

[Link README](#)

PROGRAMACION II

<JEFFERSON DAVID />

I BIMESTRE



▶ 0:00 / 3:43 ————— 🔊

1. INTRODUCCION A LA PROGRAMACION, ALGORITMOS Y DISEÑO

SEMANA 1

PREPARANDO NUESTRAS HERRAMIENTAS...

IDE

Tiene que tener:

- Compilador
- proyectos
- editor

Compilador

Java tiene un interprete para cada tipo de sistema: Windows, OS, Linux... hay que descargar el interprete para cada sistema operativo. Transforma el programa en **byte code** "ceros y unos".

INSTALAR

- EXTENSION PACK FOR JAVA
- PROJECT MANAGER FOR JAVA
- TEST RUNNER FOR JAVA

COMANDOS

- CTRL + SHIFT : Multicursor lineal vertical

GIT REPOSITORIO - ALL CODE

gh repo clone JeffersonDavid/Programacion-II

GIT

Estos son los comandos más utilizados en Git

- `git init`: Iniciar el control de versiones
- `git add .`: agrega todos los archivos al stage
- `git commit -m "mensaje"`: Mensaje clave del commit, ser puntual no detallado
- `git push`: envía los cambios a la nube
- `git status -s`: Ver estado de los archivos

Comandos necesarios pero no muy utilizados en Git

- `git config --global user.name "mi nombre"`: nombre usuario global
- `git config --global user.email "myemail@example.com"`: email del git
- `git log --oneline`: lista de todos los commits descendientemente
- `git remote add origin "link que github nos da al momento de crear el repositorio"`
- `git push -u origin master/main`: aquí podemos elegir pero por temas políticos se elige main

SEMANA 2

CONOCIMIENTOS BASICOS

Métodos

- Retornan valor : `funcion`
- no retorna valor : `procedimiento` - metodos

Structura Básica

Regla de Java:

El nombre del archivo debe ser EL MISMO que el nombre de la clase y tiene que ir en Mayusculas

CONTROL DE FLUJO

Define el comportamiento de los datos.

- if - else
- for (n)
- while(<>!=)
- Do (<>!=)
- switch

```
int a = 10;           // Tipo de dato entero
char nombre = "";     // Tipo de dato caracter

if (a == 10) {
```

```
    printf("el valor de a es 10");
} else {
    printf("el valor de a no es 10");
}
return = 0;

Funciones() <return>;    // Siempre retorna un valor
Procedimiento();        // Solo realiza una acción pero no retorna una valor
```

Tipo de datos

Propios de Java

- String
- Integer
- Bool
- Int
- ...

Primitivos

- bool
- int
- ...

Standart codif. : **camelCase**

ALGORITMIA

EJEMPLO 2:

1. **PROBLEMA:** Determinar el mayor de 2 números
2. **SOLUCIÓN:** 10, 30, 5

VARIABLES: a=10, b=30, c=5

1. **ALGORITMO** (pseudocódigo)

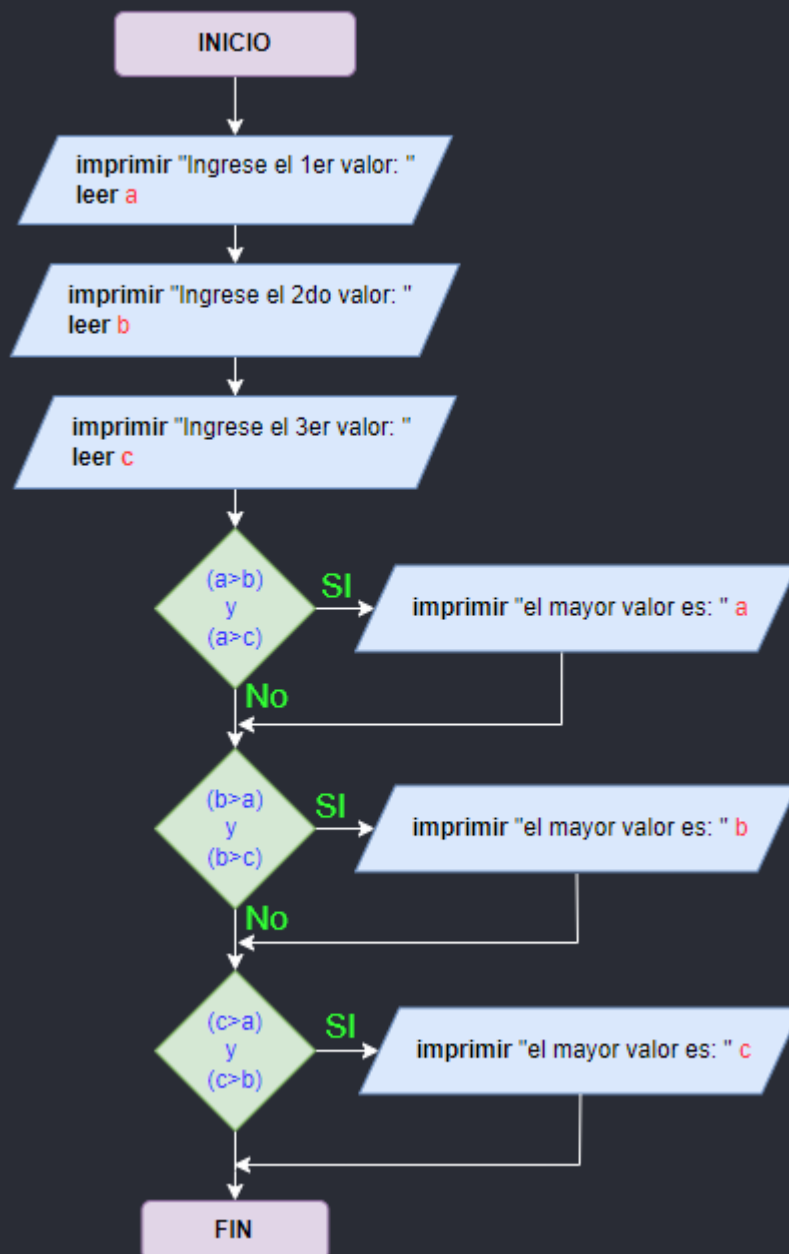
```
Imprimir    "Ingrese el primer valor: "
Leer        a
Imprimir    "Ingrese el segundo valor: "
Leer        b
Imprimir    "Ingrese el tercer valor: "
Leer        c

Si (a>=b) y (a>=c)
    imprimir "el mayor valor es: " a
    terminar
Si (b>=a) y (b>=c)
```

```
imprimir "el mayor valor es: " b
terminar
Si (c>=a) y (c>=b)
imprimir "el mayor valor es: " c
terminar
```

2. DIAGRAMA DE FLUJO

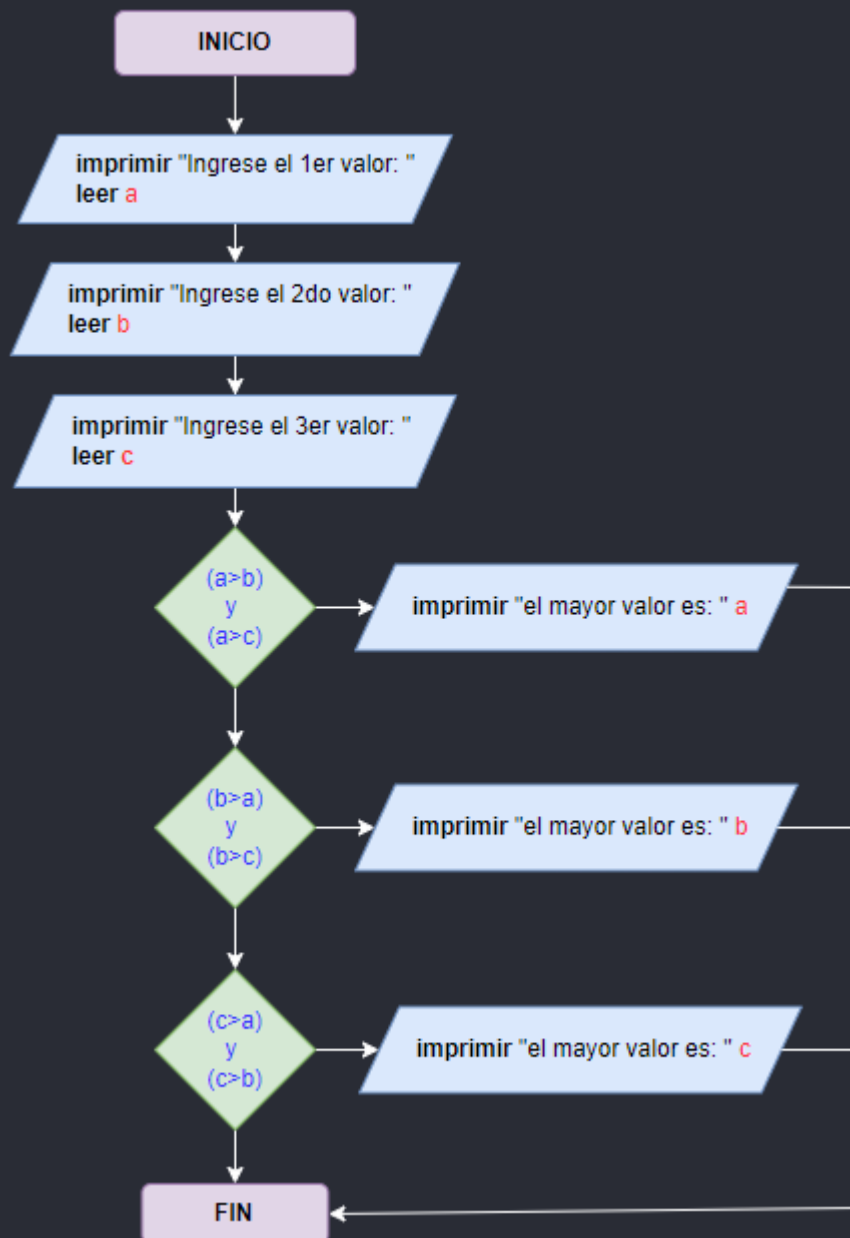
Diagrama *con error*



Tiene un error porque el programa a pesar de que puede cumplirse una condición, tiene que terminar, PERO aquí no, el programa EJECUTA TODAS LAS INSTRUCCIONES SIGUIENTES las cumpla o no.

Esto genera un CONSUMO DE RECURSOS INNECESARIOS

Diagrama *sin error*



Cuado se cumpla una condición entonces el programa termina con su ejecución

Esto NO genera un CONSUMO DE RECURSOS INNECESARIOS

3. CODING

```
// Aquí va el código
```

4. TRACE

Trace 1			
a	b	c	Salida:
15	10	5	Ingrese el 1er valor:
			15
			Ingrese el 2do valor:
			10
			Ingrese el 3er valor:
			5
			El mayor valor es: 15

Trace 2			
a	b	c	Salida:
15	22	22	Ingrese el 1er valor:
			15
			Ingrese el 2do valor:
			22
			Ingrese el 3er valor:
			22
			El mayor valor es: 22

Trace 3			
a	b	c	Salida:
12	22	25	Ingrese el 1er valor:
			12
			Ingrese el 2do valor:
			22
			Ingrese el 3er valor:
			25
			El mayor valor es: 25

2. FUNDAMENTOS DE JAVA


INTRODUCCION A JAVA

Propiedad = **Ambito** + variable

- Ambito: public, private, protect

Ejemplo:

```
public class XXXX{
    public Integer edadUsuario = 21; // propiedad
}
```

 Hay que tener MUCHO CUIDADO con el TIPO DE DATO, usar SABIAMENTE.

VARIABLE VS PROPIEDAD

```
public class XXXX{
    public Integer edad = 21; // propiedad
    public static void main(){
        String nombreUsuario = "David"; // variable
        String nombre; // ? declaración
        nombre = "Jefferson" // ? inicialización
    }
}
```

```
}  
}
```

PARAMETROS

Argumentos de una función.

```
main (String variable){...codigo}
```

```
public class App {  
    public static void main(String args[]) {  
        System.out.println("Hola Mundo :");  
    }  
}
```

Bibliotecas

Aquí algunas de las librerías más usadas y básicas, librerías del propio lenguaje, NO LIBRERIAS EXTERNAS.

Algunos metodos más comunes

String : str

- `charAt(index)` // obtener un carácter
- `length` // obtener la longitud del string
- `equals` // comparación
- `equalsIgnoreCase` // ignorar si es mayúsculas o minúscula

Scanner : std

```
Scanner stdIn = new Scanner(System.in);
```

```
< variable > = stdIn.nextLine();
```

- `nextInt()` Se salta los espacios dejados en blanco hasta que encuentra un valor de tipo int
- `nextLong()` Se salta los espacios dejados en blanco hasta que encuentra un valor de tipo long
- `nextFloat()` Se salta los espacios dejados en blanco hasta que encuentra un valor de tipo float
- `nextDouble()` Se salta los espacios dejados en blanco hasta que encuentra un valor de tipo double

“salta los espacios dejados en blanco ”

- `next()` Se salta los espacios dejados en blanco hasta que encuentra un token. Devuelve el token como un valor tipo String.

SCANNER

```
import java.util.Scanner;  
Scanner xxxx = new Scanner(System.in); // crear un objeto entrada  
int valor = xxxx.nextInt(); // usar en numeros  
string valor = xxxx.nextLine(); // usar en texto
```

😊 SEMANA 3

Creación de Métodos

Métodos sin retorno de valor

```
public void metodo(){  
    // Instrucciones  
}****
```

Detalles:

1. **public** : modificador de acceso
2. **void** : valor de retorno (puede ir int, string...)
3. **metodo** : nombre del método
4. **...** : argumentos que recibe el método

Métodos con retorno

```
public int metodo(int a, int b){  
    return a + b;  
}
```

```
public float metodo(int a, float b){  
    return a + b;  
}
```

```
public float metodo(float a, float b){  
    return a + b;  
}
```

Puede existir funciones con el mismo nombre pero DEBEN SER DE DISTINTO TIPO DE PARAMETRO
En el ejemplo anterior tenemos 3 sumas

😊 SEMANA 5

PROGRAMACION ORIENTADA A OBJETOS

HERENCIA

Permite la reutilizar código, es decir: Permite la creacion de clases apartir de otras ya existentes, heredando todas sus propiedades y métodos.

NOMENCLATURA

Clase original = clase Padre = Superclase

Clase secundaria = clase hijo = Subclase

CONSTRUCTOR

Tiene que tener el mismo nombre de la clase;

Para pasar la herencia a los hijos usamos **EXTENDS** nombreClaseHijo

Queremos heredar la clase **Hijo** apartir de la clase **Padre**.

```
// ARCHIVO PADRE
public class Padre {
    public Padre(String nombre, String ocupacion, String sexo, int edad) {
        this.nombre = nombre;
        this.ocupacion = ocupacion;
        this.sexo = sexo;
        this.edad = edad;
    }
}

// ARCHIVO HIJO
public class Hijo extends Padre(String nombre, String ocupacion, String sexo, int edad){
    super(nombre, ocupacion, sexo, edad); // Super OBLIGATORIAMENTE debe ser la
    primera linea de codigo cuando se requiere agregar. SI

    // aqui vas el codigo del Hijo
}
```

CONSTRUCTOR CON PARAMETROS

```
public Padre(String nombre, String ocupacion, String sexo, int edad) {
    this.nombre = nombre;
    this.ocupacion = ocupacion;
    this.sexo = sexo;
    this.edad = edad;
}
```

this : Referenciar variables dentro del CONSTRUCTOR de las clase.

DIAGRAMACION UML

NOTACION PARA AMBITOS

- + public
- - private
- ~ paquete
- # protect

NOTACION PARA METODOS

- + public
- - private

POO

- Mejorar el diseño
- Permite la reutilización
- Facilita la extensión
- Problema: acoplamiento

OBJETOS

La 'variable' es el objeto que tiene referencia a la memoria RAM

```
Vehiculo v = new Vehiculo();  
Coche c = new Coche();  
Barco b = new Barco();
```

En la SUBCLASE

Antes de que se ejecute el constructor

SOBRESCRIBIR METODOS DEL PADRE

☹️ SEMANA 6

MODELADO O.O UML

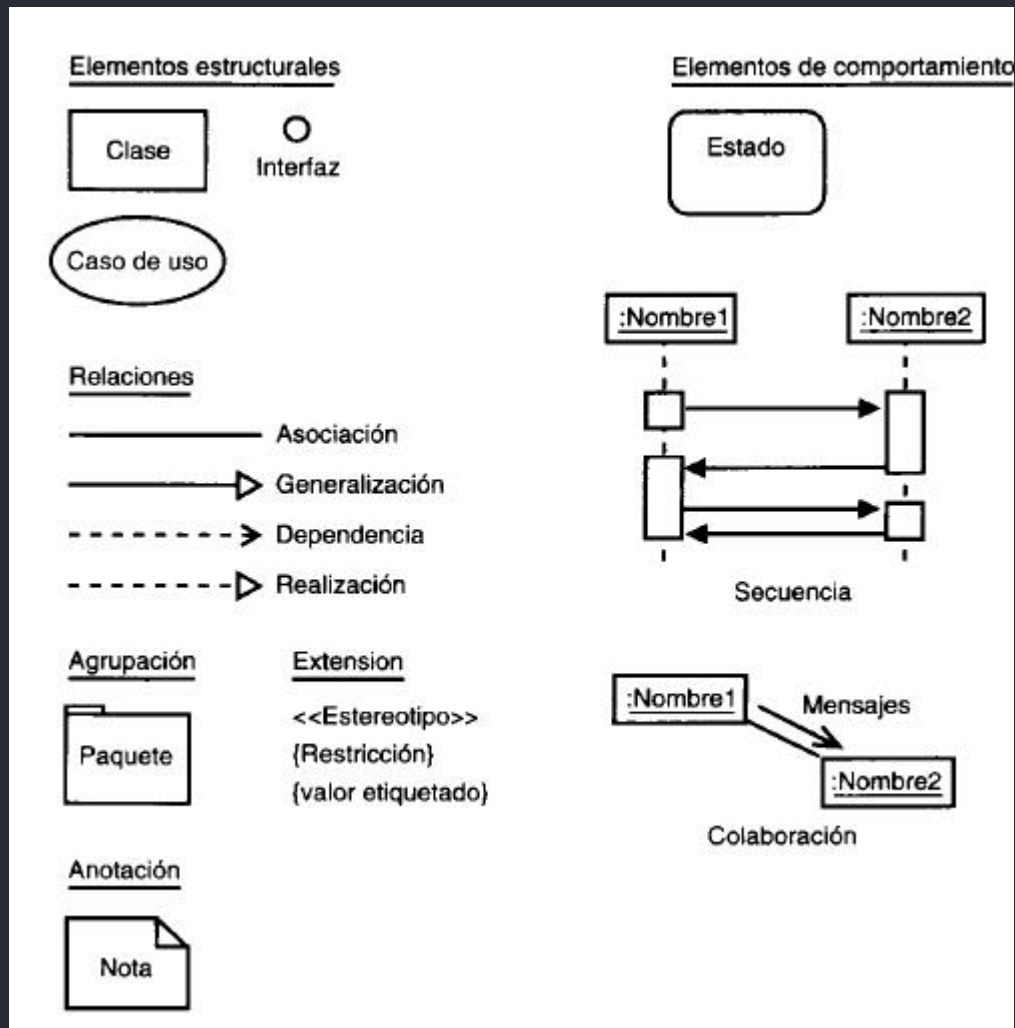
- → dirección del modelo

COMPONENTES

- **uso** "_____" bidirección "----->" línea de dirección
- **inclusión** "_____" encima de la barra lleva *include*, es como para describir más el programa.
- **extension** "----->" encima de la barra lleva *extiende*, significa que depende de algo (más opciones para el usuario ejemplo un cliente puede pagar de diferenc).
- **Generalizaciones** "->" sirve para clasificar dato.

exclude : va en dirección de la clase que la contiene, generalmente se usan entre caso de uso.

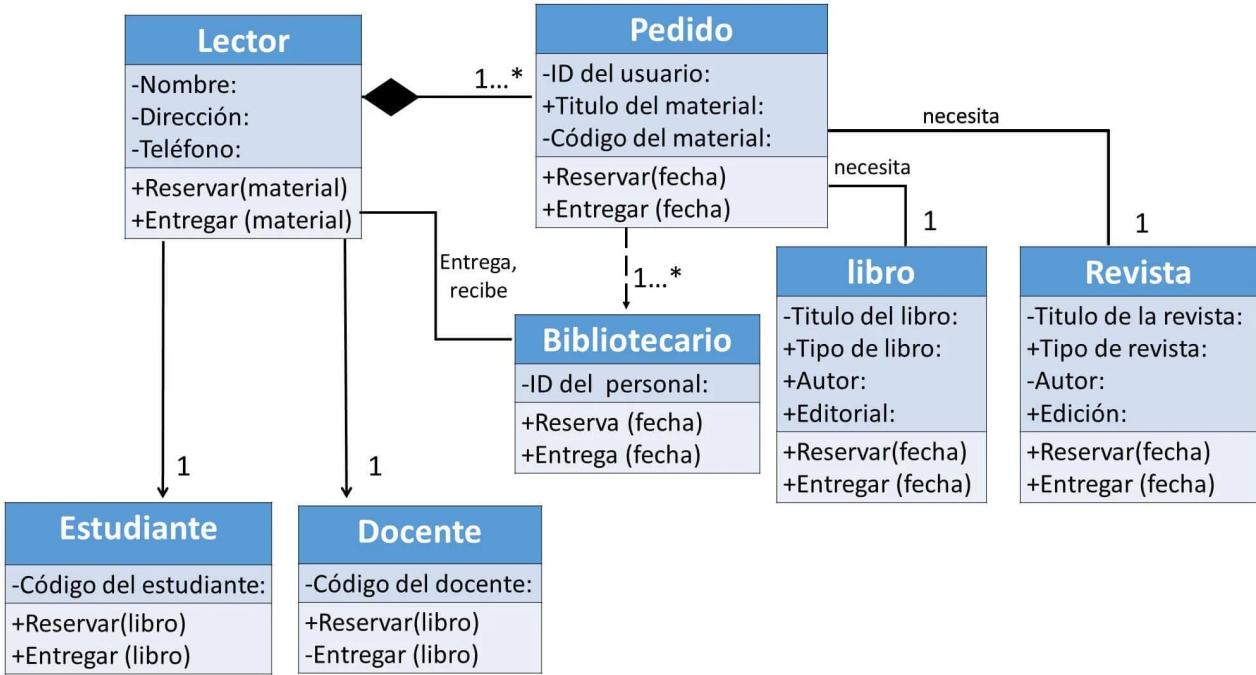
include : va en sentido de elemento opcional



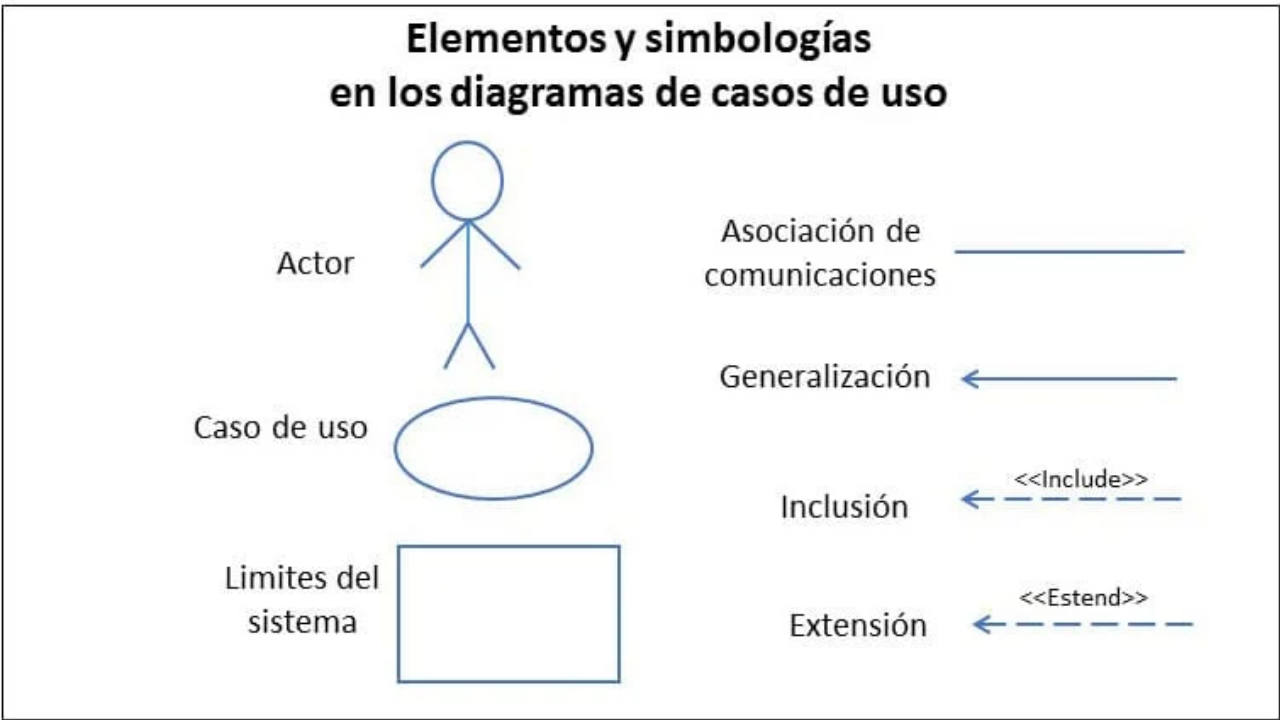
EJEMPLO DE DIGRAMA DE CLASES

Ejemplo de un biblioteca

Diagrama de clases de un sistema de servicios bibliotecarios

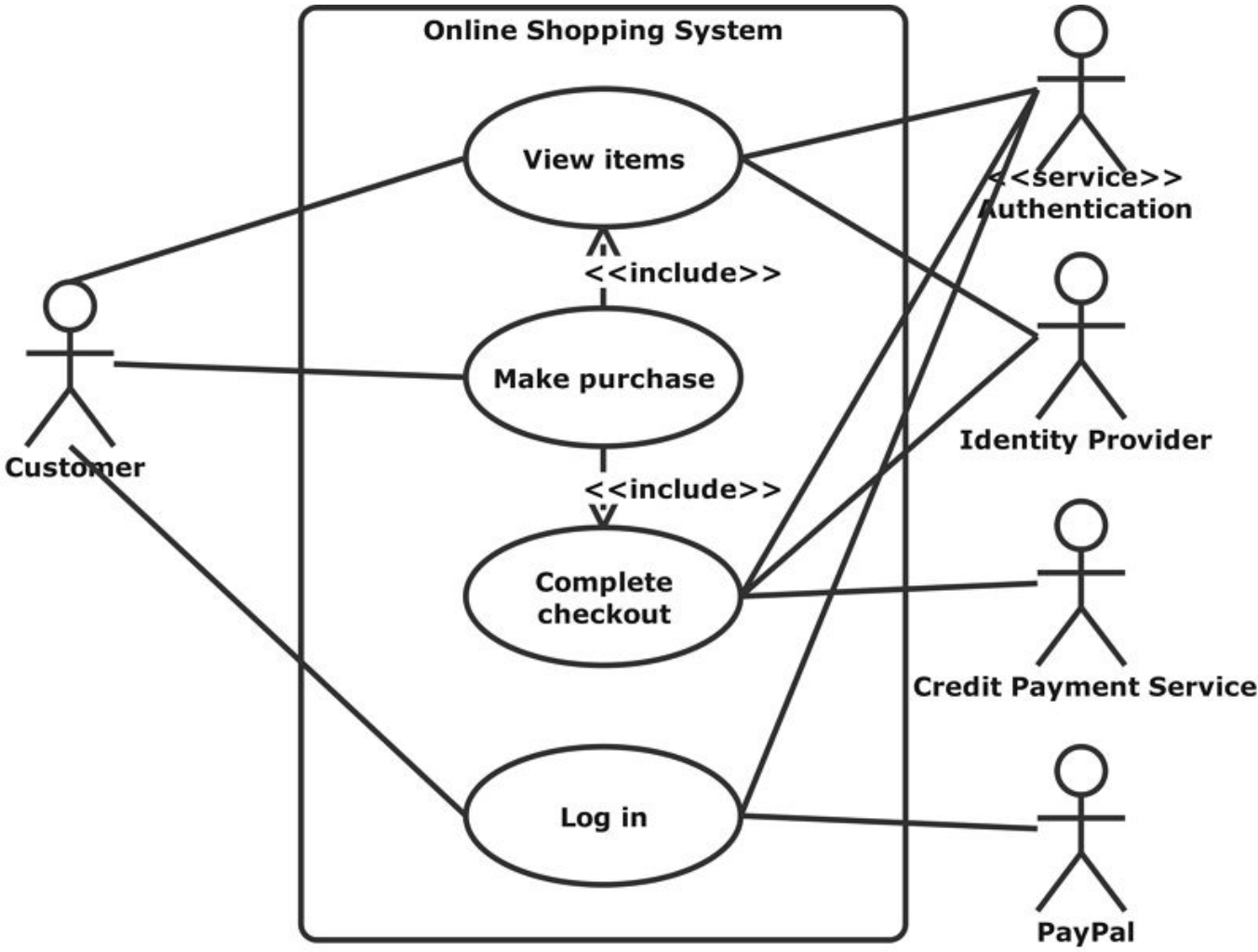


ELEMENTOS DE DIAGRAMA DE CASO DE USO



EJEMPLO DE DIAGRAMA DE CASO DE USO

Comportamiento en cada elemento y su relación



SEMANA 7

ARQUITECTURA DE SOFTWARE (N-Tire)

Preguntas que se debe hacer

- puede ser?
- debe ser?
- es?

INTERFAZ

Si nos ponemos a pensar que cuando necesitamos una acción heredada de otra clase si o si debemos

Propiedad	CLASES	INTERFAZ
Propiedades	SI	NO
Metodos	Métodos concretos heredados del padre	No tiene métodos concretos
Características	Todos los hijos tiene los mismos metodos	Cada hijo hereda los métodos y PUEDE CONTROLAR el comportamiento de cada método.

Propiedad**CLASES****INTERFAZ**

UML

SE EXTIENDE

SE IMPLEMENTA

ARQUITECTURA DE SOFTWARE

Estructura de capas

CAPAS

- PRESENTACION - GUI
- APLICACIÓN - código
- DATOS - persistencia de datos

NOTAS I BIMESTRE

PROYECTO 3 Pt

Fecha: Miercoles

- *1pt: Prototipo* : (Mockup/Figma/drawio) de todas las pantallas de aplicación
- *1pt Presentación*: Powepoint explicar
- *1pt Demo* : 20% o 30% de la app

WORKSHOP 2 Pt

Jueves 12 de Enero de 2023

- Todas las clases documentadas.
- Exportar en PDF

EXAMEN

Temas:

- Todo hasta interfaces
- Interfaces graficas - Botones