Setup

Install IntelliJ

[http://jetbrains.com/es-es/idea/download/#section=windows](http://jetbrains.com/es-es/idea/download/" \l "section=windows)

Java

<https://www.oracle.com/java/technologies/downloads/>

Maven

<https://www.javiergarzas.com/2014/06/maven-en-10-min.html>

<https://maven.apache.org/download.cgi>

<http://codigoelectronica.com/blog/instalar-maven-en-windows>

Links

<https://www.youtube.com/c/AutomationStepByStep>

<https://www.youtube.com/c/MitoCode>

<https://www.youtube.com/c/AbelQuintanaLopez/videos>

<https://testautomationu.applitools.com/>

<https://medium.com/slalom-build/quality-engineer-learning-roadmap-fddfcb77409e>

<https://www.guru99.com/>

<https://automationstepbystep.com/>

Java

Definición de una clase Java

.java – Java source code file

.class – Archivo de compilación > usando el .java el compilador de Java genera (código compilado de JVM) y lo almacena en el .class

Diagram

Description automatically generated

Java class components

* Paquete
  + Toda clase es parte de un paquete
  + Debe ser la primera definición en una clase
  + Lo recomendable es que se escriban siempre en minúsculas
* Import
  + Usados para hacer referencias que se encuentran en otro paquete
* Comentarios
  + Pueden aparecer en diferentes partes de una clase
  + El texto agregado dentro de comentarios no será tomado en cuenta por el compilador
  + Hay varios tipos:
    - Multilínea
      * Empiezas y terminan con /\* \*/
      * Pueden contener caracteres especiales
    - Simple
    - Empieza con //
* Declaración de la clase

Timeline

Description automatically generated with medium confidence

* + Define el inicio de una clase
  + Puede ser tan simple como el keyword class y el nombre de la clase
  + Está compuesta por:
    - Access modifier – controla la accesibilidad a la clase de otras clases
      * Public

Diagram

Description automatically generated with medium confidence

* + - * Protected

Chart, line chart

Description automatically generated

* + - * Private

Diagram, table

Description automatically generated

* + - * Default – cuando no se agrega ninguno de los anteriores

Diagram

Description automatically generated with low confidence

* + - Nonaccess modifiers
      * abstract
      * static
      * final
      * synchronized - puede ser accesado por múltiples hilos
      * native – llamadas a métodos creados en otros lenguajes
      * strictfp – usado en clases, interfaces y métodos – señala que un calculo de punto flotante es idéntico en diferentes plataformas
      * transient
      * volatile – puede ser modificada por múltiples hilos
    - Nombre
    - Cuerpo - todo lo incluido dentro de los {}
  + Se recomienda que el nombre de la clase inicie con mayúscula

Table

Description automatically generated

* + Variables
    - Son las características de un objeto
    - Almacenan el estado de un objeto
    - Puede ser
      * Variable de clase – accesibles en cualquier lugar de la clase
      * Variable de método – solo disponibles en el método que se crean
    - Se recomienda que tengan nombres característicos
    - Se recomienda que sean en minúsculas
  + Constructor
    - Es usado para crear o inicializar los objetos de una clase
    - Pueden existir diferentes constructores en una clase, siempre que reciba diferentes parámetros
  + Métodos
    - Definen funcionalidades de la clase
    - Puede devolver o no valores
    - Son ejecutados al ser invocados en la clase
    - El método mas conocido es el método main:
      * Es requerido para ejecutar una clase Java
      * Debe ser público y estático
      * El nombre debe ser main
      * Debe ser void y recibir un arreglo de String

Text, letter

Description automatically generated

Variables

Elemento para guardar y/o manipular un valor

Definicion de una variable

Tipo de dato + Nombre + Valor

Declaracion: proceso en que se "reserva" un espacio en memoria para guardar un valor posteriormente. Eje: int edad;

Asignacion: asignacion de un valor a la variable. Eje: edad=20;

Tipos de datos / primitivos

Primitivos – Representan valores ásicos, números reales, enteros, caracteres y booleanos

* Enteros
  + Enteros positivos o negativos
  + Byte
  + Int
  + Short
  + Long
  + Float 32 bits
  + Double 64 bits
* Caracteres
  + Char
* Boolean
  + Valores logicos

Table

Description automatically generated

Diagram

Description automatically generated

No primitivos – son objetos

* String
* Wrapper class

Table

Description automatically generated

Operadores

* +, -, \*, /
* %
* ++numero, numero++, --numero, numero-- (Si el operador esta antes primero se realiza la operación y luego se asigna el valor)
* <, >, <=, >=
* ==, !=
* =, +=, -=, \*=, /=, %=
* !
* &&
* ||

## Estructuras de control

Se pueden clasificar en:

1. Selección: permite decidir si se ejecuta un código o no
2. Repetición: permite ejecutar un bloque de sentencias muchas veces
3. Salto: permiten dar un salto o continuar con la ejecución de un programa un punto dado

Se pueden combinar sin limitaciones. Cuando se incluyen simultáneamente, se dice que son estructuras de control aninadas.

estructuras de selección

**IF**

* Se utiliza para tomar decisiones sobre un valor preexistente, si el valor se cumple el programa tomará un camino, si no, tomará otro
* Se denomina selección única
* Evalúa un valor lógico (true o false)

**IF Else**

* Se denomina selección doble porque selecciona entre dos bloques de sentencias mutuamente excluyentes
* Cuando la condición es verdadera se ejecuta el bloque if, de lo contrario se ejecuta el bloque else

[If, If..else Instrucción de Java con ejemplos - Todo sobre JAVA (quejava.com)](https://quejava.com/if-if-else-instruccion-de-java-con-ejemplos/)

**Switch**

* Estructura de selección múltiple
* La selección del bloque de sentencia depende de la evaluación de una expresión
* Para interrumpir la ejecución de las sentencias se usa BREAK
* Algunas reglas importantes para declaraciones switch:
  + Los valores duplicados de los case no están permitidos.
  + El valor para un case debe ser del mismo tipo de datos que la variable en el switch.
  + El valor para un case debe ser una constante o un literal. Las variables no están permitidas.
  + La declaración break se usa dentro del switch para finalizar una secuencia de instrucción.
  + La declaración break es opcional. Si se omite, la ejecución continuará en el siguiente case.
  + La instrucción default es opcional, y debe aparecer al final del switch.

[Sentencia Switch en Java (Ejemplos) - Java desde Cero](https://javadesdecero.es/basico/sentencia-switch-java-ejemplo/)

**Operador condicional**

* Se relaciona con la estructura if-else
* Único operador de Java con tres operandos
  + Primero: condición lógica
  + Segundo: valor en caso de resultado positivo
  + Tercero: valor en caso de resultado negativo

Estructuras de repetición

* Permiten repetir un bloque de código muchas veces o infinitamente
* Se compone de 4 partes:
  + Inicialización: declaración e inicialización de la variable
  + Condición: condición a evaluar
  + Bloque de sentencia
  + Actualización: actualización de la variable de control del bucle

1. While: no se sabe el numero de veces a iterar
   1. la condición se evalúa primero y, si devuelve verdadero, se ejecutan las declaraciones dentro del ciclo while
   2. Cuando la condición se evalúa como falsa, el control sale del ciclo y pasa a la siguiente instrucción después del ciclo while
   3. la instrucción de incremento o decremento dentro del ciclo while para que la variable del ciclo cambie en cada iteración
2. Do-while: se ejecuta al menos una vez, después comprueba condición y ejecuta si se cumple
   1. se ejecutan las sentencias dentro del ciclo y luego se evalúa la condición, si la condición se evalúa como verdadera, el control se pasa a «hacer», de lo contrario, salta a la siguiente sentencia después de do-while.
   2. Después de la ejecución de los enunciados, y la actualización del valor de la variable, la condición se verifica para el valor verdadero o falso. Si se evalúa como verdadero, comienza la siguiente iteración del ciclo.
3. For: se utiliza cuando se sabe el numero de veces a iterar o cuando se conoce el valor inicial

* Condición de inicialización: Aquí, inicializamos la variable en uso. Marca el inicio de un ciclo for. Se puede usar una variable ya declarada o se puede declarar una variable, solo local para el bucle.
* Condición de prueba: se usa para probar la condición de salida de un bucle. Debe devolver un valor booleano. También es un bucle de control de entrada cuando se verifica la condición antes de la ejecución de las instrucciones de bucle.
* Ejecución de instrucción: una vez que la condición se evalúa como verdadera, se ejecutan las instrucciones en el cuerpo del bucle.
* Incremento/Decremento: se usa para actualizar la variable para la siguiente iteración.
* Terminación de bucle: cuando la condición se vuelve falsa, el bucle termina marcando el final de su ciclo de vida.

1. Foereach

[Blucles en Java: for, while, do while Con Ejemplos - Java desde Cero](https://javadesdecero.es/basico/bucles-for-while-do-while-ejemplos/#3_Bucle_do_while)

Estructuras de Salto

* Break: interrumpe ejecución
* Continue: solo en estructura de repetición

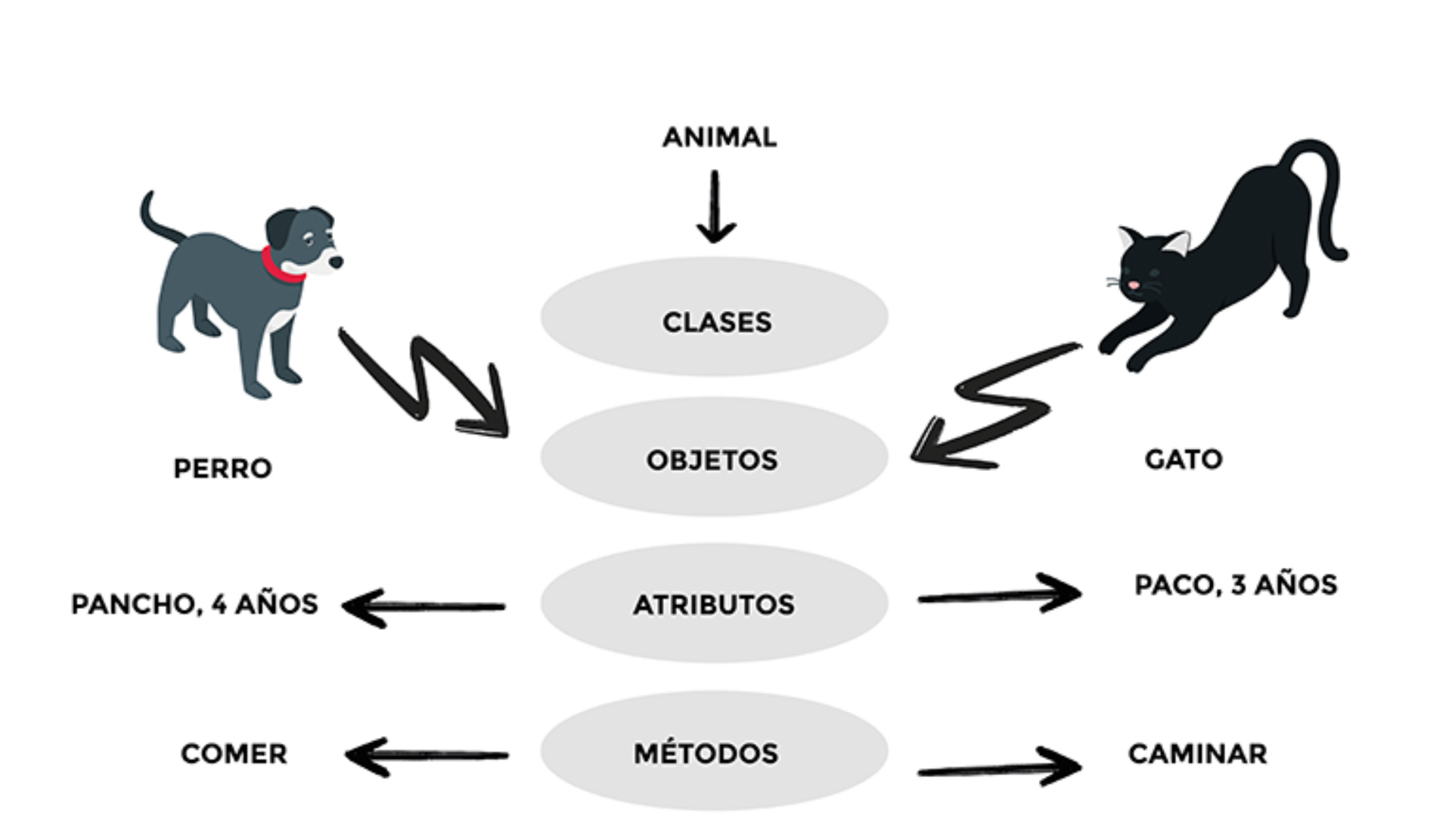
# POO

La Programación Orientada a Objetos (POO) es un [paradigma de programación](https://profile.es/blog/que-son-los-paradigmas-de-programacion/), es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él.

Se basa en el concepto de clases y objetos.

permite que el código sea reutilizable, organizado y fácil de mantener. Sigue el principio de desarrollo de software utilizado por muchos programadores [DRY (Don’t Repeat Yourself)](https://www.linkedin.com/feed/update/urn:li:activity:6724198467129896961) Además, evita el acceso no deseado a los datos o la exposición de código propietario mediante la encapsulación y la abstracción

Una clase es una plantilla. Define de manera genérica cómo van a ser los objetos de un determinado tipo.



<https://desarrolloweb.com/articulos/499.php>

<https://www.youtube.com/watch?v=cPcjiwSN1YU&t=7s>

Herencia

1. Multiple – No soportada en Java
2. Multinivel - una clase puede heredar de una clase derivada. Por lo tanto, la clase derivada se convierte en la clase base para la nueva clase.
3. Jerárquica - una clase es heredada por muchas subclases
4. Híbrida - es una combinación de herencia única y múltiple

Polimorfismo

A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So the same person possesses different behavior in different situations.

[Polymorphism explained simply!. OOP | For beginners | Dynamic vs… | by Shanika Ediriweera | Medium](https://medium.com/@shanikae/polymorphism-explained-simply-7294c8deeef7)

[Overriding in Java - GeeksforGeeks](https://www.geeksforgeeks.org/overriding-in-java/)

[Overloading in Java - GeeksforGeeks](https://www.geeksforgeeks.org/overloading-in-java/?ref=lbp)

Git

Locators

Selenium – testing

RestAssured

Postman