



# Lesson 1

12.12.2025

# Трохи про себе



УКРАЇНОПΟΥΧ  
UKSATSE



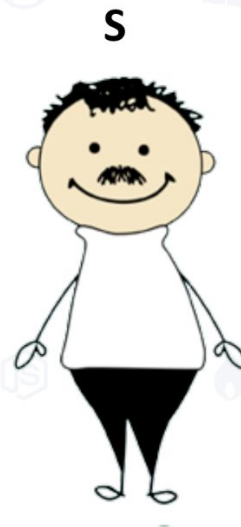
**INTERSOG**



**ANDERSEN**



hillel  
IT school



2014



3 mec

2014



6 mec

2014







**Зпрошую до гри**

У кожного свій шлях

у Java dev такий



# Java + Spring Boot Roadmap

## Step 1:

### Master Java Fundamentals

- Java 17 or 21
- OOP Concepts
- Lambdas and Streams
- Multithreading and Concurrency



## Step 3:

### Dive Into Spring Core

- Dependency Injection (DI)
- Inversion of Control (IoC)
- Bean Lifecycle
- Configuration Classes



## Step 2:

### Learn Maven or Gradle

- Dependency Management
- Profiles (Dev vs Production)
- Build and Deployment

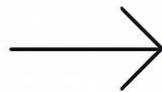




# Java + Spring Boot Roadmap

## Step 4 Build with Spring Boot

- Auto Configuration
- Starter Dependencies
- Profiles and Properties
- Actuator for monitoring



## Step 5 Build Secure REST APIs



- Create Controllers
- Use DTOs
- Validate Data
- Handle Exceptions
- Implement Rate Limiting

## Step 6 Secure Your Applications



- Spring Security Basics
- JWT Authentication
- OAuth2
- Custom Filters



# Java Developer Roadmap

## 7 Work with Databases

Spring Boot integrates smoothly with databases:

- Spring Data JPA
- Custom Queries
- Transactions
- Database Migrations with Flyway or Liquibase

## 8 Testing Best Practices

Take separates good developer from great ones. Focus on:

- Unit Testing with JUnit 5
- Mocking with Mockito
- Integration Testing with Testcontainers

## 10 Advanced Topics

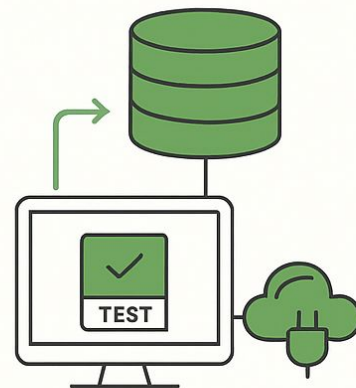
Take your backend skills to the next level by learning:

- Microservices Architecture
- Messaging with Kafka

## 9 Connect to External APIs

Take your backend skills to the next level by learning:

- Microservices Architecture
- Messaging with Kafka
- APIs with GraphQL
- Reactive Programming with WebFlux





Java<sup>TM</sup>

# 30 YEARS OF JAVA



# 7 причин вибрати Java для вивчення

№ 1 - Сильне Java-community

№ 2 - Різноманітність інструментів

№ 3 - Java virtual machine

№ 4 - Кросплатформенність Java

№ 5- Зворотна сумісність

№ 6 - Численні способи для вивчення Java

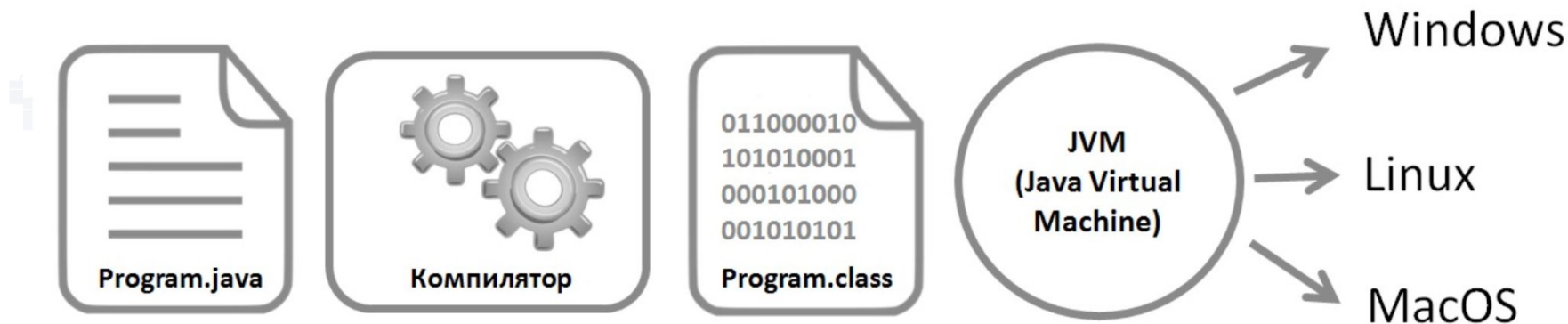
№ 7 - Можливість бути затребуваним на ринку



# Чому Java

1. "Написано одного разу, працює скрізь"
2. Дружній синтаксис
3. Об'єктно-орієнтована мова
4. Управління пам'яттю

# 1. "Написано одного разу, працює скрізь"



Коли Ви писатимете програми на мові Java, вони завжди будуть зберігати окремі файли. Причому ці файли завжди матимуть розширення **.java**

Далі, коли Ви запускаєте компілятор (javac), Ваш код з людиночитаного перетворюється на так званий байт-код (тобто в вигляді різних комбінацій 0 і 1) і код стає виключно машиночитаним. Після цього з'явиться ще один файл, який завжди матиме розширення **.class**.

Потім JVM (Java Virtual Machine) виконує байт-код.



## 2. Дружній синтаксис

Взяли все найкраще від найкращої марки "велосипеда" - мов програмування C та C++

Викинули все, що вважали зайвим і не особливо вдалим у C та C++

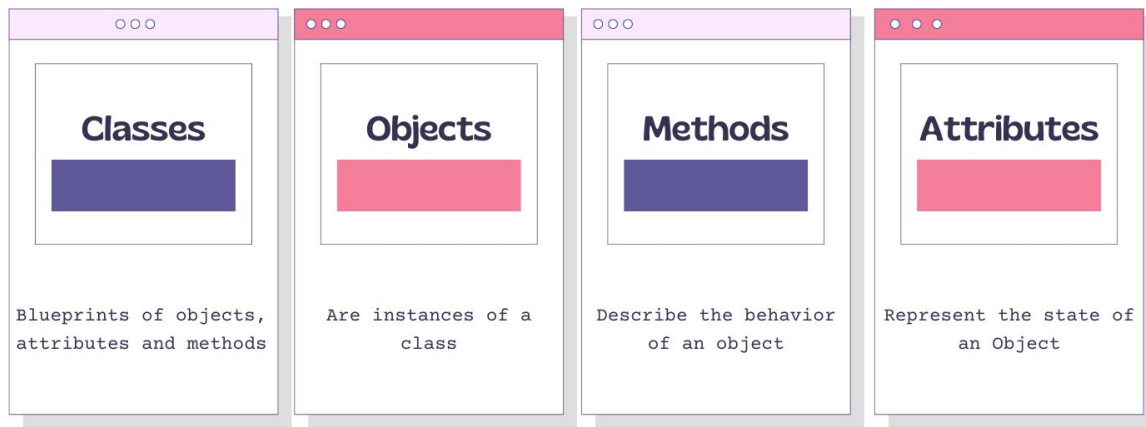
Внесли нововведення в нову мову програмування Java





### 3. Об'єктно-орієнтована мова

## Structure of Object-Oriented Programming



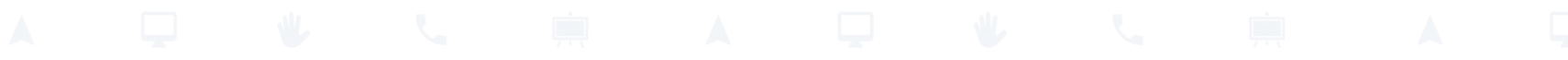
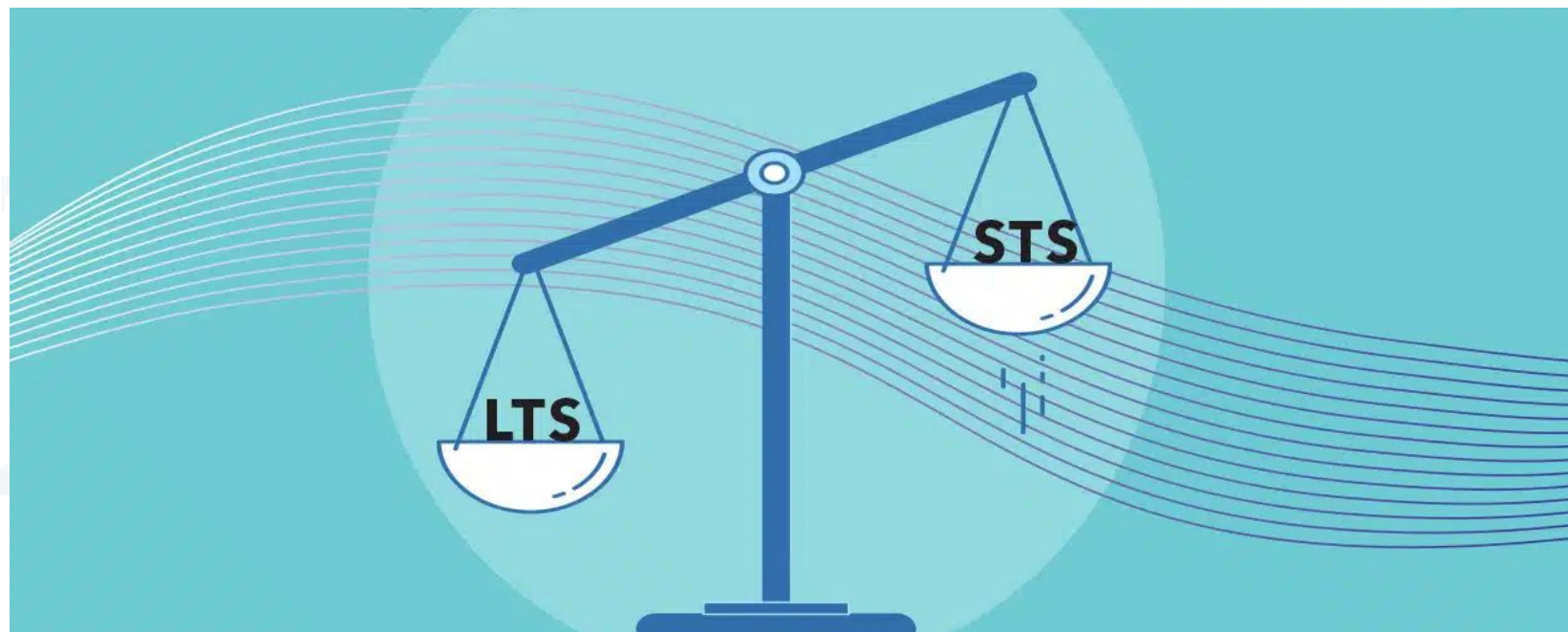


## 4. Управління пам'яттю

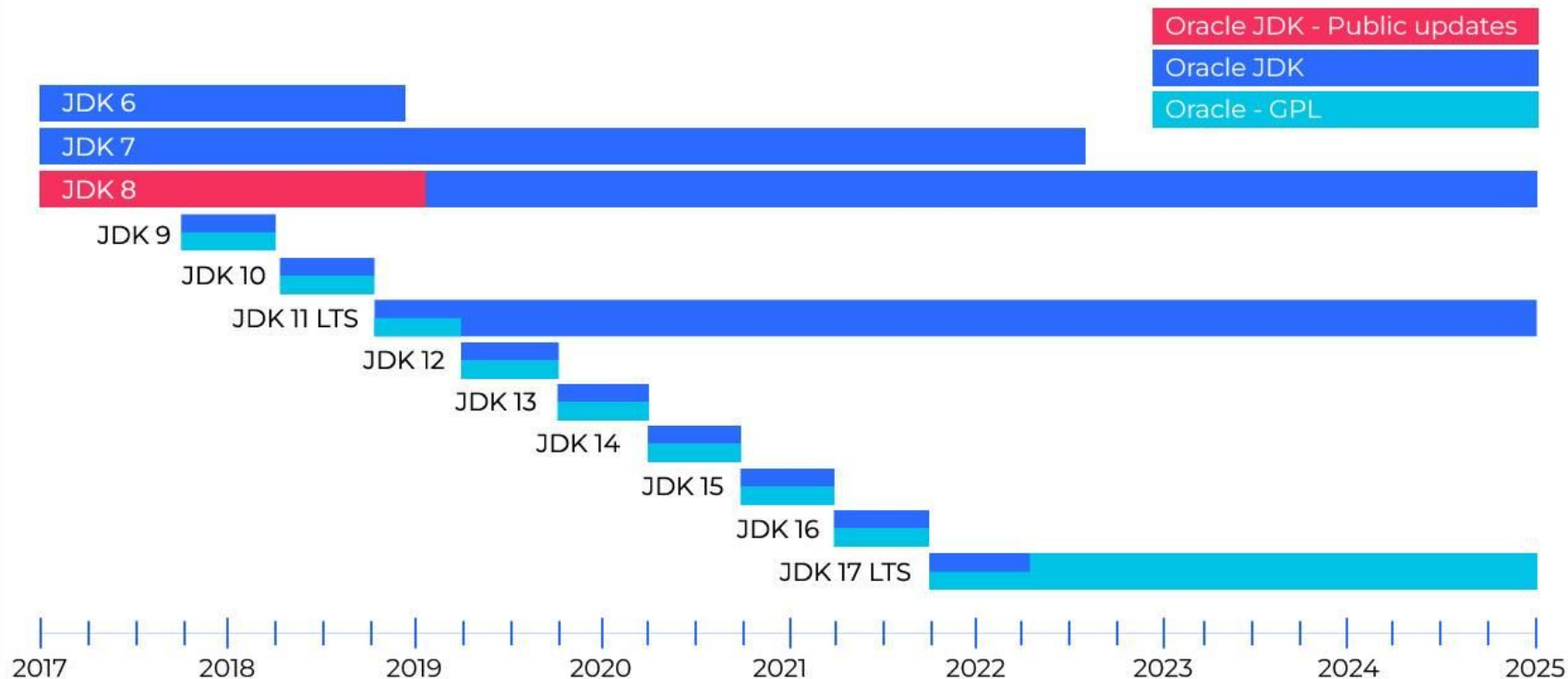


# Java Version History





# New Java Release Timeline

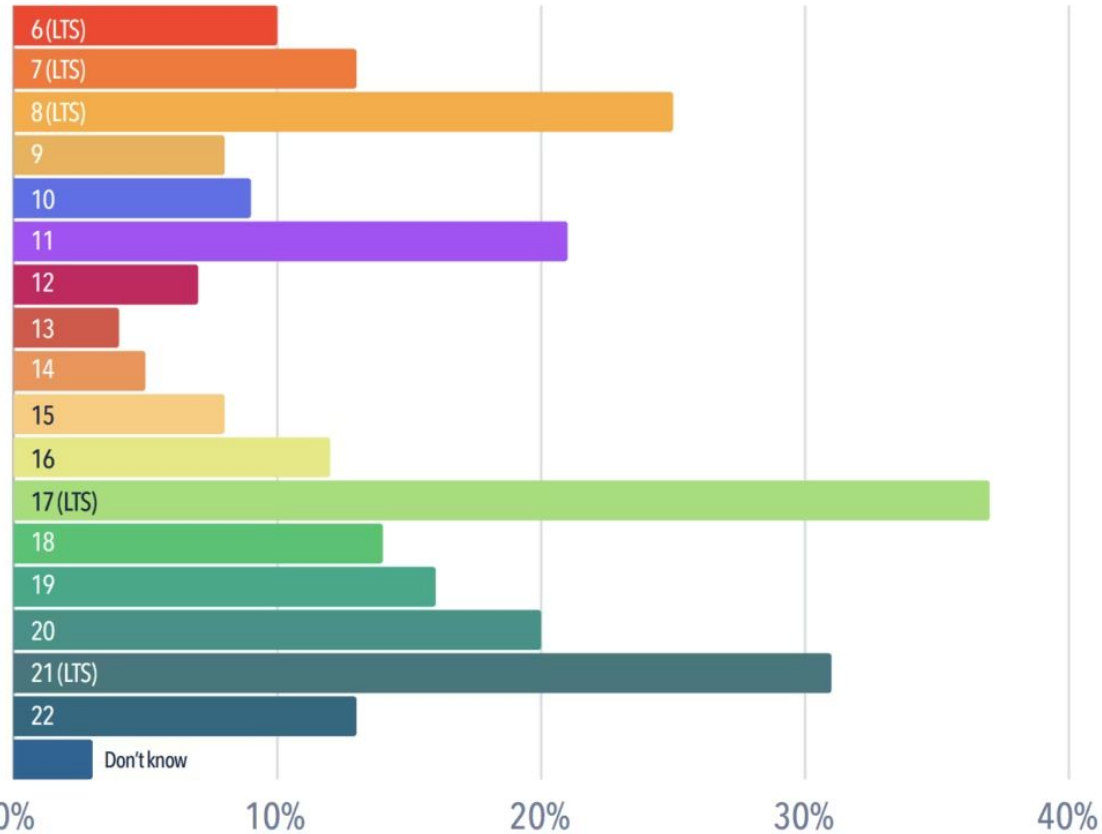


\*Dates and version numbers subject to change

LTS = Long term support release

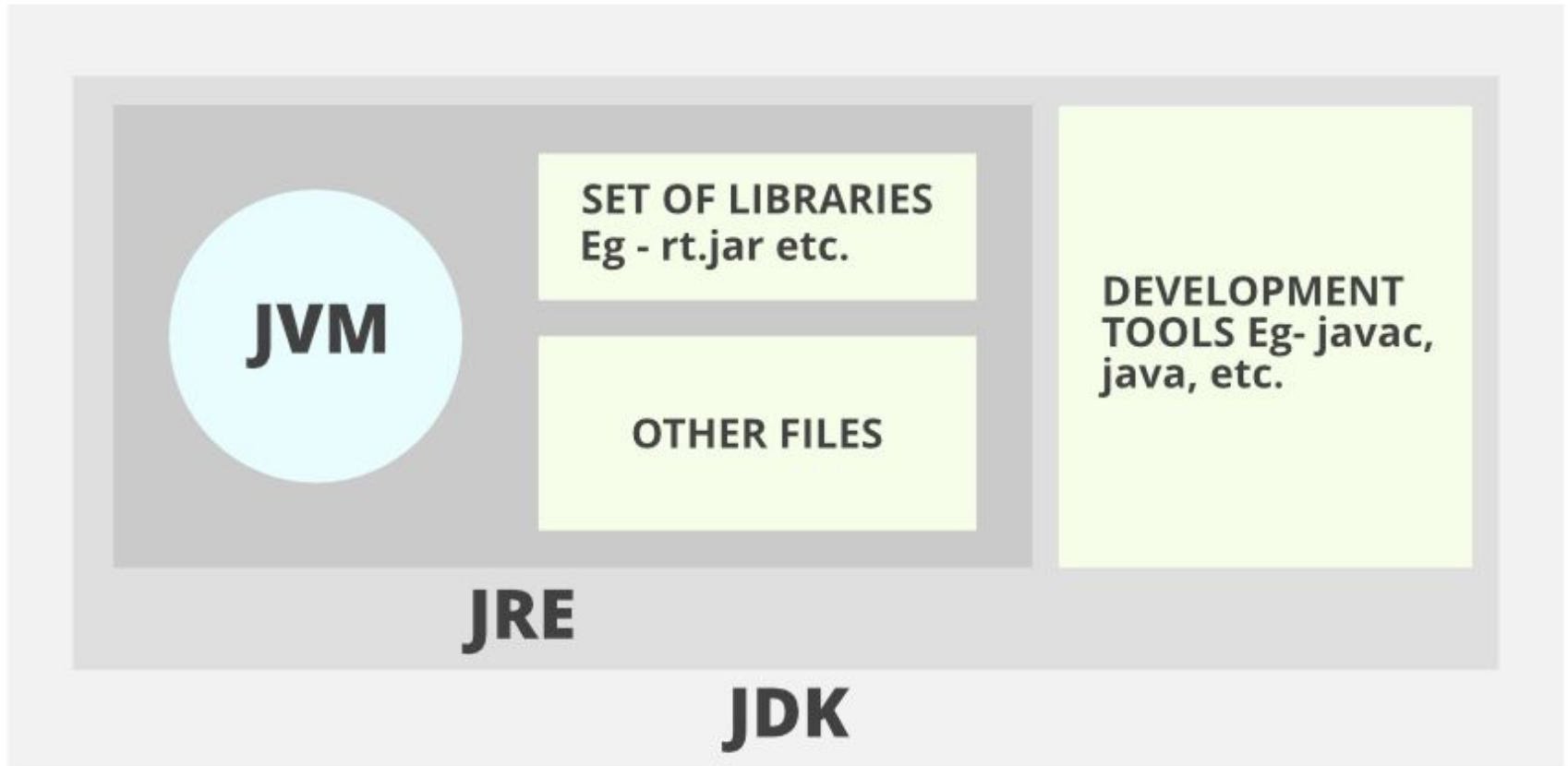



## JAVA VERSIONS





# JDK vs JRE vs JVM





**JDK:** Software Development Kit: Це набір інструментів для розробки на Java, в JRE, та компілятори (compilers) та інструменти (як JavaDoc та Java Debugger) щоб створювати та компілювати програми.

**JRE:** Java Runtime Environment (Серед виконання для Java). Фактично це віртуальна машина Java (Java Virtual Machine) на якому запуснені ваші програми Java. Вона також включає браузер плагіни для виконання Applet.



azul



OpenJDK



GraalVM.



# Встановлення JDK

**Крок 1:** Завантажити інсталяційний файл із сайту -

[www.oracle.com/technetwork/java/javase/downloads/index.html](http://www.oracle.com/technetwork/java/javase/downloads/index.html)

**Крок 2:** Встановити JDK із файлу

**Крок 3:** Встановлення змінних середовища - JAVA\_HOME та PATH



## Щоб налаштувати програмне оточення на Ubuntu, потрібно зробити кілька кроків:

**Крок №1** - Запустити консоль (поєднання клавіш Ctrl+Alt+T), а також переконатися що ви маєте права встановлювати програми на комп'ютер

**Крок №2** - Оновити списки пакетів:

```
sudo apt-get update
```

**Крок №3** - Перевірити версію Java, якщо раніше не встановлювали повернути "The program java can be found in the following packages":

```
java -version
```

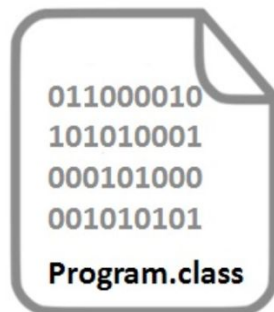
**Крок №4** - Далі, на сьогоднішній день, актуально встановити JDK 17:

```
apt install openjdk-17-jdk openjdk-17-jre
```

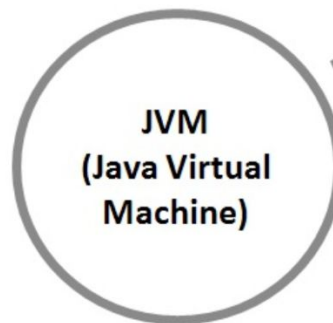
**Крок №5** - І знову перевірити версію і переконатися, що варто JDK 8:

```
java -version
```

# javac




# java




Windows

Linux

MacOS



```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello world...");  
    }  
}
```







```
javac [options] [source files]
```

```
java [options] file.class [arguments...]
```

```
java [options] -jar file.jar [arguments... ]
```

# Top Java IDE Tools



Eclipse



IntelliJ IDEA



Netbeans



Visual Studio



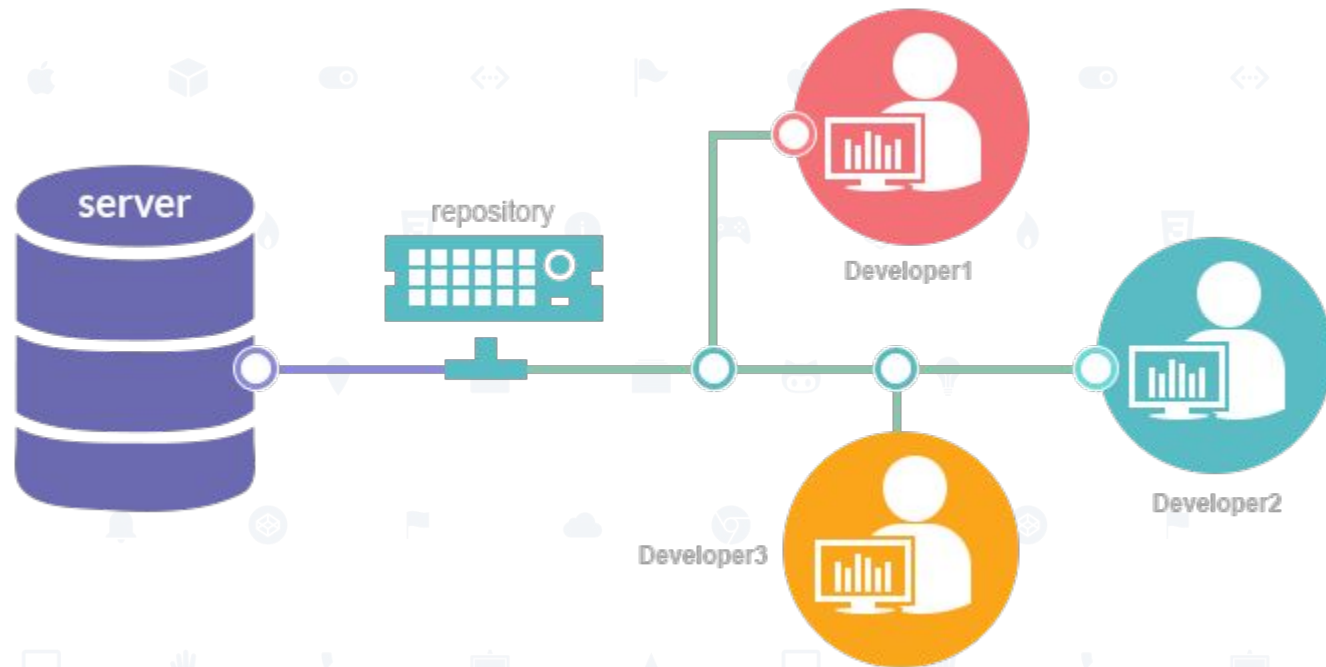
Xcode



Apache ANT

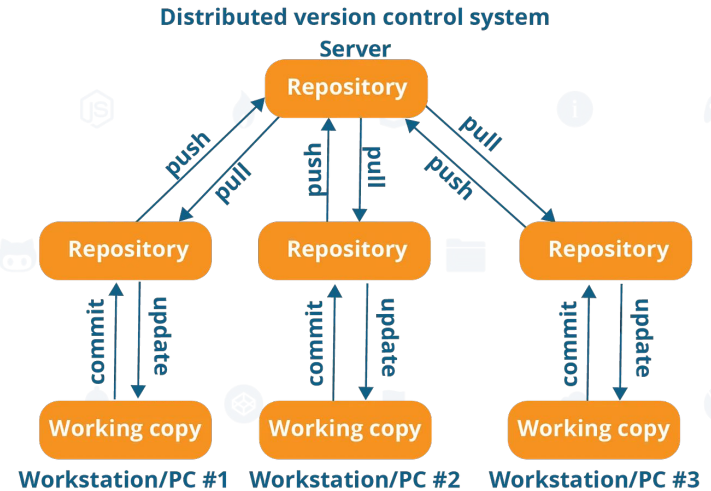
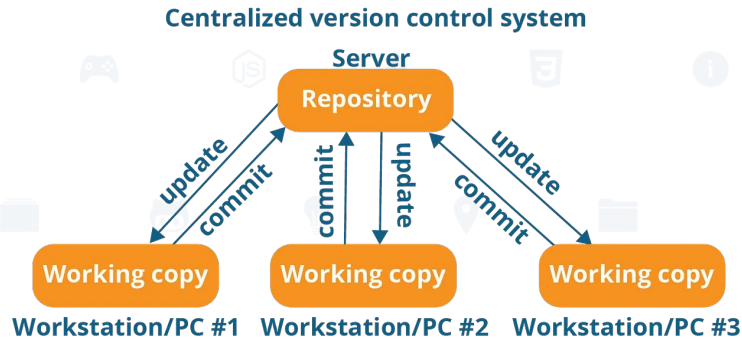
**Система контролю версій (Version Control System, VCS)** - програмне забезпечення для полегшення роботи з інформацією, що змінюється.

VCS дозволяє зберігати кілька версій одного і того ж документа, при необхідності повертатися до ранніх версій, визначати, хто і коли зробив ту чи іншу зміну, та багато іншого.



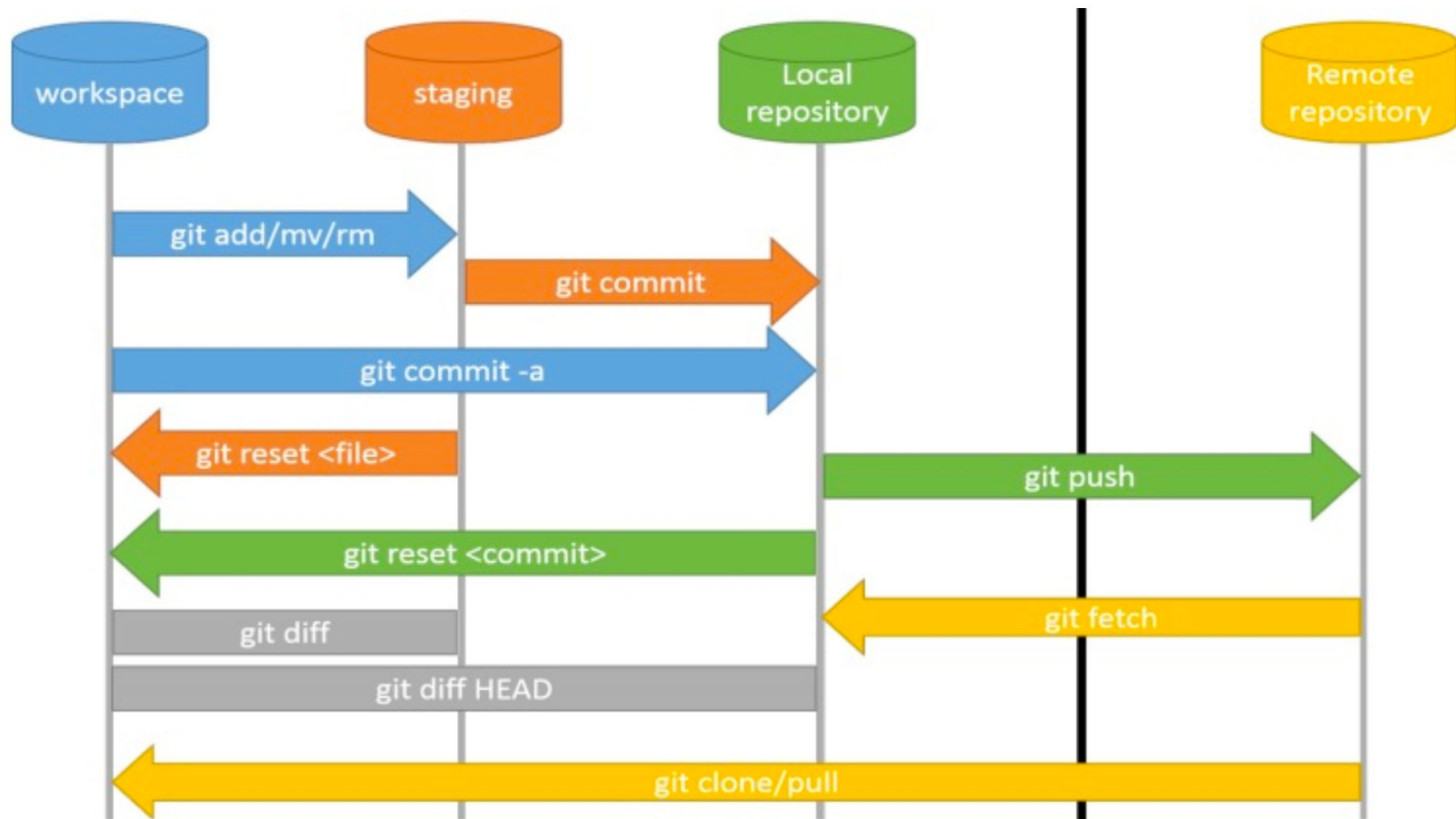
# Types of Version Control System

- Local Version Control Systems
- Centralized Version Control Systems
- Distributed Version Control Systems





<https://github.com/>





Створення репозиторію в існуючому каталозі: **\$ git init**

Клонування існуючого репозиторію: **\$ git clone [url]**

Додавання змінених файлів до індексу: **\$ git add file [, file]**

Фіксація змін: **\$ git commit**

Відкат до попереднього стану проекту: **\$ git reset – hard HEAD**

Злиття гілок: **\$ git merge**

Оновлення віддаленого репозиторію: **\$ git push**