Object-Oriented Programming

Advanced Programming

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

Computer Engineer
Lecturer
Universidad Distrital Francisco José de Caldas

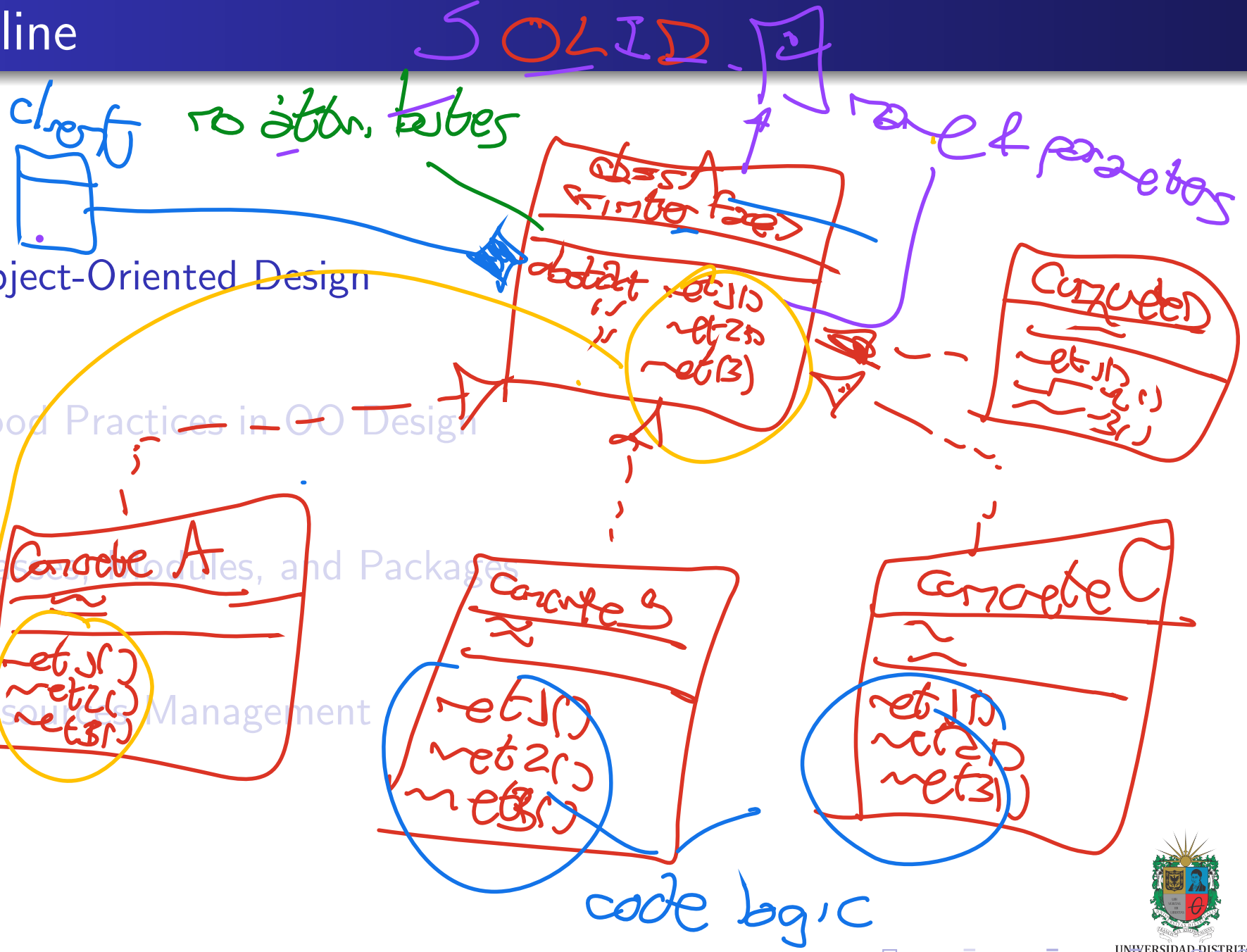
2024-III

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Outline

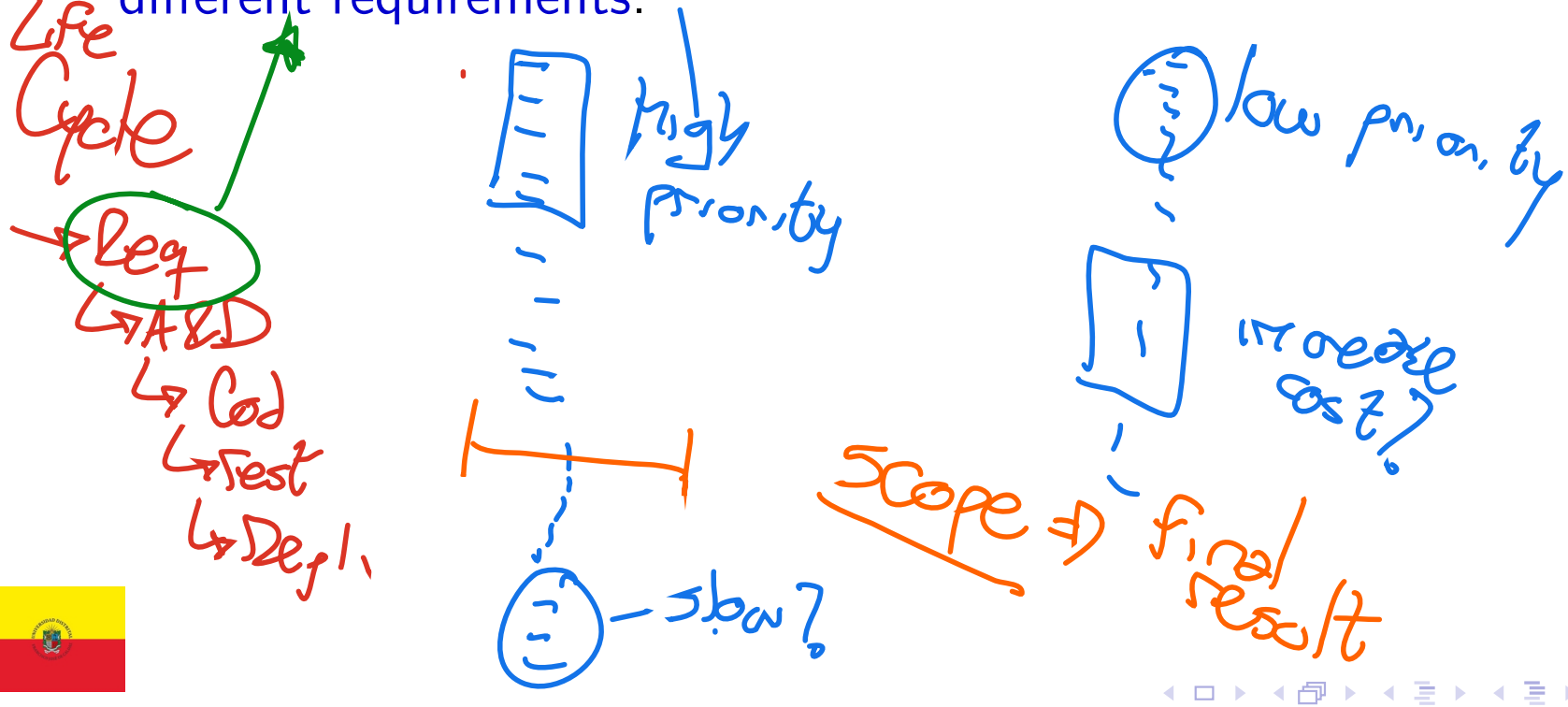
UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Outline



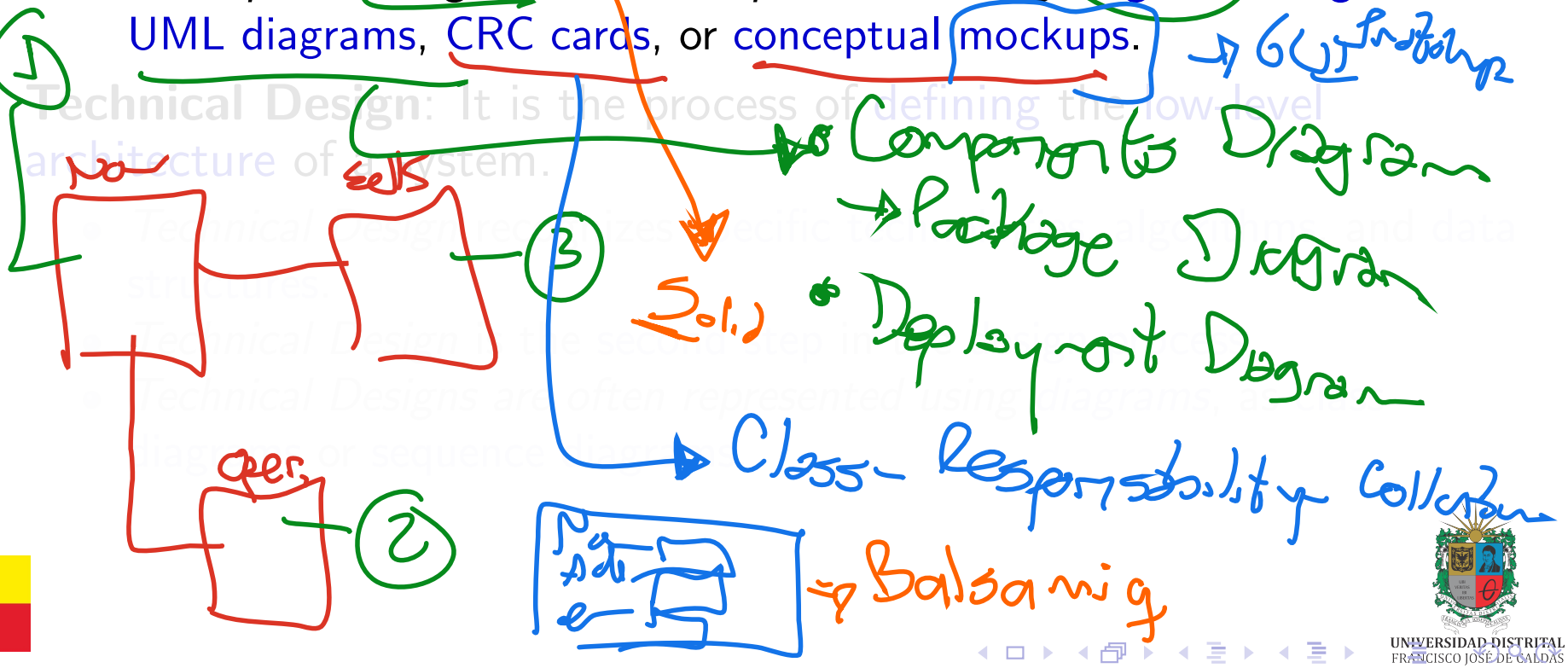
Requirements

- **Requirements Analysis**: It is the process of defining the requirements of a system.
- **Requirements**: They are the functional and non-functional specifications of a system.
- **Trade-offs**: They are the compromises that have to be made between different requirements.



Conceptual Design, Technical Design

- **Conceptual Design**: It is the process of defining the high-level architecture of a system.
  - *Conceptual Design* recognizes appropiate components, relationships, and responsabilities.
  - *Conceptual Design* is the first step in the design process.
  - *Conceptual Designs are often represented using diagrams*, as general UML diagrams, CRC cards, or conceptual mockups.

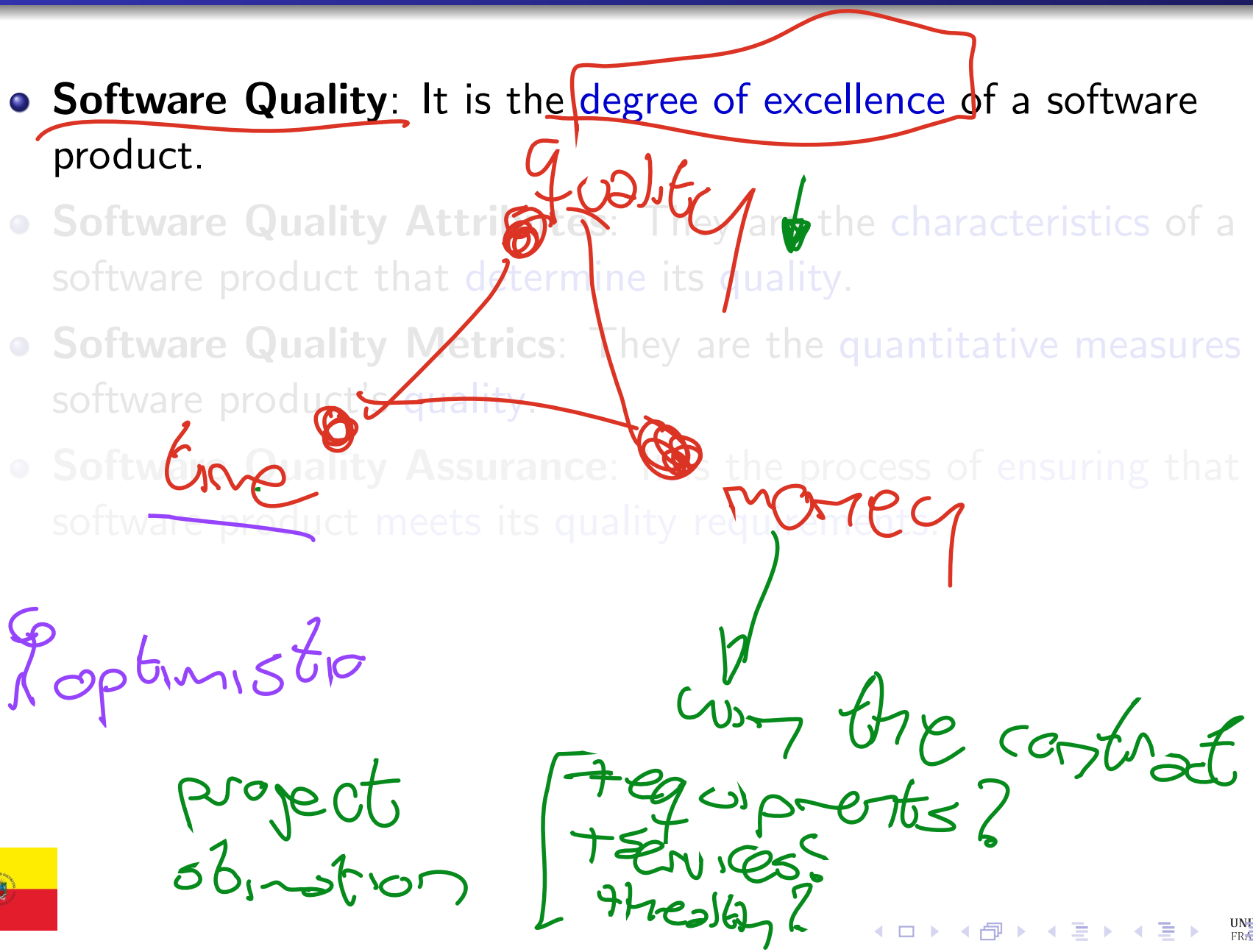


Conceptual Design, Technical Design

- **Conceptual Design**: It is the process of defining the high-level architecture of a system.
  - *Conceptual Design* recognizes appropiate components, relationships, and responsabilities.
  - *Conceptual Design* is the first step in the design process.
  - *Conceptual Designs are often represented using diagrams*, as general UML diagrams, CRC cards, or conceptual mockups.
- **Technical Design**: It is the process of defining the low-level architecture of a system.
  - *Technical Design* recognizes specific technologies, algorithms, and data structures.
  - *Technical Design* is the second step in the design process.
  - *Technical Designs are often represented using diagrams*, as class diagrams or sequence diagrams.

Software Quality

- **Software Quality**: It is the degree of excellence of a software product.
- **Software Quality Attributes**: They are the characteristics of a software product that determine its quality.
- **Software Quality Metrics**: They are the quantitative measures of a software product's quality.
- **Software Quality Assurance**: It is the process of ensuring that a software product meets its quality requirements.



Software Quality

- **Software Quality**: It is the degree of excellence of a software product.
- **Software Quality Attributes**: They are the characteristics of a software product that determine its quality.
- **Software Quality Metrics**: They are the quantitative measures of a software product's quality.
- **Software Quality Assurance**: It is the process of ensuring that a software product meets its quality requirements.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Software Quality

- **Software Quality**: It is the degree of excellence of a software product.
- **Software Quality Attributes**: They are the characteristics of a software product that determine its quality.
- **Software Quality Metrics**: They are the quantitative measures of a software product's quality.
- **Software Quality Assurance**: It is the process of ensuring that a software product meets its quality requirements.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Software Quality

- **Software Quality**: It is the degree of excellence of a software product.
- **Software Quality Attributes**: They are the characteristics of a software product that determine its quality.
- **Software Quality Metrics**: They are the quantitative measures of a software product's quality.
- **Software Quality Assurance**: It is the process of ensuring that a software product meets its quality requirements.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

CRC Cards

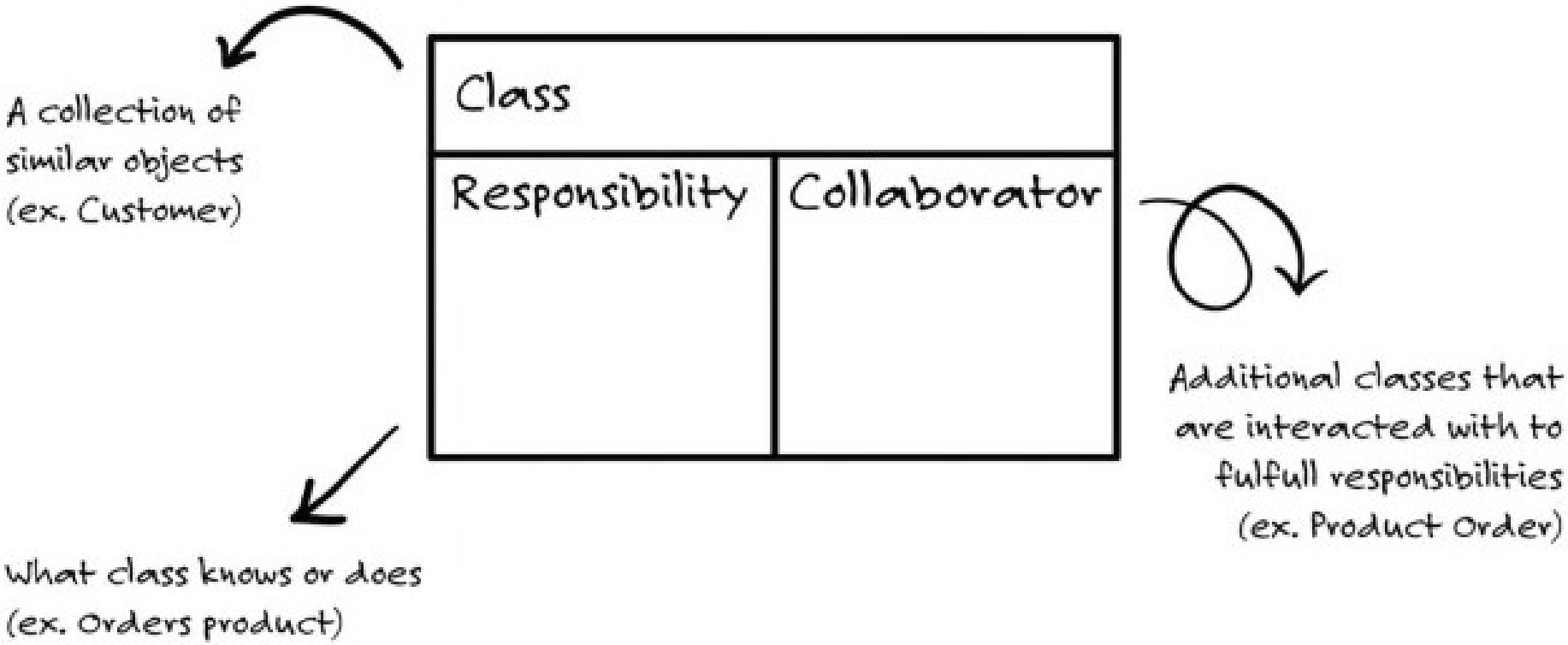


Figure: Prompt: Easy way to design with Classes & Objects.

Basics of Object-Oriented Design 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.

Basics of Object-Oriented Design I

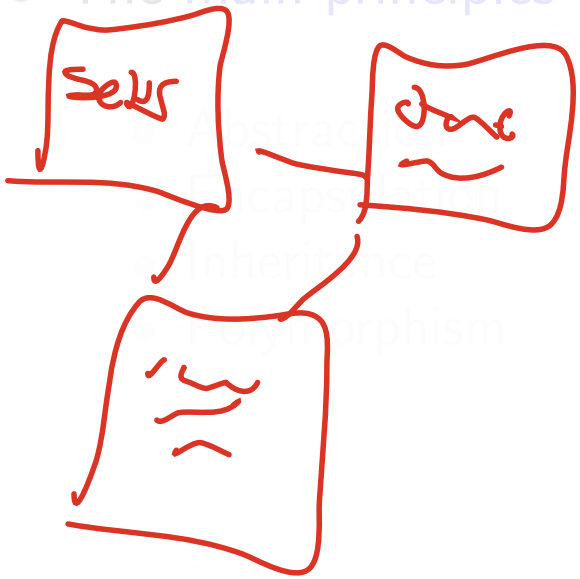
- **Object-oriented** has become one of the most traditional and popular **paradigms** in software development.
- It is based on the concept of **objects**, which can contain data, in the form of **fields** (often known as attributes or properties), and code, in the form of **procedures** (often known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

Figure: Prompt: Draw several objects sorted by size.

Basics of Object-Oriented Design 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:
  - Abstraction
  - Encapsulation
  - Inheritance
  - Polymorphism



Basics of Object-Oriented Design 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:
  - Abstraction
  - Encapsulation
  - Inheritance
  - Polymorphism

Objects Categories

- **Concrete Objects**: They are objects that can be instantiated.
- **Abstract Objects**: They are objects that cannot be instantiated.
- **Active Objects**: They are objects that can perform actions.
- **Passive Objects**: They are objects that store data.
- In general, there are three types of objects in **OOD**:
  - Entity Objects: They represent persistent information, such as the problem space.
  - Boundary Objects: They represent the interface between the system and the external world.
  - Control Objects: They represent the control of the system.

Objects Categories

- **Concrete Objects**: They are objects that can be instantiated.
- **Abstract Objects**: They are objects that cannot be instantiated.
- **Active Objects**: They are objects that can perform actions.
- **Passive Objects**: They are objects that store data.
- In general, there are three `types of objects` in **OOD**:
  - Entity Objects: They represent real-world entities, focus on the problem space.
  - Boundary Objects: They represent the interface between the system and the external world.
  - Control Objects: They represent the control logic of the system.

Objects Categories

- **Concrete Objects**: They are objects that can be instantiated.
- **Abstract Objects**: They are objects that cannot be instantiated.
- **Active Objects**: They are objects that can perform actions.
- **Passive Objects**: They are objects that store data.

Objects Categories

- **Concrete Objects**: They are objects that can be instantiated.
- **Abstract Objects**: They are objects that cannot be instantiated.
- **Active Objects**: They are objects that can perform actions.
- **Passive Objects**: They are objects that store data.
- In general, there are three `types of objects` in **OOD**:
  - Entity Objects: They represent real-world entities, focus on the problem space.
  - Boundary Objects: They represent the interface between the system and the external world.
  - Control Objects: They represent the control logic of the system.

Objects Categories

- **Concrete Objects**: They are objects that can be instantiated.
- **Abstract Objects**: They are objects that cannot be instantiated.
- **Active Objects**: They are objects that can perform actions.
- **Passive Objects**: They are objects that store data.
- In general, there are three `types of objects` in **OOD**:
  - Entity Objects: They represent real-world entities, focus on the problem space.
  - Boundary Objects: They represent the interface between the system and the external world.
  - Control Objects: They represent the control logic of the system.

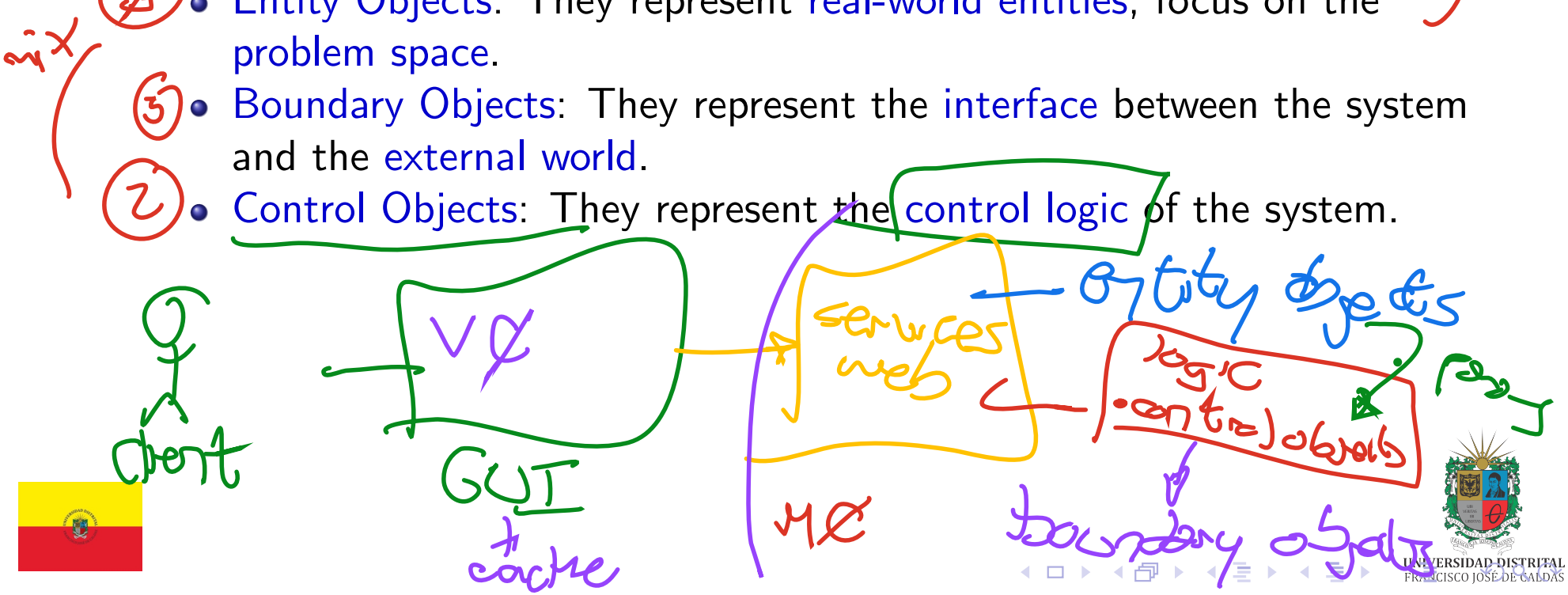
→ DB connection

→ APIs external

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

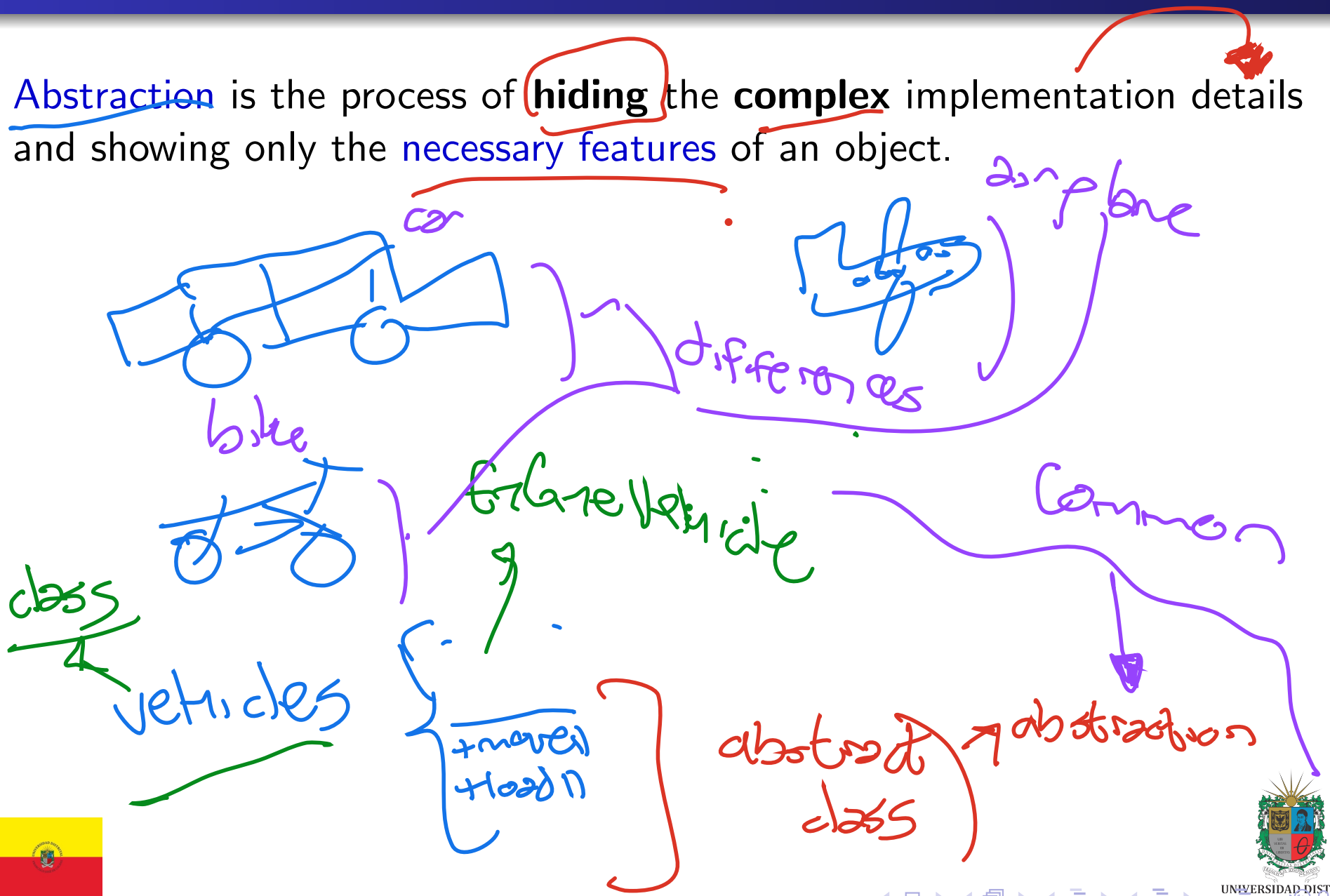
Objects Categories

- **Concrete Objects**: They are objects that can be instantiated.
- **Abstract Objects**: They are objects that cannot be instantiated.
- **Active Objects**: They are objects that can perform actions.
- **Passive Objects**: They are objects that store data.
- In general, there are three `types of objects` in **OOD**:
  - Entity Objects: They represent real-world entities, focus on the problem space.
  - Boundary Objects: They represent the interface between the system and the external world.
  - Control Objects: They represent the control logic of the system.



Abstraction in OOD

Abstraction is the process of **hiding** the **complex** implementation details and showing only the necessary features of an object.



Encapsulation in OOD

Encapsulation is the process of **hiding** the internal state of an object and requiring all interactions to be performed through an object's **methods**.

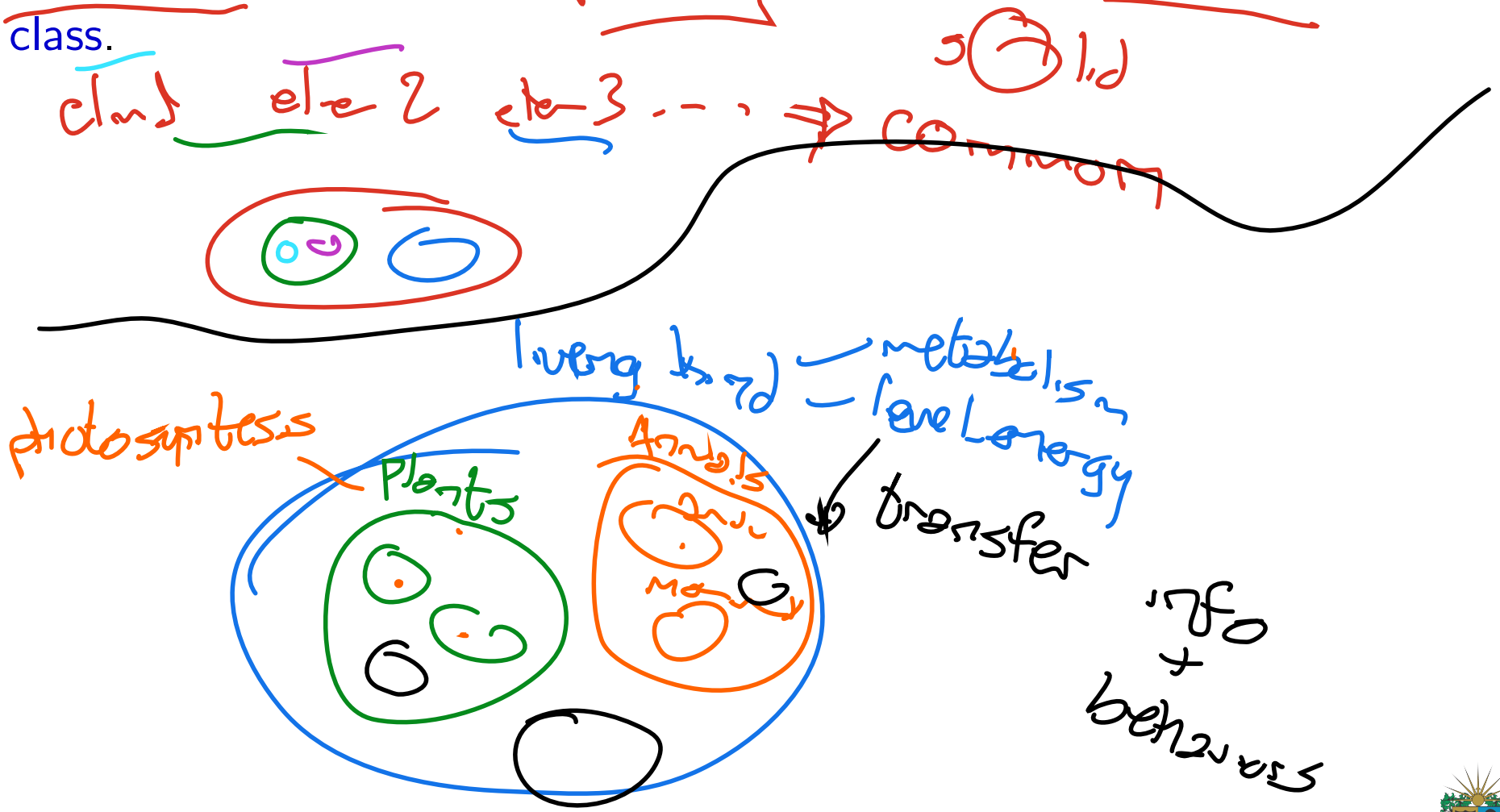
private attributes + getters & setters

```
private int age;

public void setAge(int newAge){
    if (newAge > 0){
        this.age = newAge;
    }
}

public String getCategory(){
    if(this.age<30){
        return "A";
    }else if(this.age<55){
        return "B";
    }else return "C";
}

public bool isAdult(){
    return this.age>=18? true:false;
}
```

data integrity

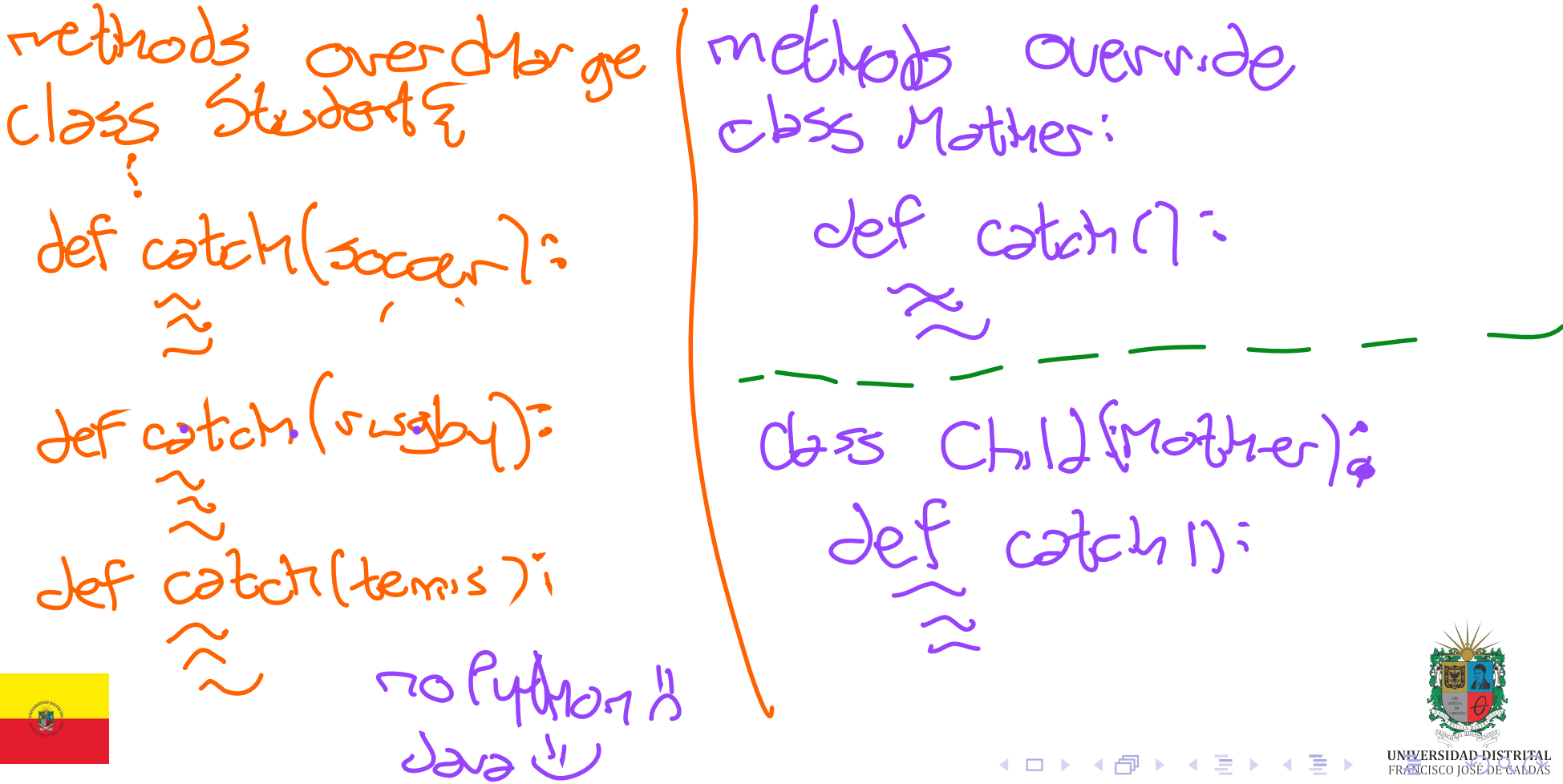
Inheritance in OOD

Inheritance is the process of creating a new class by **extending** an existing class.



Polymorphism in OOD

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

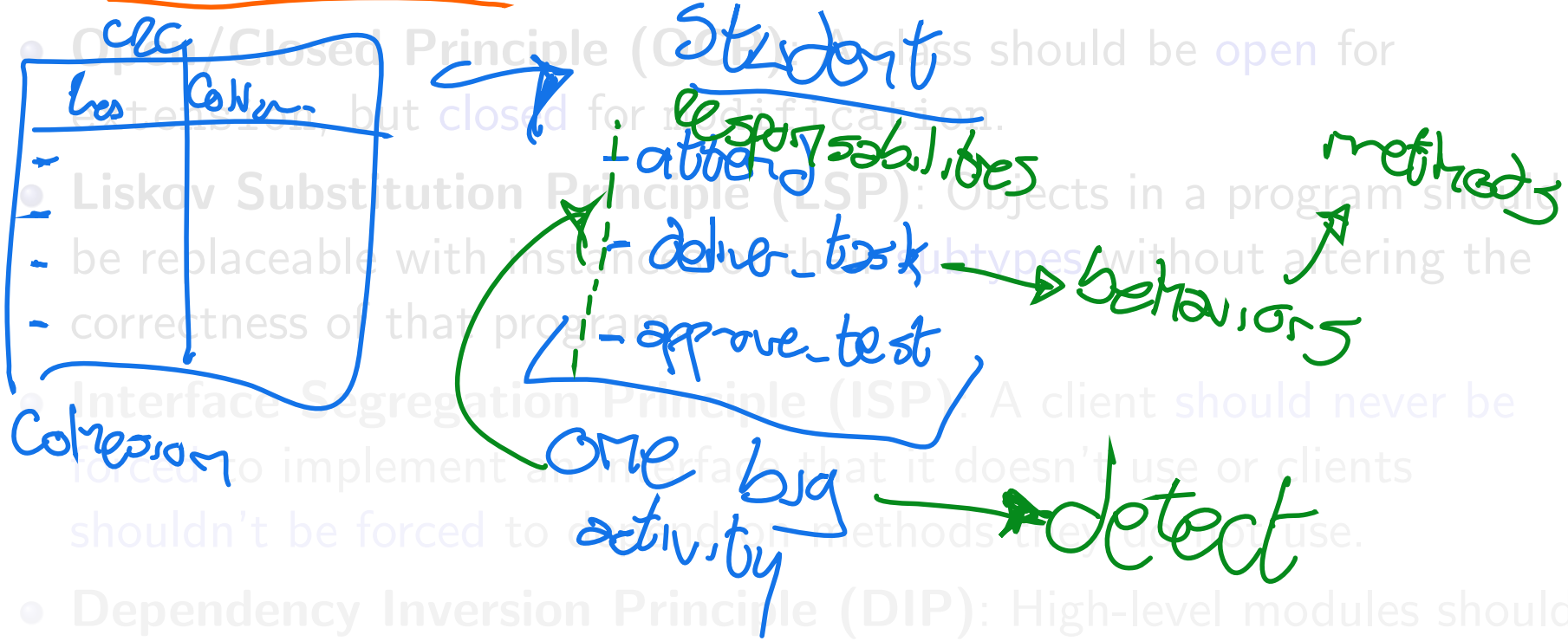


Outline

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

SOLID Principles

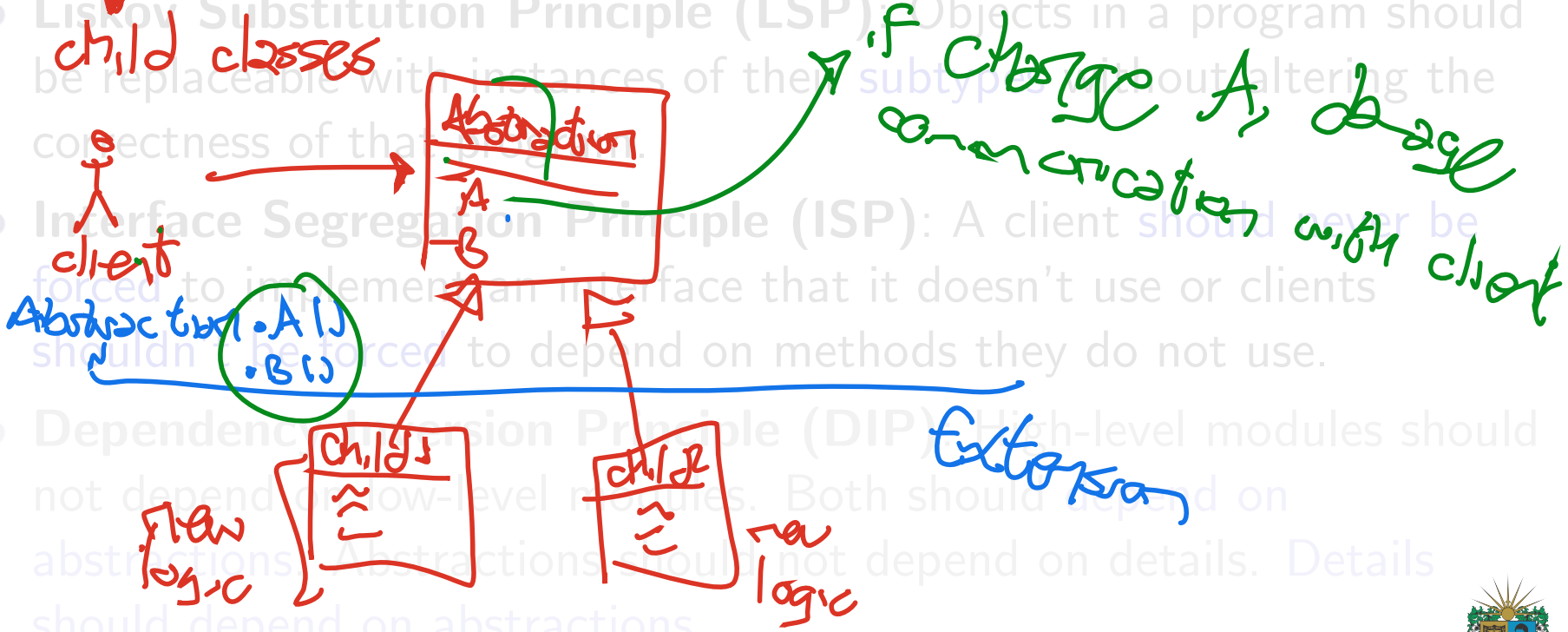
- **Single Responsibility Principle (SRP)**: A class should have only one reason to change.



- **Open/Closed Principle (OCP)**: A class should be open for extension, but closed for modification.
- **Liskov Substitution Principle (LSP)**: Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.
- **Interface Segregation Principle (ISP)**: A client should never be forced to implement an interface that it doesn't use or clients shouldn't be forced to depend on methods they do not use.
- **Dependency Inversion Principle (DIP)**: High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

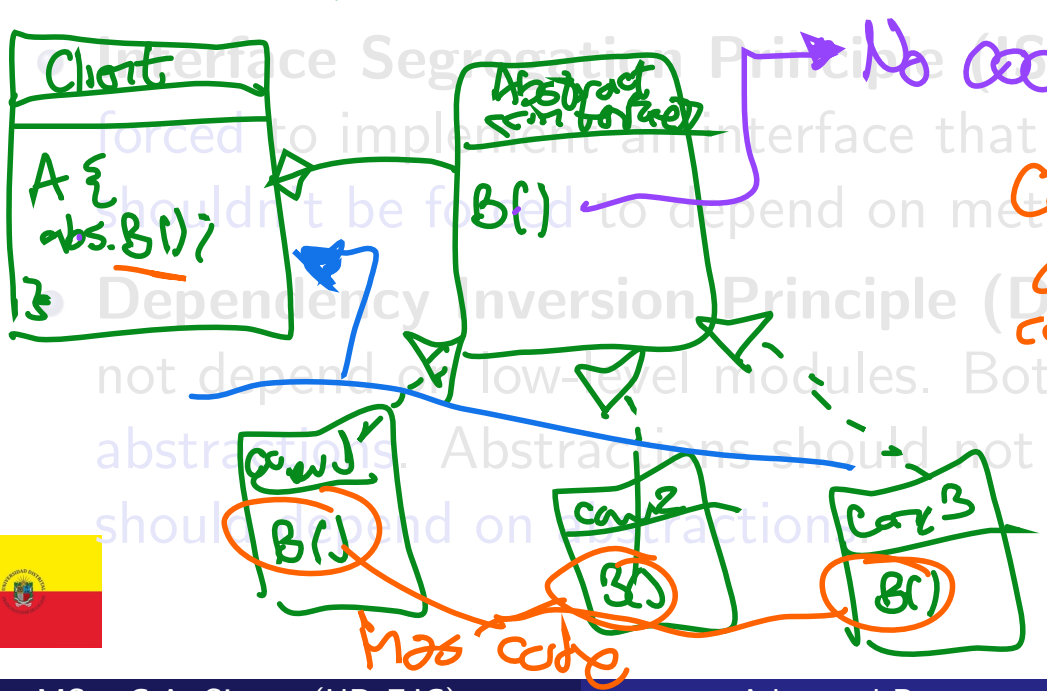
SOLID Principles

- **Single Responsibility Principle (SRP)**: A class should have only one reason to change.
- **Open/Closed Principle (OCP)**: A class should be open for extension, but closed for modification.
- **Liskov Substitution Principle (LSP)**: Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.
- **Interface Segregation Principle (ISP)**: A client should never be forced to implement an interface that it doesn't use or clients shouldn't be forced to depend on methods they do not use.
- **Dependency Inversion Principle (DIP)**: High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.



SOLID Principles

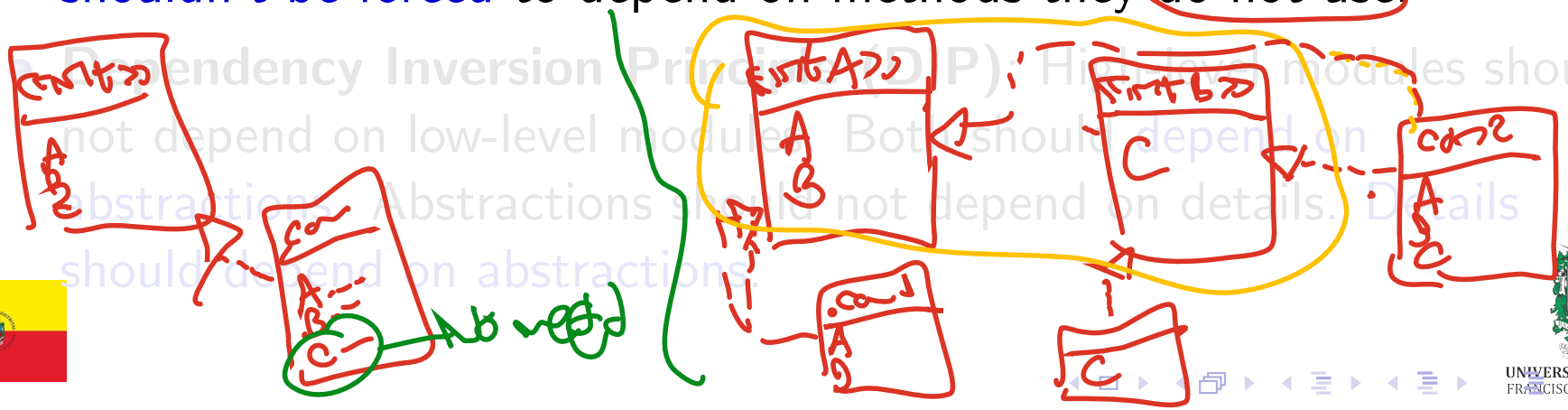
- **Single Responsibility Principle (SRP)**: A class should have only one reason to change.
- **Open/Closed Principle (OCP)**: A class should be open for `extension`, but closed for `modification`.
- **Liskov Substitution Principle (LSP)**: Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.



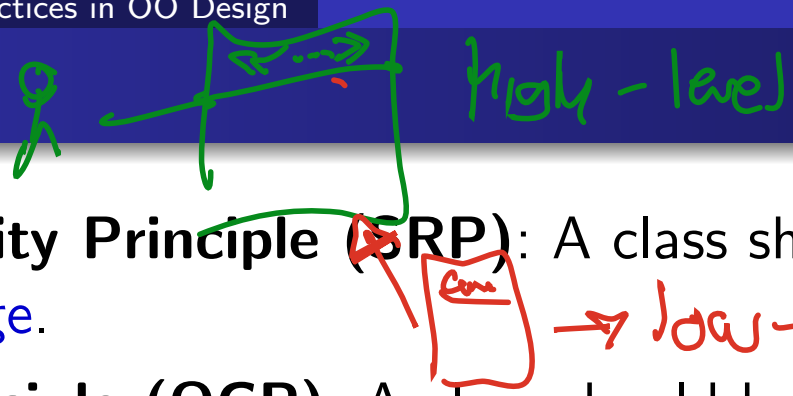
- **Interface Segregation Principle (ISP)**: A client should never be forced to implement an interface that it doesn't use or clients shouldn't be forced to depend on methods they do not use.
- **Dependency Inversion Principle (DIP)**: High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

SOLID Principles

- **Single Responsibility Principle (SRP)**: A class should have only one reason to change.
- **Open/Closed Principle (OCP)**: A class should be open for `extension`, but closed for `modification`.
- **Liskov Substitution Principle (LSP)**: Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.
- **Interface Segregation Principle (ISP)**: A client should never be forced to implement an interface that it doesn't use or clients shouldn't be forced to depend on methods they do not use.
- **Dependency Inversion Principle (DIP)**: High level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.



SOLID Principles



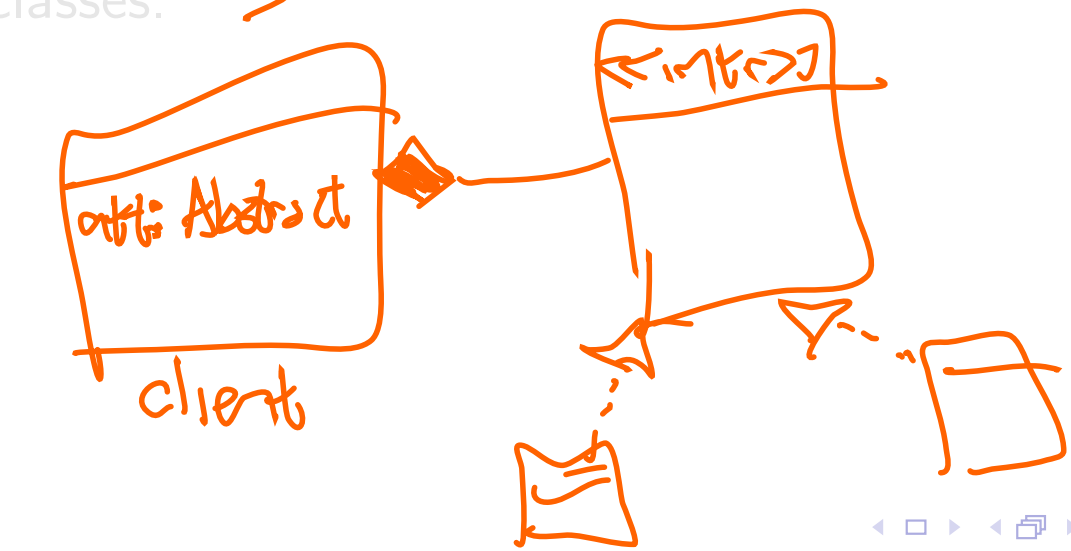
- **Single Responsibility Principle (SRP)**: A class should have only one reason to change.
- **Open/Closed Principle (OCP)**: A class should be open for `extension`, but closed for `modification`.
- **Liskov Substitution Principle (LSP)**: Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.
- **Interface Segregation Principle (ISP)**: A client should never be forced to implement an interface that it doesn't use or clients shouldn't be forced to depend on methods they do not use.
- **Dependency Inversion Principle (DIP)**: High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

↳ child implements abstractions

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

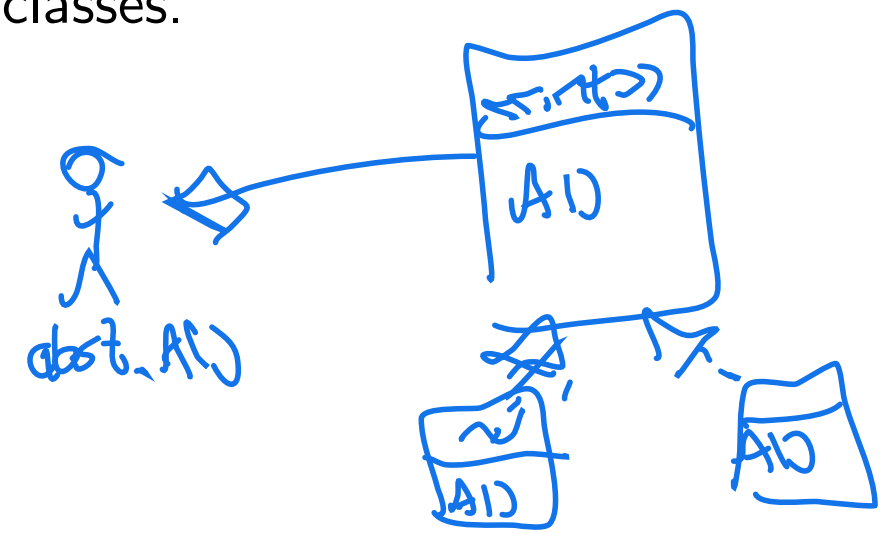
Good Practices

- **Composition over Inheritance**: Inheritance should be used only when there is a clear relationship between the base class and the derived class. In other cases, composition should be used. Inheritance is a powerful tool, but it is not always the best tool for the job. Inheritance is a way to achieve polymorphism, but it is not the only way to achieve polymorphism.
- **Code to Interfaces, not Implementations**: This principle is about designing your classes so that they depend on interfaces rather than concrete classes.



Good Practices

- **Composition over Inheritance**: Inheritance should be used only when there is a clear relationship between the base class and the derived class. In other cases, composition should be used. Inheritance is a powerful tool, but it is not always the best tool for the job. Inheritance is a way to achieve polymorphism, but it is not the only way to achieve polymorphism.
- **Code to Interfaces, not Implementations**: This principle is about designing your classes so that they depend on interfaces rather than concrete classes.



Outline

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Modules in Python

Modules are Python files that consist of Python code. They can define functions, classes, and variables.

- **Importing Modules**: To use a module, you have to import it using the `import` statement.
- **Creating Modules**: To create a module, you just have to save the code you want in a file with the file extension `.py`.
- **Built-in Modules**: Python has a set of built-in modules that you can use without installing them.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Modules in Python

Modules are Python files that consist of Python code. They can define functions, classes, and variables.

- **Importing Modules**: To use a module, you have to import it using the `import` statement.
- **Creating Modules**: To create a module, you just have to save the code you want in a file with the file extension `.py`.
- **Built-in Modules**: Python has a set of built-in modules that you can use without installing them.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Modules in Python

Modules are Python files that consist of Python code. They can define functions, classes, and variables.

- **Importing Modules**: To use a module, you have to import it using the `import` statement.
- **Creating Modules**: To create a module, you just have to save the code you want in a file with the file extension `.py`.
- **Built-in Modules**: Python has a set of built-in modules that you can use without installing them.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Packages in Python

Packages are a way of structuring `Python's module namespace` by using *dotted module names*.

- **Creating Packages**: To create a package, you just have to create a directory with an `__init__.py` file.
- **Importing Packages**: To import a package, you can use the `import` statement.
- **Virtual Environments**: Virtual environments are a way of creating isolated environments for your **Python projects**.
- **Third-party Packages**: Python has a set of third-party packages that you can install using `pip`.

Packages in Python

Packages are a way of structuring `Python's module namespace` by using *dotted module names*.

- **Creating Packages**: To create a package, you just have to create a directory with an `__init__.py` file.
- **Importing Packages**: To import a package, you can use the `import` statement.
- **Virtual Environments**: Virtual environments are a way of creating isolated environments for your **Python projects**.
- **Third-party Packages**: Python has a set of third-party packages that you can install using `pip`.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Packages in Python

Packages are a way of structuring `Python's module namespace` by using *dotted module names*.

- **Creating Packages**: To create a package, you just have to create a directory with an `__init__.py` file.
- **Importing Packages**: To import a package, you can use the `import` statement.
- **Virtual Environments**: Virtual environments are a way of creating isolated environments for your **Python projects**.
- **Third-party Packages**: Python has a set of third-party packages that you can install using `pip`.

Packages in Python

Packages are a way of structuring `Python's module namespace` by using *dotted module names*.

- **Creating Packages**: To create a package, you just have to create a directory with an `__init__.py` file.
- **Importing Packages**: To import a package, you can use the `import` statement.
- **Virtual Environments**: Virtual environments are a way of creating isolated environments for your **Python projects**.
- **Third-party Packages**: Python has a set of third-party packages that you can install using `pip`.

Packages in Java

Packages are a way of structuring `Java's namespace` by 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 `Java's namespace` by 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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Packages in Java

Packages are a way of structuring `Java's namespace` by 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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Packages in Java

Packages are a way of structuring `Java's namespace` by 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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Outline

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Resources in Computation

- **Memory Management**:It is the process of `managing` computer memory.
- **File Management**: It is the process of `managing` computer files.
- **Network Management**: It is the process of `managing` computer networks.
- **Process Management**: It is the process of `managing` computer processes.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Resources in Computation

- **Memory Management**:It is the process of `managing` computer memory.
- **File Management**: It is the process of `managing` computer files.
- **Network Management**: It is the process of `managing` computer networks.
- **Process Management**: It is the process of `managing` computer processes.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Resources in Computation

- **Memory Management**:It is the process of `managing` computer memory.
- **File Management**: It is the process of `managing` computer files.
- **Network Management**: It is the process of `managing` computer networks.
- **Process Management**: It is the process of `managing` computer processes.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Resources in Computation

- **Memory Management**:It is the process of `managing` computer memory.
- **File Management**: It is the process of `managing` computer files.
- **Network Management**: It is the process of `managing` computer networks.
- **Process Management**: It is the process of `managing` computer processes.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Objects Serialization

- **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.
- **Pickling**: It is the process of serializing an object in `Python`.
- **Unpickling**: It is the process of deserializing an object in `Python`.
- **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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Objects Serialization

- **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.
- **Pickling**: It is the process of serializing an object in `Python`.
- **Unpickling**: It is the process of deserializing an object in `Python`.
- **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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Objects Serialization

- **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.
- **Pickling**: It is the process of serializing an object in `Python`.
- **Unpickling**: It is the process of deserializing an object in `Python`.
- **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.

Objects Serialization

- **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.
- **Pickling**: It is the process of serializing an object in `Python`.
- **Unpickling**: It is the process of deserializing an object in `Python`.
- **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.

Objects Serialization

- **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.
- **Pickling**: It is the process of serializing an object in `Python`.
- **Unpickling**: It is the process of deserializing an object in `Python`.
- **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.

Memory Management

- **Garbage Collection**: It is the process of automatically *reclaiming memory* that is no longer in use.
- **Reference Counting**: It is a technique used by `Python` to manage memory.
- **Memory Profiling**: It is the process of analyzing the memory usage of a program.
- **Memory Leaks**: They are a common problem in programming where memory is allocated but never deallocated.
- **Memory Management Tools**: There are several tools available for memory management in `Python`.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Memory Management

- **Garbage Collection**: It is the process of automatically *reclaiming memory* that is no longer in use.
- **Reference Counting**: It is a technique used by `Python` to manage memory.
- **Memory Profiling**: It is the process of analyzing the memory usage of a program.
- **Memory Leaks**: They are a common problem in programming where memory is allocated but never deallocated.
- **Memory Management Tools**: There are several tools available for memory management in `Python`.

Memory Management

- **Garbage Collection**: It is the process of automatically *reclaiming memory* that is no longer in use.
- **Reference Counting**: It is a technique used by `Python` to manage memory.
- **Memory Profiling**: It is the process of analyzing the memory usage of a program.
- **Memory Leaks**: They are a common problem in programming where memory is allocated but never deallocated.
- **Memory Management Tools**: There are several tools available for memory management in `Python`.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Memory Management

- **Garbage Collection**: It is the process of automatically *reclaiming memory* that is no longer in use.
- **Reference Counting**: It is a technique used by `Python` to manage memory.
- **Memory Profiling**: It is the process of analyzing the memory usage of a program.
- **Memory Leaks**: They are a common problem in programming where memory is allocated but never deallocated.
- **Memory Management Tools**: There are several tools available for memory management in `Python`.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Memory Management

- **Garbage Collection**: It is the process of automatically *reclaiming memory* that is no longer in use.
- **Reference Counting**: It is a technique used by `Python` to manage memory.
- **Memory Profiling**: It is the process of analyzing the memory usage of a program.
- **Memory Leaks**: They are a common problem in programming where memory is allocated but never deallocated.
- **Memory Management Tools**: There are several tools available for memory management in `Python`.

Threds, 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.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Threds, 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.

Threds, 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.

Threds, 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.
- **MapReduce**: It is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster.
- **Hadoop**: It is an open-source software framework for storing and processing large data sets.
- **Spark**: It is an open-source software framework for processing large data sets.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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.
- **MapReduce**: It is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster.
- **Hadoop**: It is an open-source software framework for storing and processing large data sets.
- **Spark**: It is an open-source software framework for processing large data sets.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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.
- **MapReduce**: It is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster.
- **Hadoop**: It is an open-source software framework for storing and processing large data sets.
- **Spark**: It is an open-source software framework for processing large data sets.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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.
- **MapReduce**: It is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster.
- **Hadoop**: It is an open-source software framework for storing and processing large data sets.
- **Spark**: It is an open-source software framework for processing large data sets.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

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.
- **MapReduce**: It is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster.
- **Hadoop**: It is an open-source software framework for storing and processing large data sets.
- **Spark**: It is an open-source software framework for processing large data sets.

Outline

Thanks!

Questions?

Repo:
github.com/engandres/ud-public/tree/main/courses/advanced-programming

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS