

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 1

## GUÍA DE LABORATORIO

INFORMACIÓN BÁSICA					
<b>ASIGNATURA :</b>	Estructura de datos y algoritmos				
<b>TÍTULO DE LA PRÁCTICA:</b>					
<b>NÚMERO DE PRÁCTICA:</b>	1	<b>AÑO LECTIVO:</b>	2025 A	<b>NRO. SEMESTRE:</b>	I
<b>TIPO DE PRÁCTICA:</b>	<b>INDIVIDUAL</b>	<b>X</b>			
	<b>GRUPAL</b>		<b>MÁXIMO DE ESTUDIANTES</b>		
<b>FECHA INICIO:</b>	<b>10/05/2025</b>	<b>FECHA FIN:</b>	<b>14/05/2025</b>	<b>DURACIÓN:</b>	
<b>Alumno:</b> Silva Pino Jesus Francisco					
<b>RECURSOS A UTILIZAR:</b> Computador, lapiz y papel					
<b>DOCENTE(s):</b> Edson Luque Rene Nieto Roni Apaza					

OBJETIVOS/TEMAS Y COMPETENCIAS	
<b>OBJETIVOS:</b> Revisar los conceptos básicos de programación orientada a objetos Conocer los elementos básicos del lenguaje de programación Java	
<b>TEMAS:</b> Programación Orientada a Objetos	
<b>COMPETENCIAS</b>	C.c Diseña y desarrolla soluciones computacionales eficientes mediante el uso adecuado de estructuras de datos y algoritmos, aplicando un proceso sistemático de resolución de problemas que optimiza el uso de los recursos disponibles y cumple con especificaciones concretas, garantizando responsabilidad en su implementación

## CONTENIDO DE LA GUÍA

### I. MARCO CONCEPTUAL

#### Java: ¿Qué es OOP?

OOP significa Programación Orientada a Objetos .

La programación procedimental consiste en escribir procedimientos o métodos que realicen operaciones sobre los datos, mientras que la programación orientada a objetos consiste en crear objetos que contengan tanto datos como métodos.

La programación orientada a objetos tiene varias ventajas sobre la programación procedimental:

La programación orientada a objetos es más rápida y fácil de ejecutar

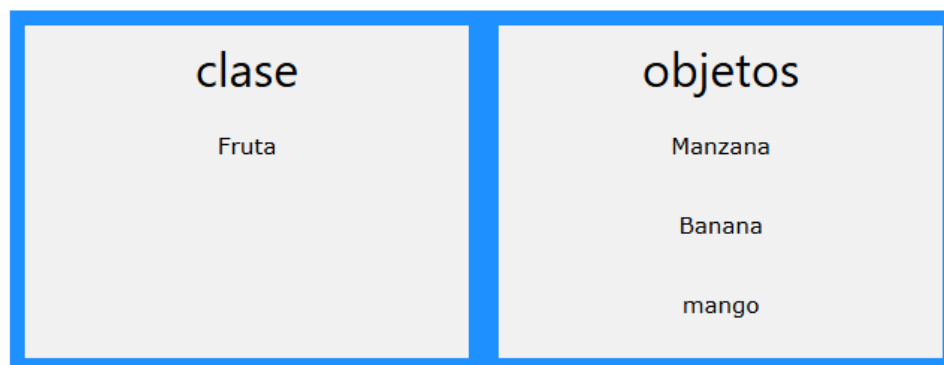
La programación orientada a objetos proporciona una estructura clara para los programas.

La programación orientada a objetos ayuda a mantener el código Java DRY ("No te repitas") y hace que el código sea más fácil de mantener, modificar y depurar.

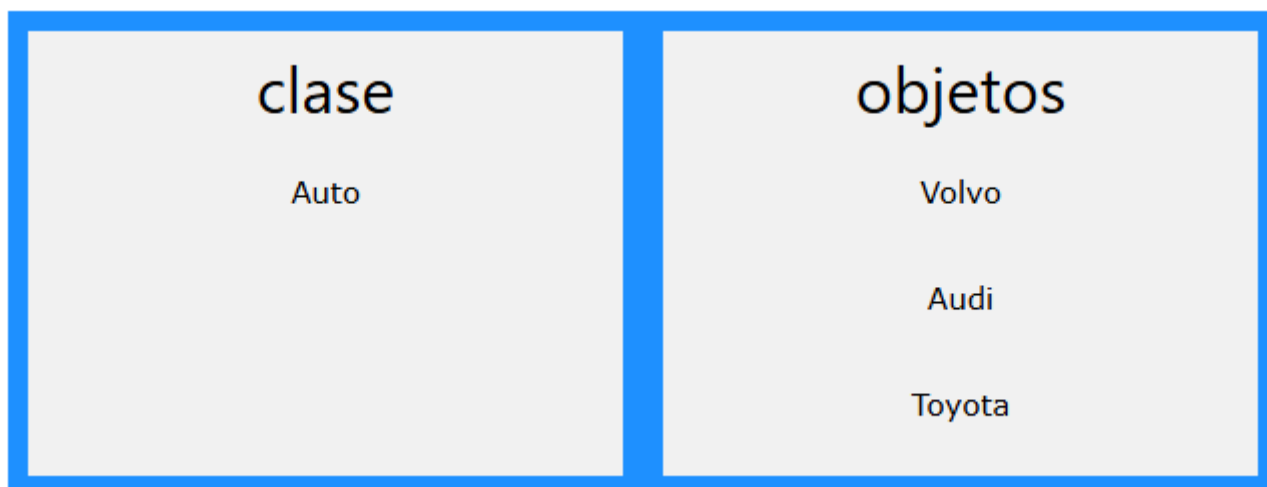
La programación orientada a objetos permite crear aplicaciones totalmente reutilizables con menos código y un tiempo de desarrollo más corto.

#### Java: ¿Qué son las clases y los objetos?

Las clases y los objetos son los dos aspectos principales de la programación orientada a objetos.



Observe la siguiente ilustración para ver la diferencia entre clase y objetos:



Otro ejemplo:

Entonces, una clase es una plantilla para objetos y un objeto es una instancia de una clase.

Cuando se crean los objetos individuales, heredan todas las variables y métodos de la clase.

### Clases/Objetos de Java

Java es un lenguaje de programación orientado a objetos.

En Java, todo está asociado a clases y objetos, junto con sus atributos y métodos. Por ejemplo, en la vida real, un coche es un objeto. Este coche tiene atributos , como el peso y el color, y métodos , como la tracción y el freno.

Una clase es como un constructor de objetos, o un "plano" para crear objetos.

#### Crear una clase

Para crear una clase, utilice la palabra clave class

#### Crear un objeto

En Java, un objeto se crea a partir de una clase. Ya hemos creado la clase llamada Main, así que ahora podemos usarla para crear objetos.

Para crear un objeto de Main, especifique el nombre de la clase, seguido del nombre del objeto y utilice la palabra clave new:

#### Uso de múltiples clases

También puedes crear un objeto de una clase y acceder a él desde otra. Esto suele usarse para una mejor organización de las clases (una clase contiene todos los atributos y métodos, mientras que la otra contiene el main()método (código a ejecutar)).

Recuerde que el nombre del archivo Java debe coincidir con el nombre de la clase.

#### Atributos de clase de Java

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 4

En el capítulo anterior, usamos el término "variable" en el ejemplo (como se muestra a continuación). En realidad, es un atributo de la clase. O bien, se podría decir que los atributos de clase son variables dentro de una clase

### **Acceso a atributos**

Puede acceder a los atributos creando un objeto de la clase y utilizando la sintaxis de punto (.):

### **Métodos de clase Java**

Aprendió en el capítulo Métodos Java que los métodos se declaran dentro de una clase y que se utilizan para realizar determinadas acciones

### **Estático vs. Público**

A menudo verá programas Java que tienen atributos static y public.

### **Constructores de Java**

Un constructor en Java es un método especial que se utiliza para inicializar objetos. Se llama al crear un objeto de una clase. Permite establecer valores iniciales para los atributos de un objeto

### **Parámetros del constructor**

Los constructores también pueden tomar parámetros, que se utilizan para inicializar atributos.

### **Modificadores**

A estas alturas, ya estás bastante familiarizado con la palabra public clave que aparece en casi todos nuestros ejemplos:

La palabra public clave es un modificador de acceso, lo que significa que se utiliza para establecer el nivel de acceso para clases, atributos, métodos y constructores.

Dividimos los modificadores en dos grupos:

Modificadores de acceso : controlan el nivel de acceso

Modificadores sin acceso : no controlan el nivel de acceso, pero proporcionan otras funciones



### **Modificadores de acceso**

Para las clases, puedes utilizar cualquiera de los dos publico el valor predeterminado :

- public
- default

Para atributos, métodos y constructores, puede utilizar uno de los siguientes:

- public
- private
- default

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p align="center"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p align="right"><b>Página:</b> 5</p>

- protected

### **Modificadores sin acceso**

Para las clases , puedes utilizar uno de los dos final o abstract

- final
- abstract

Para los atributos y métodos , puede utilizar uno de los siguientes:

- final
- static
- abstract
- transient
- synchronized
- volatile

### **Encapsulación**

El propósito de la encapsulación es garantizar que los datos confidenciales permanezcan ocultos a los usuarios. Para lograrlo, debe declarar variables/atributos de clase como private. Proporcionar métodos públicos de obtención y configuración para acceder y actualizar el valor de una variable private.

#### **Obtener y configurar**

Private solo se puede acceder a las variables dentro de la misma clase (una clase externa no tiene acceso a ellas). Sin embargo, es posible acceder a ellas si proporcionamos métodos públicos get y set .

El método get devuelve el valor de la variable y el método set establece el valor.

La sintaxis para ambos es que comienzan con get o set, seguido del nombre de la variable, con la primera letra en mayúscula:

#### **¿Por qué encapsulación?**

Mejor control de los atributos y métodos de clase

Los atributos de clase pueden ser de sólo lectura (si solo usa el método get) o de sólo escritura (si solo usa el método set)

- Flexible: el programador puede cambiar una parte del código sin afectar otras partes
- Mayor seguridad de los datos

### **Paquetes y API de Java**

En Java, un paquete se usa para agrupar clases relacionadas. Imagínalo como una carpeta en un directorio de archivos . Usamos paquetes para evitar conflictos de nombres y escribir código más fácil de mantener. Los paquetes se dividen en dos categorías:

- Paquetes integrados (paquetes de la API de Java)

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 6

- Paquetes definidos por el usuario (crea tus propios paquetes)

### **Paquetes integrados**

La API de Java es una biblioteca de clases preescritas, de uso gratuito, incluidas en el entorno de desarrollo de Java.

La biblioteca contiene componentes para la gestión de entradas, programación de bases de datos y mucho más. La lista completa se encuentra en el sitio web de Oracle:  
<https://docs.oracle.com/javase/8/docs/api/> .

La biblioteca se divide en paquetes y clases . Esto significa que puede importar una sola clase (con sus métodos y atributos) o un paquete completo que contenga todas las clases que pertenecen al paquete especificado.

Para utilizar una clase o un paquete de la biblioteca, debe utilizar la importpalabra clave

### **Importar una clase**

Si encuentra una clase que desea utilizar, por ejemplo, la Scannerclase que se utiliza para obtener la entrada del usuario

### **Importar un paquete**

Hay muchos paquetes disponibles. En el ejemplo anterior, usamos la Scannerclase del java.util paquete. Este paquete también incluye funciones de fecha y hora, un generador de números aleatorios y otras clases de utilidad.

Para importar un paquete completo, termine la oración con un asterisco ( \*). El siguiente ejemplo importará TODAS las clases del java.util paquete

### **Paquetes definidos por el usuario**

Para crear tu propio paquete, debes comprender que Java utiliza un directorio del sistema de archivos para almacenarlos. Al igual que las carpetas de tu ordenador



## **II. EJERCICIO/PROBLEMA RESUELTO POR EL DOCENTE**

Ejercicio 01 Crear una clase Main con una variable x:

```
public class Main {
    int x = 5;
}
```

Ejercicio 02 Crea un objeto llamado " myObj" e imprime el valor de x:

```
public class Main {
    int x = 5;
```

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p align="center"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p align="right"><b>Página:</b> 7</p>

```
public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
}
}
```

Ejercicio 03 Crea dos objetos de Main

```
public class Main {
    int x = 5;

    public static void main(String[] args) {
        Main myObj1 = new Main(); // Object 1
        Main myObj2 = new Main(); // Object 2
        System.out.println(myObj1.x);
        System.out.println(myObj2.x);
    }
}
```

Ejercicio 04 Crear dos archivos en el mismo directorio uno llamado Main.java y otro llamado Second.java, hacer que la clase Second llame a la clase Main

Main.java

```
public class Main {
    int x = 5;
}
```

Segundo.java

```
class Second {
    public static void main(String[] args) {
        Main myObj = new Main();
        System.out.println(myObj.x);
    }
}
```

Ejercicio 05 Crea una clase llamada " Main" con dos atributos: x y y:

```
public class Main {
    int x = 5;
    int y = 3;
}
```

Ejercicio 06 Crea un objeto llamado " myObj" e imprime el valor de x:

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 8</p>

```
public class Main {
    int x = 5;

    public static void main(String[] args) {
        Main myObj = new Main();
        System.out.println(myObj.x);
    }
}
```

Ejercicio 07 Establezca el valor de x a 10 al principio y después cámbielo a 25, usando la propiedad del atributo del objeto myObj

```
public class Main {
    int x = 10;

    public static void main(String[] args) {
        Main myObj = new Main();
        myObj.x = 25; // x is now 25
        System.out.println(myObj.x);
    }
}
```

Ejercicio 08 Utilize el modificar final para evitar la modificación de un atributo

```
public class Main {
    final int x = 10;

    public static void main(String[] args) {
        Main myObj = new Main();
        myObj.x = 25; // will generate an error: cannot assign a value to a final variable
        System.out.println(myObj.x);
    }
}
```

Ejercicio 09 Cambie el valor de x a 25 en myObj2y déjelo sin modificar en myObj1

```
public class Main {
    int x = 5;

    public static void main(String[] args) {
        Main myObj1 = new Main(); // Object 1
        Main myObj2 = new Main(); // Object 2
        myObj2.x = 25;
        System.out.println(myObj1.x); // Outputs 5
        System.out.println(myObj2.x); // Outputs 25
    }
}
```



	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p align="center"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 9</p>

```
}
}
```

### III. EJERCICIOS/PROBLEMAS PROPUESTOS

Ejercicio 1, utilizando una clase abstracta crear una clase Animal con el método dormir y de esta se extienda una clase perro que pueda ladrar, implementar y mostrar sus métodos en una clase main

```
%%writefile Ejercicio1.java
abstract class Animal {

    public void dormir() {
        System.out.println("El animal está durmiendo");
    }

    public abstract void hacerSonido();
}

class Perro extends Animal {

    public void hacerSonido() {
        System.out.println("¡Guau guau!");
    }
}

public class Ejercicio1 {
    public static void main(String[] args) {
        Perro miPerro = new Perro();
        miPerro.dormir();
        miPerro.hacerSonido();
    }
}
```

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p align="center"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 10</p>

```
[2]    public void dormir() {
        System.out.println("El animal está durmiendo");
    }

    public abstract void hacerSonido();
}

class Perro extends Animal {

    public void hacerSonido() {
        System.out.println("¡Guau guau!");
    }
}

public class Ejercicio1 {
    public static void main(String[] args) {
        Perro miPerro = new Perro();
        miPerro.dormir();
        miPerro.hacerSonido();
    }
}
```

Writing Ejercicio1.java

```
%cd ../Ejercicio1/
!javac Ejercicio1.java && java Ejercicio1
```

```
/content/Laboratorio1/Ejercicio1
El animal está durmiendo
¡Guau guau!
```

Ejercicio 2, Averiguar como se utiliza interface en java y crear una clase vehiculo con los métodos iniciar y detener implementar una clase auto que use dichos métodos y mostrarlos en una clase main

```
%%writefile ./Laboratorio1/Ejercicio2/Ejercicio2.java
```

```
interface Vehiculo {
    void iniciar();
    void detener();
}
```

```
class Auto implements Vehiculo {
```

```
    public void iniciar() {
        System.out.println("El auto está encendiendo");
    }
```

```
    public void detener() {
        System.out.println("El auto está apagándose");
    }
```

```
}
```

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p align="center"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p align="right"><b>Página:</b> 11</p>

```
public class Ejercicio2 {
    public static void main(String[] args) {
        Auto miAuto = new Auto();
        miAuto.iniciar();
        miAuto.detener();
    }
}
```

```
[4] public void iniciar() {
    System.out.println("El auto está encendiendo");
}

    public void detener() {
        System.out.println("El auto está apagándose");
    }
}

public class Ejercicio2 {
    public static void main(String[] args) {
        Auto miAuto = new Auto();
        miAuto.iniciar();
        miAuto.detener();
    }
}
```

```
➦ Writing ./Laboratorio1/Ejercicio2/Ejercicio2.java
```

```
[8] %cd Laboratorio1/Ejercicio2
```

```
➦ [Errno 2] No such file or directory: 'Laboratorio1/Ejercicio2'
/content/Laboratorio1
```

```
[9] %cd Ejercicio2/
```


```
➦ /content/Laboratorio1/Ejercicio2
```

```
[10] !javac Ejercicio2.java && java Ejercicio2
```

```
➦ El auto está encendiendo
El auto está apagándose
```

Ejercicio 3, Crear una clase auto con los atributos modelo y año, parametrizar el constructor con estos atributos, crear un clase sin argumentos y con argumentos

```
%%writefile Ejercicio3.java
class Auto {
    private String modelo;
    private int año;
```

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 12

```

public Auto() {
    this.modelo = "Desconocido";
    this.año = 0;
}

public Auto(String modelo, int año) {
    this.modelo = modelo;
    this.año = año;
}



public void mostrarInfo() {
    System.out.println("Modelo: " + modelo + ", Año: " + año);
}

}

public class Ejercicio3 {
    public static void main(String[] args) {
        Auto auto1 = new Auto();
        Auto auto2 = new Auto("Toyota Corolla", 2020);

        auto1.mostrarInfo();
        auto2.mostrarInfo();
    }
}

```

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 13

```

%%writefile Ejercicio3.java
class Auto {
    private String modelo;
    private int año;

    public Auto() {
        this.modelo = "Desconocido";
        this.año = 0;
    }

    public Auto(String modelo, int año) {
        this.modelo = modelo;
        this.año = año;
    }

    public void mostrarInfo() {
        System.out.println("Modelo: " + modelo + ", Año: " + año);
    }
}

public class Ejercicio3 {
    public static void main(String[] args) {
        Auto auto1 = new Auto();
        Auto auto2 = new Auto("Toyota Corolla", 2020);

        auto1.mostrarInfo();
        auto2.mostrarInfo();
    }
}

Overwriting Ejercicio3.java

[21] !javac Ejercicio3.java && java Ejercicio3
Modelo: Desconocido, Año: 0
Modelo: Toyota Corolla, Año: 2020

```

## Ejercicio 4, Investigar que son los Enums y mostrar su utilidad en un ejemplo simple

### 1. Enums (Enumeraciones)



#### ¿Qué son?

Los **Enums** (enumeraciones) son un tipo de dato especial en Java que permite definir un conjunto fijo de constantes con nombres descriptivos. Son útiles cuando necesitas representar un grupo de valores que no cambiarán, como días de la semana, estados de un pedido, colores, etc.

#### Características:

- Son **tipos seguros** (evitan errores al asignar valores no válidos).
- Pueden tener **métodos, constructores y atributos**.
- Se usan con switch, lo que mejora la legibilidad del código.
- Implementan automáticamente Comparable y pueden usarse en colecciones.

#### Utilidad:

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 14

- **Legibilidad:** DiaSemana.MIERCOLES es más claro que 3 (si usáramos números).
- **Seguridad:** Evita errores como día = 99 (que no existe).
- **Mantenibilidad:** Si añades un nuevo día, solo modificas el Enum.

Ejemplo

```
%%writefile Ejercicio4.java
```

```
enum DiaSemana {
    LUNES, MARTES, MIERCOLES, JUEVES, VIERNES, SABADO, DOMINGO
}
```

```
public class Ejercicio4 {
    public static void main(String[] args) {
        DiaSemana hoy = DiaSemana.MIERCOLES;

        System.out.println("Hoy es " + hoy);

        switch(hoy) {
            case LUNES:
                System.out.println("¡Comienza la semana!");
                break;
            case VIERNES:
                System.out.println("¡Por fin es viernes!");
                break;
            case SABADO: case DOMINGO:
                System.out.println("¡Disfruta el fin de semana!");
                break;
            default:
                System.out.println("Día laboral normal");
        }

        System.out.println("\nTodos los días de la semana:");
        for (DiaSemana dia : DiaSemana.values()) {
            System.out.println(dia);
        }
    }
}
```

```

0s [23]
    System.out.println("Comienza la semana!");
    break;
case VIERNES:
    System.out.println("¡Por fin es viernes!");
    break;
case SABADO: case DOMINGO:
    System.out.println("Disfruta el fin de semana!");
    break;
default:
    System.out.println("Día laboral normal");
}

System.out.println("\nTodos los días de la semana:");
for (DiaSemana dia : DiaSemana.values()) {
    System.out.println(dia);
}
}
}

Writing Ejercicio4.java

2s !javac Ejercicio4.java && java Ejercicio4

Hoy es MIERCOLES
Día laboral normal

Todos los días de la semana:
LUNES
MARTES
MIERCOLES
JUEVES
VIERNES
SABADO
DOMINGO

```

Ejercicio 5, Investigar y crear un paquete simple

## 2. Paquetes (Packages)


### ¿Qué son?

Los **paquetes** son una forma de organizar clases e interfaces relacionadas en Java. Funcionan como **carpetas** en un sistema de archivos, evitando conflictos de nombres y facilitando la modularidad.

### Características:

- **Evitan colisiones de nombres:** Dos clases pueden llamarse igual si están en paquetes distintos.
- **Control de acceso:** Permiten usar modificadores como public, protected y private.
- **Jerarquía:** Se pueden anidar (ej: com.empresa.proyecto.util).

### Utilidad:

	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 16

- **Organización:** Agrupa clases relacionadas (ej: utilidades, modelos, controladores).
- **Reutilización:** Puedes importar paquetes en otros proyectos.
- **Encapsulación:** Oculta clases internas con default (solo accesibles dentro del paquete).

Ejemplo

Estructura del paquete

Ejercicio5/

└─ utilidades

└─ Calculadora.java

### Archivo Calculadora.java

```
package utilidades;
```

```
public class Calculadora {
    public static int sumar(int a, int b) {
        return a + b;
    }
}
```

```
    public static int restar(int a, int b) {
        return a - b;
    }
}
```

Clase Ejercicio5 para testear el paquetes

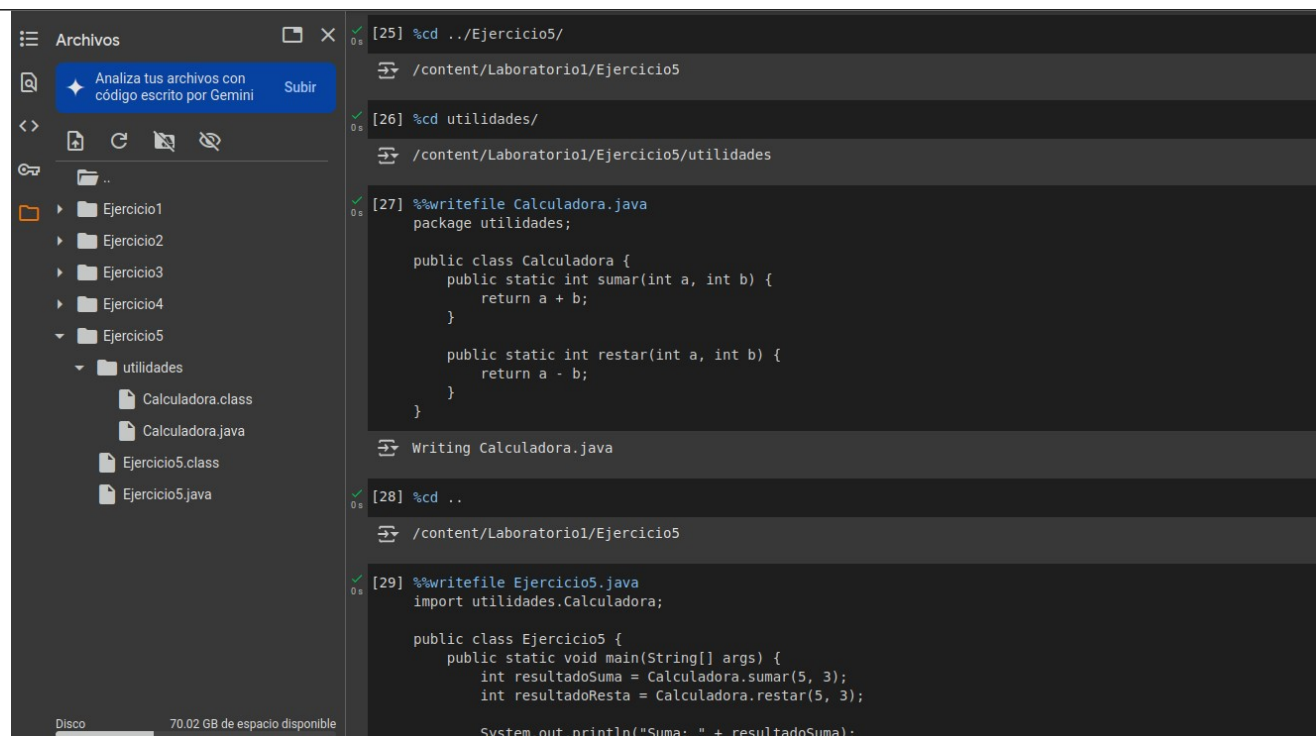
```
import utilidades.Calculadora;
```

```
public class Main {
    public static void main(String[] args) {
        int resultadoSuma = Calculadora.sumar(5, 3);
        int resultadoResta = Calculadora.restar(5, 3);

        System.out.println("Suma: " + resultadoSuma);
        System.out.println("Resta: " + resultadoResta);
    }
}
```



	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 17



```

[25] %cd ../Ejercicio5/
/content/Laboratorio1/Ejercicio5

[26] %cd utilidades/
/content/Laboratorio1/Ejercicio5/utilidades

[27] %%writefile Calculadora.java
package utilidades;

public class Calculadora {
    public static int sumar(int a, int b) {
        return a + b;
    }

    public static int restar(int a, int b) {
        return a - b;
    }
}

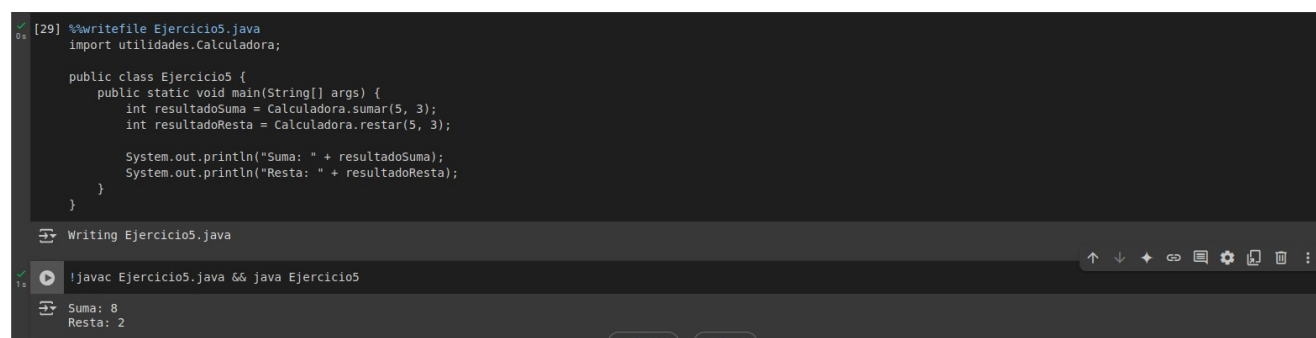
Writing Calculadora.java

[28] %cd ..
/content/Laboratorio1/Ejercicio5

[29] %%writefile Ejercicio5.java
import utilidades.Calculadora;

public class Ejercicio5 {
    public static void main(String[] args) {
        int resultadoSuma = Calculadora.sumar(5, 3);
        int resultadoResta = Calculadora.restar(5, 3);

        System.out.println("Suma: " + resultadoSuma);
    }
}
  
```



```

[29] %%writefile Ejercicio5.java
import utilidades.Calculadora;

public class Ejercicio5 {
    public static void main(String[] args) {
        int resultadoSuma = Calculadora.sumar(5, 3);
        int resultadoResta = Calculadora.restar(5, 3);

        System.out.println("Suma: " + resultadoSuma);
        System.out.println("Resta: " + resultadoResta);
    }
}

Writing Ejercicio5.java

!javac Ejercicio5.java && java Ejercicio5

Suma: 8
Resta: 2
  
```

## IV. CUESTIONARIO

Pregunta 1 ¿Existen otros lenguajes orientados a objetos? Nombrellos

Sí, existen muchos otros lenguajes de programación orientados a objetos además de Java. Algunos de los más importantes son:

1. **C++** (extensión orientada a objetos de C)
2. **Python** (soporta múltiples paradigmas incluyendo POO)
3. **C#** (desarrollado por Microsoft, similar a Java)
4. **Ruby** (orientado a objetos puro)
5. **JavaScript** (usa prototipos para POO)

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y</b>  <b>SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 18

## 6. PHP (soporta programación orientada a objetos), etc

Pregunta 2 ¿Qué es la conversión de tipos?

La conversión de tipos (o "type casting") es el proceso de convertir un valor de un tipo de dato a otro. En Java existen dos tipos principales:

### 1. Conversión implícita (automática/widening):

- Ocurre cuando convertimos un tipo más pequeño a uno más grande
- Ejemplo: int a long, float a double
- No hay pérdida de información

```
int numInt = 10;
long numLong = numInt; // Conversión automática
```

### 2. Conversión explícita (manual/narrowing):

- Cuando convertimos un tipo más grande a uno más pequeño
- Requiere sintaxis especial: (tipoDestino)valor
- Puede haber pérdida de información

```
double numDouble = 9.78;
int numInt = (int) numDouble; // numInt será 9 (pierde decimales)
```

También existe la conversión entre tipos de objetos (cuando hay herencia) y la conversión mediante métodos como `parseInt()` para convertir Strings a números.

Pregunta 3 ¿Cuál es el tipo de dato mas grande en Java?


En Java, el tipo de dato primitivo más grande es:

- **Para números enteros:** long (8 bytes, rango: -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807)
- **Para números decimales:** double (8 bytes,  $\pm 1.79769313486231570E+308$ )

Si son necesarios números aún más grandes, Java proporciona clases especiales en el paquete `java.math`:

- BigInteger (para enteros arbitrariamente grandes)
- BigDecimal (para números decimales de precisión arbitraria)

## V. REFERENCIAS Y BIBLIOGRAFÍA RECOMENDADAS:

	<b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b> <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b> <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 19

[1] Especificación de la API de la plataforma Java™, edición estándar 8, 2025, [https://docs-oracle-com.translate.goog/javase/8/docs/api/?\\_x\\_tr\\_sl=en&\\_x\\_tr\\_tl=es&\\_x\\_tr\\_hl=es&\\_x\\_tr\\_pto=tc](https://docs-oracle-com.translate.goog/javase/8/docs/api/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc)

## TÉCNICAS E INSTRUMENTOS DE EVALUACIÓN

TÉCNICAS: <i>Ejercicios propuestos</i>		INSTRUMENTOS: <i>Rúbricas</i>		
CRITERIOS DE EVALUACIÓN				
<i>Criterio de evaluación / Niveles de Expectativa</i>	<i>1 = Insatisfactorio</i>	<i>2 = En proceso</i>	<i>3 = Satisfactorio</i>	<i>4 = Sobresaliente</i>
<i>Informe</i>	<i>El informe es difícil de leer y no cuenta con la información pedida (0)</i>	<i>El informe incluye la mayor parte de la información solicitada, pero cuesta comprenderlo (2)</i>	<i>El informe incluye la información solicitada y es comprensible (4)</i>	<i>El informe está claramente detallado e incluye toda la información solicitada (8)</i>
<i>Cantidad de Información</i>	<i>Uno o más de los temas no han sido tratados (0)</i>	<i>Todos los temas han sido tratados y la mayor parte de las preguntas han sido contestadas, como mínimo, con una frase (2)</i>	<i>Todos los temas han sido tratados y la mayor parte de las preguntas han sido contestadas, como mínimo con dos frases cada una (3)</i>	<i>Todos los temas han sido tratados y todas las preguntas han sido contestadas con tres o más frases cada una (6)</i>
<i>Calidad de información</i>	<i>La información tiene poco que ver con el tema principal (0)</i>	<i>La información está relacionada con el tema principal (2)</i>	<i>La información está claramente relacionada con el tema principal (3)</i>	<i>La información está claramente relacionada con el tema principal y presenta otros ejemplos (6)</i>