JAVA-9 - Programación orientada a objetos - Fundamentos

Objetivos de aprendizaje:

Al final de la sesión, seré capaz de:

- Describir qué son las clases Java.
- Declarar e implementar clases en Java.
- Definir los atributos de las clases usando tipos primitivos y otros objetos.
- Definir el comportamiento de las clases utilizando métodos.
- Explicar e instanciar objetos Java.
- Describir la diferencia entre objetos, clases e instancias.
- Definir constructores de clase.

Objetos, clases, métodos y constructores en Java.

Un ejemplo para clases y objetos en Java

Vamos a imaginar que somos los dueños o administradores de una fábrica de galletas, vendemos diferentes tipos de galletas con diferentes formas, sabores y colores, evidentemente al tener diferentes formas para nuestras galletas necesitaremos varios moldes para las galletas que fabricamos (un molde para cada forma), necesitaremos diferentes recetas para el sabor de cada una y por último diferentes maneras de darle color a las galletas.

¿Qué son las clases en Java?

Una clase en Java se puede entender como un prototipo que define las variables y los métodos comunes a un cierto tipo de instancias, una clase define todo lo que caracteriza y pueden hacer una o varias instancias.

En nuestro ejemplo de la fábrica de galletas, la clase sería uno de los moldes para galletas, junto con los métodos para colorearlas y la receta (método para prepararlas). Quiere decir que en nuestro programa de Java tendremos una manera de especificar las características de la galleta dado que hay diferentes moldes de galletas.

En java las clases son las matrices de las que luego se pueden crear múltiples instancias del mismo tipo. La clase define las variables y los métodos comunes a las instancias de ese tipo (el tipo de la clase creada), pero luego, cada instancia de esta clase tendrá sus propios valores (su propio molde, color y receta) y compartirán las mismas funciones.

En java, primero deberemos crear una clase antes de poder crear instancias o ejemplares de esa clase. Evidentemente primero necesitamos los moldes y demás para poder hacer las galletas.

¿Qué son los objetos en Java?

En Java, un objeto es básicamente una instancia de una clase (las instancias de las que hablábamos hace un momento). Para el ejemplo de la fábrica de galletas, los objetos vendrían siendo cada una de las diferentes galletas obtenidas de los moldes definidos (clases), creados por medio de un proceso o "constructor" de galletas.

¿Cómo crear una clase en Java?

Como dije anteriormente la clase es la que nos dice los componentes del ejemplar que vamos a crear, es decir, una clase contiene los atributos y los métodos que conformarán al ejemplar o instancias, de este modo al momento de crear una clase en Java, debemos especificar el tipo y el nombre (como mínimo) de los atributos y adicionalmente debemos especificar (si existen) los métodos o funciones, el tipo de dato que retornan, el nombre y los parámetros que reciben dichos métodos.

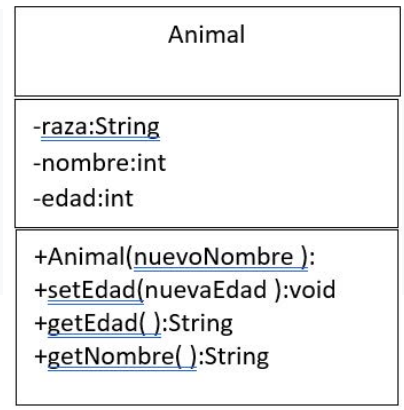
Estructura básica de una clase en Java


```java
//Le damos un nombre "MiClase" a la clase
public class MiClase
{
    //Atributos de la clase
    private String atributo1;
    private int atributo2;
    private float atributo3;

    //Constructor con el mismo nombre de la clase
    public MiClase(){}

    //Métodos de la clase
    public void metodo1()
    {
        //Método vacio
    }

    public String metodo2()
    {
        return "metodo2";
    }
}
```

En el ejemplo anterior hemos creado una clase en Java llamada "MiClase" la cual posee un total de tres atributos (todos ellos privados) y son de tipo String, int y float respectivamente. Adicionalmente esta clase tiene un constructor (que siempre por norma, debe tener el mismo nombre de la clase) el cual no recibe ningún parámetro, aunque pueden recibir todos los parámetros que nosotros deseemos, también tiene un método llamado "metodo1" que no retorna valor alguno (es de tipo void) y otro método llamado "metodo2" que retorna una cadena de caracteres (es de tipo String) con el valor "metodo2". Cabe resaltar que una clase en Java puede tener o no métodos y atributos, sin embargo lo más normal en la mayoría de los casos es que tenga tanto métodos como atributos que la caractericen.

Veamos ahora un ejemplo un poco más completo e ilustrativo.

Ejemplo de clases en Java




```java
package misClases; //Se le declara un paquete

public class Animal
{
    private String raza;
    private String nombre;
    private int edad;

    public Animal(String nuevoNombre)
    {
        nombre = nuevoNombre; //Se le da un nombre al animal
    }

    //Método para obtener la edad del animal
    public int getEdad()
    {
        return edad;
    }

    //Método para establecer la edad del animal
    public void setEdad(int nuevaEdad)
    {
        edad = nuevaEdad;
    }

    //Método para obtener el nombre del animal
    public String getNombre()
    {
        return nombre;
    }
}
```

La clase Java que hemos creado para este ejemplo tiene como nombre "Animal", pertenece al paquete "misClases" y posee los atributos raza, nombre y edad, adicionalmente tenemos un constructor que recibe un nombre y se lo asigna al animal y tres métodos encargados de obtener y establecer la edad del animal y el restante para obtener el nombre.

Objetos en Java

Objetos en Muy bien, ya hemos aprendido a la perfección cómo crear clases en java, y cuáles son sus componentes esenciales, ahora vamos a aprender a crear objetos en Java, cómo se hace, veremos brevemente la utilidad y funcionamiento de los constructores y como usar los métodos

¿Cómo crear objetos en Java?

Al momento de crear objetos en Java, debemos tener claras dos cosas indispensables, la primera es el nombre de la clase para la cual vamos a crear el objeto y segundo el constructor que dicha clase posee, es decir, si el constructor recibe o no parámetros.

Para crear objetos en Java, el lenguaje nos proporciona el comando new, con este comando le decimos a Java que vamos a crear un nuevo objeto de una clase en especifico y le enviamos los parámetros (en caso de ser necesario) según el constructor, veamos un ejemplo.

Ejemplo de objetos en java

Vamos a crear un objeto o instancia en Java para la clase que hemos creado al comienzo llamada MiClase. Esta clase tiene un constructor que no recibe parámetros, por lo cual no es necesario enviar algún tipo de valor al momento de crear el objeto, veamos entonces la sintaxis para crear un objeto del tipo MiClase en java.

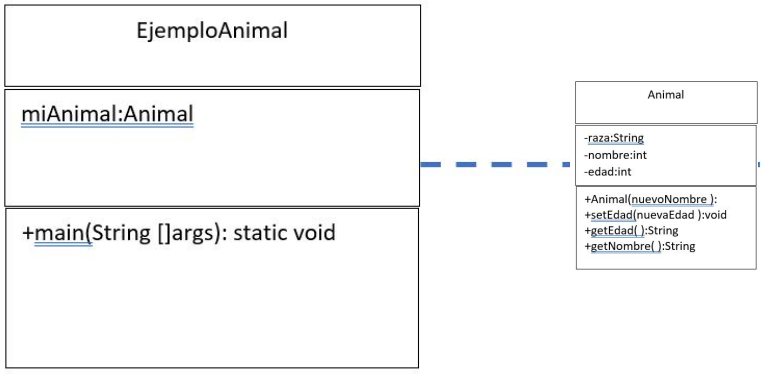
```
MiClase miObjeto;
//Declaramos una variable del tipo de la clase
miObjeto = new MiClase();
//Aquí ya hemos creado un objeto de MiClase
```

Otra forma de hacer y que también es válida es la siguiente:

```
MiClase miObjeto = new MiClase();
//Declaramos y creamos el objeto en una= línea
```

Ejemplo 2 de objetos en Java

Vamos ahora a crear un objeto o instancia para la clase Animal, en esta ocasión tenemos un constructor que recibe un parámetro (el nombre del animal) y que posee tres métodos de los cuales haremos uso para este ejemplo.




```java
import misClases.Animal; //Importamos la clase Animal para poder usarla

public class Ejemplo
{
    public static void main(String[] args)
    {
        //Creamos un animal cuyo nombré será Falco
        Animal miAnimal = new Animal("Falco");
        //Le establecemos 3 años de edad a Falco.
        miAnimal.setEdad(3);
        //Mostraremos el nombre del animal por pantalla
        System.out.println("El nombre es: " + miAnimal.getNombre());
        //Mostramos la edad del animal por pantalla
        System.out.println(" y tiene " + miAnimal.getEdad() + " años");
        //Este código deberia imprimir "El nombre es: Falco y tiene 3 años"
    }
}
```

Nota: Es importante notar que es similar hacer uso de método set para establecer atributos y usar el constructor enviandole parámetros, es decir, el resultado es el mismo al usar un constructor vacío y luego establecer los valores de los atributos por medio de una función set o usar un constructor que reciba una serie de parámetros que usará para establecer valores a los atributos. La diferencia radica esencialmente en la agilidad y número de líneas, al usar el constructor y enviar los parámetros, el constructor hace todas las asignaciones por nosotros y usamos una sola línea de código (normalmente) mientras que al usar métodos set debemos usar un método set por cada atributo y si tenemos cinco o más atributos ya se vuelve un poco molesto hacerlo además de tener que usar un constructor de todas formas. Veamos un corto ejemplo para estar más claros.

Constructor vs mutador (set)

En programación se conoce como método set al método encargado de establecerle un valor a un atributo según un parámetro recibido (ver setEdad(...) en el ejemplo de clases en Java).

Una pregunta que nos puede surgir es si debemos usar un constructor con parámetros o los métodos set de los atributos para establecer los valores de los atributos del objeto. En realidad es lo mismo en cuanto a resultados u términos generales veamos:

Ejemplo 1: Usando constructor

```java
public class Aclaracion
{
    private int atributo1;
    private int atributo2;
    private String atributo3;

    //Declaramos un constructor
    public Aclaracion(int attr1, int attr
                                      r2, String attr3)
    {
        atributo1 = attr1;
        atributo2 = attr2;
        atributo3 = attr3;
    }

    public static void main(String[] args
                                        s)
    {
        Aclaracion ac = new Aclaracion(5,  , 10, "x");//Creamos un objeto enviando parámetros al constructor

        System.out.println(ac.atributo1 +  + ", " + ac.atributo2 + ", " +ac.atributo3);//Mostramos el valor de los atributos
        //Imprime '5, 10, x'
    }
}
```

En este ejemplo hemos hecho uso del constructor para establecer le los valores a los atributos del objeto. Ahora hagamos lo mismo usando métodos set o accediendo directamente a cada atributo.

Ejemplo 2: Usando métodos set

```
public class Aclaracion{
   private int atributo1;
   private int atributo2;
   private String atributo3;
   public void setAtributo1(int attr1){
      atributo1 = attr1;
   }
public void setAtributo2(int attr2){
      atributo2 = attr2;
   }
   public void setAtributo3(String attr3){
      atributo3 = attr3;
   }
public static void main(String[] args) {
Aclaracion ac1 = new Aclaracion();//Creamos un objeto con constructor vacío
Aclaracion ac2 = new Aclaracion();//Creamos otro objeto con constructor vacío
//Establecemos los valores de los atributos usando métodos set de cada uno
ac1.setAtributo1(5);
ac1.setAtributo2(10);
ac1.setAtributo3("x");
//Establecemos los valores de los atributos accediendo directamente a cada uno
ac2.atributo1 = 5;
ac2.atributo2 = 10;
ac2.atributo3 = "x";
 System.out.println(ac1.atributo1 + ", " + ac1.atributo2 + ", " +ac1.atributo3);
//Mostramos el valor de los atributos de ac1
        //Imprime '5, 10, x'
System.out.println(ac2.atributo1 + ", " + ac2.atributo2 + ", " +ac2.atributo3);
//Mostramos el valor de los atributos de ac2
        //Imprime '5, 10, x'
```

Como podrás evidenciar hacer ambas cosas nos permiten llegar al mismo resultado, sin emabrgo en el ejemplo 2, podemos ver como el número de líneas aumentó considerablemente y fue más engorroso programarlo. En todo caso siempre es buena práctica declarar los métodos set y get para cada atributo, pues si debemos modificar el valor de un atributo cualquiera, debemos hacerlo por medio de un método get y no creando un objeto nuevo.

Nota: En el ejemplo 2, podrás ver que hicimos uso de los métodos set para establecer valores en el objeto ac1 mientras que para el objeto ac2 hicimos asignaciones directas. Esto lo pudimos hacer debido a que los atributos son privados y estamos accediendo a ellos desde la propia clase. Eso mismo no se podría hacer desde otra clase, pues deberíamos usar el método set (que son públicos), (publico, privado modificadores de acceso).

En este punto debemos saber cómo crear una clase, qué son atributos y métodos de una clase en Java, qué es y para qué sirve un constructor (forma muy básica), cómo crear objetos en Java y cómo hacer uso de los métodos de una clase desde los objetos de la misma.

Watch Video At: https://youtu.be/cUj8CiYIbyI

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

Clase
Una clase es un plano 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.

Objeto
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 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.

INSTANCIAS
Una instancia es un elemento tangible (ocupa memoria durante la ejecución del programa) generado a partir de una definición de clase. Todos los objetos empleados en un programa han de pertenecer a una clase determinada.

Aunque el término a veces se emplea de una forma imprecisa, un objeto es una instancia de una clase predefinida en Java o declarada por el usuario y referenciada por una variable que almacena su dirección de memoria.

Modificadores de Acceso
Como su nombre indica, los modificadores de acceso en Java ayudan a restringir el alcance de una clase, constructor, variable, método o miembro de datos. Hay cuatro tipos de modificadores de acceso disponibles en Java:

Default – No se requiere palabra clave
Private
Protected
Public

Modificador/Acceso	Clase	Paquete	Subclase	Todos
public	Sí	Sí	Sí	Sí
protected	Sí	Sí	Sí	No
default	Sí	Sí	No	No
private	Sí	No	No	No

El acceso a las propiedades y métodos se determina mediante las palabras reservadas de los modificadores de acceso, en Java hay cuatro modificadores de acceso que definen ámbitos de visibilidad de más restrictivos a menos restrictivos:

private: únicamente la clase puede acceder a la propiedad o método.
private (valor por defecto si no se indica ninguno): solo las clases en el mismo paquete pueden acceder a la propiedad o método.
protected: las clases del mismo paquete y que heredan de la clase pueden acceder a la propiedad o método.
public: la propiedad o método es accesible desde cualquier método de otra clase.