Programación Orientada a Objetos
Profesor: Rodolfo Baspineiro

Módulo 1: Introducción a la Programación Orientada a Objetos

Sesión 1: ¿Qué es Java?
Octubre 18 de 2021

Java:
- Lenguaje de Programación de uso general
- De alto nivel
- Orientado a objetos
- Fuertemente tipado
- Compilado (a bytecode) e interpretado (por la JVM)
- Independiente de la plataforma

Tipos de datos en Java

Lenguaje fuertemente tipado: Exige una declaración explícita de la variable antes de empezar a usarla.

```
int valor;
double coeficiente;
String nombre;
```

Para obtener resultados reales al operar con enteros:

```
int valor = 15;
double cociente;
cociente = valor/2.0;

int valor1 = 15;
int valor2 = 2;
double cociente;
cociente = (double)valor1/valor2;
```

Declaración de variables:

```
string nombre = “Andy”;
```

Tipos de datos primitivos

Tipos de datos primitivos	
byte	Números enteros entre -128 y 127
short	Números enteros entre -32768, 32767
int	Números enteros entre -2147483648 y 2147483647
long	Enteros muy grandes, entre -9223372036854775808 y 9223372036854775807
float	Número con coma -3.402823e38 a 3.402823e38
double	Número con coma, mayor capacidad -1.79769313486232e308 a -1.79769313486232e308
string	Cadena de caracteres
char	Un carácter (Ej: 'a') Unicode
boolean	Verdadero o falso (true /false)

Pregunta:

- ¿Cómo instalar una nueva versión de Java? Actualmente tengo 1.8.0_111.

Sincrónico

Sesión 2: Introducción a Java
Octubre 20 de 2021

String, Integer y Float

Las clases Integer y Float son equivalentes a los tipos primitivos, pero proporcionan funcionalidades adicionales.

- Para comparar objetos de una clase:
`nombre.equals(“Juan”);` // El operador == solo aplica para tipos primitivos

- Para comparar por relaciones de orden:
`.compareTo()`

- Un objeto que no se ha inicializado tiene por defecto el valor `null`.

- Package: Paquete que permite agrupar clases.

Ingreso de datos, Scanner

```
Scanner scanner;
scanner = new Scanner(System.in);
int num1;
float coeficiente;
String nombre;
char inicial;

// Lectura de un número entero
System.out.println(“Ingrese primer valor”);
num1= scanner.nextInt();

// Lectura de un número decimal
System.out.println(“Ingrese el coeficiente”);
coeficiente= scanner.nextFloat();

// Lectura de una línea de texto
System.out.println(“Ingrese su nombre”);
nombre= scanner.nextLine();
inicial= nombre.charAt(0);
```

Métodos de Strings

```
.length()
.toUpperCase()
.equals()
.charAt()
```

Clase Integer

Declaración:

```
Integer valor = 0;
Integer valor = new Integer(1);
```

Comparar dos enteros:

```
valor1.equals(valor2); // Devuelve true si los valores son iguales, o false en caso contrario

valor1.compareTo(valor2); // Si valor1 > valor2 retorna 1; si valor1 < valor2, retorna -1. Si los valores son iguales retorna 0.
```

Clase Float

Declaración:

```
Float coeficiente = 2.5f;
Float num = new Float(0.5);
```

Clase Date

```
Date fecha = new Date(); // Crea un objeto de clase Date cuyo valor es la fecha actual.

Date fecha2 = new Date(120, 11, 5); // Corresponde al 5 de diciembre de 2020. Enero es el mes 0 y diciembre el mes 11. El año se cuenta a partir de 1900.
System.out.println(fecha2.toString()); // Devuelve 5/12/2020.
```

Funciones:

Función con return:

```
int suma(int num1, num2)
{
     return num1,num2;
}
```

Función sin return:

```
void mostrarMensaje(String mensaje)
{
     System.out.println(mensaje);
}
```

Array

Los arrays son estructuras estáticas que permiten almacenar elementos del mismo tipo.
Los arrays tienen longitud fija, que debe definirse al momento de crear el objeto. No es posible eliminar elementos.

```
String[] nombres = new String[5];
```

Para recorrer un array:

```
for(String nombre : nombres) {
     System.out.println(nombre);
}
```

Sesión 3
Octubre 21 de 2021

Sesión 4: Objetos y UML
Octubre 25 de 2021

Clases e Instancias
Modelar los aspectos de la vida real que influyen en nuestro contexto.

Atributos, responsabilidades y constructor

Mascota
```
nombre: String
color: String
especie: String
```

Una clase es un nuevo tipo de dato.

Constructor:

```
Mascota (nombre: String, color: String, especie: String) {
}
```

Encapsulamiento
Nadie puede ver o modificar directamente la información que forma la estructura interna del objeto, solamente el propio objeto.

- Los atributos de los objetos se declaran como privados.
- Los métodos que sean públicos serán vistos por los otros objetos.
- Se utilizan métodos públicos para permitir ver o modificar las características de los objetos.
- Para cambiar el valor de un atributo se usa un método set, por ejemplo, para cambiar el nombre será setNombre(String)
- Para obtener el valor de un atributo se usa un método get, por ejemplo, para saber el nombre será getNombre(): String.

Diagrama UML

UML: Unified Modeling Language. Es un lenguaje de modelado, de propósito general, usado para la visualización, especificación, construcción y documentación de sistemas orientados a objetos.

Diagrama de clases

Es un diagrama UML de estructura que muestra una vista estática de la estructura del sistema, o de una parte de este, describiendo qué atributos y comportamiento debe desarrollar con los métodos necesarios para llevar a cabo las operaciones del sistema. El diagrama de clases muestra qué clases forman el sistema y las relaciones entre ellas.

Nombre
Atributos. Se indica la visibilidad, el nombre y el tipo de dato.
Métodos. Se indica la visibilidad, el nombre, los parámetros y el tipo de dato que retorna el método.

Niveles de visibilidad:
- Privado
+ Público
# Protegido

Software para diagramas UML: Drawio
http://app.diagrams.net

Sincrónico

Normalmente, no se escriben los setters y getters en el diagrama UML. En ocasiones, no se muestra el constructor.

Sesión 5: Clases
Octubre 27 de 2021

Nombres en Java:
atributos, métodos y objetos: camelCase
Clases: MayusculaInicial
paquetes: minuscula
constantes: MINUSCULAS_SEPARADAS_CON_GUION_BAJO

Ejemplo de una clase en Java:

Los nombres de las clases inician con mayúscula y deben estar en singular. El nombre del archivo .java debe coincidir con el nombre de la clase.

```
public class Articulo{

	// atributos
	private String descripcion;
	private double precioVenta;
	private int stock;

	// constructor
	public Articulo(String descripcion, int cantidad, double
precio) {
		this.descripcion=descripcion;
		precioVenta=precio;
		stock=cantidad;
	}

	// getters
	public String getDescripcion() {
		return descripcion;
	}
	public double getPrecioVenta() {
		return precioVenta;
	}
	public int getStock() {
		return stock;
```

```
    }

    // métodos
    public boolean hayStock(){
        return stock>0;
    }
    public double consultarPrecio(){
        return precioVenta;
    }

}
```

Crear una instancia

```
Articulo articulo = new Articulo(“Artículo 1”, 50, 1400.00);
```

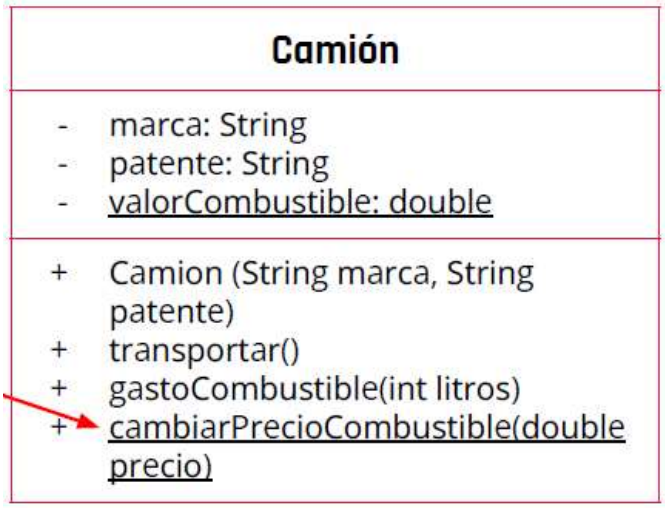
Otra forma:

```
Articulo articulo = new Articulo(descripcion: “Artículo 1”, stock:
50, precioVenta: 1400.00);
```

Variables y métodos de Clase

Variables de clase
Son aquellas variables o atributos que guardan valores comunes a todos los objetos. Se muestran subrayadas en el diagrama UML.

Métodos de clase
Son métodos que se pueden utilizar directamente con la clase, sin necesidad de instanciar o crear un objeto. Aparecen subrayados en el diagrama UML.



Para definir atributos o métodos de clase se utiliza la palabra reservada static. Los atributos y métodos static no se pueden usar con una instancia, sino que operan directamente al nivel de la clase.

Ejemplo:

```
public class Camion {
	private String marca;
	private String patente;
	static private double valorCombustible; // Atributo de clase
	public Camion(String marca, String patente){
		this.marca=marca;
		this.patente=patente;
	}
	public double gastoCombustible(int litros){
		return litros*Camion.valorCombustible;
	}
	static public void cambiarPrecioCombustible(double precio){
		Camion.valorCombustible=precio; // Método de clase
	}
}

public class Main {
	public static void main(String[] args) {
		Camion miCamion = new Camion("Ford",”AB XXX CD”);
		Camion.cambiarPrecioCombustible(98.50);
		Sytem.out.println("Gasto " +
	miCamion.gastoCombustible(40));
	}
}
```

Sincrónico

Sesión 6

Sincrónico

La clase Date es obsoleta. Luego se creó la clase Calendar y actualmente se usa LocalDate.