

# Historia de la informática

Wednesday, August 21, 2024 9:15 PM

- Ábaco en mesopotamia y china.
- Blaise Pascal invento la primera calculadora mecanica.
- Charles Babbage invento la Máquina Diferencial y la **Máquina Analítica (preursora de la computadora)**
- Ada Lovelace, fue la primera programadora conocida, ayudo a Babbage
- El término inteligencia artificial se acuño en 1956 en la conferencia de Dartmouth, ahí comenzó su investigación formal.
- John McCarthy (no Occonor) es considerado el padre de la IA.
- En 1990 surgieron las primeras computadoras personales como la Altair 8800, la Apple 1 y el Apple 2.
- En 1960, ARPANET (Advance Research Project Agency Network), una red de comunicación del gobierno de USA sentó las bases de lo que hoy es la internet, usaba el protocolo TCP/IP.
- En 1989 Tim Berners-Lee creó la Wold Wide Web.
- Microsoft fue fundada en 1975 por Bill Gates y Paul Allen.
- Google fue fundada por Larry Page y Sergey Brin en 1998.

# Hardware

Wednesday, August 21, 2024 9:16 PM

## Componentes esenciales

- Procesador (CPU): es el cerebro de la computadora, realiza las operaciones.
- Memoria RAM: Random Access Memory, almacenamiento rápido y temporal.
- Unidad de almacenamiento (DD, SSD): almacenamiento lento y permanente.
- Placa Base, Tarjeta gráfica.

## Servidores

- Están diseñados para proporcionar servicios, recursos o funcionalidades a otros dispositivos o usuarios en una red.
- Están optimizados para tareas de almacenamiento, procesamiento y gestión de grandes cantidades de datos y tráfico de red.
- Pueden ser **servidores web, servidores de bases de datos, servidores de correo electrónico, servidores de archivos, etc.**
- Suelen operar de forma continua y están configurados para ser seguros y confiables.

## Sistemas embebidos

- Son sistemas informáticos diseñados para realizar funciones específicas dentro de dispositivos más grandes o sistemas integrados.
- Tienen recursos limitados en comparación con las computadoras de escritorio o servidores, pero están optimizados para tareas específicas.
- Se utilizan en una amplia gama de aplicaciones, como dispositivos médicos, electrodomésticos inteligentes, sistemas de control industrial, automóviles, etc.
- Suelen tener un diseño compacto, consumo de energía eficiente y pueden operar en condiciones ambientales adversas.

# Software

Wednesday, August 21, 2024 9:17 PM

La parte intangible de un sistema informatico, es el conjunto de programas, instrucciones y datos necesarios para realizar diversas tareas en una computadora.

## Lenguaje de Máquina

Es el conjunto de instrucciones directamente interpretable por el hardware de una computadora. Consiste en una secuencia de códigos binarios, representados por 0 y 1, que le indican al hardware operaciones elementales como sumas, restar, mover datos, entre otros.

Cada procesador tiene su propio conjunto de instrucciones en lenguaje máquina, conocido como conjunto de instrucciones de arquitectura (ISA). Estas instrucciones se ejecutan en la CPU y controlan el flujo de datos y las operaciones realizadas por la computadora.

Aunque es el lenguaje más básico y difícil de entender para los humanos, todos los programas de software eventualmente se traducen a lenguaje de máquina para poder ser ejecutados por el procesador.

## Lenguaje de Ensamblador

- Tambien conocido como Assembly.
- Proporciona una representación simbólica de las instrucciones de máquina.
- Es mas legible que el lenguaje máquina.
- Es como un traductor entre lenguaje máquina y lenguaje humano
- Cada instrucción en lenguaje ensamblador representa una instrucción del lenguaje máquina correspondiente.
- Ofrece mayor control sobre el hardware y la optimización de programas para tareas específicas.

## Sistemas operativos

- Corazón de cualquier dispositivo informático.
- Intermediario entre Hardware y software.
- Provee una interfaz usuario-máquina, y servicios como gestión de archivos, seguridad y multitarea.

# Servidores

Wednesday, August 21, 2024 9:17 PM

- Actuan como el corazón de las redes informaticas, proporcionando servicios, almacenamiento y recursos a otras computadoras en la red.
- Un servidor es una computadora dedicada a proporcionar servicios, recursos y datos a otras computadoras en una red (cliente).
- Diseñados para administrar multiples solicitudes de manera simultanea.

## Servidores Web

- Alojan y entregan sitios web y aplicaciones web a través de internet.
- Utilizan protocolos como HTTP y HTTPS para comunicarse con navegadores web.

## Servidores de correo electronico

- Gestionan la entrega y recepcion de emails.
- Utilizan protocolos como SMTP, POP3 e IMAP para enviar, recibir y almacenar emails.

**Servidores de Base de Datos:** Almacenan y gestionan datos estructurados, proporcionan acceso concurrente a multiples usuarios y apps.

**Servidores de archivos:** almacenan y comparten archivos en una red. Permiten a los usuarios acceder y compartir archivos de forma centralizada.

**Servidores de aplicaciones:** Los servidores de aplicaciones ejecutan y gestionan aplicaciones empresarialess y servicios en una red. Proporcionan entornos de ejecucion para aplicaciones web y empresariales, facilitando el desarrollo, implementacion y administracion de software.

Los servidores desempeñan funciones en una red como el almacenamiento y distribucion de archivos, la gestion de bases de datos, la provision de servicios de email y hospedaje de sitios web. Tambien pueden proporcionar seguridad, control de acceso, monitoreo y administracion de la red.

CLI: Command Line Interface

GUI: Grafic User Interface (iconos, ventanas, menus)

NUI: Native User Interface (gestos, voz y otros)

# Terminal (CLI)

Wednesday, August 21, 2024 9:18 PM

## Consola, terminal y CLI

**Consola:** Es el conjunto de teclado y pantalla, fisico o virtual (ventana de consola)

**Terminal:** Es el software o la aplicación que proporciona una interfaz de linea de comandos para interactuar con el OS. En sistemas Linux, la terminal es la principal forma de acceder a la CLI.

**CLI:** Interfaz que permite al usuario interactuar con un programa or sistema, emitiendo comandos de texto simples. La CLI se utiliza a menudo a travez de la terminal.

Ventajas de la terminal

- Eficiencia
- Automatización
- Acceso Remoto
- Potencia y flexibilidad

## Bash

- directorio-interno/ ---> Esto es un directorio, porque tiene el / al final
- **mkdir:** para crear directorio
- **touch:** para crear un archivo
- **rm:** para borrar un archivo
- **mv:** mover archivo -> Ejemplo: **mv mi.txt directorio-interno/**
- **cp:** copiar archivo
- **pwd:** path absoluto de doonde estoy
- **ls | grep "no":** (grep: patron) lista todos los archivos que incluyen "no"
- **ls > listado.txt:** crea un archivo de texto con el nombre de todos los archivos que listaria el comando ls.
- **cat archivo.txt:** permite leer el texto que hay dentro del archivo.
- **cmod:** cambia los permisos de acceso de archivos directorios
- **chown:** cambia al propietarios y al grupo de archivos y subdirectorios.
- **find:** busca archivos y directorios

## Bash VS Linux

- mkdir - mkdir
- clear - cls
- touch archivo.txt - New-Item archivo.txt -ItemType File
- rm archivo.txt - del archivo.txt
- rm folder - rmdir folder
- ls - dir
- \_ - ren renombrar.txt renombrado.txt
- cp - copy original.txt copia.txt

# Git

Wednesday, August 21, 2024 9:19 PM

Git es el VCS (version control manager) más conocido.

## Configuración inicial

git config --global user.name

git config --global user.email

Las credenciales serán usadas de manera global, en el caso de trabajar con diferentes empresas, debes ingresar las credenciales correspondientes.

## Estados de archivos

1. Untracked: Existen en el directorio de trabajo pero no han sido agregados al staging área. Git no rastrea los cambios en estos archivos.
2. Unmodified: Archivos que no han sido modificados desde el ultimo commit.
3. Modified: Archivos que se han modificado pero aún no se agregan al área de staging.
4. Staged: Archivos añadidos al área de preparación (staging) mediante el comando git add.

## Comandos Básicos

1. # git init: inicia git
2. # git add: agrega repositorios al stage
3. # git commit: confirma los cambios realizados en el repositorio creando un nuevo punto en la historia del proyecto
4. # git config: se utiliza para configurar variables de entorno específicas de Git como usuario y email. Es útil para personalizar la configuración de git en un proyecto específico.
5. # git rm --cached: elimina el archivo del staging.
6. # git restore: elimina los cambios realizados restaurando el archivo a la versión del último commit.
7. # git log --oneline
8. # git status -s
9. # git diff: funciona para ver las diferencias entre archivos que aún no están en el stage.

## Ignorar y borrar archivos del repositorio

- # git ignore: ignora los archivos que no deseamos incluir en el repositorio.
- # git rm archivo.txt: elimina archivos que han sido agregados al repositorio, pero que no queremos incluir anymore.

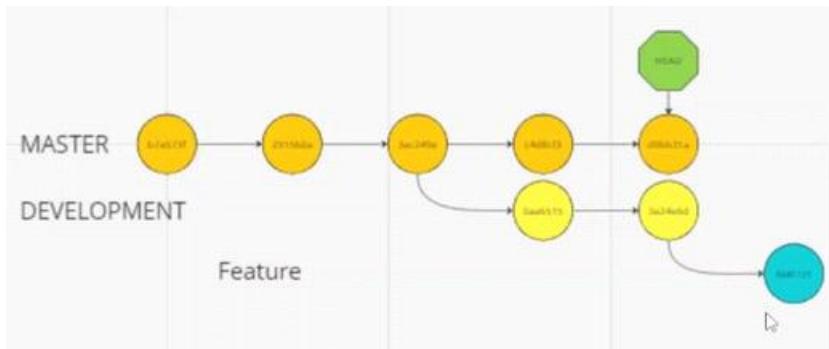
## Otros comandos

- # git config --global core.editor "code --wait"
- # git config --global core.autocrlf
  - Input (mac/linux): configuración para que los saltos de línea sean como linux
  - true (windows): configuración para que los saltos de línea sean como windows
  - false (windows): configuración para que los saltos de línea sean como linux (recomendado)

## Ramas Git

- Rama principal (master/main)
- Ramas de características (feature branches)
- Ramas de corrección (bugfix branches)
- Ramas de versión (release branches)
- Ramas de desarrollo (development branches)

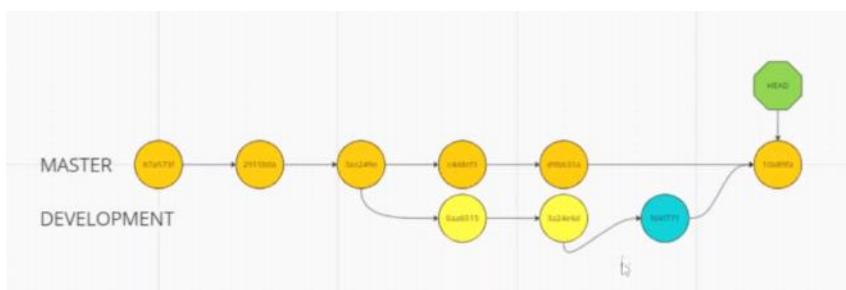
# git checkout -b development -> Comando para cambiar de rama y crearla en caso que no exista.



# git switch development -> Comando para cambiar de rama, se diferencia de git checkout en que git switch es exclusivo para ramas, git checkout permite también cambiar entre commits (puntos de ramas)

#git merge feature/agregar-file10 -> Utilizamos git merge desde la rama que queremos conservar, en el ejemplo anterior desde la rama Development

#git branch -> Comando para listar las ramas existentes.



# GitHub

Wednesday, August 21, 2024 10:26 PM

Git es el software de control de versiones, GitHub es la plataforma en la nube que permite alojar repositorios git.

En otras palabras, GitHub es un servicio de alojamiento para proyectos que utilizan Git como sistema de control de versiones.

# git push -u origin master -> En este comando, origin hace referencia al remote seteado y master (o main) a la rama local.

# git push -u origin rama2 -> Con este comando subimos a GitHub la nueva rama2 creada en nuestro repositorio local.

# git fetch lee la información que hay en el repositorio remoto. No hace ningún cambio, simplemente lee si la rama remota esta por delante de la rama local.

```
● PS C:\Users\sergi\Desktop\proyecto-git-digitalhouse> git log --oneline
  22b0539 (HEAD -> master, origin/master, origin/HEAD) Se agrega file2
  2c600eb Se agrega texto en file1
  951c907 Se agrega el primer archivo
● PS C:\Users\sergi\Desktop\proyecto-git-digitalhouse> git branch
* master
  rama2
● PS C:\Users\sergi\Desktop\proyecto-git-digitalhouse> git fetch
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 986 bytes | 89.00 KiB/s, done.
From https://github.com/digitalhouse-content/proyecto-git-digitalhouse
  22b0539..8958350 master -> origin/master
● PS C:\Users\sergi\Desktop\proyecto-git-digitalhouse> git log --oneline
  22b0539 (HEAD -> master) Se agrega file2
  2c600eb Se agrega texto en file1
  951c907 Se agrega el primer archivo
○ PS C:\Users\sergi\Desktop\proyecto-git-digitalhouse> █
```

Git cree que origin (repo remoto) está en el commit  
22b0539

Hacemos Git Fetch

Ahora que hicimos git fetch, trajimos la información  
del repo local, y ahora Git sabe que origin no está  
en el commit 22b0539 (sino un commit adelante).  
Sin embargo Git Fetch no hace cambios y  
permanecemos en el mismo commit.

# git pull trae el commit del repositorio remoto al repositorio local.

En resumen:

- # git fetch: informa en que estado está el repositorio local en comparación con el remoto (si esta actualizado o no)
- # git pull: trae al repositorio local todos los cambios del repositorio remoto
- # git push: envia todos los cambios del repositorio local al repositorio remoto.

Para hacer push de una rama, el head debe estar dentro de la rama.

**En git push origin main, origin es el repositorio remoto y main es la rama local que estamos subiendo al repositorio remoto.**

# git branch -d feature/mejora1

- Este comando elimina la rama **feature/mejora1** del repositorio local.

# git push origin -d feature/mejora1

- Este comando actualiza el repositorio remoto eliminando la rama **feature/mejora1**

# git fetch --prune

- Es necesario usar este comando si eliminamos la rama desde GitHub (ejemplo rama2), de esta forma el repositorio local actualiza la información.

- Despues de esto podemos usar de forma normal el comando # git branch -d rama2.

# Arquitectura de computadoras

Thursday, August 22, 2024 9:08 PM

La arquitectura de computadoras es la estructura interna de un sistema informático. Determina como interactúa hardware y software para procesar datos.

1. CPU
2. Memoria
3. Disp. E/S
4. Comunicación entre los componentes

## Modelo de Von Neumann

Modelo de Bon Neumann, es una descripción conceptual de la arquitectura de las computadoras digitales, que ha servido como base para la mayoría de las computadoras modernas.

Introdujo la idea de almacenar tanto datos como instrucciones en la misma memoria de la computadora, y esto permitió la programabilidad de las computadoras.

1. Unidad de procesamiento (CPU): Ejecuta instrucciones y realiza cálculos.
2. Memoria: Almacena tanto datos como instrucciones, los datos pueden ser interpretados como instrucciones y las instrucciones almacenadas como datos.
3. Program Counter: Es un registro especial de la CPU que indica la dirección de la siguiente instrucción a ser ejecutada.
4. Bus de datos y control: Facilitan la transferencia de datos e instrucciones entre la CPU y la memoria, así como entre la CPU y los dispositivos de E/S.

## CPU

Responsable de ejecutar instrucciones de programas almacenados en la memoria.

Consiste en una combinación de hardware y microcircuitos que interpretan y ejecutan instrucciones de programas, realizan cálculos y gestionan los datos.

1. **Unidad Aritmético Lógica (ALU):** Realiza operaciones aritméticas (suma, resta, multiplicación y división) y lógicas (and, or, not) necesarias para **procesar datos**. Opera en datos binarios, manipulando bits y bytes para realizar operaciones matemáticas y lógicas.
2. **Unidad de Control:** Controla y coordina las operaciones de la CPU. Se encarga de interpretar las instrucciones del programa almacenadas en la memoria decodificarlas y ejecutarlas secuencialmente. También controla el flujo de datos entre la CPU y otros componentes del sistema.
3. **Registros:** Son pequeñas áreas de almacenamiento de alta velocidad ubicadas dentro de la CPU. Se utiliza para almacenar datos temporales, direcciones de memoria y otros valores necesarios para ejecutar las instrucciones del programa. Los registros son fundamentales para el buen funcionamiento de la CPU, ya que ofrecen almacenamiento interno de acceso rápido.

## Funcionamiento y coordinación de la CPU

## **1. Ejecución de instrucciones**

Comienza cuando la unidad de control lee una instrucción del programa almacenada en la memoria.

La instrucción se decodifica para determinar la operación que el procesador debe ejecutar.

La unidad de control coordina la ejecución de la instrucción, enviando señales a la ALU y otros componentes según sea necesario.

## **2. Operaciones Aritméticas y lógicas**

La unidad de control envía los operandos desde los registros a la ALU, la ALU realiza la operación, almacena el resultado en un registro designado.

## **3. Acceso a la memoria**

En algunas instrucciones la CPU necesita acceder a los datos en memoria, para esto la Unidad de Control coordina el acceso a la memoria, enviando direcciones de memoria y controlando la transferencia de datos entre la memoria principal y los registros de la CPU.

## **4. Ciclo de instrucción**

El proceso de ejecución de instrucciones se repite continuamente en lo que se conoce como Ciclo de Instrucción. En cada ciclo la CPU

Busca  
Interpreta  
Ejecuta  
Actualiza  
una instrucción del programa.

## **CPU vs GPU**

La CPU realiza muchas tareas y operaciones de forma concurrida, y tiene un número limitado de hilos. Las GPU están especializadas en procesar gráficos para programas de diseño o video Juegos entre otros. En IA se usan GPU en lugar de CPU ya que las GPU procesan miles de hilos al mismo tiempo (para procesar gráficos), y esta característica es muy útil para procesos de machine learning.

# Memoria

Thursday, August 22, 2024 11:41 PM

Es la capacidad de un sistema para almacenar, retener y recuperar información

La memoria puede ser Volátil (como la memoria RAM) o persistente (como la memoria SSD).

## Memoria RAM (Random Access Memory)

Es conocida como la memoria principal, es una memoria de acceso aleatorio, lo que significa que puede acceder a cualquier ubicación de memoria de forma directa y rápida, sin necesidad de pasar ubicaciones adyacentes.

La memoria RAM es volátil, lo que significa que se pierde cuando se apaga la computadora.

La memoria RAM es mucho más rápida que la memoria secundaria, lo que permite un acceso rápido por parte del procesador.

## Jerarquía de memorias

1. Memoria cache del procesador (muy muy rápida, poca poca capacidad)
2. Memoria RAM (rápida pero tiene poca capacidad)
3. Memoria secundaria (es más lenta que la memoria RAM, pero es la que tiene mayor capacidad).

## Memoria secundaria

La memoria secundaria se refiere al almacenamiento de datos en dispositivos de almacenamiento no-volátiles como Discos Duro (HDD) o unidades de estado Sólido (SSD)

Información del sistema: Win + R - msinfo32

# Sistema Operativo

Friday, August 23, 2024 12:06 AM

Es un software que conecta el hardware con los programas de aplicación.

Gestiona los recursos de hardware. Proporciona servicios a los programas de software

- Intermediario (capa de abstracción) entre Hardware y Software
- Gestión de recursos
- Interfaz de usuario
- Ejecución de programas
- Servicios de sistema
- Entorno de desarrollo
- Seguridad y protección
- Portabilidad.

## Operating Systems

1. Windows
2. macOS
3. Linux
4. Android
5. iOS

# Lenguajes y Paradigmas de Programación

Friday, August 23, 2024 12:18 AM

Los lenguajes de programación son conjuntos de instrucciones y reglas utilizados para escribir programas informáticos.

## Paradigmas de programación

Enfoques o estilos para resolver problemas mediante la codificación

**Imperativo:** Se centra en la descripción de los pasos o procedimientos necesarios para alcanzar un resultado.

**Declarativo:** El enfoque declarativo se centra en lo que quiere lograr, y no como lograrlo. Los programas declarativos especifican las relaciones entre diferentes entidades y permiten que el sistema determine la mejor manera de llegar al resultado deseado.

**Orientado a objetos:** Se basa en la representación de entidades como objetos que tienen propiedades (atributos) y comportamientos (métodos).

### Funcional:

- Construcción de programas
- Funciones puras (no tienen efectos secundarios y siempre se obtiene el mismo resultado al invocarlas con los mismos argumentos).
- Aplicación de funciones matemáticas.
- Programación basada en expresiones.

## Lenguajes de Alto Nivel

- Alejados del lenguaje máquina.
- Cercanos al lenguaje humano.
- Utilizan abstracción y estructuras de datos complejas para facilitar su escritura y comprensión.
- Independientes de la arquitectura del hardware
- Puede escribirse en diferentes tipos de computadoras (OS).

## Lenguajes de Bajo nivel

- Cercanos al lenguaje máquina.
- Alejados de lenguaje humano.
- Diseñados para interactuar con el hardware de la computadora
- Permiten control más preciso sobre el hardware de la computadora.
- Mas eficientes en cuanto a recursos.

Alto Nivel	Bajo Nivel
Mayor Portabilidad	Mayor control sobre el hardware
Mayor legibilidad y mtto	Mayor eficiencia en términos de rendimiento
Desarrollo Rápido	-
Python, Java, C#	Ensamblador, C, C++

# Intérpretes, Compiladores y Código Fuente

Friday, August 23, 2024 10:32 PM

## Interpretes

- Lee y ejecuta instrucciones
- Línea por línea
- Convierte código fuente en instrucciones ejecutables.
- Ex. Python, Ruby, JavaScript

## Compiladores

- Convierten código fuente a lenguaje de máquina (binario).
- Se realiza antes de la ejecución
- Ex. C, C++, Rust y Java (a bytecode)

## Código fuente

- Código escrito en cualquier lenguaje de programación, antes de ser transformado a un formato ejecutable (binario, bytecode, etc.)
- Permite colaboración, revisión, depuración y mantenibilidad.

# IDE's

Friday, August 23, 2024 10:58 PM

## Integrated development environment (IDE)

- Apps que proveen herramientas de desarrollo.
- Incluyen editores de texto, compiladores, depuradores, administradores de proyecto, etc.
  - o Eclipse
  - o NetBeans
  - o IntelliJ IDEA

## Máquinas virtuales

- Known as VM
- Simulan un sistema informático completo (hardware, software, etc.)
- Permiten ejecutar múltiples sistemas operativos y aplicaciones en el servidor físico

## Importancia en el despliegue y desarrollo de software en VM's

1. Ambientes aislados
2. Portabilidad
3. Administración de recursos
4. Escalabilidad
5. Respaldo y Recuperación

## Practica máquina virtual

1. Descargar VMWare
2. Crear nueva máquina virtual
3. Proveer el ISO del SO que deseamos instalar
4. Personalizar credenciales del OS (Ubuntu)
5. Asignar recursos de la máquina virtual

# Redes

Saturday, August 24, 2024 7:45 AM

- Interconexión entre dispositivos
- Permiten comunicación
- Intercambio de datos

Las redes utilizan medios como:

- Cables coaxiales (cobre)
- Fibra óptica
- Conexiones inalámbricas (wifi)
- LAN
- WAM
- WIFI

## Componentes de una red

### Switch

Actúan como un centro del conexión para dispositivos dentro de una red local

### Router

Actúa como un puente entre diferentes redes, dirigiendo tráfico de manera eficiente

### Repetidores

Reciben la señal de la red, la amplifican y la retransmiten

### Modem

Son el enlace entre los dispositivos que usan la red y la internet. Convierten señales analógicas en digitales para transmitir datos y viceversa.

## Clasificación de redes

### Por tamaño

1. LAN: red de área local, limitada a una geografía pequeña como casa, oficina o campus.
2. MAN: red de área metropolitana, abarca una geografía más grande como una ciudad.
3. WAN: red de área amplia, extiende a una geografía amplia, como un país o continente.

### Por topología

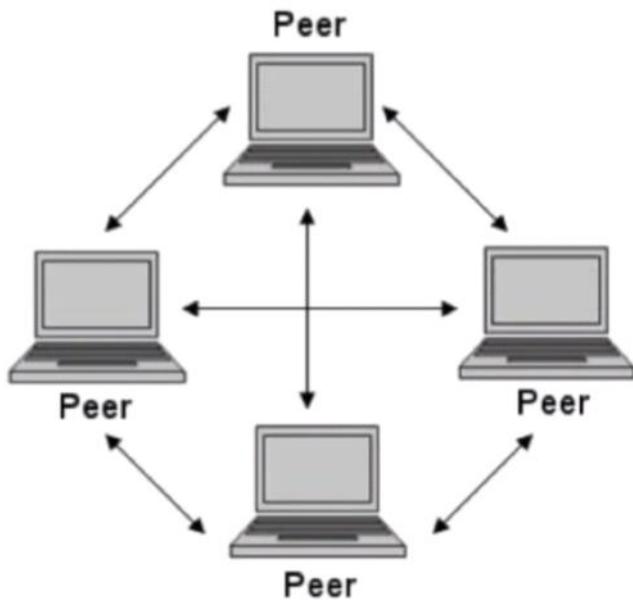
1. Estrella: Todos los dispositivos conectados a un nodo central (como un switch o un router)
2. Bus: Todos los dispositivos comparten un solo canal de comunicación
3. Anillo: Cada dispositivo se conecta a otros dos dispositivos, formando un anillo cerrado de conexiones.

### Por método de acceso

1. Ethernet: protocolo de acceso múltiple | detección de colisiones (CSMA/CD)
2. Token Ring: Evita colisiones utilizando un Token para controlar el acceso
3. Wifi: Utiliza estándar IEEE 802.11 para redes inalámbricas.

### Modelo Peer to peer (P2P)

En esta red cada dispositivo actúa como cliente y servidor



### Modelo Cliente-Servidor

- Los dispositivos se dividen en Cliente / Servidor
- El cliente solicita servicios o recursos
- El Servidor proporcionan dichos servicios y recursos

Características:

- Centralización
- Especialización de roles
- Seguridad
- Escalabilidad

### Medios de transmisión de datos

- Conductos físicos o inalámbricos que transportan físicamente los datos
- Cableado: Coaxial, trenzado, fibra óptica
- Inalámbrico: Bluetooth, Wifi, 4G, 5G
- La frecuencia de Wifi 2.4GHz tiene mayor alcance / penetración que la red 5GHz

# Protocolos de Internet

Saturday, August 24, 2024 9:40 AM

Reglas que permiten la comunicación dentro de una red informática  
Se dividen en capas:

- **Capa de Aplicación (Application):** Se encuentran los protocolos que permiten a las aplicaciones comunicarse entre sí.
  - o HTTP - Web
  - o SMTP - Correo electrónico
  - o FTP - Transferencia de Archivos
  - o DNS - Resolución de nombres de dominio
- **Capa de Transporte (Transport):** Se encuentran protocolos especializados en la transmisión de los datos.
  - o TCP - Transmition Control Protocol
    - Proporciona una conexión orientada a la fiabilidad (paquetes). Utilizado para transferencia de datos almacenables.
  - o UDP - User Datagram Protocol
    - Es más ligero y veloz. Usado para la transmisión de video en tiempo real.
- **Capa de Red (Network):** Se encuentra el protocolo de internet
  - o IP - Internet Protocol - Asigna una dirección única a cada dispositivo en la red, y determina como se deben enviar datos de un dispositivo a otro.
- **Capa de Enlace de Datos (Data Link):** define protocolos para transferencia confiable de datos entre dispositivos directamente conectados
  - o Ethernet
  - o Wifi
- **Capa Física (Physical Layer):** Es la capa más baja, se encarga de la transferencia física de bits a través de medios de comunicación
  - o Cable Coaxial
  - o Fibra Óptica
  - o Señales de Radio

## Direcciones MAC

- MAC (Media Access Control)
- Identificador **único** asignado a cada dispositivo de red
- Grabado en el hardware
- 48 bits (6 Bytes)
- Representación hexadecimal (00:1A:2B:3C:4D:5E)
- Identifica de manera única un dispositivo dentro de una red local
- Utilizado en la Data Link

## Direcciones IPv4

- 32 bits (4 Bytes)
- Representación decimal (192:168:0:1)
- Ha alcanzado su límite, por lo cual se usa la NAT
- NAT: Network Address Translation

### **Direcciones IP Fijas:**

- Asignadas manualmente
- Permanecen constantes
- Útiles para servidores

### **Direcciones IP Dinámicas:**

- Asignadas automáticamente
- Servidor DHCP (Protocolo de configuración dinámica de Host)
- Temporales, pueden cambiar cada vez que el dispositivo se conecta a la red.

### **Mascara de Subred**

- Número binario
- Divide una red IP en partes más pequeñas
- Define que parte de la dirección IP pertenece a la red y que parte pertenece a los host dentro de esa red.
- Ejemplo **255.255.255.0** (en negrita la red, en light el host)
  - Dirección IP: 192.168.1.100
  - Mascara de red: 255.255.255.0
  - La subred sería: 192.168.1.0

### **Clases de direcciones IP**

- A, B, C, D y E
- D y E son direcciones IP reservadas para otros propósitos.
- Cada clase de dirección IP tiene máscaras de subred predeterminadas:
  - Clase A: 255.0.0.0
  - Clase B: 255.255.0.0
  - Clase C: 255.255.255.0

### **Enrutamiento**

Proceso de seleccionar el mejor camino para que los datos viajen desde origen al destino.

#### **Componentes:**

- Router
- Tabla de enrutamiento
- Protocolos de enrutamiento (algoritmos) RIP, OSPF, BGP

#### **Tipos:**

- Enrutamiento estático
- Enrutamiento dinámico

#### **Estrategias:**

- Vector de distancia
- Estado de enlace

## **Modelo OSI (Open System Interconnection)**

Describe funciones de una red con el modelo de 7 capas lógicas:

### **Physical Layer**

Define medios físicos y eléctricos para transmitir datos entre dispositivos

Dispositivos: Cables, conectores, concentradores

### **Data Link Layer**

Controla acceso al medio, proporciona detección y corrección de errores en la capa física  
Dispositivos: Switches, Bridges

### **Network Layer**

Se encarga del enrutamiento y conmutación de datos  
Dispositivos: Routers, Gateways

### **Transport Layer**

Proporciona servicios de extremo a extremo, para el transporte de datos  
Segmentación y control de flujo  
Protocolos: TCP, UDP

### **Session Layer**

Establece, mantiene y termina las conexiones entre aplicaciones en diferentes dispositivos.  
Controla puertos y sesiones

### **Presentation Layer**

Asegura que los datos estén en un formato usable y es donde comienza la encriptación

### **Application Layer**

Es la capa de interacción humano-máquina, donde las aplicaciones pueden acceder a los servicios de red

- Las capas interactúan entre si mediante interfaces
- Ventajas:
  - o Modularidad
  - o Estandarización
  - o Resolución de problemas

### **Proxy**

- Servidor intermediario Cliente-Servidor
- Mejora la eficiencia, ya que cachea contenido frecuentemente solicitado
- Ayuda a bloquear tráfico no deseado
- Ayuda a acceder a recursos bloqueados por ubicación geográfica

### **Proxy Web**

- Intercepta tráfico web
- Filtra contenido
- Realiza cache de páginas web

### **Proxy de Reenvío**

- Envía solicitudes en nombre del cliente
- Renvía respuestas a los clientes

### **Proxy Inverso**

- Sirve como punto de entrada para los clientes
- Distribuye solicitudes a servidores backend

# Proxy

Saturday, August 24, 2024 2:27 PM

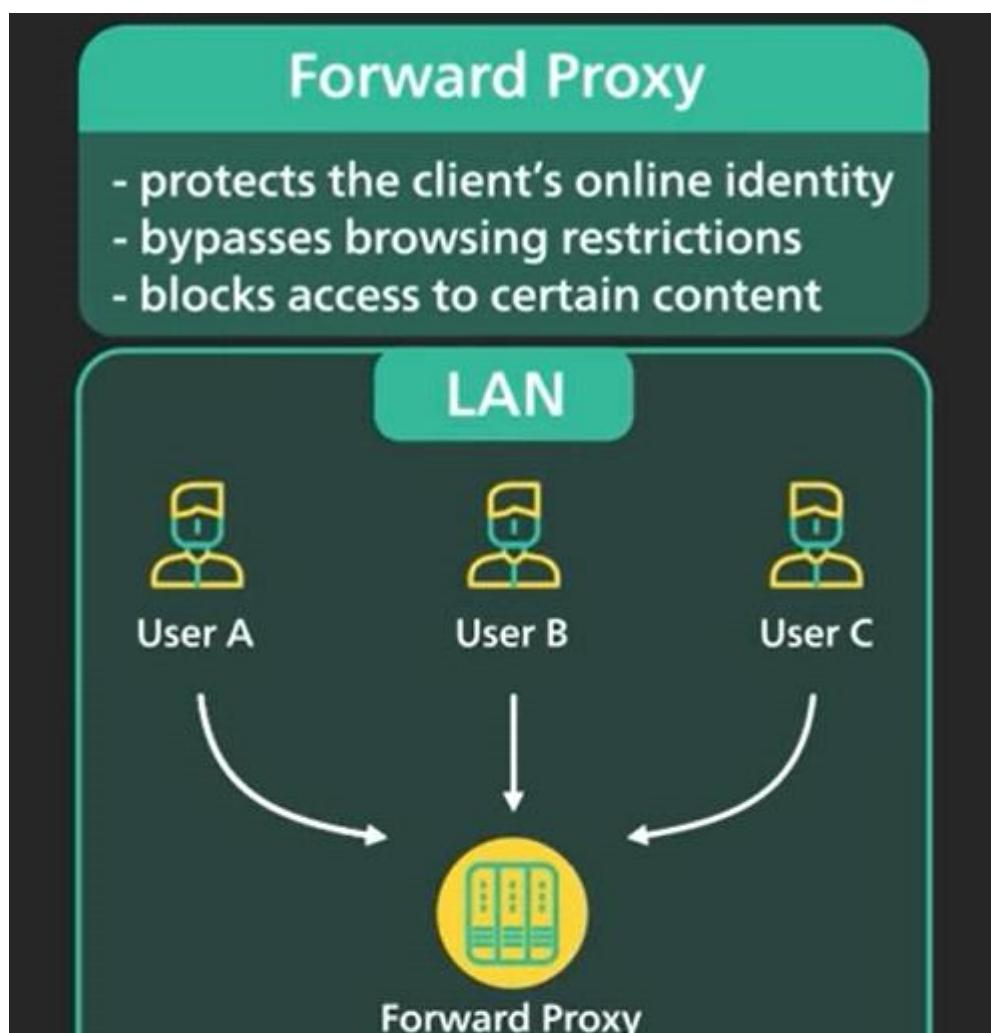
## Funciones principales

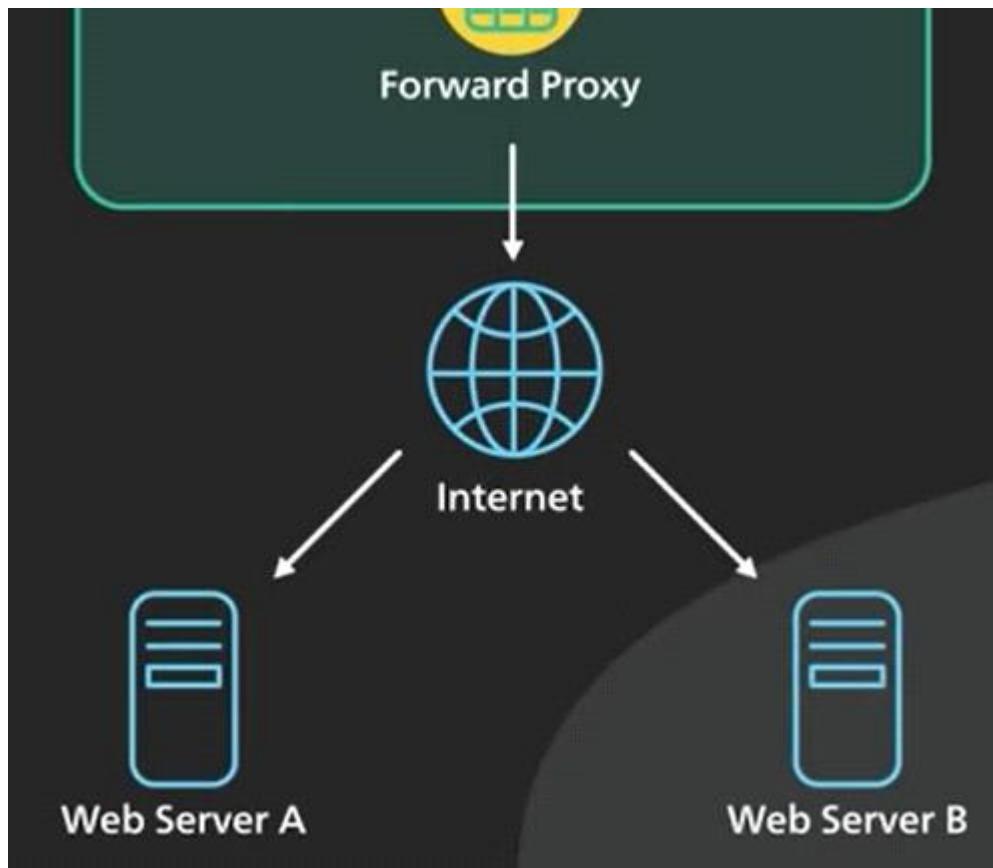
1. Filtrado de contenidos: restricción de acceso a servicios/contenidos
2. Control de accesos: bloquear accesos a las personas
3. Caché web: muy utilizado por proveedores de internet, aumenta el rendimiento
4. Eliminar restricciones geográficas: permiten cambiar la IP para acceder.

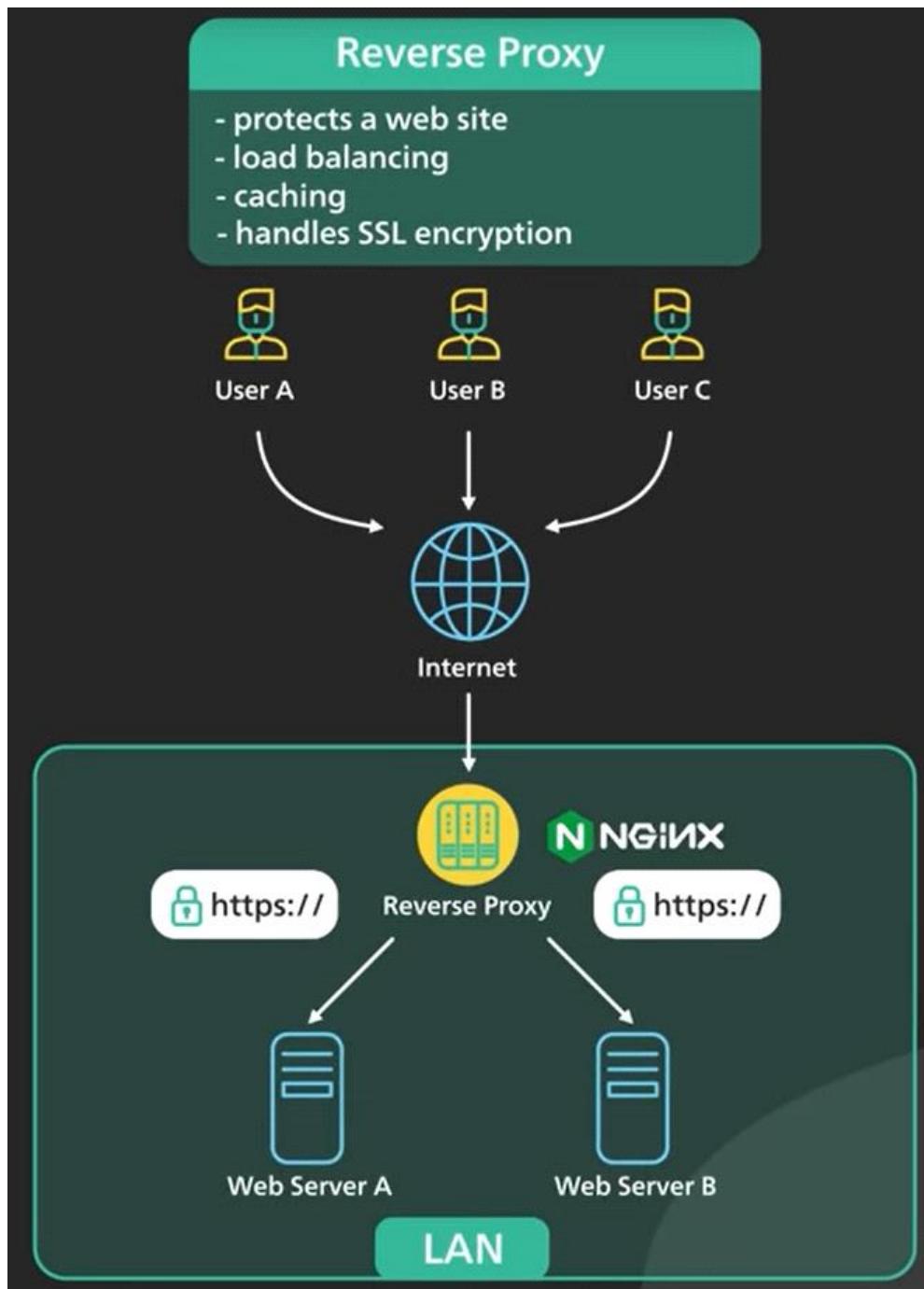
## Proxy inverso

## Funciones principales

1. Anonimato por parte de los servidores backend.
2. Sistemas de protección adicionales.
3. Cifrado (certificado SSL/TLS) entre el proxy inverso y el navegador del cliente, le quita esta carga a los servidores destino de backend.
4. Balanceo de carga: reparte las peticiones entre los diferentes servidores backend que están repartidos por los diferentes data centers.
5. Caché: almacena internamente contenido solicitado frecuentemente.







## VPN (Virtual Private Network)

- Permite enviar y recibir datos a través de una red pública como si fuera una red privada.
- Proporciona seguridad y privacidad
- Cifrado de tráfico de red
- Oculta la IP
- Acceso seguro a redes privadas desde redes públicas

### VPN de acceso remoto

Permite a dispositivos remotos conectarse a una red privada.

**VPN de sitio a sitio**

Conecta de manera segura dos redes privadas a través de internet

**VPN de acceso de usuario.**

Permite acceder a recursos de red privados mediante login, independiente de la ubicación.

## Proceso VPN

1. Inicio de sesión en la VPN
2. Establecer conexión
3. Transmitir datos
4. Salida a internet

# Introducción a la programación orientada a objetos en JAVA

Saturday, August 24, 2024 5:53 PM

## Datos primitivos:

- En Java los datos primitivos tienen una contraparte Objeto, ejemplo:

Primitivo	Objeto
int	Integer
double	Double

## Firma de un Método

- En una función estática como main, solo puedo usar funciones estáticas. Ejemplo:

```
public class Main {  
  
    public static void main(String[] args) {  
  
        addTwoNumbers( numberOne: 2.0, numberTwo: 5.5 )  
    }  
  
    1 usage  
    public static Double addTwoNumbers(Double numberOne,Double numberTwo){  
  
        return numberOne + numberTwo;  
    }  
}
```

## Scanner

```
Scanner myScanner = new Scanner(System.in);  
  
Integer miEntero = myScanner.nextInt();
```

## Objeto

En OOP un objeto se define principalmente por su **responsabilidad** dentro del contexto.

Por ejemplo:

- La responsabilidad de una silla en contexto general es permitir sentarse.
- La responsabilidad de una silla en el contexto de un videojuego puede ser usarla como arma, con puntos de daño, etc.

La responsabilidad de un usuario:

- Caemos en la trampa de decir que la responsabilidad de un usuario es hacer login, registrarse, etc.
- En la realidad, la **responsabilidad** de un usuario depende del contexto, por ejemplo:
  - o Instagram: dar clic, enviar mensajes, compartir fotos, etc.
  - o TESO: atacar, equipar arma, usar poción, etc.
  - o InDriver: Pedir servicio, agregar medio de pago, enviar mensajes al conductor.

Los **atributos** están ahí para ayudarle al objeto a cumplir sus funciones/responsabilidad.

- Uber: el atributo tarjetas ayuda a cumplir la responsabilidad de pagar.
- Facebook: necesita atributos diferentes, las tarjetas no serían atributos necesarios.

"Un **objeto** es un molde de la realidad y se define por su responsabilidad dentro de la aplicación,"

"La **responsabilidad** de un objeto se define por su contexto"

"Los **atributos** de un objeto ayudan a cumplir su función o responsabilidad"

### Objeto vs. Instancia

- Un objeto es un molde de la realidad en el contexto que nosotros estamos intentando programar.
- La instancia es una unidad del objeto, por un solo objeto vamos a tener muchas instancias.

### Variables y métodos de clase

Son métodos y variables que no se le pueden atribuir a una instancia, sino que se le atribuyen a toda la clase. Por ejemplo:

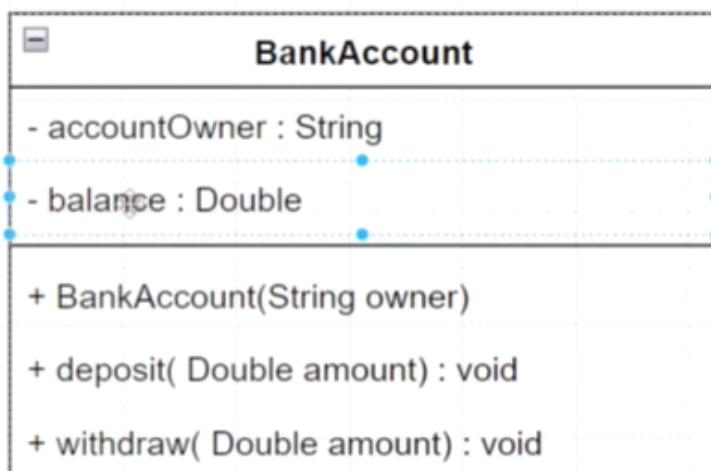
```
Integer miNum = 9;
```

miNum.parseInt() **X X X** <--- parseInt es un método de clase, por lo cual no funciona con instancias.

Integer miNewNum = Integer.parseInt("5") **✓ ✓ ✓** <--- método de clase en acción

Con la palabra **static** podemos construir estos métodos y atributos de clase.

## Clase JAVA



### Set

```
public void setName(String name){  
    this.name = name;  
}
```

### Get

```
public String getName(){  
    return this.name;  
}
```

Creando un breakpoint en el último bracket () de un bloque podemos ver el estado del programa en ese punto, antes de que el bloque de código termine.

# Relaciones entre clases Java

Tuesday, August 27, 2024 6:18 PM

## Relación de asociación

Es una relación entre dos clases que permite que una clase utilice la funcionalidad de otra sin necesidad de heredar de ella.

En teoría es la relación de "has-a" o "tiene-un".

Es más entendible ver la relación de asociación como **usa** en lugar de **tiene**.

- Cuanto tiempo yo conozco al objeto?
- Cuando tienes (conoces) un objeto, lo tienes como atributo.
- Cuando usas un objeto, lo recibes para cumplir con la responsabilidad ( se recibe como parámetro de un método).

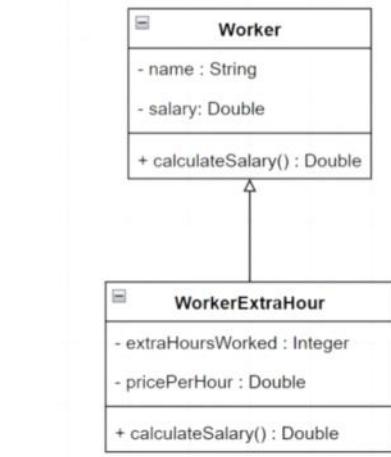


## Relación de herencia

Es una relación entre clases donde una clase hereda atributos y métodos de otra clase.

En teoría es la relación de "is-a" o "es-un"

Mediante herencia, una clase puede heredar atributos y responsabilidades de la clase padre.



```
public class WorkerExtraHour extends Worker {

    public WorkerExtraHour(
        String name,
        Double salary,
        Integer hours,
        Double pricePerHour)
    {
        super(name,salary);
        this.extraHoursWorked=hours;
        this.pricePerHour=pricePerHour;
    }
}
```

## Polimorfismo

Es la capacidad de un objeto de reaccionar de manera diferente hacia el mismo mensaje

Dog dog = new Animal() X ↗ X ↘ X

Animal animal = new Dog()

animal.hacerSonido() -> hará lo que diga el método en la clase Dog()

## Casteo

Donde hay un parente puede ir cualquiera de sus hijos, pero donde hay un hijo no puede ir su parente.

Para hacer un casteo las clases deben ser de la misma familia.

## Clase Object

Todos los objetos de Java derivan de la clase Object.

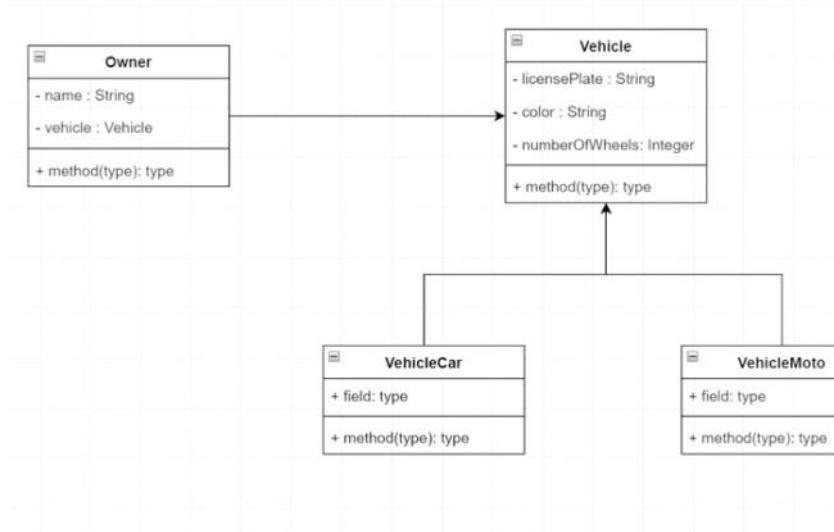
Tiene 3 métodos:

1. `ToString()`: editable para definir que contiene la versión String del objeto.
2. `Equals()`: editable para definir que hace que un objeto sea equivalente a otro.
3. `HashCode()`: editable para definir un código único que identifica al objeto.

## Herencia con Asociación

Donde hay un parente, debe poder entrar cualquiera de sus hijos. (**Principio de Liskov - LSP**)

- Utilizar el nombre de la clase parente antes del nombre de la clase hija, es considerado una buena práctica. Ex: `VehicleCar`

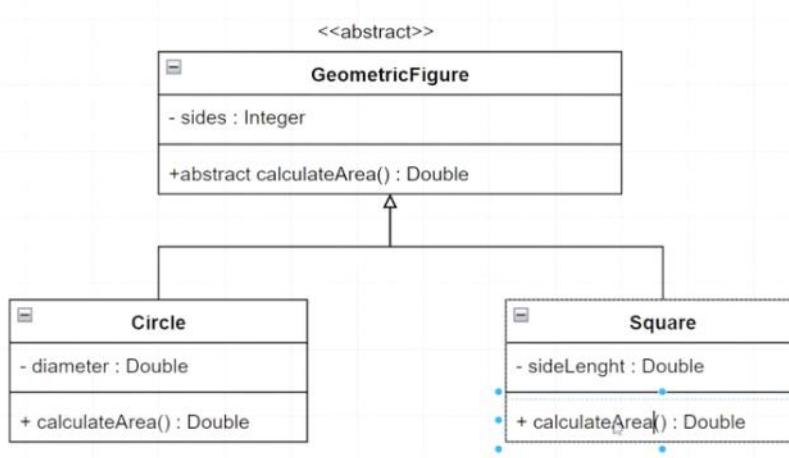


## Clases Abstractas

No pueden ser instanciadas.

Los métodos abstractos se declaran sin cuerpo, ejemplo:

```
public Double calculateArea();
```



## Sobrecarga vs. sobreescritura

La sobrecarga no necesita implementar ninguna anotación, ya que la firma del método cambia.  
La sobreescritura necesita la anotación `@Override`

Wow esto es muy Abstracto, si uso el método

```

@Override
public void withdraw(Double amount) {
    super.withdraw(amount);
}

```

Pero al mismo tiempo, el método `super.withdraw(amount)` usa el método `canWithdraw(amount)` como aquí:

```

public void withdraw(Double amount) {
    if(canWithdraw(amount))
        this.balance -= amount;
}

```

El método `canWithdraw(amount)` va a utilizar la implementación sobreescrita (si la hay) de la clase hija, por ejemplo, en este caso en este caso la clase hija sobrescribe este método como:

```

@Override
public Boolean canWithdraw(Double amount) {
    return amount <= getBalance() + overdraftAmount;
}

```

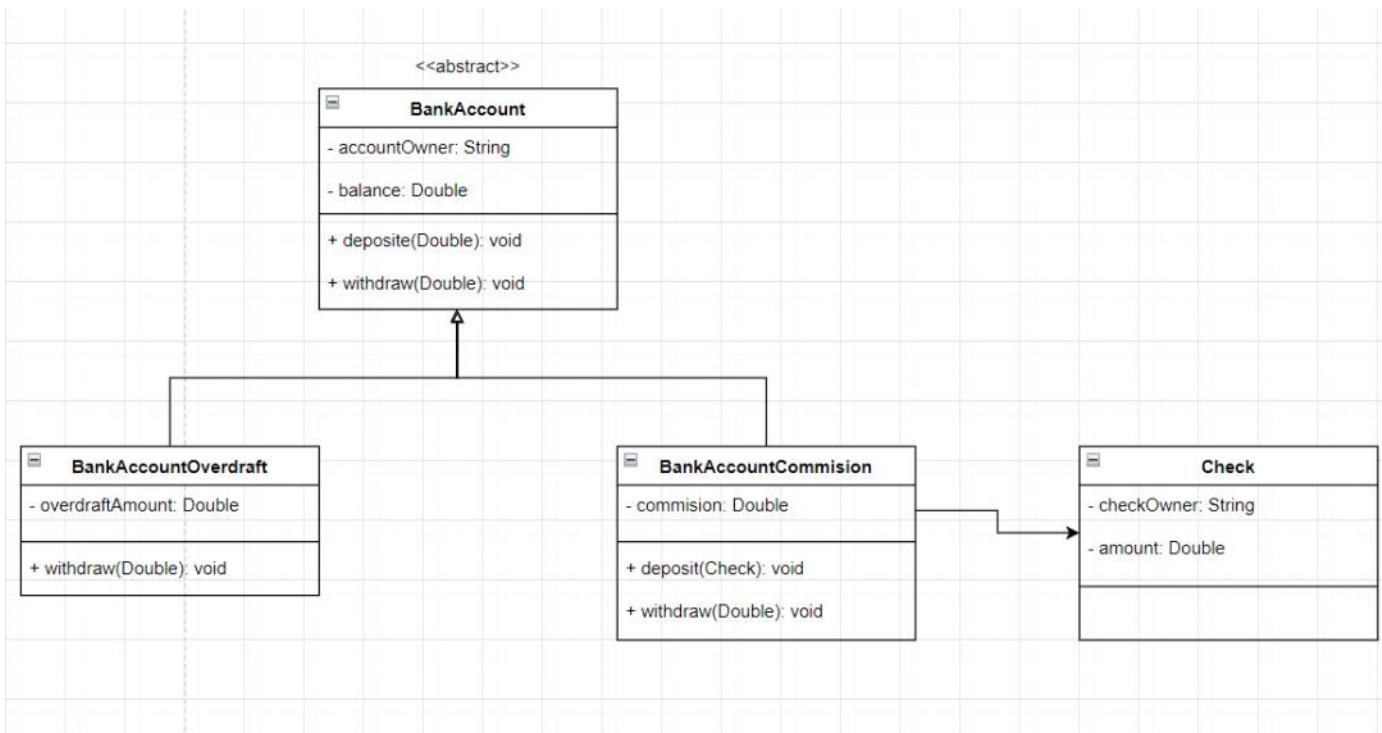
Para más contexto, revisar el proyecto [sobreescritura\\_transaccionesBanco](#).

Clase padre:

```
© BankAccount.java × © BankAccountOverdraft.java © BankAccountCommision.java © Main.java © Check.java  
1  public abstract class BankAccount { 2 usages 2 inheritors  
7      this.balance = balance;  
8  }  
9  
10     public void deposit(Double amount){ 1 usage  
11         this.balance += amount;  
12     }  
13  
14 ⚡     public void withdraw(Double amount){ 2 usages 2 overrides  
15         if(canIWithdraw(amount))  
16             this.balance -= amount;  
17     }  
18  
19     public Double getBalance() { 2 usages  
20         return balance;  
21     }  
22  
23 ⚡     public Boolean canIWithdraw(Double amount){ 1 usage 1 override  
24         return amount <= this.balance;  
25     }  
26 }  
27
```

Clase hija:

```
© BankAccount.java × © BankAccountOverdraft.java × © BankAccountCommision.java © Main.java © Check.java  
1  public class BankAccountOverdraft extends BankAccount { 2 usages  
2      private Double overdraftAmount; 2 usages  
3  
4      public BankAccountOverdraft(String accountOwner, Double balance, Double overdraftAmount) { 1 usag  
5          super(accountOwner, balance);  
6          this.overdraftAmount = overdraftAmount;  
7      }  
8  
9      @Override 3 usages  
10     ⚡     public void withdraw(Double amount) {  
11         super.withdraw(amount);  
12     }  
13  
14     @Override 1 usage  
15     ⚡     public Boolean canIWithdraw(Double amount) {  
16         return amount <= getBalance() + overdraftAmount;  
17     }  
18 }  
19
```



Métodos de clase