Review OF OOP And UML

Presentation By Parsa Attaran
Instructor: Dr.Mojtaba Vahidi Asl
1404 Ap Fall

TABLE OF CONTENTS

01 Why OOP

02 Constructors

03 Methods

04 UML

01

Why OOP?

INTRODUCTION

Object-oriented programming (**OOP**) is a programming paradigm based on the concept of “objects”, which can contain data, in the form of fields or attributes, and code, in the form of methods or functions. A feature of objects is an object’s methods that can access and often modify the data fields of the object with which they are associated (objects have a notion of “this” or “self”). In OOP, computer programs are designed by making them out of objects that interact with one another. OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

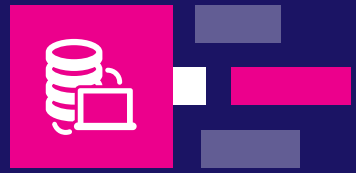
1. Everything Is An Object.
2. Objects communicate by sending and receiving messages (in terms of objects).
3. Objects have their own memory (in terms of objects).
4. Every object is an instance of a class (which must be an object).
5. The class holds the shared behavior for its instances (in the form of objects in a program list)

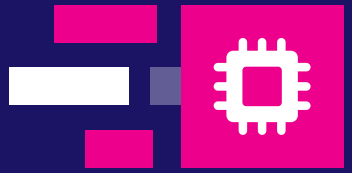
Alan Kay

OOP

OOP can make software development more modular

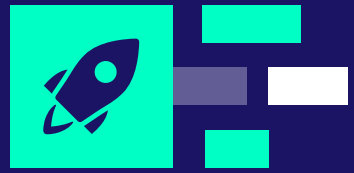
OOP systems can be easily scaled from small to large applications, provided that their design quality supports such growth





Which can make it easier to upgrade and update

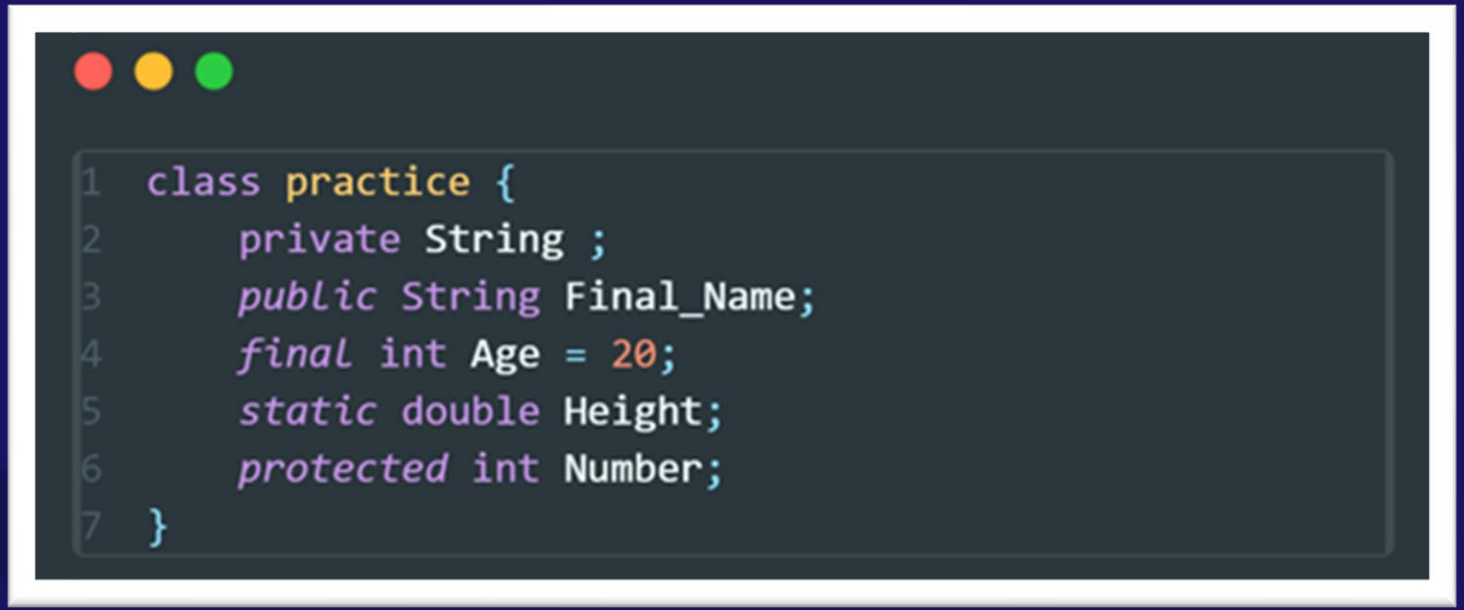
It is very easy to partition the work in a project



Class Attribute

- **Java class attributes** are the variables that are bound in a class
- A class attribute defines the state of the class during program execution
- A class attribute is accessible within class methods by default.

Example



```
class practice {
    private String ;
    public String Final_Name;
    final int Age = 20;
    static double Height;
    protected int Number;
}
```

Access Modifiers Review

Public

Private

Protected

02

Methods

Class Methods

- A method in Java is a set of instructions that can be called for execution using the method name
- Getters and setters are used to protect your data

Example

```
1       private String ;
2       public String Final_Name;
3       final int Age = 20;
4       static double Height;
5       protected int Number;
6       double Change_Grade(double Grade){
7           return Grade*2;
8       }
9       String add(String Name, String Final_Name){
10          return Name + " " + Final_Name;
11      }
```

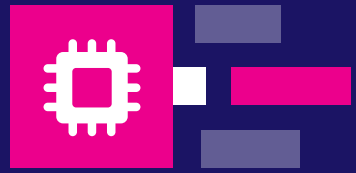
03

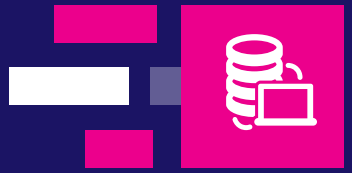
Constructors

Constructors

A constructor is a special method in a class that initializes new objects.

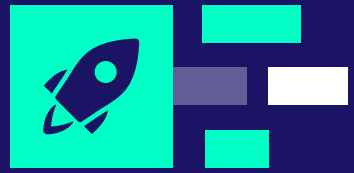
OOP systems can be easily upgraded from small to large systems





In most programming languages, the constructor has the same name as the class.(such as JAVA)

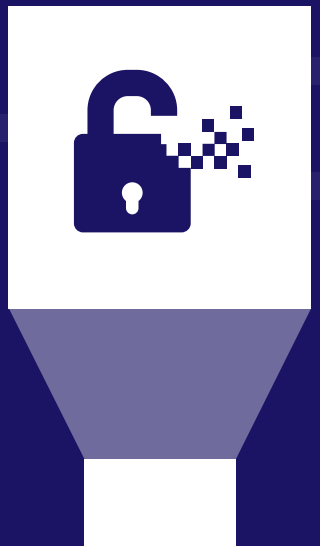
It is very easy to partition the work in a project



What are types of constructors

No Arguments

Default

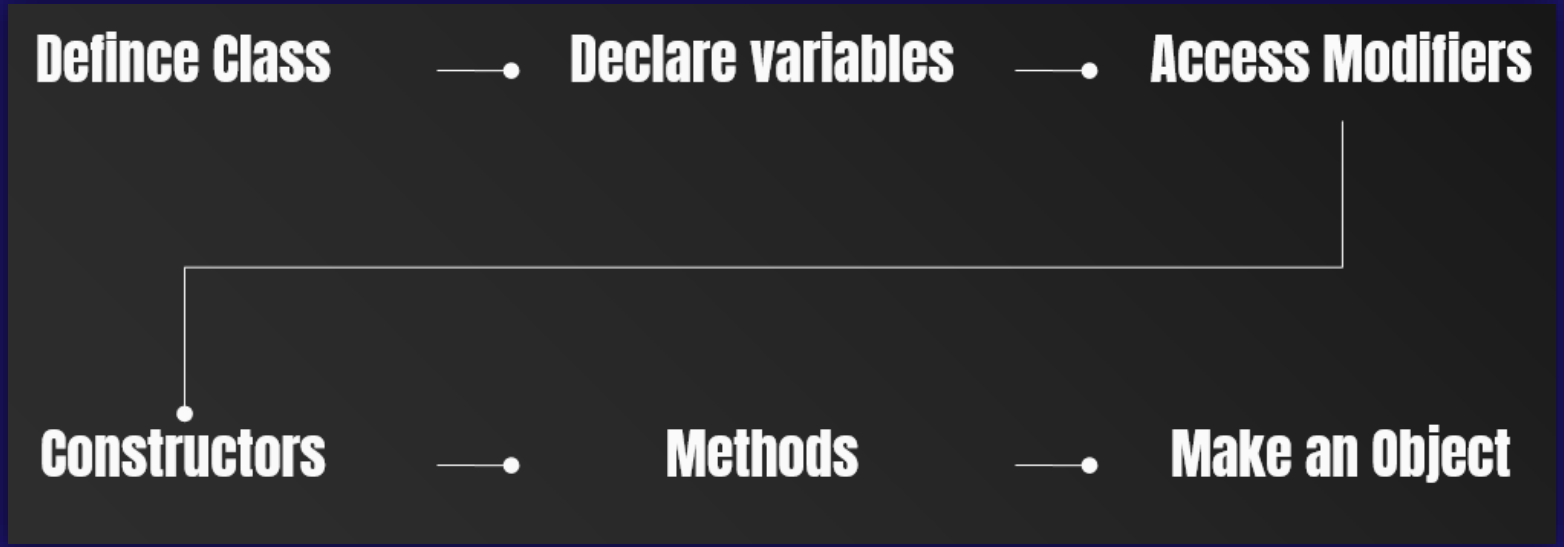


Parametrized

Example

```
1       practice (String n){
2           Name = n;
3       }
4
5       practice(String Name , String Final_Name , double Grade){
6           this.Grade = Grade;
7           this.Final_Name=Final_Name;
8           this.Name = Name;
9       }
10
11      practice(String Name , String Final_Name , int Age){
12          this(Name);
13      }
```

Sequence


Defince Class
Declare variables
Access Modifiers
Constructors
Methods
Make an Object

04

UML Diagram

UML

1

The UML Class diagram is a graphical notation (Unified Modeling Language)

2

Is used to construct and visualize object oriented systems

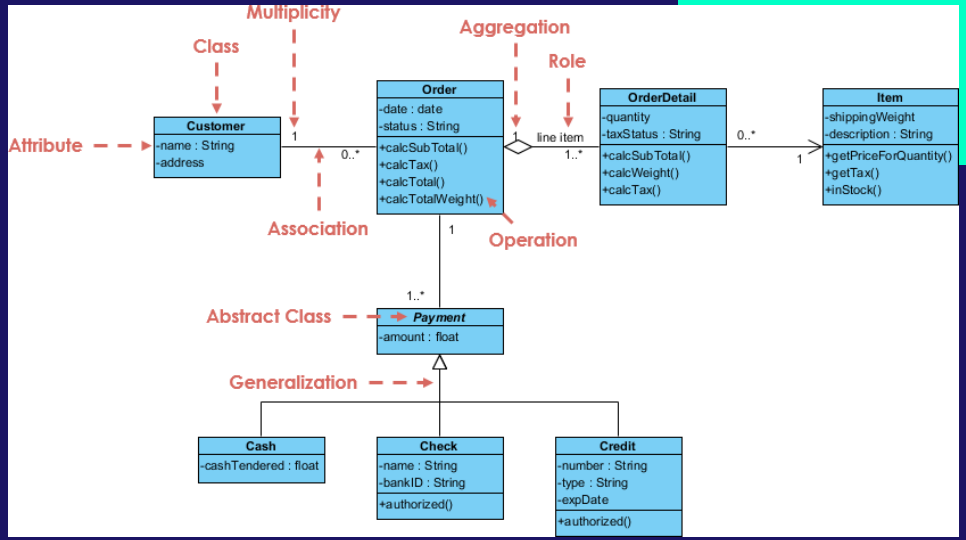
3

Is a type of static structure diagram

4

Describes the structure of a system

Why do we need UML?

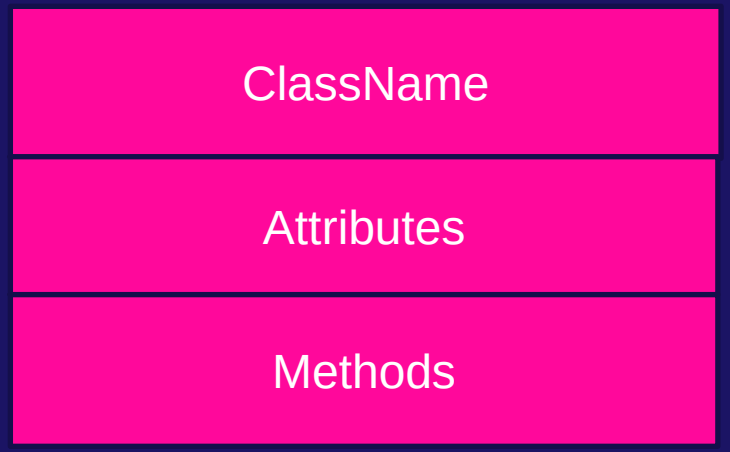


UML provides a standardized way to visualize the design of a system

How to show classes with UML

Info Shown

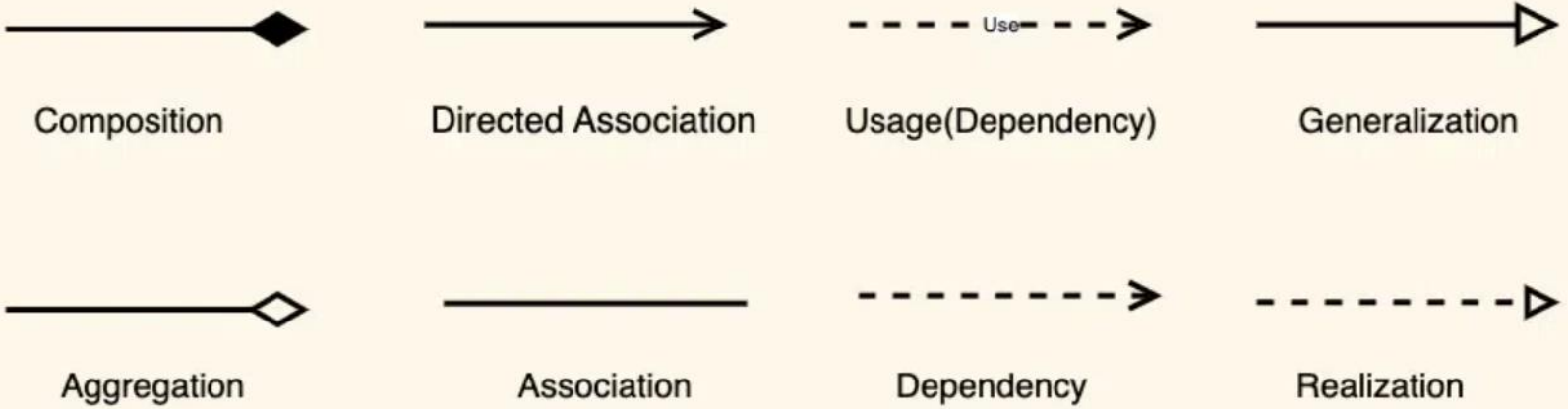
- Variable Names
- Access modifiers
- Methods and Their Arguments



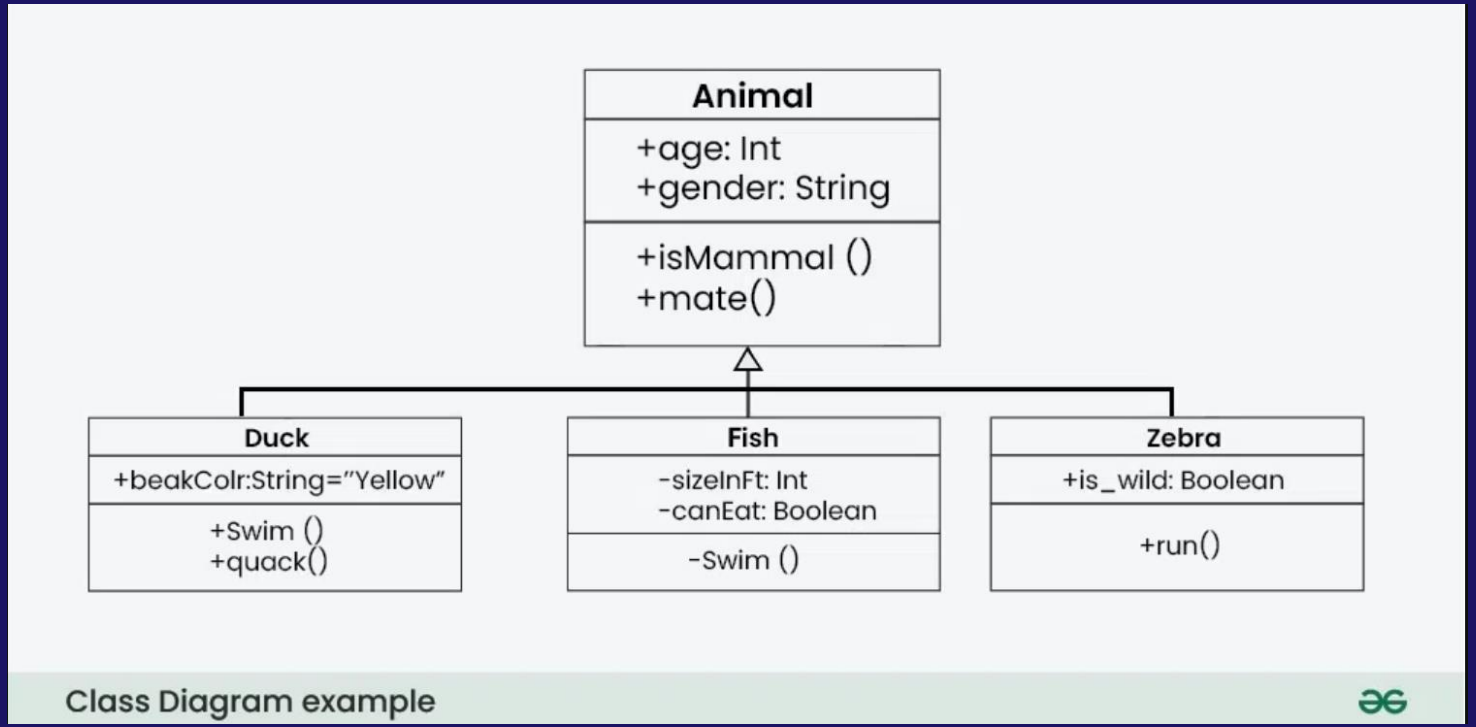
Important Marks

Public	Private	Protected
+	-	#
Package	Derived	Static
~	/	_

Class Diagram Relationships



An example


Animal
+age: Int
+gender: String
+isMammal ()
+mate()
Duck
+beakColr:String="Yellow"
+Swim ()
+quack()
Fish
-sizeInFt: Int
-canEat: Boolean
-Swim ()
Zebra
+is_wild: Boolean
+run()
Class Diagram example

THANKS !

Do you have any questions?
parsa.attaran84@gmail.com
@In_depthSci

CREDITS: This presentation template was created by Slidesgo, incluiding icons by Flaticon, and infographics & images by Freepik.