Object-Oriented Introduction

Advanced Programming

Author: Eng. Carlos Andrés Sierra, M.Sc.
cavirguezs@udistrital.edu.co

Computer Engineer
Lecturer
Universidad Distrital Francisco José de Caldas

2024-III

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Outline

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Outline

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Basics of Software Architecture I

- It is important to **develop** innovative and sophisticated **software** to provide a nice **solution** for **end users needs**.
- Software architecture brings **innovation** and **robust** structures.
- The goal of software architecture is to **minimize the human efforts** required to build and maintain the expected system.

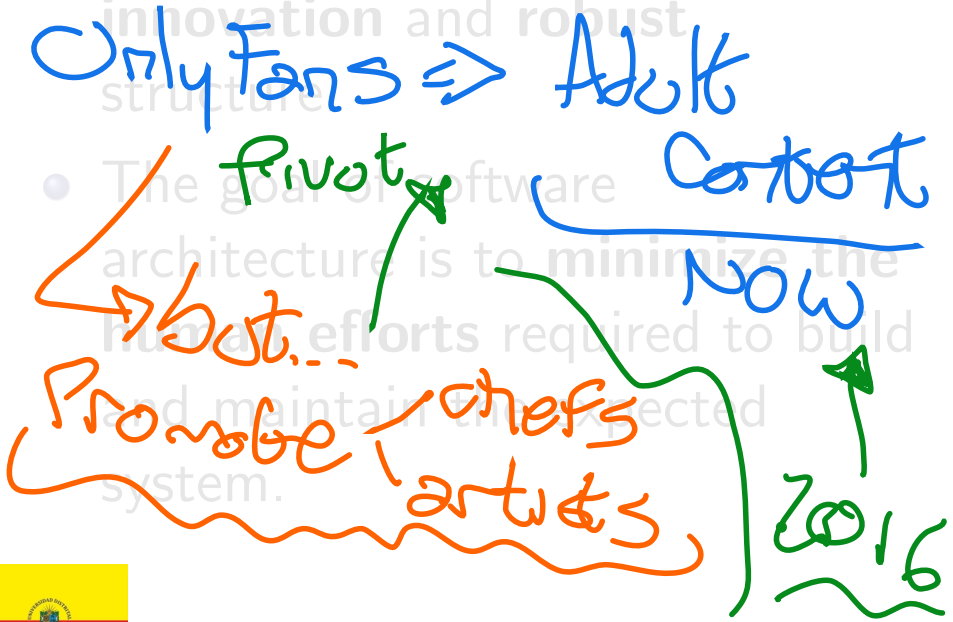


Figure: Prompt: A python developer watching a building architecture draws.

Basics of Software Architecture I

- It is important to **develop** innovative and sophisticated **software** to provide a nice **solution** for **end users needs**.
- Software architecture brings **innovation** and **robust** structure.
- The goal of software architecture is to **minimize the human efforts** required to build and maintain the expected system.

Figure: Prompt: A python developer watching a building architecture draws.

Basics of Software Architecture I

- It is important to **develop** innovative and sophisticated **software** to provide a nice **solution** for **end users needs**.
- Software architecture brings **innovation** and **robust** structure.
- The goal of software architecture is to **minimize the human efforts** required to build and maintain the expected system.

Figure: Prompt: A python developer watching a building architecture draws.

Basics of Software Architecture II

Figure: Prompt: A python developer watching a building architecture draws.

- A software architecture is the **skeleton** for a complete **software system**. It leads the system to be **scalable, reliable, and maintainable**. Also it helps to take better **technical decisions**.
- There are some **software architecture styles**, each one with pros/cons, and specific use cases. However, they try to provide a **reference solution** for a high-level structure of a software system.

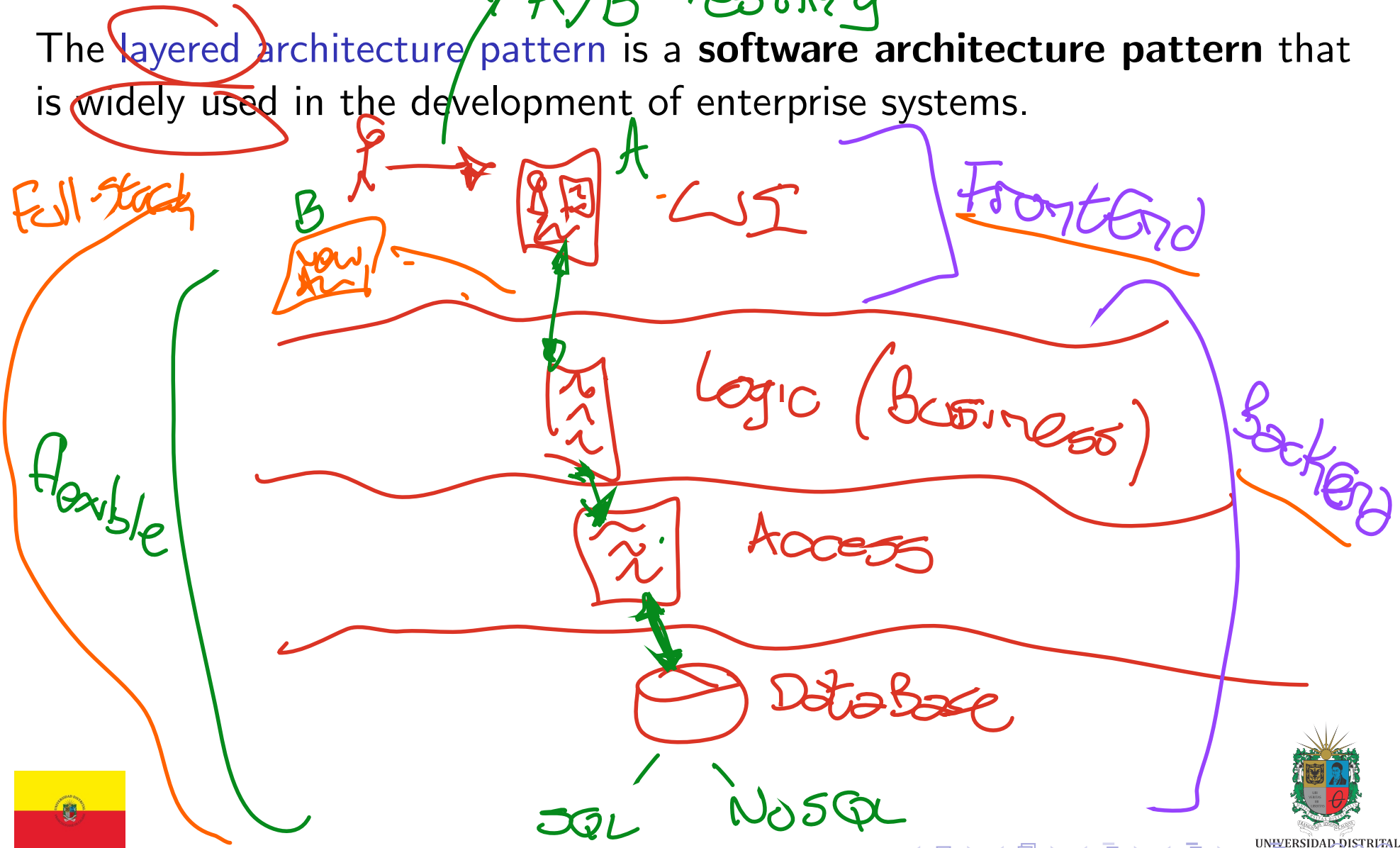
Basics of Software Architecture II

Figure: Prompt: A python developer watching a building architecture draws.

- A software architecture is the **skeleton** for a complete **software system**. It leads the system to be **scalable, reliable, and maintainable**. Also it helps to take better **technical decisions**.
- There are some **software architecture styles**, each one with pros/cons, and specific use cases. However, they try to provide a **reference solution** for a high-level structure of a software system.

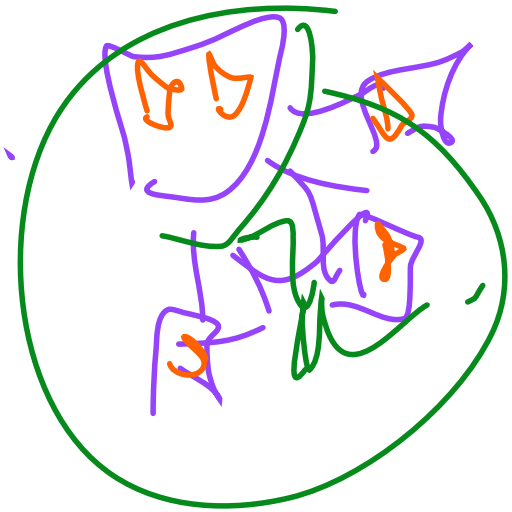
Layered Architecture Pattern

The layered architecture pattern is a **software architecture pattern** that is widely used in the development of enterprise systems.



Software Design Roles

The software architect is the person who is responsible for the **high-level design** of the system. He/she is the person who makes the **technical decisions** and is responsible for the **overall quality** of the system.



Outline

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Object-Oriented Paradigm I

- **Object-oriented** has become one of the most traditional and popular **paradigms** in software development.

Figure: Prompt: Draw several objects sorted by size.

Object-Oriented Paradigm I

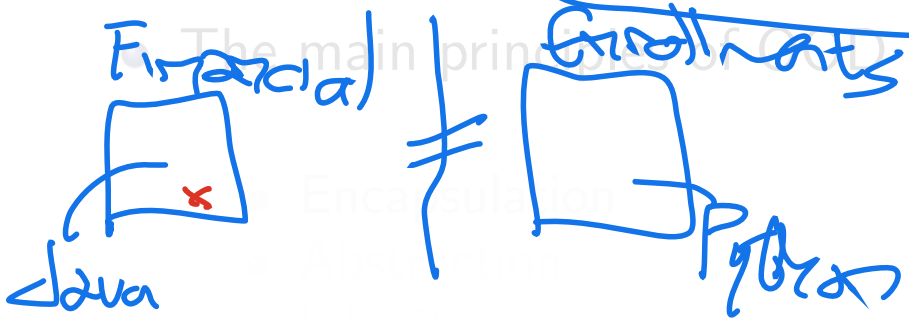
- **Object-oriented** has become one of the most traditional and popular **paradigms** in software development.
- It is based on the concept of **objects**, which can contain data, in the form of **fields** (often known as *attributes* or *properties*), and code, in the form of **procedures** (often known as *methods*).

Figure: Prompt: Draw several objects sorted by size.

Object-Oriented Paradigm II

Figure: Prompt: Draw several objects sorted by size.

- The idea is to design a **system modularly**, and to make it easier to maintain, and to understand. Also the idea is emphasize the **reuse of code**.



Object-Oriented Paradigm II

Figure: Prompt: Draw several objects sorted by size.

- The idea is to design a **system modularly**, and to make it easier to maintain, and to understand. Also the idea is emphasize the **reuse of code**.
- The main principles of OOD are:
  - Encapsulation
  - Abstraction
  - Inheritance
  - Polymorphism

Object-Oriented Paradigm II

Figure: Prompt: Draw several objects sorted by size.

- The idea is to design a **system modularly**, and to make it easier to maintain, and to understand. Also the idea is emphasize the **reuse of code**.
- The main principles of OOD are:
  - Encapsulation
  - Abstraction
  - Inheritance
  - Polymorphism

Object-Oriented Paradigm II

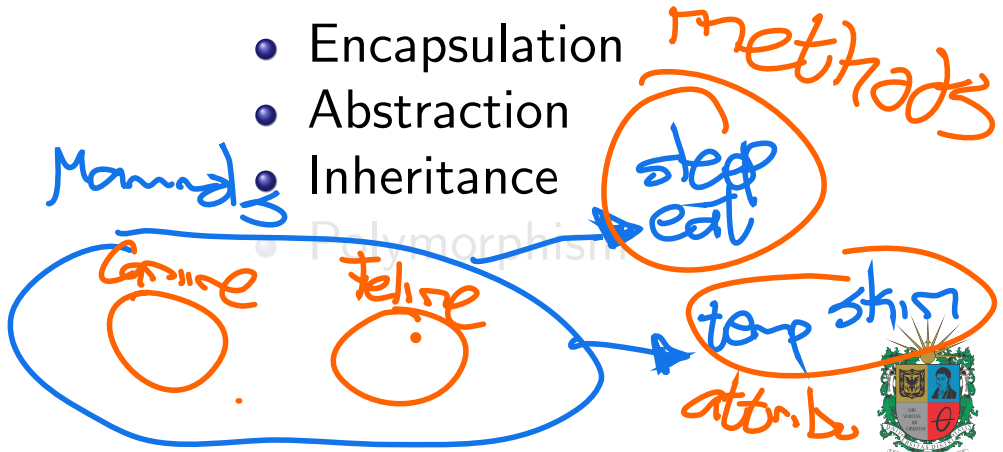
Figure: Prompt: Draw several objects sorted by size.

- The idea is to design a **system modularly**, and to make it easier to maintain, and to understand. Also the idea is emphasize the **reuse of code**.
- The main principles of OOD are:
  - Encapsulation
  - Abstraction
  - Inheritance
  - Polymorphism

Object-Oriented Paradigm II

Figure: Prompt: Draw several objects sorted by size.

- The idea is to design a **system modularly**, and to make it easier to maintain, and to understand. Also the idea is emphasize the **reuse of code**.
- The main principles of OOD are:
  - Encapsulation
  - Abstraction
  - Inheritance
  - Polymorphism



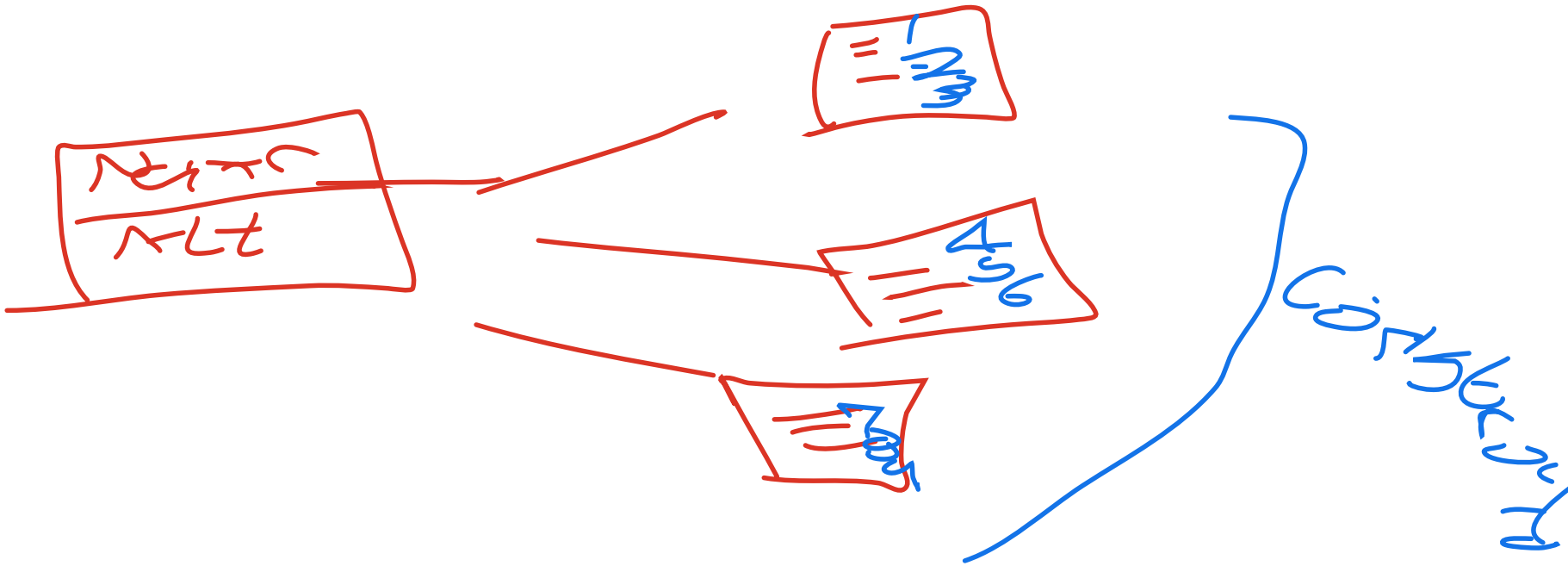
Object-Oriented Paradigm II

Figure: Prompt: Draw several objects sorted by size.

- The idea is to design a **system modularly**, and to make it easier to maintain, and to understand. Also the idea is emphasize the **reuse of code**.
- The main principles of OOD are:
  - Encapsulation
  - Abstraction
  - Inheritance
  - Polymorphism

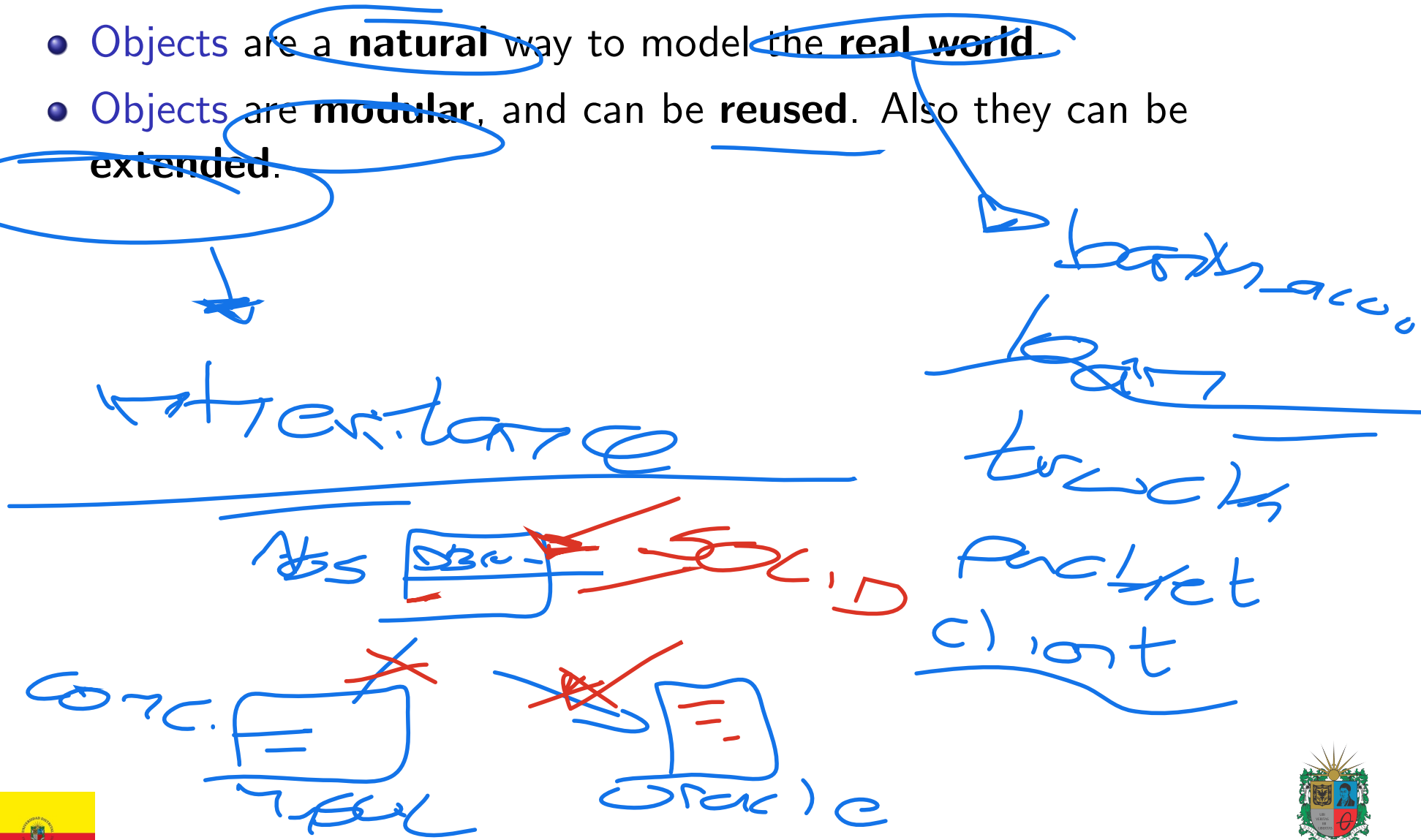
What is a class?

A class is a **blueprint** for creating objects (a particular **data structure**), providing initial values for **state** (member variables or attributes), and implementations of **behavior** (member functions or methods).



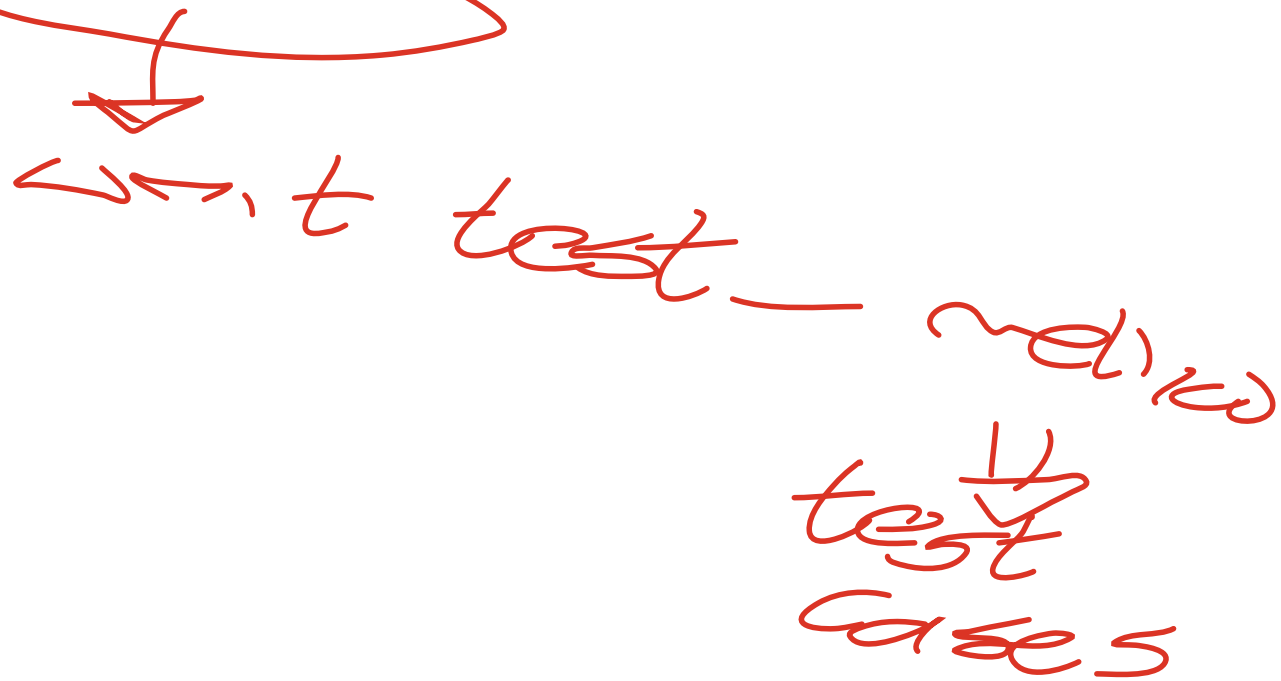
Why Objects?

- Objects are a **natural** way to model the **real world**.
- Objects are **modular**, and can be **reused**. Also they can be **extended**.



Why not Objects?

- Objects are **complex** to understand and to design in big systems.
- Objects are **expensive** in terms of memory and processing.
- Objects sometimes are **difficult** to test.

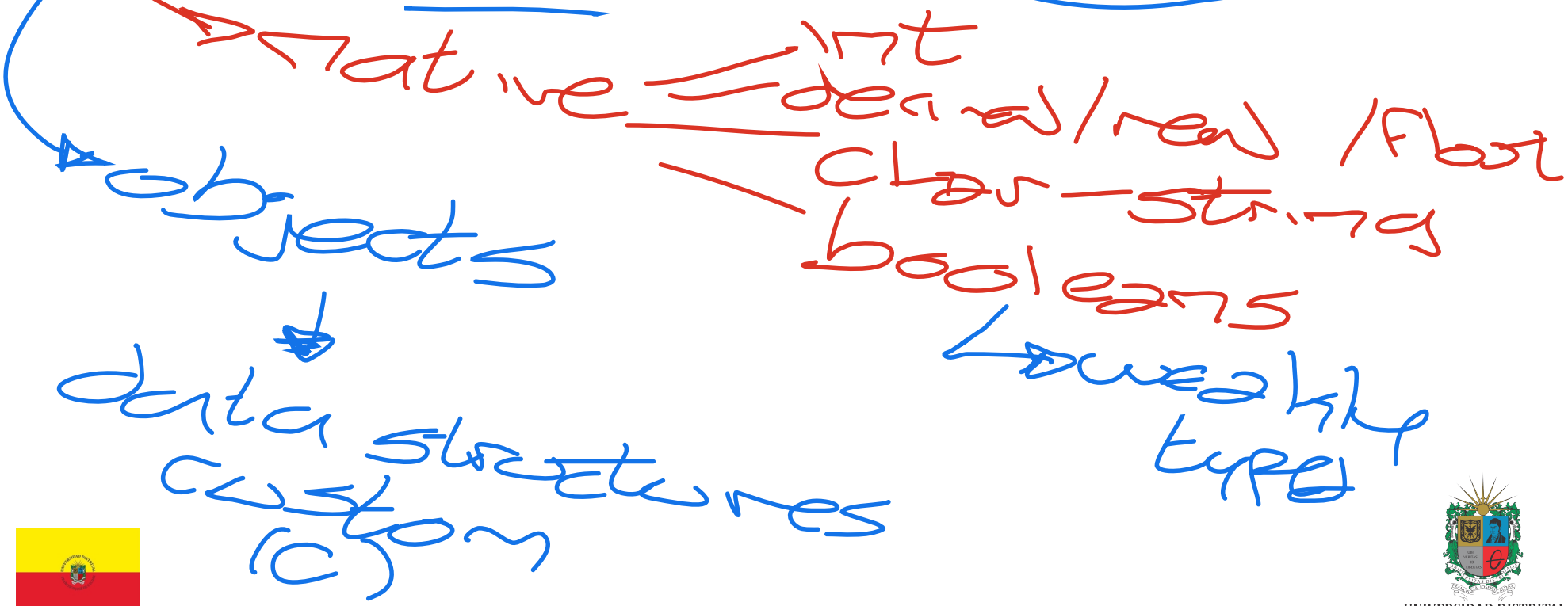


Outline

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Data Types and Abstract Data Types

- A data type is a **classification** that specifies which type of **value** a variable can hold.
- An abstract data type is a **mathematical model** for a certain class of data structures that have similar behavior. It is a **data type** that is defined by its **behavior** from the point of view of a user.



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

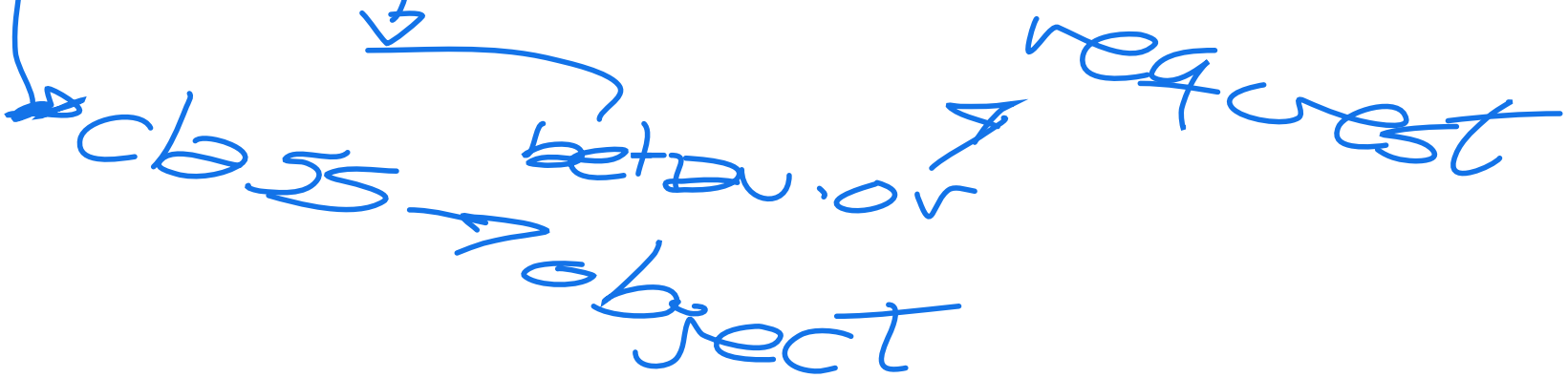
Behaviours: Methods and Functions

- A function is a **procedure** that is defined outside of a **class**.
- A method is a **function** that is associated with a **class**. It is a **procedure** that has access to the **object** and its *data*.

object → actions, behavior ⇒ algorithm

Entities and Duties

- An entity is a **thing** that exists in the **real world** and can be **distinguished** from other things.
- A duty is a **task** that someone is required to perform.
- In object-oriented programming, an **entity** is an **object** and a **duty** is a **method**.



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Software Life Cycle

The software life cycle is a **process** that is used to design, develop, and test **high-quality software**.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Code Review, Pair Programming, and Linters

- A code review is a **systematic examination** of **computer source code**.
- Pair programming is an **agile software development technique** in which two programmers work together at one workstation.
- A lint is a **tool** that is used to **analyze source code** and flag **programming errors**, **bugs**, **stylistic errors**, and **suspicious constructs**.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Code Review, Pair Programming, and Linters

- A code review is a **systematic examination** of **computer source code**.
- Pair programming is an **agile software development technique** in which two programmers work together at one workstation.
- A lint is a **tool** that is used to **analyze source code** and flag **programming errors**, **bugs**, **stylistic errors**, and **suspicious constructs**.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Code Review, Pair Programming, and Linters

- A code review is a **systematic examination** of **computer source code**.
- Pair programming is an **agile software development technique** in which two programmers work together at one workstation.
- A lint is a **tool** that is used to **analyze source code** and flag **programming errors**, **bugs**, **stylistic errors**, and **suspicious constructs**.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Unit Testing and Test-Driven Development

- Unit testing is a **software testing method** by which individual units of source code are tested to determine whether they are fit for use.
- Test-driven development is a **software development process** that relies on the repetition of a very short development cycle.

Software Quality and Metrics

- Software quality is a field of study and practice that describes the **desirable attributes** of **software products**.
- Software metrics are a **measure** of some **property** of a piece of **software** or its **specifications**.

Outline

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Thanks!

Questions?

Repo: *https://github.com/EngAndres/ud-public/tree/main/courses/advanced-programming*

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS