el programa principal donde se incluyen las demás clases, por medio de la clase Scanner se tiene acceso al flujo de datos proporcionado por el teclado, en este caso para poder elegir entre las opciones que se tienen en el menú de "Estacionamiento UPIITA".

Carro.java

```
package aplicacion01;
public class Carro {
   private String Propietario;
   private String Marca;
   private int Modelo;
  Carro(String P,String Ma, int Mo)
  {
   Propietario=P;
   Marca=Ma;
   Modelo=Mo;
  }
  Carro()
  {
   Propietario="NULO";
   Marca="GENERICO";
   Modelo=1990;
  }
  public void setModelo(int Mo)
  {
      Modelo=Mo;
  }

  public int getModelo()
  {
    return Modelo;
  }
}
```

Observaciones: Este código contiene a la clase Carro la cual tiene como atributos dos tipos de datos declarados como privados: el string (Propietario y Marca) y el otro tipo int (Modelo), en cuanto a sus métodos contiene "setModelo" con el cual se puede modificar el modelo del carro, y el método "getModelo" el cual obtiene el valor del modelo del carro.

Parking.java

```
package aplicacion01;
import java.util.Vector;
public class Parking {
   Vector <Carro> Coches;
   Parking()
   {
   }
   public void addCar(Carro C)
   {
      Coches.add(C);
   }
```

```
    public int getNumCar()
    {
       return Coches.size();
    }

     public Carro getCar(int index)
    {
       if (index<getNumCar())
       {
        return Coches.get(index);
       }
       return null;
    }
}
```

Observaciones: Este código contiene a la clase Parking, en esta clase fue usada la clase vector en la cual guardamos a la clase Carro, teniendo así la posibilidad de tener la cantidad deseada por el usuario de carros que se pueden modificar. Contiene los métodos addCar para agregar elementos tipo Carro al vector, getNumCar para conocer el número de elementos que se tienen y getCar para poder encontrar el carro deseado en el vector.

**Resultados y conclusiones**

En la práctica se observa como por medio de un lenguaje orientado a objetos se logran tener todas las funcionalidades que se tienen en un lenguaje de programación estructurada con la diferencia que en el programa principal solo se ve el resultado de las acciones que se quieren realizar sin tener que ver a fondo todas las partes del programa para comprender como opera este.

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 la clase Carro se tienen 2 constructores, en uno de ellos el usuario da los atributos de la clase, y en el otro constructor se asignan datos por defecto proporcionados por el programado.

En ambas clases Parking y Carro se usan los métodos de la clase String y Vector para poder gestionar la memoria y la edición de la clase.

En conclusión el uso de las clases facilita tanto al programador la forma de compresión de su programa, dándole la facilidad de usar otras clases para administrar datos de sus propias clases, así como al equipo que se esté usando, ya que tiene se administra de una mejor forma la memoria y los procesos que se realizan en este.

**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.