



UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERIA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE PROGRAMACIÓN DE COMPUTADORAS

DESARROLLO DE SOFTWARE II
1LS116

PROFESOR :
ING. JOSÉ JAVIER CHIRÚ F.

II SEMESTRE
2022

PROGRAMACIÓN ORIENTADA A OBJETOS CON JAVA

MÓDULO 2

PROGRAMACIÓN



Un programa es una secuencia lógica de instrucciones escritas en un determinado lenguaje de programación que dicta a la computadora las acciones que debe llevar a cabo. Una instrucción es una orden que se le da a la máquina para que ejecute una acción



Programación orientada a objetos se puede como una técnica o estilo de programación que utiliza objetos como bloques esenciales de construcción. Los elementos básicos de la POO son: objetos, mensajes, métodos y clases.

Características de la Programación Orientada a Objetos



Abstracción

Encapsulamiento

Herencia

Polimorfismo

Abstracción

La abstracción consiste en captar las características y funcionalidades que un objeto desempeña y estos son representados en clases por medio de atributos y métodos de dicha clase.

La abstracción depende principalmente del interés del observador, permitiendo abstracciones muy diferentes de la misma realidad, debido a esto algunos la clasifican un tanto subjetiva.

Abstracción

Qué características podemos abstraer de la clase persona? O también podríamos preguntarnos, ¿Qué características semejantes tienen todas las personas?, no importando si son 1, 10 ó 1 millón de personas, nosotros podemos abstraer las características y funcionalidades que en un momento en particular nos sirven para darle solución a un problema, para este caso se definen como características

Abstracción

Características (Atributos)

- El nombre
- Apellidos
- Edad
- Dirección

Como funcionalidades (Métodos)

- Hablar
- Comer
- Dormir
- Trabajar.

Encapsulamiento



Encapsular consiste en hacer que los atributos o métodos internos a una clase no se puedan acceder ni modificar desde fuera, sino que tan solo el propio objeto pueda acceder a ello.



El encapsulamiento es la propiedad que permite asegurar que el contenido de la información de un objeto será accedido correctamente

Herencia

La herencia básicamente consiste en que una clase puede heredar sus variables y métodos a varias subclases (la clase que hereda es llamada superclase o clase padre). Esto significa que una subclase, aparte de los atributos y métodos propios, tiene incorporados los atributos y métodos heredados de la superclase.

Polimorfismo

Se refiere a la posibilidad de definir clases diferentes que tienen métodos o atributos denominados de forma idéntica, pero que se comportan de manera distinta. El concepto de polimorfismo se puede aplicar tanto a funciones como a tipos de datos. Así nacen los conceptos de funciones polimórficas y tipos polimórficos

ESTRUCTURA GENERAL DE UNA APLICACIÓN OO

Estructura general de un programa OO

- Import: Declaración para importar clases desde paquetes
- nombre del programa: El nombre del archivo fuente debe coincidir con el nombre de la clase principal.
- Atributos
- Métodos



Estructura general de un programa OO

```
//Importación de Paquetes y Librerías  
package Ejemplo;
```

```
public class Suma {  
    //Declaración de Atributos  
    2 usages  
    private int num1, num2;  
  
    //Declaración de métodos  
    public void asignar(int num_1, int num_2){  
        num1=num_1;  
        num2=num_2;  
    }  
  
    //Declaración de métodos  
    public int sumar(){  
        //Declaración de variables  
        int res;  
        //Sentencias  
        res=num1+num2;  
        return res;  
    }  
}
```

DIAGRAMA UML

El Lenguaje unificado de modelado (UML) desempeña un papel importante en el desarrollo de software, pero también en otros sistemas de muchos sectores de la industria, ya que es un medio de mostrar visualmente el comportamiento y la estructura de un sistema o un proceso. UML ayuda a identificar posibles errores en las estructuras de la aplicación, el comportamiento del sistema.

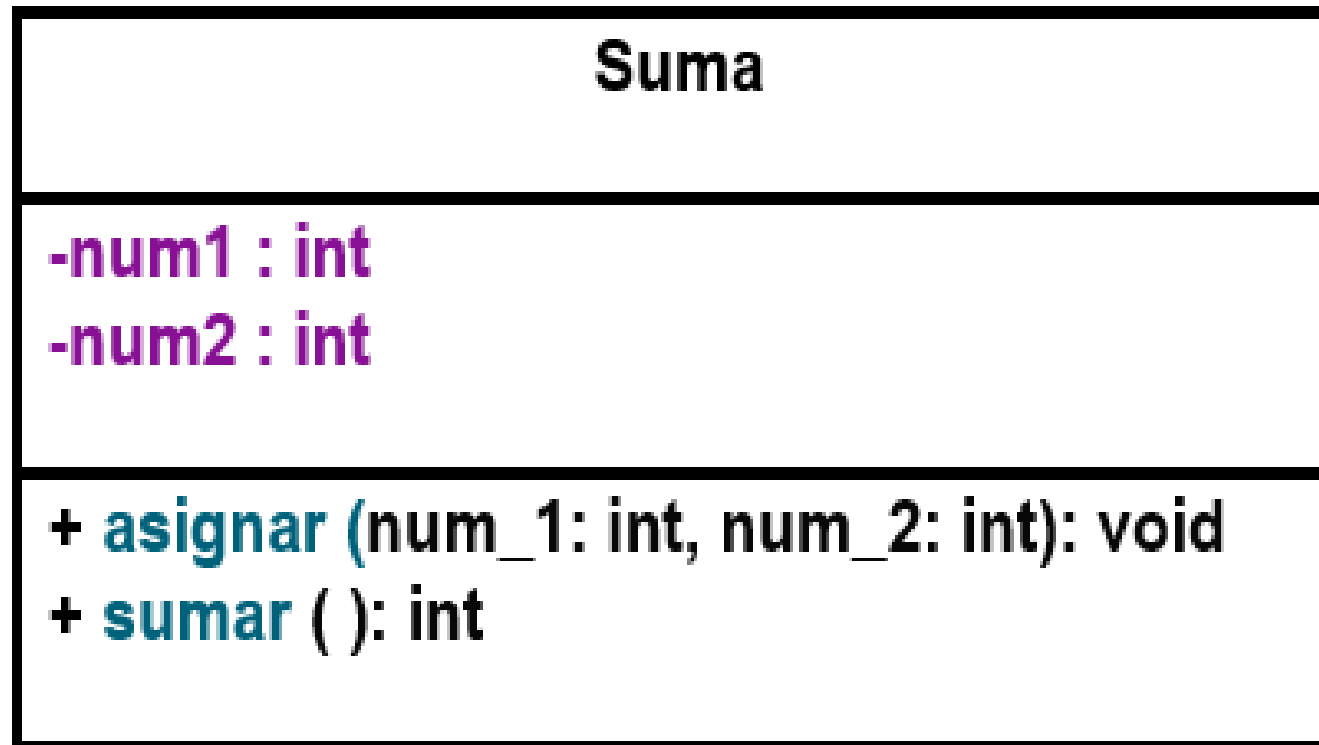
DIAGRAMA UML

- **Nombre de la Clase**
- **Atributos:** Describen las características de los objetos: tipo de acceso y tipo de dato
- **Métodos:** las instrucciones necesarias para realizar un proceso o tarea específicos.
- **Los tipos de acceso**
 - ❖ **Públicos:** Son accesibles desde cualquier parte del programa. Se representan con el signo más (+).
 - ❖ **Privados:** Son accesibles sólo por los métodos que pertenecen a la clase. Se representan con el signo menos (−).
 - ❖ **Protegidos:** Se utilizan sólo con la herencia. Se representan por el signo de número (#).

DIAGRAMA UML



DIAGRAMA UML



ELEMENTOS BÁSICOS DE LA PROGRAMACIÓN EN JAVA

Comentarios

Los comentarios, son líneas de código, que no son tenidas en cuenta por el compilador en el momento de ejecutar nuestra aplicación (es como si no estuviesen allí), por lo tanto no están sujetas a restricciones de sintaxis ni nada similar y podremos escribir cualquier cosa en éstas.

Comentarios de varias líneas

Los comentarios de varias líneas se incluyen entre los símbolos `/*` y `*/`, como en C y C++.

```
/* Este es un ejemplo de  
un comentario de varias Líneas.
```

```
*/
```

Comentarios de una sola línea

Para comentar una sola línea se utiliza la doble diagonal **//**. El comentario se inicia cuando se encuentra la doble diagonal y continua hasta el final de la línea.

// Este es un comentario de una sola linea

// Este es otro comentario

LOS IDENTIFICADORES

Son palabras creadas por el programador para denominar los elementos que necesita declarar en un programa, tales como variables, clases, métodos, objetos y estructuras de datos.

Los identificadores deben crearse de acuerdo al estilo de escritura **camelCase**

Los identificadores de clases comienzan con mayúscula; para cualquier otro elemento deben iniciar con minúscula.

Algunos ejemplos de identificadores aplicando **camelCase** son: estado, edad, nombre, numSeguro, calcularSuma, cPersona.

TIPOS DE DATOS EN JAVA

Java cuenta con un pequeño conjunto de tipos de datos primitivos. Podríamos considerarlos fundamentales, ya que la mayor parte de los demás tipos, los tipos estructurados o complejos, son composiciones a partir de estos más básicos. Estos tipos de datos primitivos sirven para gestionar los tipos de información más básicos, como números de diversas clases o datos de tipo verdadero/falso (también conocidos como "valores booleanos" o simplemente "booleanos").

TIPOS DE DATOS EN JAVA

Tipo de dato	Representación	Tamaño (bytes)	Rango de valores	Valor por defecto.	Clase Asociada.
byte	Número entero con signo	1	-128 a 127	0	Byte
short	Número entero con signo	2	-32,768 a 32,767	0	Short
Int	Número entero con signo	4	-2147483648 a 2147483647	0	Integer
long	Número entero con signo	8	-9223372036854775808 a 9223372036854775807	0	Long
float	Número en coma flotante de precisión simple	4	$\pm 3.4 \times 10^{-38}$ a $\pm 3.4 \times 10^{38}$	0.0	Float
double	Numérico en coma flotante de precisión doble	8	$\pm 1.8 \times 10^{-308}$ a $\pm 1.8 \times 10^{308}$	0.0	Double
char	Carácter Unicode	2	\u0000 a \uFFFF	\u0000	Character
Boolean	Dato lógico	-	true ó false	false	Boolean

Variables

Para declarar variables en JAVA utilizaremos el siguiente formato:

TipoDato nombreIdentificador

- ❖ **tipo de dato** : tipo de datos que se pueden almacenar en esta
- ❖ **Nombre identificador**: nombre dado a la variable

Constantes en Java

Una constante es una variable del sistema que mantiene un valor inmutable a lo largo de toda la vida del programa. Las constantes en Java se definen mediante el modificador **final**.

La estructura sería:

```
final tipoDato nombreConstante = valor;
```

Ejemplo :

```
final int DIASSEMANA = 7;
```

OPERADORES ARITMÉTICOS

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
-	operador unario de cambio de signo	-4	-4
+	Suma	2.5 + 7.1	9.6
-	Resta	235.6 - 103.5	132.1
*	Producto	1.2 * 1.1	1.32
/	División (tanto entera como real)	0.050 / 0.2 7 / 2	0.25 3
%	Resto de la división entera	20 % 7	6

OPERADORES ARITMÉTICOS COMBINADOS

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
+=	Suma combinada	$a+=b$	$a=a+b$
-=	Resta combinada	$a-=b$	$a=a-b$
=	Producto combinado	$a=b$	$a=a*b$
/=	División combinada	$a/=b$	$a=a/b$
%=	Resto combinado	$a\%=b$	$a=a\%b$

OPERADORES DE RELACIÓN

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
<code>==</code>	igual que	<code>7 == 38</code>	false
<code>!=</code>	distinto que	<code>'a' != 'k'</code>	true
<code><</code>	menor que	<code>'G' < 'B'</code>	false
<code>></code>	mayor que	<code>'b' > 'a'</code>	true
<code><=</code>	menor o igual que	<code>7.5 <= 7.38</code>	false
<code>>=</code>	mayor o igual que	<code>38 >= 7</code>	true

OPERADORES LÓGICOS

Operador	Significado
	OR lógico (ó)
&&	AND lógico (y)
!	NOT lógico (no)

JERARQUÍA DE LAS OPERACIONES

$$5 + (6 + 2) - 4 \div \sqrt{16} =$$

- 1 Paréntesis.
- 2 Potencias y raíces.
- 3 Multiplicaciones y divisiones.
- 4 Sumas y Restas.



JERARQUÍA
DE
OPERADORES

SENTENCIAS DE ASIGNACIÓN

Una sentencia de asignación va a permitir poder asignar (almacenar) como su mismo nombre lo dice un valor a una variable que nosotros hayamos definido con anterioridad. Por lo tanto, lo que se defina al lado derecho del signo igual se almacenará en la variable situada al lado izquierdo del signo igual. Para ello, es que nosotros usamos la sentencia de asignación, para almacenar valores a las variables.

Una sentencia de asignación tiene la siguiente forma:

variable = **expresión**;

SENTENCIAS DE ASIGNACIÓN

```
// Declara las variables p, q y r  
int p, q, r;  
// Asigna el valor 2 a la variable p  
p = 2;  
// Asigna una copia del valor de p a la variable q  
q = p;  
// Evalúa el valor de la expresión 2*p + q se asigna a la variable r  
r = 2*p + q;
```


Construcción de clases y objetos en Java

Las clases en Java son plantillas para la creación de objetos, en lo que se conoce como programación orientada a objetos, considerada uno de los principales paradigmas de desarrollo de software en la actualidad.

Dentro de una **clase** se definen los datos y el código que actúa sobre esos datos. El código está contenido en métodos. Tanto las clases, como los objetos y los métodos son fundamentales para Java.

Miembros de una clase

Las clases también son denominadas tipos abstractos de datos. Cada clase contiene datos, así como un conjunto de funciones que manipulan estos datos. A los datos que componen una clase se les llama **datos miembro**. A los métodos que componen una clase se les llama **métodos miembro**.



Miembros de una clase

```
//Nombre de la Clase
public class Suma {
    // Declaracion de Variables
    //Datos miembro
    private int num1, num2;
    //Declaracion de metodos
    //métodos miembro.
    public void asignar(int num_1,int num_2){
        //Setencias
        num1=num_1;
        num2=num_2;
    }
    //Declaracion de metodos
    //métodos miembro.
    public int sumar(){
        //Setencias
        int res;
        res=num1+num2;
        return res;
    }
}
```

MÉTODOS

Un método es un conjunto de instrucciones definidas dentro de una clase, que realizan una determinada tarea y a las que podemos invocar mediante un nombre.

La estructura general de un método Java es la siguiente:

```
tipoDeAcceso tipoDeDato nombreMetodo([lista parámetros])  
{  
    // instrucciones  
}
```

MÉTODOS

Dependiendo del tipo de dato que enviemos al método en Java, podemos diferenciar dos comportamientos:

- **Paso por valor:** Se crea una copia local de la variable dentro del método.
- **Paso por referencia:** Se maneja directamente con el objeto, los cambios realizados dentro del método le afectarán también fuera.

MÉTODOS

```
//método tipo sin retorno, con dos parametros  
public void asignar(int num_1,int num_2){  
    //Setencias  
    num1=num_1;  
    num2=num_2;  
}
```

MÉTODOS

```
//método de retorno entero, sin parametros  
public int sumar(){  
    //Setencias  
    int res;  
    res=num1+num2;  
    return res;  
}
```


Métodos Get y Set

Los métodos get y set, son simples métodos que usamos en las clases para mostrar (get) o modificar (set) el valor de un atributo. El nombre del método siempre será get o set y a continuación el nombre del atributo, su modificador siempre es public ya que queremos mostrar o modificar desde fuera la clase.

Por ejemplo, getNombre o setNombre.

Métodos Get y Set

Esta es la sintaxis de cada uno:

```
public tipo_dato_atributo getAtributo (){  
    return atributo;  
}  
public void setAtributo (tipo_dato_atributo variable){  
    this.atributo = variable;  
}
```



Métodos Get y Set

```
// Declaracion de Variables
private int num1, num2;

public int getNum1() {
    return num1;
}

public void setNum1(int num1) {
    this.num1 = num1;
}

public int getNum2() {
    return num2;
}

public void setNum2(int num2) {
    this.num2 = num2;
}
```



CONSTRUCTOR

Un constructor es un método especial de una clase que se llama automáticamente siempre que se declara un objeto de esa clase.

CONSTRUCTOR

```
//Nombre de la Clase
public class Suma {

    // Declaracion de Variables
    private int num1, num2;
    // Constructor de la clase
    public Suma() {
    }
}
```

SOBRECARGA DE MÉTODOS

La sobrecarga de métodos es la creación de varios métodos con el mismo nombre, pero con diferente lista de tipos de parámetros. Java utiliza el número y tipo de parámetros para seleccionar cuál definición de método ejecutar.

Java diferencia los métodos sobrecargados con base en el número y tipo de parámetros o argumentos que tiene el método y no por el tipo que devuelve.

```
//Sobre Carga de Metodos
```

```
public int sumar(){
```

```
    //Setencias
```

```
    int res;
```

```
    res=num1+num2;
```

```
    return res;
```

```
}
```

```
//Sobre Carga de Metodos
```

```
public double sumar(double val1,double val2,double val3)
```

```
    //Setencias
```

```
    double res;
```

```
    res=val1+val2+val3;
```

```
    return res;
```

```
}
```

SOBRECARGA DE MÉTODOS

DECLARACIÓN Y CREACIÓN DE UN OBJETO

Una vez que se tiene definida la clase a partir de la cual se crearán los objetos se está en la posibilidad de instanciar los objetos requeridos.

El operador new

El operador new crea una instancia de una clase asignando la cantidad de memoria necesaria de acuerdo al tipo de objeto

```
Clase nombre_objeto = new Clase();
```


ACCESO A DATOS Y MÉTODOS

```
public class Main {  
  
    public static void main(String[] args) throws IOException {  
        //Declaracion de Variables  
        int val1=50, val2=20,res;  
        //Instanciando la clase con el objeto  
        Suma obj= new Suma();  
        //enviando los valores al metodo asignar mediante el objeto  
        obj.asignar(val1,val2);  
        // se asignar el resultado del metodo sumar mediante el objeto  
        res=obj.sumar();  
        System.out.println("La Suma de Respuesta "+res);  
    }  
}
```