

# FULLSTACK APPLICATION FUNDAMENTALS

## Object-Oriented Programming

Author: Eng. Carlos Andrés Sierra, M.Sc.  
[cavirguezs@udistrital.edu.co](mailto:cavirguezs@udistrital.edu.co)

Full-time Adjunct Professor  
Computer Engineering Program  
School of Engineering  
Universidad Distrital Francisco José de Caldas

2025-III



# Outline

- ① Layered Architecture
- ② Data Layer
- ③ Backend Layer
- ④ FrontEnd Layer
- ⑤ Computing Resources



# Outline

1 Layered Architecture

2 Data Layer

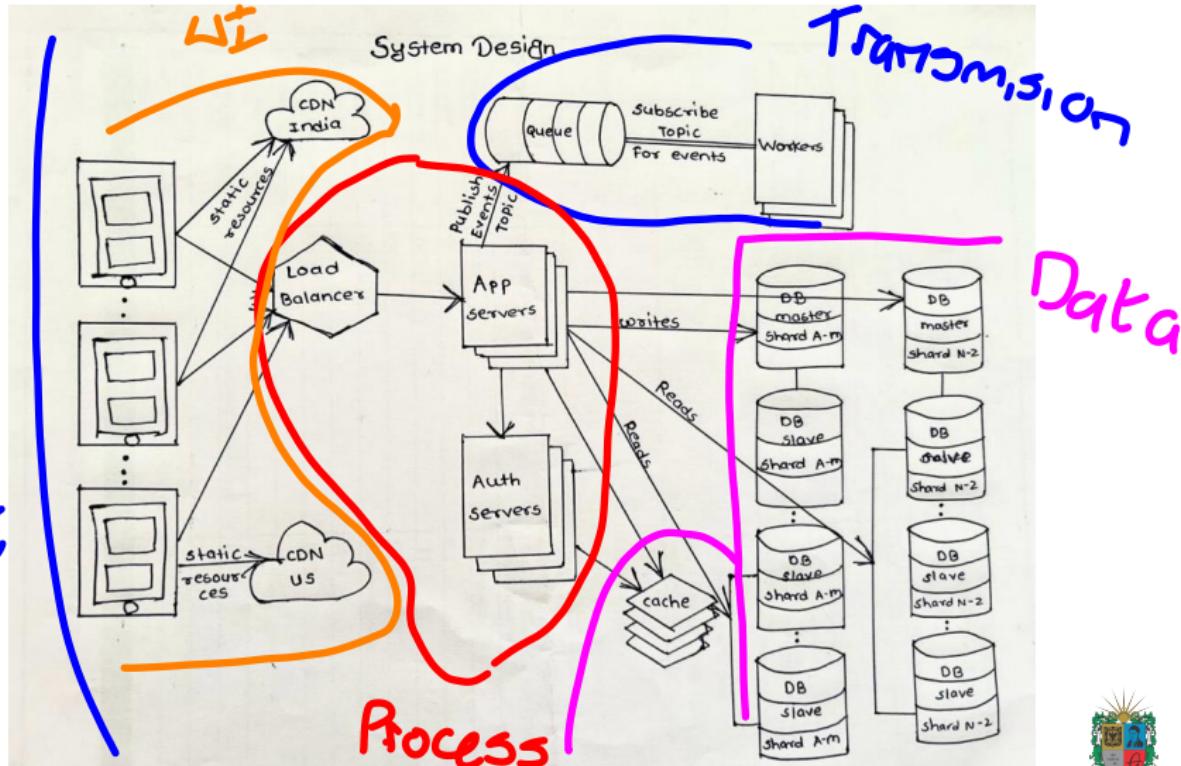
3 Backend Layer

4 FrontEnd Layer

5 Computing Resources



# Systems Design applied to Software Architectures



# What is a System Architecture?

- A system architecture is the structure of a system that defines its components, interactions, and relationships. → O.O.P.
- A system architecture is the blueprint of a system that guides its development and implementation. ↗ template + recipe
- A system architecture is the foundation of a system that ensures that it meets the needs of its users.

solve the problem

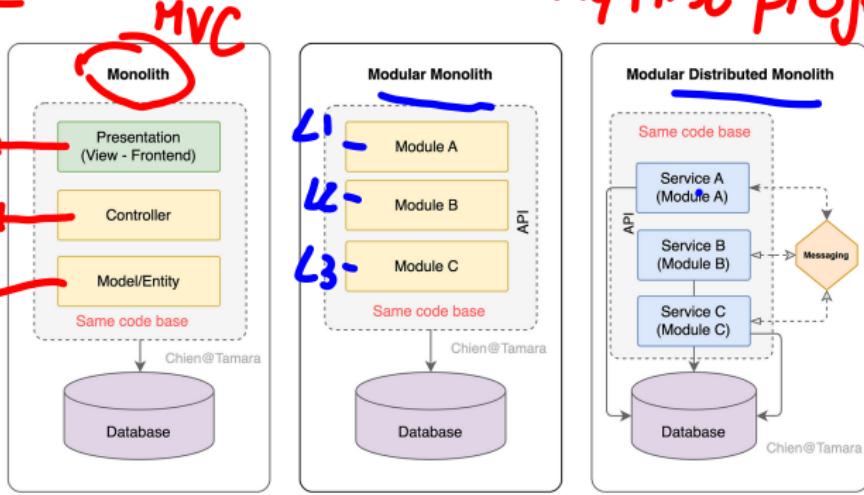


# Monolithic System Architecture

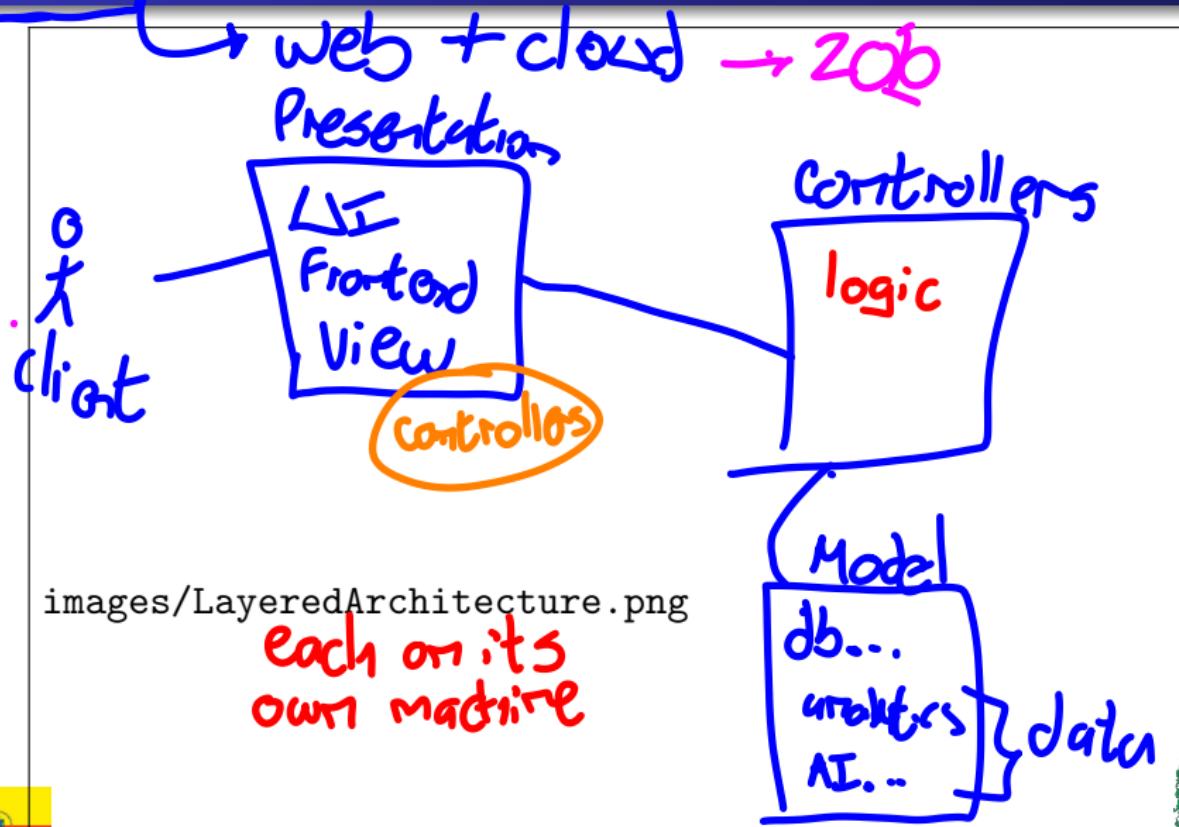
- A **monolithic system architecture** is a **single-tier** architecture that consists of a **single unit** that performs all the **functions** of the system. → **1 project → 1 machine**
- It is **simple**, **easy to develop**, and **maintain**, but it is **not scalable** and flexible. It is typically used for **small systems** that do not require high performance or reliability.

→ **my first project**

**each Folder**  
**L1** ← **UI**  
**Logic** ←  
**Data Access** ←



# Layered Architecture Pattern



# Packages in Java

**Packages** are a way of structuring the Java namespace using dotted package names. → **Folder structure**

- **Creating Packages:** To create a package, you just have to create a directory with a package-info.java file.
- **Importing Packages:** To import a package, you can use the import statement.
- **Third-party Packages:** Java has a set of third-party packages that you can use in your projects.
- **Maven:** Maven is a build automation tool used primarily for Java projects.

## Demo time!

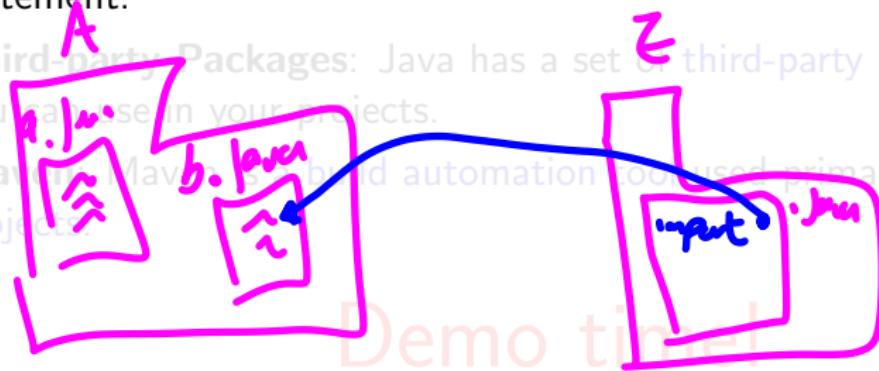


# Packages in Java

**Packages** are a way of structuring the Java namespace using *dotted package names*.

- **Creating Packages:** To create a package, you just have to create a **directory** with a `package-info.java` file.
- **Importing Packages:** To **import a package**, you can use the `import` statement.

- **Third-party Packages:** Java has a set of third-party packages that you can use in your projects.
- **Maven**: Maven is a build automation tool used primarily for Java projects.



# Packages in Java

**Packages** are a way of structuring the Java namespace using *dotted package names*.

- **Creating Packages:** To create a package, you just have to create a **directory** with a `package-info.java` file.
- **Importing Packages:** To **import a package**, you can use the `import` statement.
- **Third-party Packages:** Java has a set of **third-party packages** that you can use in your projects.
- **Maven:** Maven is a build automation tool used primarily for Java projects.

↳ `util`

`Math`

Demo time!

• `Spring Boot`



# Packages in Java

**Packages** are a way of structuring the Java namespace using *dotted package names*.

- **Creating Packages:** To create a package, you just have to create a [directory](#) with a `package-info.java` file.
- **Importing Packages:** To [import a package](#), you can use the `import` statement.
- **Third-party Packages:** Java has a set of [third-party packages](#) that you can use in your projects.
- **Maven:** Maven is a [build automation tool](#) used primarily for [Java](#) projects.

project management  
packages

Demo time!



# Outline

1 Layered Architecture

2 Data Layer

3 Backend Layer

4 FrontEnd Layer

5 Computing Resources



# Data System Concepts

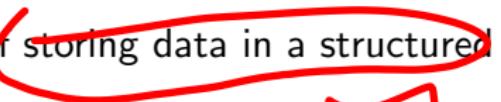
## Key Points of Data Systems:

- **Data modeling** is the process of designing the **structure** and organization of data.
- Data storage is the process of storing data in a structured or unstructured format.  
*what Format sources transform*
- Data retrieval is the process of accessing and retrieving data from a storage system.
- Data integration is the process of modifying and transforming data.
- Data security is the process of protecting data from unauthorized access and ensuring its **integrity** and **confidentiality**.



# Data System Concepts

## Key Points of Data Systems:

- **Data modeling** is the process of designing the **structure** and organization of data.
- **Data storage** is the process of storing data in a **structured** or unstructured **format**. 
- **Data retrieval** is the process of accessing and retrieving data from a storage system. 
- Data manipulation is the process of modifying and **transforming** data.
- Data security is the process of protecting data from unauthorized access and ensuring its **integrity** and **confidentiality**.



# Data System Concepts

## Key Points of Data Systems:

- Data modeling is the process of designing the **structure** and organization of data.
- Data storage is the process of storing data in a structured or unstructured **format**.
- Data retrieval is the process of **accessing** and **retrieving** data from a storage system.  
*→ resources → extract*
- Data manipulation is the process of modifying and transforming data.
- Data security is the process of protecting data from unauthorized access and ensuring its **integrity** and **confidentiality**.



# Data System Concepts

## Key Points of Data Systems:

- **Data modeling** is the process of designing the **structure** and organization of data.
- **Data storage** is the process of storing data in a structured or unstructured **format**.
- **Data retrieval** is the process of **accessing** and **retrieving** data from a storage system.
- **Data manipulation** is the process of **modifying** and **transforming** data.
- **Data security** is the process of protecting data from unauthorized access and ensuring its **integrity** and **confidentiality**.



# Data System Concepts

## Key Points of Data Systems:

- **Data modeling** is the process of designing the **structure** and organization of data.
- **Data storage** is the process of storing data in a structured or unstructured **format**.
- **Data retrieval** is the process of **accessing** and **retrieving** data from a storage system.
- **Data manipulation** is the process of modifying and **transforming** data.
- **Data security** is the process of protecting data from unauthorized access and ensuring its integrity and confidentiality.

*no damage*



# Data Access Objects and Data Transfer Objects

Data Access Objects (DAOs) and Data Transfer Objects (DTOs) are design patterns used to separate the data access logic from the business logic in an application.

- A Data Access Object (DAO) is an object that provides an abstract interface to some type of database or other persistence mechanism.
- The DAO pattern is used to separate the data access logic from the business logic in an application.



Demo time!



# Data Access Objects and Data Transfer Objects

Data Access Objects (DAOs) and Data Transfer Objects (DTOs) are design patterns used to separate the data access logic from the business logic in an application.

- A Data Access Object (DAO) is an object that provides **an abstract interface** to some type of database or other persistence mechanism.
- The DAO pattern is used to **separate the** data access logic from the business logic in an application.
- A Data Transfer Object (DTO) is an object that **carries data** between processes in an application.
- The DTO pattern is used to **transfer data** between processes in an application.

→ database only

List of  
Objects

Demo time!

List of  
rows

One	Two	Three	Four
Row 1			
Row 2			
Row 3			



# Objects Persistence

- **Serialization:** It is the process of converting an object into a stream of bytes. *Object vs. stream* → **fast, flyweight**
- **Deserialization:** It is the process of converting a stream of bytes into an object.

## Demo time!

- **JSON:** It is a lightweight data-interchange format that is easy for *humans* to read and write and easy for *machines* to parse and generate.

## Demo time!



# Objects Persistence

- **Serialization:** It is the process of **converting** an **object** into a **stream of bytes**.
- **Deserialization:** It is the process of **converting** a **stream of bytes** into an **object**.

## Demo time!

- **JSON:** It is a **lightweight data-interchange format** that is easy for *humans* to read and write and easy for *machines* to parse and generate.

## Demo time!



# Outline

1 Layered Architecture

2 Data Layer

3 Backend Layer

4 FrontEnd Layer

5 Computing Resources



# Backend Concepts

## Key Points of Backend Systems:

- A **backend system** is a software system that provides the **logic** and functionality to support the front-end of an application.
- A **backend system** typically consists of a **server**, a **database**, and an **application server**.
- A **server** is a computer that provides **services** to other computers over a network.
- An **application server** is a software framework that provides an environment for **running web applications**.
- A **database** is a collection of data that is **organized** and stored in a defined format.



# Backend Concepts

## Key Points of Backend Systems:

- A **backend system** is a software system that provides the **logic** and functionality to support the front-end of an application.
- A **backend system** typically consists of a **server**, a **database**, and an **application server**.
- A **server** is a computer that provides **services** to other computers over a network.
- An **application server** is a software framework that provides an environment for **running web applications**.
- A **database** is a collection of data that is **organized and stored** in a defined format.



# Connection with Data Layer

- The **backend layer** is responsible for managing the **data layer** and providing the **logic** and **functionality** to *support the front-end* of an application.
- The **connection** between the backend and data layers is typically managed through an **application programming interface (API)**.
- An **API** is a set of **rules** and **protocols** that allows different software applications to **communicate** with each other.
- The **API** provides a way for the front-end of an application to **interact** with the backend and access the data stored in the database.
- **ORM frameworks** such as **SQLAlchemy** are often used to manage the **connection** between the backend and data layers.



# Connection with Data Layer

- The **backend layer** is responsible for managing the **data layer** and providing the **logic** and **functionality** to *support the front-end* of an application.
- The **connection** between the backend and data layers is typically managed through an **application programming interface (API)**.
- An **API** is a set of **rules** and **protocols** that allows different software applications to **communicate** with each other.
- The **API** provides a way for the front-end of an application to **interact** with the backend and access the data stored in the database.
- **ORM frameworks** such as **SQLAlchemy** are often used to manage the **connection** between the backend and data layers.



# Connection with Data Layer

- The **backend layer** is responsible for managing the **data layer** and providing the **logic** and **functionality** to *support the front-end* of an application.
- The **connection** between the backend and data layers is typically managed through an **application programming interface (API)**.
- An **API** is a set of **rules** and **protocols** that allows different software applications to **communicate** with each other.
- The **API** provides a way for the front-end of an application to **interact** with the backend and access the data stored in the database.
- **ORM frameworks** such as **SQLAlchemy** are often used to manage the **connection** between the backend and data layers.



# Outline

1 Layered Architecture

2 Data Layer

3 Backend Layer

4 FrontEnd Layer

5 Computing Resources



# FrontEnd Layer

- The **front-end** is the **client-side** of the application and everything that the user interacts with.
- The **front-end** is the **presentation layer** of the application.
- The **front-end** is the **user interface** and the **user experience**.



# FrontEnd Layer

- The **front-end** is the **client-side** of the application and everything that the user interacts with.
- The **front-end** is the **presentation layer** of the application.
- The **front-end** is the **user interface** and the **user experience**.



# Model-View-Controller (MVC) Pattern

- The **Model-View-Controller** (MVC) is a software architectural pattern that separates the application into three main components: **model**, **controller**, and **view**.
- The **model** is responsible for managing the **data** and **business logic** of the application.
- The **view** is responsible for **displaying** the data to the user and providing a way for the user to interact with the application.
- The **controller** is responsible for **handling user input** and **updating** the model and view.



# Model-View-Controller (MVC) Pattern

- The **Model-View-Controller** (MVC) is a software architectural pattern that separates the application into three main components: **model**, **controller**, and **view**.
- The **model** is responsible for managing the **data** and **business** logic of the application.
- The **view** is responsible for **displaying** the data to the user and providing a way for the user to interact with the application.
- The **controller** is responsible for **handling** user **input** and updating the model and view.



# Model-View-Controller (MVC) Pattern

- The **Model-View-Controller** (MVC) is a software architectural pattern that separates the application into three main components: **model**, **controller**, and **view**.
- The **model** is responsible for managing the **data** and **business** logic of the application.
- The **view** is responsible for **displaying** the data to the user and providing a way for the user to interact with the application.
- The **controller** is responsible for **handling** user **input** and updating the model and view.



# Model-View-Controller (MVC) Pattern

- The **Model-View-Controller** (MVC) is a software architectural pattern that separates the application into three main components: **model**, **controller**, and **view**.
- The **model** is responsible for managing the **data** and **business** logic of the application.
- The **view** is responsible for **displaying** the data to the user and providing a way for the user to interact with the application.
- The **controller** is responsible for **handling** user **input** and **updating** the model and view.



# Java Swing & Java FX

- **Java Desktop Applications** are applications that **run on a user's computer** and provide a graphical user interface (GUI) for the user to interact with.
- **Java Swing** is a set of **GUI components** that can be used to create **desktop applications** in Java.
- **JavaFX** is a set of **GUI components** that can be used to create **desktop applications** in Java.
- **JavaFX** is the successor to **Java Swing** and provides a more modern, flexible way to create desktop applications.
- **JavaFX Scene Builder** is a visual layout tool that allows developers to create JavaFX user interfaces without writing any code.



# Java Swing & Java FX

- **Java Desktop Applications** are applications that **run on a user's computer** and provide a graphical user interface (GUI) for the user to interact with.
- **Java Swing** is a set of **GUI components** that can be used to create **desktop applications** in Java.
- **JavaFX** is a set of **GUI components** that can be used to create **desktop applications** in Java.
- **JavaFX** is the successor to **Java Swing** and provides a more modern, **flexible** way to create **desktop applications**.
- **JavaFX Scene Builder** is a visual layout tool that allows developers to create JavaFX user interfaces without writing any code.



# Java Swing & Java FX

- **Java Desktop Applications** are applications that [run on a user's computer](#) and provide a graphical user interface (GUI) for the user to interact with.
- **Java Swing** is a set of [GUI components](#) that can be used to create [desktop applications](#) in Java.
- **JavaFX** is a set of [GUI components](#) that can be used to create [desktop applications](#) in Java.
- **JavaFX** is the successor to **Java Swing** and provides a more modern, [flexible](#) way to create [desktop applications](#).
- **JavaFX Scene Builder** is a visual layout tool that allows developers to create [JavaFX user interfaces](#) without writing any code.



# Java FX Components

- **JavaFX Components** are the building blocks of a JavaFX user interface.
- **JavaFX Components** are organized into a **scene graph**, which is a hierarchical structure that represents the **layout** and **organization** of the user interface.
- **JavaFX Components** can be customized and styled using CSS.
- **JavaFX Components** can be animated and transformed using the JavaFX animation and transformation APIs.
- **JavaFX Components** can be controlled using event handlers.
- **JavaFX Components** can be bound to data using the JavaFX binding APIs.

## Demo time!



# Java FX Components

- **JavaFX Components** are the building blocks of a JavaFX user interface.
- **JavaFX Components** are organized into a **scene graph**, which is a hierarchical structure that represents the **layout** and **organization** of the user interface.
- **JavaFX Components** can be **customized** and **styled** using CSS.
- **JavaFX Components** can be **animated** and **transformed** using the JavaFX animation and transformation APIs.
- **JavaFX Components** can be **controlled** using event handlers.
- **JavaFX Components** can be **bound** to data using the JavaFX binding APIs.

## Demo time!



# Java FX Components

- **JavaFX Components** are the building blocks of a JavaFX user interface.
- **JavaFX Components** are organized into a **scene graph**, which is a hierarchical structure that represents the **layout** and **organization** of the user interface.
- **JavaFX Components** can be **customized** and **styled** using CSS.
- **JavaFX Components** can be **animated** and transformed using the JavaFX animation and transformation APIs.
- **JavaFX Components** can be **controlled** using event handlers.
- **JavaFX Components** can be **bound** to data using the JavaFX binding APIs.

## Demo time!



# Java FX Components

- **JavaFX Components** are the building blocks of a JavaFX user interface.
- **JavaFX Components** are organized into a **scene graph**, which is a hierarchical structure that represents the **layout** and **organization** of the user interface.
- **JavaFX Components** can be **customized** and **styled** using CSS.
- **JavaFX Components** can be **animated** and transformed using the JavaFX animation and transformation APIs.
- **JavaFX Components** can be **controlled** using event handlers.
- **JavaFX Components** can be **bound** to data using the JavaFX binding APIs.

## Demo time!



# Outline

- 1 Layered Architecture
- 2 Data Layer
- 3 Backend Layer
- 4 FrontEnd Layer
- 5 Computing Resources



# Memory Management

- **Garbage Collection:** It is the process of automatically *reclaiming memory* that is **no longer in use**.
- **Memory Profiling:** It is the process of analyzing the **memory usage** of a program.
- **Memory Leaks:** A common problem in programming where memory is allocated but **never deallocated**.
- **Memory Management Tools:** There are several tools available for memory management in Java.

## Demo time!



# Memory Management

- **Garbage Collection:** It is the process of automatically *reclaiming* memory that is **no longer in use**.
- **Memory Profiling:** It is the process of analyzing the **memory usage** of a program.
- **Memory Leaks:** A common problem in programming where memory is allocated but **never deallocated**.
- **Memory Management Tools:** There are several tools available for memory management in Java.

Demo time!



# Memory Management

- **Garbage Collection:** It is the process of automatically *reclaiming* memory that is **no longer in use**.
- **Memory Profiling:** It is the process of analyzing the **memory usage** of a program.
- **Memory Leaks:** A common problem in programming where memory is allocated but **never deallocated**.
- **Memory Management Tools:** There are several tools available for **memory management** in Java.

## Demo time!



# Threads, Processes, and Concurrency

- **Threads:** They are the **smallest unit of execution** that can be scheduled by an operating system.
- **Processes:** They are the **largest unit of execution** that can be scheduled by an operating system.
- **Parallelism:** It is the ability of a program to execute **multiple tasks simultaneously**.
- **Synchronization:** Synchronization is the process of coordinating the execution of multiple threads or processes.



# Threads, Processes, and Concurrency

- **Threads:** They are the **smallest unit of execution** that can be scheduled by an operating system.
- **Processes:** They are the **largest unit of execution** that can be scheduled by an operating system.
- **Parallelism:** It is the ability of a program to execute **multiple tasks simultaneously**.
- **Synchronization:** Synchronization is the process of coordinating the execution of **multiple threads or processes**.



# Threads, Processes, and Concurrency

- **Threads:** They are the **smallest unit of execution** that can be scheduled by an operating system.
- **Processes:** They are the **largest unit of execution** that can be scheduled by an operating system.
- **Parallelism:** It is the ability of a program to execute **multiple tasks simultaneously**.
- **Synchronization:** Synchronization is the process of coordinating the execution of multiple threads or processes.



# Threads, Processes, and Concurrency

- **Threads:** They are the **smallest unit of execution** that can be scheduled by an operating system.
- **Processes:** They are the **largest unit of execution** that can be scheduled by an operating system.
- **Parallelism:** It is the ability of a program to execute **multiple tasks simultaneously**.
- **Synchronization:** Synchronization is the process of **coordinating** the execution of **multiple threads** or processes.



# Parallel Computation

- **Parallel Computation:** It is the process of executing **multiple tasks simultaneously**.
- **Distributed Computation:** It is the process of executing multiple tasks on multiple machines.

## Demo time!



# Parallel Computation

- **Parallel Computation:** It is the process of executing **multiple tasks simultaneously**.
- **Distributed Computation:** It is the process of executing **multiple tasks on multiple machines**.

## Demo time!



# Outline

- 1 Layered Architecture
- 2 Data Layer
- 3 Backend Layer
- 4 FrontEnd Layer
- 5 Computing Resources



# Thanks!

## Questions?



Repo: <https://github.com/EngAndres/ud-public/tree/main/courses/object-oriented-programming>

