# Semana 05

## Clases y Objetos en Java

Las Clases y los Objetos son conceptos básicos de la Programación Orientada a Objetos que giran en torno a entidades de la vida real.

### Clases

1. La clase es un conjunto de objetos que comparten características/comportamiento y propiedades/atributos comunes.
2. La clase no es una entidad del mundo real. Es solo una plantilla, un plano o un prototipo a partir del cual se crean los objetos.
3. La clase no ocupa memoria.
4. La clase es un grupo de variables de diferentes tipos de datos y un grupo de métodos.

Una clase en java puede contener:

- Miembro de datos
- Métodos
- Constructores
- Clase anidada e
- Interfaces

**Sintaxis**

Syntax to declare a class:

```
access_modifier class<class_name>
{
   data member;
   method;
   constructor;
   nested class;
   interface;
}
```

**Ejemplo**

```
class Student {
```

```
        int id; // data member (also instance variable)
        String name; // data member (also instance variable)

        public static void main(String args[])
        {
                Student s1 = new Student(); // creating an object of
                                                                // Student
                System.out.println(s1.id);
                System.out.println(s1.name);
        }
}
```

Una clase es un proyecto o prototipo definido por el usuario a partir del cual se crean objetos. Representa el conjunto de propiedades o métodos que son comunes a todos los objetos de un tipo. En general, las declaraciones de clase pueden incluir estos componentes, en orden:

1. **Modificadores:** una clase puede ser pública o tener acceso predeterminado.
2. **Palabra clave de clase:** la palabra clave de clase se utiliza para crear una clase.
3. **Nombre de la clase:** El nombre debe comenzar con una letra inicial (en mayúscula por convención).
4. **Superclase (si existe):** el nombre del padre de la clase (superclase), si existe, precedido por la palabra clave extends. Una clase solo puede extender (subclase) un padre.
5. **Interfaces (si las hay):** una lista separada por comas de las interfaces implementadas por la clase, si las hay, precedida por la palabra clave implements. Una clase puede implementar más de una interfaz.
6. **Cuerpo:** el cuerpo de la clase está rodeado por llaves, { }.

Los constructores se utilizan para inicializar nuevos objetos. Los campos son variables que proporcionan el estado de la clase y sus objetos, y los métodos se utilizan para implementar el comportamiento de la clase y sus objetos.

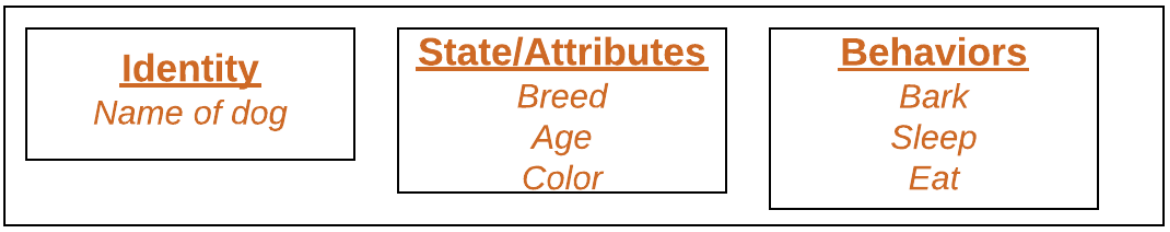
## Objetos

Es una unidad básica de Programación Orientada a Objetos y representa entidades de la vida real. Un programa típico de Java crea muchos objetos que, como sabe, interactúan invocando métodos. Un objeto consta de:

1. **Estado:** Está representado por los atributos de un objeto. También refleja las propiedades de un objeto.
2. **Comportamiento:** Está representado por los métodos de un objeto. También refleja la respuesta de un objeto con otros objetos.

3. **Identidad:** Da un nombre único a un objeto y permite que un objeto interactúe con otros objetos.

**Ejemplo de un objeto: perro**

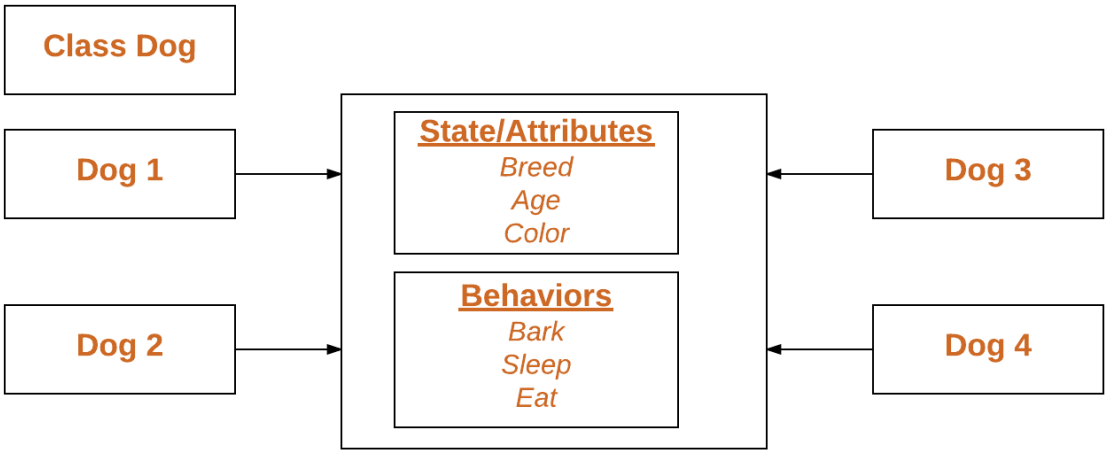



Los objetos corresponden a cosas que se encuentran en el mundo real. Por ejemplo, un programa de gráficos puede tener objetos como "círculo", "cuadrado" y "menú". Un sistema de compras en línea puede tener objetos como "carrito de compras", "cliente" y "producto".

## Declaración de objetos (conocido como instancia de clase)

Cuando se crea un objeto de una clase, se dice que la clase está instanciada. Todas las instancias comparten los atributos y el comportamiento de la clase. Pero los valores de esos atributos, es decir, el estado son únicos para cada objeto. Una sola clase puede tener cualquier número de instancias.

**Ejemplo:**




Como declaramos variables como (type name;). Esto notifica al compilador que usaremos el nombre para referirnos a datos cuyo tipo es tipo. Con una variable primitiva, esta declaración también reserva la cantidad adecuada de memoria para la variable. Entonces, para la variable de referencia, el tipo debe ser estrictamente

un nombre de clase concreto. En general, **no podemos crear objetos** de una clase abstracta o una interfaz.

Dog filadelfio;

Si declaramos una variable de referencia (filadelfio) como esta, su valor será indeterminado (nulo) hasta que se cree un objeto y se le asigne. Simplemente declarar una variable de referencia no crea un objeto.

## Inicialización de un objeto

El operador new instancia una clase asignando memoria para un nuevo objeto y devolviendo una referencia a esa memoria. El operador new también invoca al constructor de clases.

```
// Class Declaration

public class Dog {
	// Instance Variables
	String name;
	String breed;
	int age;
	String color;

	// Constructor Declaration of Class
	public Dog(String name, String breed, int age,
			String color)
	{
		this.name = name;
		this.breed = breed;
		this.age = age;
		this.color = color;
	}

	// method 1
	public String getName() { return name; }

	// method 2
	public String getBreed() { return breed; }

	// method 3
	public int getAge() { return age; }
```

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

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

	public static void main(String[] args)
	{
		Dog fildelfio
			= new Dog("filadelfio", "pug", 5, "cafe");
		System.out.println(tuffy.toString());
	}
}
```

Salida

```
Hola mi nombre es filadelfio.
Mi raza, edad y color son pug,5,cafe
```

## Inicialización usando un método/función

```
/*package whatever //do not write package name here */
public class GFG {
	// sw=software
	static String sw_name;
	static float sw_price;

	static void set(String n, float p)
	{
		sw_name = n;
		sw_price = p;
	}

	static void get()
	{
		System.out.println("El nombre del Software es: " + sw_name);
		System.out.println("El precio del Software es: "
						+ sw_price);
```

```
        }

        public static void main(String args[])
        {
                GFG.set("Visual studio", 0.0f);
                GFG.get();
        }
}
```

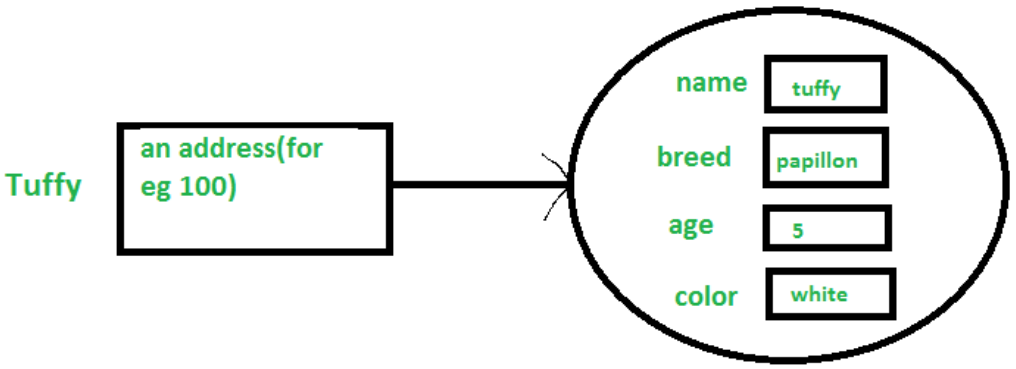
Salida

```
El nombre del Software es: Visual studio
El precio del Software es: 0.0
```

- Esta clase contiene un único constructor. Podemos reconocer un constructor porque
- s usa el mismo nombre que la clase y no tiene tipo de retorno. El compilador de Java diferencia los constructores según el número y el tipo de argumentos. El constructor de la clase Perro toma cuatro argumentos. La siguiente declaración proporciona "filadelfio", "pug", 5, "cafe" como valores para esos argumentos:

  Dog filadelfio = new Dog("filadelfio","pug",5, "cafe");

- El resultado de ejecutar esta declaración se puede ilustrar como:




**Nota:** Todas las clases tienen al menos **un constructor**. Si una clase no declara explícitamente ninguna, el compilador de Java proporciona automáticamente un

constructor sin argumentos, también llamado constructor predeterminado. Este constructor predeterminado llama al constructor sin argumentos del padre de la clase (ya que contiene solo una declaración, es decir, super();), o al constructor de la clase Object si la clase no tiene otro padre (ya que la clase Object es el padre de todas las clases directamente o indirectamente).

## Formas de crear un objeto de una clase

Hay cuatro formas de crear objetos en Java. Estrictamente hablando, solo hay una forma (mediante el uso de una nueva palabra clave), y el resto usa internamente una nueva palabra clave.

- **Usando la palabra reservada new:** Es la forma más común y general de crear un objeto en Java. Ejemplo:

  ```
  // creating object of class 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 la clase con el nombre de cadena dado. Tenemos que dar un nombre completamente calificado para una clase. Al llamar al nuevo método Instance () en este objeto Class, se devuelve una nueva instancia de la clase con el nombre de cadena dado.

  ```
  // creating object of public class Test
  // consider class Test present in com.p1 package
  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.

  ```
  // creating object of class Test
  Test t1 = new Test();

  // creating clone of above object
  Test t2 = (Test)t1.clone();
  ```

- **Deserialización:** La deserialización es una técnica de lectura de un objeto del estado guardado en un archivo.

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

```
ObjectInputStream in = new ObjectInputStream(file);
Object obj = in.readObject();
```

## Creación de múltiples objetos por un solo tipo (una buena práctica)

- En tiempo real, necesitamos diferentes objetos de una clase en diferentes métodos. Crear una cantidad de referencias para almacenarlas no es una buena práctica y, por lo tanto, declaramos una variable de referencia estática y la usamos cuando sea necesario. En este caso, el desperdicio de memoria es menor. Los objetos a los que ya no se hace referencia serán destruidos por Garbage Collector de java. Ejemplo:

  ```
  Test test = new Test();
  test = new Test();
  ```

- En el sistema de herencia, usamos la variable de referencia de la clase principal para almacenar un objeto de subclase. En este caso, podemos cambiar a diferentes objetos de subclase usando la misma variable referenciada. Ejemplo:

  ```
  class Animal { }

  class Dog extends Animal { }
  class Cat extends Animal { }

  public class Test{

      // using Dog object
      Animal obj = new Dog();

      // using Cat object
      obj = new Cat();
  }
  ```

## Diferencias entre Clase y Objeto

| Clase | Objeto |
| --- | --- |
| La clase es el modelo de un objeto. Se | Un objeto es una instancia de la clase. |

| | |
|---|---|
| utiliza para crear objetos. | |
| No se asigna memoria cuando se declara una clase. | La memoria se asigna tan pronto como se crea un objeto. |
| Una clase es un grupo de objetos similares. | Un objeto es una entidad del mundo real, como un libro, un automóvil, etc. |
| La clase es una entidad lógica. | Un objeto es una entidad física. |
| Una clase solo se puede declarar una vez. | Los objetos se pueden crear muchas veces según los requisitos. |
| Un ejemplo de clase puede ser un automóvil. | Los objetos de la clase coche pueden ser BMW, Mercedes, Ferrari, etc. |