Temario parte 1

Descarga e instalación de Java y Eclipse.

Introducción a Java.

Estructuras principales del lenguaje

POO

- Objetos y Clases. Herencia.

- Polimorfismo. Encapsulación

- Clases Abstractas

- Etc.

Programación de interfaces gráficas

- Swing

- JavaFX

Eventos

Archivos JAR

- Ejecutables

- Distribuibles

Tratamientos de errores (Excepciones y depuración)

Programación genérica

Colecciones

Programación multihilo. Threads.

Programación con archivos externos

SOCKETS

Programación para BBDD. JDBC.

JavaBeans

Jakarta EE

  - JSP

  - Servlets

  - JSTL

  - Persistencia de datos

  - Web sockets

  - Seguridad

- Despliegue

etc

Caracteristicas de java.

- De 1991 a 1994 intentaron vender la tecnólogia a diferentes empresas del ambito tecnológico.

  No hubo éxito.

- El proyecto liderado por gosling y naughton queda en stand-by

- Corría el año 1994 e Internet se hacía más grande… Pensaron que las características de Java

se ajustaban como un guante a la naturaleza de Internet.

JAVA es comunmente utilizado para:

- En el sentido de que se intentó quitar las características más “Complejas” de otros lenguajes­ de programación (artimética de punteros, ficheros de encabezado etc) y sobre todo que ocupa poco.

- Orientado a objetos.

- Distribuido: Buen tratamiento de redes. Buena programación para internet.

- Seguro: Como fue pensado para programar en red (internet) se hizo seguro.

- Leer o escribir ficheros sin permiso.

- Desbordar la pila de ejecución.

- Corrupción de memoria.

- Independiente respecto a plataforma (Write Once, run everywhere).

- Adaptable: tipo de datos primitivos iguales en todas las plataformas.

- Compilado e intrerpretado.

- Alto rendimiento. Multihilo.

- Soporte para desarrollo empresarial.

- Compatibilidad hacia atrás.

- Amplia comunidad y soporte.

11-08-2025 video 4

- Anatomia de un programa java sencillo.

- Estructuras principales.

11-08-2025 video 5

- Tipos primitivos

- Declaración de variables

- Inferencia de tipos

Tipos primitivos

- Enteros

  - Int: 4 bytes de espacio para almacenamiento. Desde -2, 147, 483, 684, hasta 2, 147, 483, 647

  - Short: 2 bytes de espacio de almacenamiento. Desde -32, 768 hasta 32,767

  - Long: 8 bytes de espacio para almacenamiento. Una barbaridad. Sufijo L

  - Byte: 1 byte de espacio para almacenamiento. Desde -128 hasta 127.

- Coma flotante (decimales)

  - Float: 4 bytes de espacio para almacenamiento. Aproximadamente 6 a 7 cifras decimales   significativas.

  - Double: 8 bytes de espacio para almacenamiento. Aproximadamente 15 cifras decimales   significativas.

- CHAR: para representar caracteres.

- Boolean: 2 únicos valores. True/False

Variables en java.

¿Qué es una variable? Espacio en la memoria del ordenador donde se almacenará un valor que podrá cambiar durante la ejecución del programa.

¿Por qué hay que utilizar variables? Porque a la hora de crear un programa surge la necesidad de guardar datos temporalmente que necesitarás utilizar en el futuro en ese mismo programa.

¿Comó se crea una variable en Java? Especificando el tipo de dato que almacenará en su interior + el nombre de la variable. P. ej: int salario;

¿Qué es inicializar una variable? Es darle un valor. Nombre\_variable=valor. p.ej:

Salario = 2000; Java no permite utilizar variables que no se hayan iniciado.

¿Qué es la inferencia de tipos?

- Es una carácteristica de algunos lenguajes de programación que permite al compilador deducir automáticamente el tipo de una variable en función del valor que se le asigna, sin que el programador tenga que especificarlo explícitamente.

- El compilador analiza el valor asignado y deduce el tipo más adecuado.

¿Dónde se puede usar “var” en java?

- Variables locales dentro de los métodos

- En bucles for-each y for

- En try-with-resources

¿Dónde no se puede usar?

- Parámetros de métodos

- Variables de clase o atributos de instancia

- Tipos genéricos.

¿Y por qué se inculyó en java?

- Reducir la verbosidad.

- Hacer el código más legible.

- Evitar repeticiones en tipos evidentes.

- Alinear Java con tendencias modernas de programación.

- Mayor productividad en código complejo.

- Mejor integración con lambdas y streams.

- Facilitar cambios de tipos.

18-08-2025 y 19-08-2025

- Inferencia de Tipos – poder declarar solo una vez la variable y utilizar distintas veces el valor para guardar, cambiar el valor o parsear entre los diversos tipos de java.

- Sintaxis de todo lo anterior. Inserción de comentarios.

- Operadores en Java numéro 1:

Aritméticos:

1. + : Suma

2. - : Resta

3. \* : Multiplicación

4. / : división

5. % : Modulo

- Lógicos, relacionales y booleanos.

> : Mayor que .

< : Menor que.

>= : Mayor o igual que.

<= : Menor o igual que.

!= Distinto que.

------ Continuación

== : Igual que.

&& : Y lógico.

||: o lógico.

No es valido en java comparar con el objeto String, ya que como bien se menciona es un objeto.

String nombre = “Juan”;

String nombre2 = “Ana”;

if (nombre == nombre2) no esta bien hacerlo de esta forma.

- Incremento y decremento

++ : incremento

- - : Decremento

+= n.º : incremento

-= n.º : decremento

int edad = 15;

edad++; esto vale 16 ya que incrementa en 1 el valor;

- Concatenación (textos, textos y números)

- + : une o concatena.

Operadores en Java 2.

Bit a bit (Bitwise)

& : And bit and bit

| : OR bit a bit

^ : XOR bit a bit

~ : complemento bit a bit.

« : desplazamiento izquierda

» : desplazamiento derecha

»>: desplazamiento derecha sin signo.

- Operador ternario

  ? : Condicionales.

- Asignación compuesta

  \*= Multiplicación

  /= División y asignación

  %= : Módulo y asignación

video 8 pildoras informaticas 20-08-2025.

-- - - - - -

- Declaración múltiple de variables.

- Declaración de constantes. Uso “final”

Promoción de tipos en java

- Si una operación involucra un tipo menor a int (byte, short, char), se promueve a int antes de realizar la operación.

- Si una operación involucra un int y un long, el resultado es long.

- Si involucra un float, el resultado el float.

- Si involucra un double, el resultado es double.

Clases en Java

Existen las clases creadas por uno mismo llamadas clases propias.

Y tambíen existen clases llamadas Clases Predefinidas: ejemplo – String, Math, Array, Thread etc etc....

Video 9 – 20-08-2025

Organización API Java.

Módulos, Paquetes, Clases que se dividen en clases con métodos y clases con atributos.

Si el método es estatico, quiere decir que debes poner el nombre de la clase primero y después el nombre de la clase que se quiere ejemplo Math.sqrt() ..

Video 10 – 20-08-2025

- Cálculos Con Clase Math.

- Casting

Conversiones

hay 2 tipos en java, las implicitas es cuando se convierte un tipo más pequeño a uno más grande widening casting, y la explicita es de más grande a mas pequeño (Narrowing casting) este tipo lo haces tu.

Video 11 – 20-08-2025

- Cálculos con Clase Math.

- Casting

- Métodos habituales:

  Pow()

  Random()

  abs()

  max()

Video 12 – 20-08-2025

- Manipulación de cadenas con clase String

String no es un tipo primitivo.

¿ Como almacenar una cadena de caracteres ?

- String mi\_nombre = “Juan”; donde mi\_nombre es un objeto (instancia, ejemplar) de la clase String.

Métodos (más usados) de la clase String para manipulación de cadenas de texto:

- length(): devuelve la longitud de una cadena de caracteres.

- charAt(n): devuelve la posición de un carácter dentro de una cadena. (Las posiciones empiezan a contar de 0).

- substring(x,y): devuelve una subcadena dentro de la cadena, siendo X el carácter a partir del cuál se extrae e Y en el nº de caracteres que se requieren extraer.

- equals(cadena): devuelve true si dos cadenas que se comparan son iguales y false si no lo son. Distingue mayúsculas y minúsculas.

- equalsIgnoreCase(cadena): Igual que el anterior pero sin tener en cuentra mayúsculas y minúsculas.

Video 13 – 20-08-2025

// API DE JAVA E STRINGS INMUTABLES.

    String saludo="Hola";

    saludo=saludo+" Mundo cruel";

    System.out.println(saludo);/\*Losobjetosdetipostring son inmutables,esdecirno se

                  pueden eliminar ni modificar en la memoria ocuapada. Stack and Heap \*/

Api de Java. Paquetes.

¿Por qué son necesarios los paquetes? ¿ Por qué se inventaron ?

- Para organizar las clases

- Para evitar conflictos de nombres

- Para controlar la visibilidad de las clases.

El paquete más importante de todos en java es java.lang y por lo mismo no se importa en la clase java a lo contrario de los otros que tienes que agregar la palabra import más el paquete que vayas a utilizar import – Java.Util.\*.

¿Para que sirven los módulos?

- Encapsulación fuerte.

- Reducción del tamaño de aplicaciones.

- Mejor organización del código.

- Mayor seguridad.

- Mejor rendimiento.

Ejemplos de modulos:

Java.base

Java.desktop

Java.sql

Java.xml

Java.naming

Jdk.jartool

Jdk.attach

Jdk.jdi

Jdk.jshell

Jdk.jfr

Java.smartcardio

los modulos se incluyen en un archivo llamado module.info.

Video 14 : 25-08-2025 - 40 min

Entrada de datos con Scanner.

Nos permitira ingresar datos desde la consola del sistema.

Herencia:

Java.lang.Object esta heredando de la clase padre object.

Java.util.Scanner

Implements palabra reservada permite que una clase de java adopte comportamientos definidos en las interfaces que le indica la documentación.

Cuando son métodos estáticos requiere de llamar la clase de donde provienen los métodos Math.sqrt(“\_”); como en este ejemplo.

Cuando no son estaticos, para poder utilizar estos métodos de la clase, crear un objeto o una instancia que pertenezca a la clase que no es estática.

Constructor:

Construye, se tiene que definir en esos objetos, cual es su estado inicial de los objetos, su característica principal es que tiene el mismo nombre que la clase.

Podemos tener varios constructores con el mismo nombre que se denomina sobrecarga de constructores.

El objetivo es darle a los objetos Scanner diferentes estados iniciales.

La clase Scanner sirve para leer datos de entrada desde diferentes fuentes, ejemplo desde la consola de Java, también permite leer datos desde archivos, desde cadenas de texto tipo String.

Leer datos de la consola recibe un InputStream.

El metodo, devuelve un String del tipo Scanner.

Video 15 26-08-2025 29:01 min

Entrada de datos con Joption pane y expansion de la clase

scanner con enteros.

Video 16 26-08-2025 44:38 min  
  
Estructuras de control de flujo.  
  
Condicionales I. Condicional IF.

Nota nueva del video 15 actualizando con los comentarios del autor del curso:  
si quieres puedes utilizar unicamente el método que vayas a usar, para mejor legibilidad en el código.  
  
También podemos utilizar en el código el paquete y la clase que utilizaremos en el código sin hacer la importación cotidiana que se hace, esa es otra forma de agregar el método que se va a utilizar, en este caso para el ejemplo del video 15 que es la Clase JoptionPane.

Clase envolvente se utiliza para sentidos especificos de código para que te devuelva un Objeto de lo que necesites en ves de utilizar un tipo primitivo.

Flujo de ejecución – El orden en el que se van ejecutando las instrucciones por parte del interprete java.

Sintaxis:  
if(condicion){

- Código a ejecutar si la condición es verdad (true);

}

si la condición es falsa, no se cumple y no entra a la instrucción.

Y hay otra opción para el if.

if(condicion){

- Código a ejecutar si la condición es verdad (true);

} else {

- Código a ejecutar si la condición es falsa.

}

solamente se construye el else si es necesario.

Video 17  
27-08-2025

# Condicionales II. Ternario y Switch: novedades.

Scanner no puede volver a ser abierto una vez que la entrada por teclado en system.in haya sido cerrada por el metodo “objeto”.close(); ya que esto significa que la entrada o el buffer de lectura ya no esta permitiendo modificar o entrar los datos por teclado.

Puedes abrir varios scanner creando el espacio creando el objeto pero el método de entrada debe de cambiar.  
  
Ejemplo: Scanner entrada2 = new Scanner(InputFileStream); aca ya no va declarado System.in por el cierre del scanner en código anterior …..

Condicional Switch

- Switch (Valor a evaluar ) {

- Case valor1:

1. Código a ejecutar;

Break;

- Case valor2:

2. Código a ejecutar;

Break;

}

String resultado = switch(dia\_semana){

// Segunda forma del switch a partir del java 14 se modifico la utilización del switch.

/\* Se guarda el valor ahora en la variable que recibe del switch \*/

case "Lunes", "Martes", "Miercoles", "Jueves", "Viernes" -> "Laborable";

case "Sabado", "Domingo" -> "Festivo";

// case "Martes"-> "Laborable";

//

// case "Miercoles"-> "Laborable";

//

// case "Jueves"-> "Laborable";

//

// case "Viernes"-> "Laborable";

//

// case "Sabado" -> "Festivo";

//

// case "Domingo" -> "Festivo";

//

default->{

System.***out***.println("Procesando...");

yield "Día no valido";

// la novedad es que puedes mandar varias lineas de código en el switch

// colocando la instrucción yield;

}

};

System.***out***.println(resultado);

}

Situación usar if vs usar switch.

Comparar rangos o condiciones ok not

comparar valores exactos ok ok

Ejecutar varias instrucciones ok ok

Evaluar expresiones complejas ok not

}

Video 18

37:00 min 27-08-2025

BUCLES

Indeterminados:

sin ejecutar el programa, no sabes cuantas veces lo va a ejecutar ya que revisando el código no sabes cuantas veces va a repetir la condicion que le haya o hayan impuesto tu u otros.

- While

- Do-while

Determinados //sin ejecutar el programa tu sabes cuantas veces lo va a ejecutar.

- For

- For-each

While:

- while (condición){   
  
// cuando la condición es false, deja de repetir las lineas de código y finaliza la ejecución de la estructura.

-Línea 1

-Línea 2

-Línea 3

-Línea 4

-Línea 5

}

Video 19 30:00 min 28/08/2025

Estructuras de control de flujo:  
  
- Planteamiento del ejercicio.

Se deberá adivinar el número aleatorio pidiendo al usuario un número entre el 1 y el 100, guardando el número de intentos mostrándolos en consola.

Video 20 27:54 min   
29-08-2025

Bucles 3 While vs Do-While

Se utiliza el ejemplo anterior de pedirle al usuario un número aleatorio, para cambiar la estructura de un while a un do-While.

Se elaborara un programa que nos de el peso ideal para mujer y para hombre desde un JoptionPane para pedir, genero y altura solo con la letra M y H y si lo pide de otra forma, que te vuelva a pedir el dato.

Y el otro Joption para pedir la altura en cm sin repetirse.

Video 21 33:00 min   
29-08-2025

Bucles IV FOR:

El bucle for se puede ejecutar las veces que sea necesario, y lo puedes repetir definida o indefinidamente, se utiliza para recorrer, caracteres, cadenas, listas, arreglos, matrices, valores de bases de datos al igual que el while, recorrer colecciones, etc.

Sintaxis:

- for (inicio bucle; condición; contador bucle){

Linea 1

Linea 2

Linea 3

Linea 4

}

for(int i = 25; i<35; i++) {

//

// System.out.println("Leonardo " + i + " numero de veces impresas");

// // el contador i nos da el número de veces que imprimimos el valor en la consola.

//

// }

// System.out.println("Leonardo");

// System.out.println("Leonardo");

// System.out.println("Leonardo");

// System.out.println("Leonardo");

// System.out.println("Leonardo");

// System.out.println("Leonardo");

// System.out.println("Leonardo");

// System.out.println("Leonardo");

Video 22 Bucles V. 24:33 min  
Bucle For II  
01-09-2025

Se realiza la corrección del programa del video 21 donde si tiene más de una arroba, es valido aun así, y para un correo tener 2 o más arrobas no es correcto.

También se realizará un programa que calcule el factorial de un número.

Se utilizará un JoptionPane para pedir el número a calcular.

Video 23   
01-09-2025

Arrays I 33:32 min

- Utilidad y sintaxis básica.

- Novedades y alternativas a los Arrays en versiones 9 y 14 de java.

- ¿ Qué es ?

- Estructura de datos que contiene una colección de valores del mismo tipo.

- ¿ Para que sirven ?

Para almacenar valores que normalmente tienen alguna relación entre si.

- Sintaxis  
 int[] miArray = new int[10];

miArray[0] = 15;

miArray[1] = 25;

miArray[2] = 8;

miArray[3] = -7;

miArray[4] = 92;

viene marcado por un indice, la primera posición empieza en el número 0.

forma resumida del array int[] miArray={15, 25, 8, -7, 92};

Java9. API java.util.List.of(…)

Alternativa ligera.

-List<String> nombres = List.of(“Ana”, “Luis”, “Ana”);

Ya no se ven los indices en este tipo de lista.

¿Cuándo usarlo en lugar de Arrays?

- Cuando se vayan a crear listas inmutables.

- Mejor integración con Streams y APIs modernas.

- Labores de testing.

¿Cuando no usarlo?

- si necesitas modificar la lista.

- si vas a trabajar con primitivos.

- si los datos provienen de entradas de usuario, bules etc.

JAVA 14. Records y Arrays.

- Desde java 14 podemos crear estructuras que incluyan arrays de forma más sencilla y limpia: los records.

- public record Alumno(string nombre, int[] notas){}

- Alumno a = new Alumno(“Luis”, new int []{8, 9, 10});

Video 24 02-09-2025

Arrays-II Bucle for-each

ejerciciós foreach 25:51 min

Video 25 02-09-2025

Arrays III. Arrays bidimensionales.

34:00 min

Los arrays bidimensionales tienen el ancho y el alto.

Se almacena en un array normal, en cada una de las posiciones, otro array.

Para identificar cada una de las posiciones utilizamos 2 indices.

0,1 0,2 0,3 0,4 0,5 0,6 0,7 0,8

1,0 1,1 1,2 1,3 1,4 1,5 1,6 1,7

int[][] nombre\_array = new int[5][3];

Especificamos cuantos elementos almacenará en la primera y segunda dimension por eso se utilizan los corchetes.

Video 26 33:12 min.

Arrays IV

Ejemplo practico con array bidimensional.  
04-09-2025

10% 11% 12% 13% 14% 15%

10,000 10,000 10,000 10,000 10,000 10,000

11,000 11,100 11,200 11,300 11,400 11,500

12,100 12,321 12,544 12,769 12,996 13,255

13,310 13,676.31 14,049.28 14,428.97 14,815.44 15,208.75

14,641 15,180.70 15,735.19 16,304.74 16,689.60 17,490.06

Video 27 36:56 min.

04-09-2025

# **¿Qué es la POO?**

Programación orientada a objetos.

- ¿Qué es?

- ¿Por qué surge?

- ¿Qué características tiene?

Paradigmas de la programación se divida en 2 ramas:

Programación imperativa y estructurada.

Programación Orientada a objetos.

Programación imperativa:

Algunos ejemplos de lenguajes: Fortran, Cobol, Basic etc.

Desventajas:

- Unidades de código muy grandes en aplicaciones complejas.

- En aplicaciones complejas el código resultaba difícil de descifrar.

- Poco re-utilizable.

- Si existe fallo en alguna línea del código, es muy probable que el programa caiga.

- Aparición frecuente de código espagueti.

- Difícil de depurar por otros programadores en caso de necesidad o error.

Programación Orientada a Objetos

- ¿En qué consiste?

- Trasladar la naturaleza de los objetos de la vida real al código de programáción.

- ¿Cuál es la naturaleza de un objeto de la vida real?

- Los objetos tienen un estado, un comportamiento (¿Qué puede hacer?), y unas propiedades.

- Pongamos un ejemplo: El objeto coche.

- ¿Cuál es el estado de un coche? Un coche puede estar parado, cicurlando, aparcado etc.

- ¿Qué propiedades tiene un coche? Un coche tiene un color, un peso, un tamaño etc.

¿Qué comportamiento tiene un coche? Un coche puede arrancar, frenar, acelerar, girar etc.

Programación Orientada a objetos:

- Alguno ejemplos de lenguajes: C++, java, visual.NET etc.

- Ventajas:

- Programas divididos en “Trozos”, “Partes”, “módulos” o “clases”. Modularización.

- Muy re-utilizable. Herencia.  
  
- Si existe fallo en alguna línea del código, el programa continuará con su funcionamiento. Tratamiento de Excepciones.

- Encapsulamiento.

P.O. Procedimientos – Programación orientada.

Después surgio la programación orientada a objetos.

Clase 1, Clase 2, Clase 3, Clase 4, Clase 5.

Modularización: Se utiliza sobretodo para la reutilización de código de un programa a otro sin perder funcionalidad.

Vocabulario de la POO

- Clase

- Objeto

- Ejemplar de clase. Instancia de clase. Ejemplarizar una clase. Instanciar una clase.

- Modularización

- Encapsulamiento / encapsulación.

- Herencia

- Polimorfismo.

Clases y Objetos.

Video 28 39:17 min

04-09-2025

Clase

- Modelo donde se redactan las características comunes de un grupo de objetos.

Objeto

- Ejemplar perteneciente a una clase, cada uno tiene sus elementos diferenciadores, derivan de esa clase con sus propias características.

Objeto:

- Tiene propiedades (atributos):

Color

Peso

Alto

Largo

- Tiene un comportamiento (¿Qué es capaz de hacer?):

Arrancar

Frenar

Girar

Acelerar

- Accediendo a propiedades de objeto desde código (pseudocódigo):

Renault.Color = “Rojo”;

Renault.peso = 1500;

Renault.ancho = 2000;

Renault.alto = 900;

- Accediendo a comportamiento de objeto desde código (Pseudocódigo):

Renault.arranca();

Renault.frena();

Renault.gira();

Renault.acelera();

nom\_obj ”.” propiedad = “valor de prop”.

nom\_obj “.” nomb\_comportamiento.

Los métodos de una clase definen el comportamiento de futuros objetos.

A las clases creadas llevan un método constructor, que es un método especial, se encarga de darle un estado o un comportamiento inicial al objeto, tiene que estar definido.

Si no hay constructor, la clase lo define como, constructor por defecto, los objetos usan camelCase para crear la clase.

El constructor tiene exactamente el nombre de la clase y tiene que tener el modificador de acceso public.

Puedes crear los objetos desde la misma clase o desde otra clase.

Si creas otra clase puedes crearla con PascalCase o UpperCamelCase para ser una clase de objeto valida y también puedes usar el uso del plural.

Video 29 Modularización y Encapsulamiento.

08-09-2025

41:00 min

Modularización:

Termino que se utiliza para realizar el código sin depender de otros métodos, clases o sin depender de otras instancias en el programa o aplicación creada.

Encapsulamiento:

Termino que se utiliza para almacenar funciones especificas de código privado para que no se pueda utilizar en otro modulo, clase o método del programa o aplicación.

Video 30 Getters, Setters y Constructores.

08-09-2025

29:40 min.

Setter: permite establecer el valor de un campo.

Getter: permite obtener y ver el valor del campo establecido con setter.

Método constructor: Tendrá el mismo nombre de la clase y al principio de ella.

Con esto defines cual es el estado inicial en el que estará asignado el objeto de esa clase constructor.

Video 31  
25:00 min

10-09-2025

Uso del .this y parámetros por referencia y por valor.

Video 32

43:00 min

11-09-2025

Creación de Objetos I.

los setters, casi nunca se ocupan en otras clases más que en la que se esta utilizando, podemos encapsular métodos public como private para evitar alguna modificación desde el main.

Video 33

29:13 min 11-09-2025

Creación de Objetos II,

Constructor con parámetros.

Se construirá una app con tres valores de tipo objeto,

nombre, sueldo, fecha de alta y subir Sueldo de un empleado.

Patrón de diseño: Diseño al crear alguna aplicación con el objetivo de que las personas desarrolladoras de aplicaciones puedan saber donde se encuentra cada objeto de manera organizada y como esta estructurada la aplicación.

Patron Modelo, vista controlador:  
todas las clases que tengan que ver con el manejo y acceso de bases de datos, deben estar en el modelo.

Todas las clases que tengan que ver con la interfaz visual, que es lo que el usuario ve en pantalla tendrán que estar en una parte del patrón que se llama vista.

Y en el controlador es un nexo o unión de comunicación entre las clases que están en el modelo y las clases que están en la vista.

Y para ello tendremos un paquete para modelo, un paquete para la vista, y uno para controlador, cada uno de ellos con sus clases en el interior.

Si necesito 10,15 o más clases, todas deben de ir en cada uno de los paquetes.

Por convención para crear paquetes, es obtener el dominio y el nombre del paquete.  
  
Ejemplo:

es.pildoras.poo

Video 34

32:23 min 11-09-2025

Creación de Objetos III,

Constructor de parámetros continuación del video 33,

Se crean los setters y getters para la continuación del programa empleado agregando la clase GregorianCalendar y hereda la clase getTime.

Video 35

24:58 min 12-09-2025

Java POO IX. Construcción objetos IV.

Se finalizo el programa para dar información del contrato de cada Empleado más aumento de sueldo con objetos y arreglos.

Video 36

20:07 min 15-09-2025

Constantes uso final.

Se realizará un programa que sea capaz de crear instancias u objetos de tipo empleado en los cuales tenga su nombre y el área a la que pertenece su empresa.

Video 37

20:07 min 15-09-2025

Constantes Uso-Static

A los empleados le podemos dar un número id para cada uno de ellos.

Static: se refiere a compartir una sola variable para los objetos que se crean, sin que cada uno tenga una copia de cada variable creada para cada objeto, si se tiene una clase que tiene un estado inicial con un objeto id, a cada nueva instancia; Se le crea la copia de las variables que se crearon en la clase.

Para poder utilizar una variable static no se utiliza un objeto, se utiliza el nombre de la clase mas la variable o la constante Ejem: Math.PI, Empleados.Id.

Video 38

17:02 min 15-09-2025

Métodos Static Sobrecarga de Constructores.

Método estático nunca puede acceder a las variables de clase, solamente si son constantes del mismo tipo estático.

Métodos static.

No actúan sobre objetos.  
No acceden a campos de ejemplar (variables/constantes declaradas en la clase), a menos que estas sean también static.

Para llamarlos se utiliza el nombre\_clase.nombre\_metodo

Video 39

19:31 min 17-09-2025

Sobrecarga de constructores

Una sobrecarga de constructores una misma clase puede tener varios métodos con el mismo nombre de una clase y si se utiliza una sobre carga de constructores los métodos constructores deben tener diferente número de argumentos.

Video 40

23:15 min 17-09-2025  
Herencia

¿Qué es la herencia?

La herencia es la reutilización de código que permite utilizar elementos ya declarados sin tener que volver a agregarlos, ya que pueden compartir los mismos elementos, pero no todos, puesto que se tienen otras características que se agregan.  
  
Una camioneta puede tener lo mismo que un coche, pero no un coche puede tener lo mismo que una camioneta, hay algunos, no todos pero posiblemente se agreguen más parámetros.

En java las clases que heredan de lo que tienen las otras clases, no solo heredan las características sino también heredan los métodos.

Además java no admite la herencia múltiple, como en los diversos casos en la vida cotidiana.

Video 41

21:08 min 17-09-2025  
Herencia II

Se crea una instancia de tipo furgoneta, y a través de esto podemos acceder a los métodos de la Super Clase Coche más los datos de la clase de tipo furgoneta.

Diseño de la herencia. La regla “Es un”…

coche → furgoneta

Clase principal Vehículo, programar las propiedades generales que pueden tener todos los vehículos que vemos.

Coche – furgoneta - camión – Moto.

Video 42

22:32 min 17-09-2025

Herencia III Diseñando la herencia.

Dándole seguimiento a la clase Empleado, en el cual se creo dando los datos como el nombre, sueldo, fecha de contrato y subeSueldo. Con un ID único para la creación de objetos de tipo empleado.

¿Y si queremos crear Jefes?

¿Y si estos Jefes reciben además del sueldo, un incentivo?

¿Me sirve la clase Empleado para crear Jefes?

Crear nueva clase que diferencie quien es un empleado y quien es un jefe y además nos permita dar un incentivo a los jefes.

Jefatura, se llamará la clase y herede las características de empleado.

Recuerda: Regla “Es un…”

¿Un Jefe es un Empleado? (siempre)

¿Un Empleado es un jefe? (no).

Superclase será empleado y la clase Jefe es una subclase.

Video 43

24:39 min 18-09-2025

La herencia IV Polimorfismo

Enlazado dinámico.

¿Qué es el polimorfismo y el principio de sustitución?

Principio de sustitución: se puede utilizar un objeto de la subclase siempre que el programa espere un objeto de la Superclase.

O lo que es lo mismo: Un Objeto se puede comportar de diferente forma dependiendo del contexto. Las variables objeto son polimorfas.

Video 44

23:19 min 18-09-2025

La herencia V

Refundición de objetos (Casting) clases y Métodos final.

Podemos hacer el casting entre objetos, como en el ejemplo de la clase Uso\_Empleado basándonos en la regla de “Es un”, ya que al ser un jefe un empleado, podemos hacer el casting adecuado para pasar una instancia de tipo empleado a una instancia de tipo Jefatura.

double num1 = 7.5;

Int num2 = (int) num1;   
  
// Este tipo de casteo es similar a lo que se hace cuando es de tipo objeto como en este caso.

Jefatura jefa\_Finanzas = (Jefatura) misEmpleados[5];

Ya que mis empleados[5] tienen argumentos o parámetros definidos como instancia almacenados en la posición 5 para visualizar los métodos creados en la subclase Jefatura y poder utilizar como en el ejemplo del ejercicio el método.

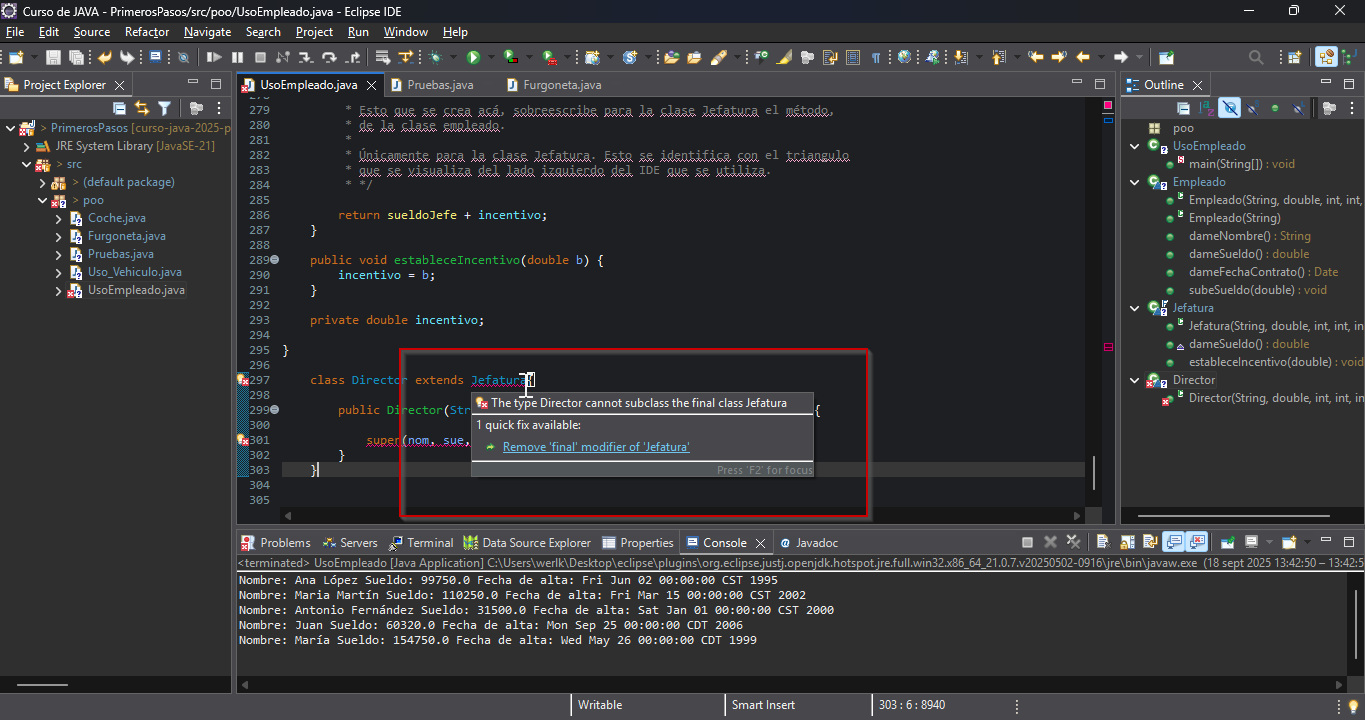
jefa\_Finanzas.estableceIncentivo(55000);

Jefatura jefe\_compras=(Jefatura) misEmpleados[1];

esta instrucción es la misma que la de jefe\_finanzas solo que da error porque no puede ser empleado un jefe.

Clases de tipo final, se detiene el flujo de herencia.

Clase empleado → Clase Jefe → Clase Director. Si no quieres crear y heredar alguna otra clase se define el tipo final en la última clase para que no se realice la herencia.



class Director extends Jefatura{

//Error   
 public Director(String nom, double sue, int anio, int mes, int dia) {

super(nom, sue, anio, mes, dia);

}

}

final class Jefatura extends Empleado { //por el elemento final.

}

También podemos definir métodos de tipo final para que no se pueda hacer la sobre escritura de métodos, para que no se puedan construir métodos con el mismo nombre y que funcione diferente al que construimos.

Para ello en el ejemplo que tenemos para la clase empleado, le ponemos final al método dameSueldo() para que no se pueda hacer la sobre escritura para los siguientes programadores;

puedes hacer otros métodos para que los hereden con nombres diferentes en la clase.

Clase calendar de la API de java y otros métodos, se tienen como final.

Ejemplo: public final Date getTime() de

java.util.Calendar;

Video 45

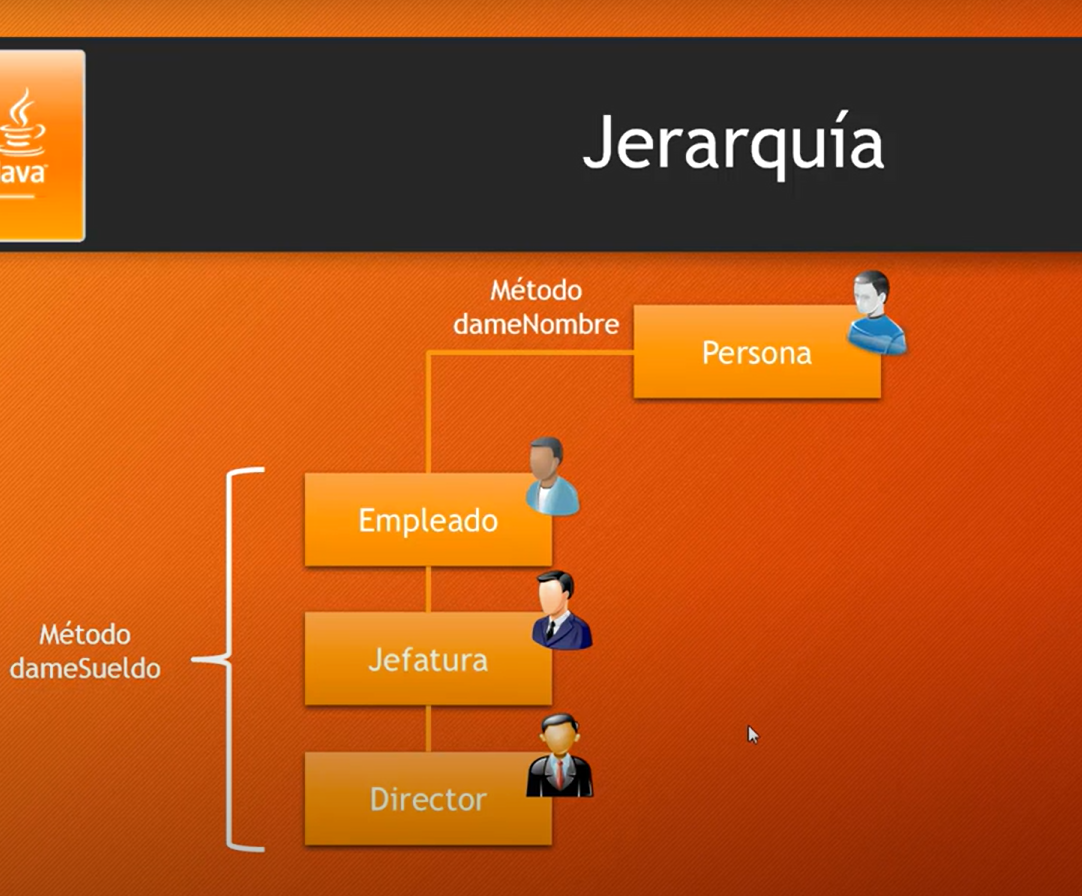
16:25 mín 18-09-2025

La herencia VI, Clases Abstractas.

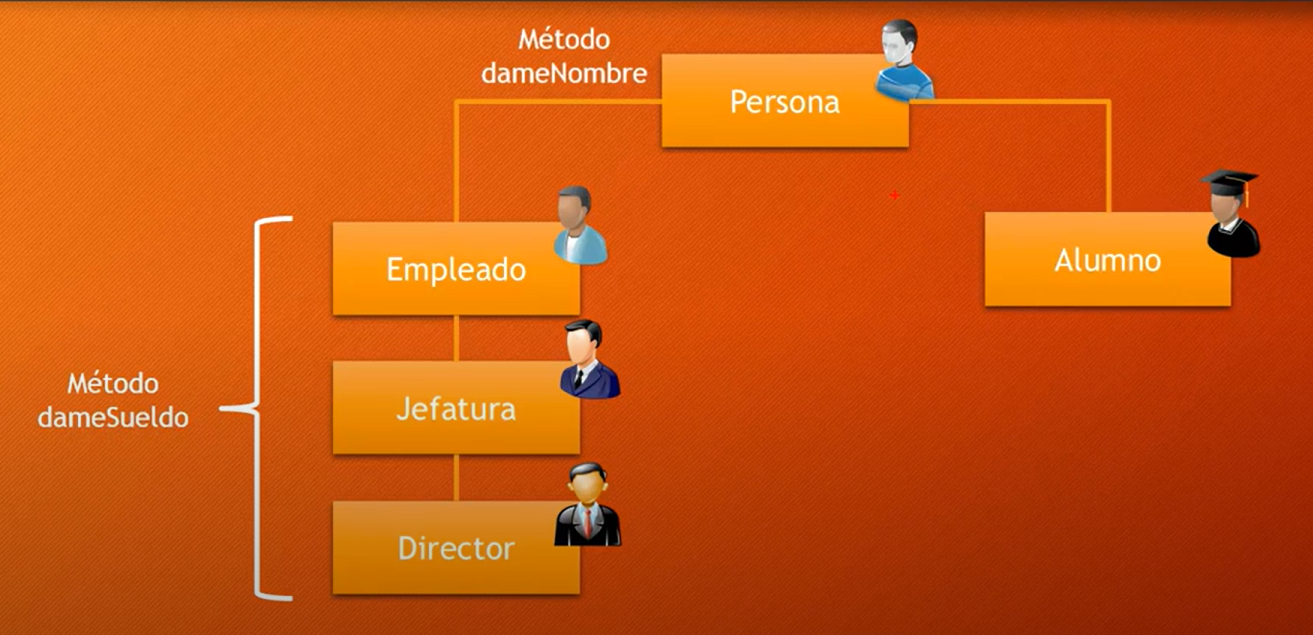
Podemos tener una clase maestra, líder o potente con más cosas que realizará para las demás clases.

La subclase que es más especializada, es la ultima, ya que tiene más métodos de los cuales la mayoría los heredo de las clases antecesoras a esta, la clase máster.

Si el flujo de herencia es al contrario, subiendo por las clases, las clases van siendo menos especializadas, más genéricas o mejor conocidas como abstractas.



Si al final se quiere agregar otra clase, debería de heredar de la clase persona, pero no de empleado, jefatura o director, porque no tiene un sueldo pero si tiene nombre, heredando de persona pero no tiene nada que ver con las otras 3, apartada de ellas siguiendo la regla de “Es un”.



si se quiere dar una descripción no es la misma que llevan las clases empleado, jefatura, director a la de alumno, en estos casos se declara el método como abstracto.

Declaración de método abstracto:  
un método abstracto no lleva llaves ni recibe parámetros. Solamente se definen.

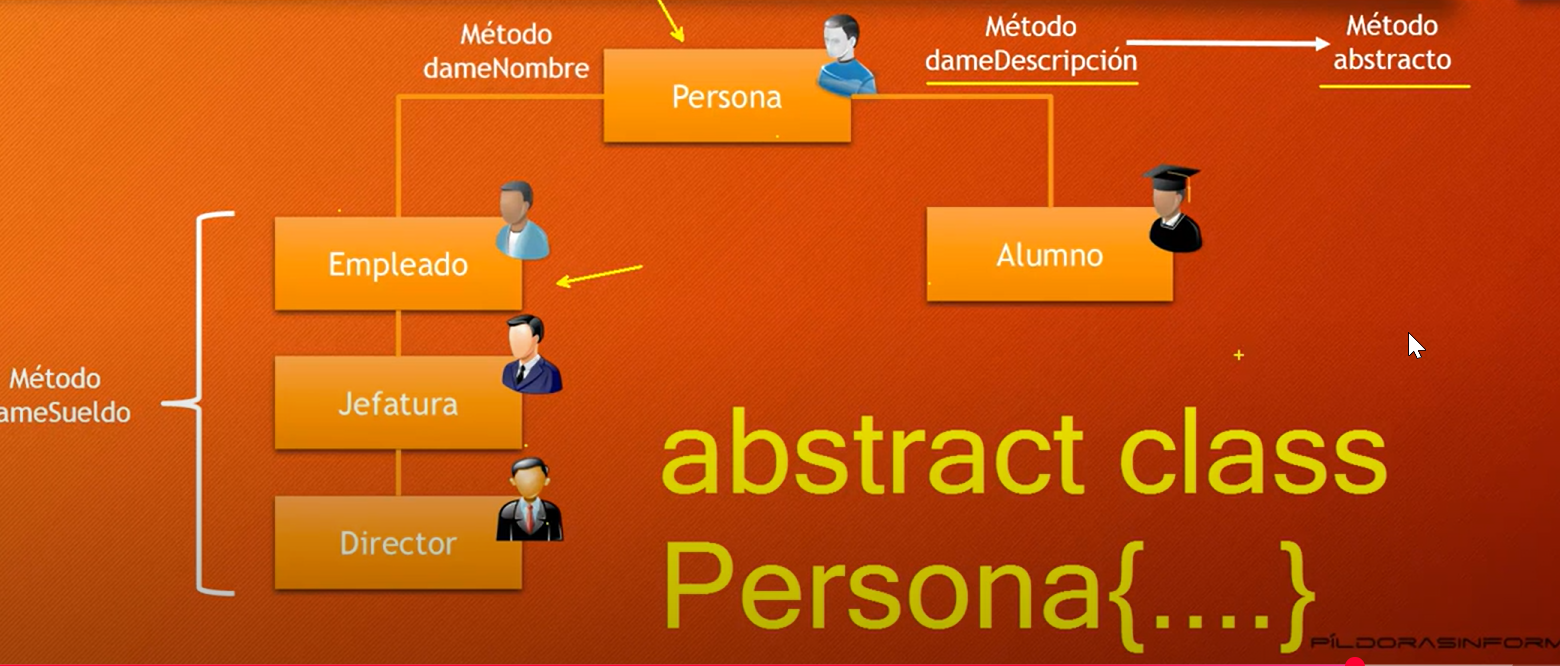
public abstract String dameDescripción();

por regla, en cuanto declaras un método abstracto, estas obligado a declarar también como abstracta la clase general.

Declaración de Clase Abstracta.

abstract class Persona {…}

esto implica que todas las clases que heredan de persona, empleado, jefatura, director tengan que sobre escribir el método que heredan, dameDescripción.



Public String dame...{

…}

Se crea un patrón de diseño que se debe de seguir obligatoriamente en tu jerarquía de herencia, porque será diferente el método para cada una de las clases.