Conceptos POO

¿Qué es una Clase en Java?

La clase es la esencia de Java. Es la base sobre la cual se construye todo el lenguaje Java porque la clase define la naturaleza de un objeto. Como tal, la clase forma la base para la programación orientada a objetos en Java.

1. Qué es una Clase

¿Qué es una clase? Una clase es una plantilla que define la forma de un objeto. Especifica los datos y el código que operará en esos datos. Java usa una especificación de clase para construir

2. La forma general de una clase

```
class NombreClase{

   //Declarar variables de instancia

   tipo var1;

   tipo var2;

   //..

   //Declarar métodos     Hasta ahora hemos visto el método MAIN

   tipo metodo1(parámetros){

      //Cuerpo del método

   }

   tipo metodo2(parámetros){

      //Cuerpo del método

   }

}
```

Conceptos POO

3. Definición de una Clase

```
class Vehiculo{

   int pasajeros;  //número de pasajeros

   int capacidad;  //capacidad en galones

   int mpg;       //consumo de combustible en millas por galón

}
```

4. ¿Cómo se crea un objeto?

```
Vehiculo minivan = new Vehiculo(); //Creando un objeto vehículo llamado minivan
```

```
Objeto.Miembro
```

```
minivan.capacidad = 16
```

Conceptos POO

```
/* Un programa que usa la clase Vehiculo
 El archivo se llama DemoVehiculo.java
 */
class Vehiculo {
   int pasajeros;  //números de pasajeros
   int capacidad;  //capacidad del combustible en galones
   int mpg;       //combustible consumido en millas por galon
}
//Esta clase declara un objeto de tipo Vehiculo
class DemoVehiculo {
   public static void main(String[] args) {
      Vehiculo minivan = new Vehiculo();
      int rango;
      //asignando valores a los campos de minivan
      minivan.pasajeros = 9;
      minivan.capacidad = 15;
      minivan.mpg = 20;
      //Calcular el rango asumiendo un tanque lleno
      rango = minivan.capacidad * minivan.mpg;
      System.out.println("La Minivan puede llevar " + minivan.pasajeros
+ " pasajeros con un rango de " + rango + " millas");
   }
}
```

Salida:

```
La Minivan puede llevar 9 pasajeros con un rango de 300 millas
```

5. Creación de varios objetos

```
//Este programa crea dos objetos Vehiculo
class Vehiculo {
   int pasajeros;  //números de pasajeros
   int capacidad;  //capacidad del combustible en galones
   int mpg;       //combustible consumido en millas por galon
}
//Esta clase declara un objeto de tipo Vehiculo
class DosVehiculo {
   public static void main(String[] args) {
      Vehiculo minivan = new Vehiculo();
      Vehiculo sportscar = new Vehiculo();
      int rango1, rango2;
      //asignando valores a los campos de minivan
      minivan.pasajeros = 9;
      minivan.capacidad = 15;
      minivan.mpg = 20;
      //asignando valores a los campos de sportscar
      sportscar.pasajeros = 10;
      sportscar.capacidad = 25;
```

```
        sportscar.mpg = 30;

        //Calcular el rango asumiendo un tanque lleno

        rango1 = minivan.capacidad * minivan.mpg;

        rango2 = sportscar.capacidad * sportscar.mpg;

        System.out.println("La Minivan puede llevar " + minivan.pasajeros
+ " pasajeros con un rango de " + rango1 + " millas");

        System.out.println("El Sportscar puede llevar " +
sportscar.pasajeros + " pasajeros con un rango de " + rango2 + "
millas");

    }

}
```

Salida:

```
La Minivan puede llevar 9 pasajeros con un rango de 300 millas

El Sportscar puede llevar 10 pasajeros con un rango de 750 millas
```

6. Variables de referencia y asignación

- Cuando se asigna una variable de tipo primitivo a otra, la situación es sencilla. La variable de la izquierda recibe una copia del valor de la variable a la derecha.
- Al asignar una variable de referencia de objeto a otra, la situación es un poco más complicada porque está cambiando el objeto al que se refiere la variable de referencia. El efecto de esta diferencia puede causar algunos resultados contraintuitivos.

```
Vehiculo auto1 = new Vehiculo();

Vehiculo auto2 = auto1;
```

```
auto1.mpg = 28;
```

```
System.out.println(auto1.mpg);

System.out.println(auto2.mpg);
```

¿Adivina qué? Se muestra el mismo valor: 28.

```
Vehiculo auto1 = new Vehiculo();

Vehiculo auto2 = auto1;

Vehiculo auto3 = new Vehiculo();
```

```
auto2 = auto3; //Ahora auto2 y auto3 se refieren al mismo objeto
```

Conceptos POO

Introducción a Objetos en Java

1. Qué es una Clase

- **Modificadores**: una clase puede ser pública o tener acceso predeterminado (default). (Veremos otros más adelante)
- **Nombre de clase**: el nombre debe comenzar con una letra (en mayúscula por convención).
- Superclase (si corresponde): el nombre del elemento primario de la clase (superclase), si lo hay, precedido por la palabra clave **extends**. Una clase solo puede extender (subclase) a uno de los padres.
- **Interfaces** (si corresponde): una lista de interfaces separadas por comas implementadas por la clase, si las hay, precedidas por la palabra clave **implements**. Una clase puede implementar más de una interfaz.
- **Cuerpo**: El cuerpo de la clase rodeado de llaves: {}.

2. Qué es un Objeto

- **Estado**: está representado por atributos de un objeto. También refleja las propiedades de un objeto.
- **Comportamiento**: se representa mediante métodos de un objeto. También refleja la respuesta de un objeto con otros objetos.
- **Identidad**: le da un nombre único a un objeto y permite que un objeto interactúe con otros objetos.

Ejemplo de un objeto: Perro

Identidad: nombre del perro

Estado/atributo: Color, Edad, Raza

Comportamiento: Dormir, comer, ladrar

Conceptos POO

3. Declaración de objetos

```
Perro clifford;
```

4. Inicializando un objeto

Ejemplo:

```
// Declaración de clase

public class Perro
{
   // Variables de instancia
   String nombre;
   String raza;
   int edad;
   String color;

   // Declaración del constructor de clase
   public Perro(String nombre, String raza,
                int edad, String color)
   {
      this.nombre = nombre;
      this.raza= raza;
      this.edad = edad;
      this.color = color;
   }

   // método 1
   public String getNombre()
   {
      return nombre;
   }

   // método 2
   public String getRaza()
   {
      return raza;
```

```
    }

    // método 3
    public int getEdad()
    {
        return edad;
    }

    // método 4
    public String getColor()
    {
        return color;
    }

    @Override
    public String toString()
    {
        return("Hola mi nombre es "+ this.getNombre()+
                ".\nMi raza, edad y color son: " +
                this.getRaza()+"," + this.getEdad()+
                ","+ this.getColor());
    }

    public static void main(String[] args)
    {
        Perro clifford = new Perro("clifford","pitbull", 5, "blanco");
        System.out.println(clifford.toString());
    }
}
```

Salida:

```
Hola mi nombre es clifford.

Mi raza, edad y color son: pitbull,5,blanco
```

```
Perro clifford = new Perro("clifford","pitbull", 5, "blanco");
```

Conceptos POO

5. Maneras de crear el objeto de una clase

- Usar palabra clave (keyword) new: es la forma más común y general de crear objetos en Java.

Ejemplo:

```
// creando un objeto de clase Test

Test t = new Test ();
```

- Usando el método Class.forName (String className): hay una clase predefinida en el paquete java.lang con el nombre Class. El método forName (String className) devuelve el objeto Class asociado con con el nombre de la clase. Tenemos que dar el nombre completo para una clase. Al llamar al método new Instance() en este objeto Class, se devuelve una nueva instancia de la clase con el nombre de cadena proporcionado.

```
// crear un objeto de public class Test

// considerar la clase Test presente en el paquete com.p1

Test obj = (Test)Class.forName("com.p1.Test").newInstance();
```

- Usando el método clone(): el método clone() está presente en la clase Object. Crea y devuelve una copia del objeto.

```
// creando el objeto de la clase Test

Test t1 = new Test ();
```

```
// creación del clon del objeto anterior

Test t2 = (Test)t1.clone();
```

- Deserialización: Deserialización es la técnica de leer un objeto desde el estado guardado en un archivo. Hablaremos más adelante sobre Serialización/Des-serialización en Java.

```
FileInputStream file = new FileInputStream(filename);

ObjectInputStream in = new ObjectInputStream(file);
```

```
Object obj = in.readObject();
```

6. Crear objetos múltiples solo por un tipo (Una buena práctica)

Ejemplo:

```
Test test = new Test();

test = new Test();
```

```
class Animal {}
class Dog extends Animal {}
class Cat extends Animal {}
public class Test
{
   // usando el objeto Dog
   Animal obj = new Dog();
   // usando el objeto Cat
   obj = new Cat();
}
```

7. Objetos anónimos

- Se usan para llamadas de método inmediato.
- Serán destruidos después de llamar al método.
- Son ampliamente utilizados en diferentes bibliotecas. Por ejemplo, en las bibliotecas AWT se utilizan para realizar alguna acción al capturar un evento (por ejemplo, presionar una tecla).

```
btn.setOnAction(new EventHandler()
{
   public void handle(ActionEvent event)
   {
      System.out.println("Hola Mundo!");
   }
});
```

Conceptos POO