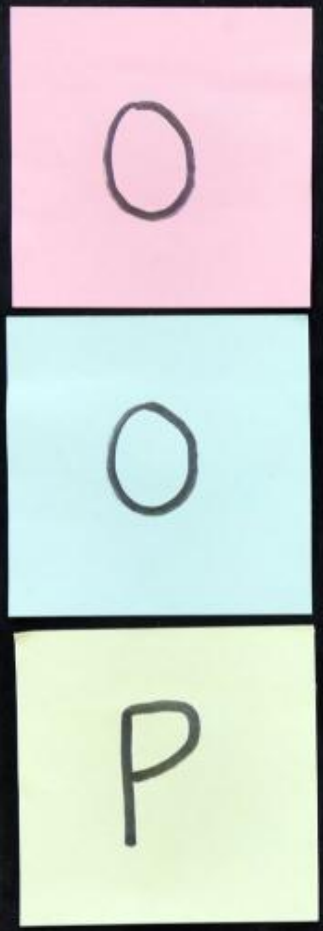
OBJECT ORIENTED PROGRAMMING IN

software engineering



Object Oriented Programming

TABLE OF CONTENTS

“I made up the term ‘object-oriented’, and I can tell you I didn’t have C++ in mind.”

—ALAN KAY

01

INTRODOCTION OF OOP

INTRODUCTION

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

CONCEPTS

OBJECT-ORIENTED PROGRAMMING

It’s the closest planet to the Sun and the smallest one in the Solar System

FUNCTIONAL PROGRAMMING

Venus has a beautiful name and is the second planet from the Sun

02

HISTORY OF OOP

OOP TIMELINE

1

1940

Low level languages like machine code and assembly

2

1950

first popular high-level languages
(Fortran) && (ALGOL)

3

1962-1963

Ivan Sutherland's seminal Sketchpad

4

1965

first programming language recognized as "object oriented"
Simula

5

1983

OOP come whit C++

03.

ADVANTAGES OF OOP

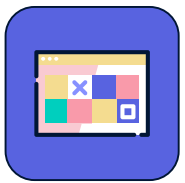
FEATURES OF OOP

RE-USABILITY

It means reusing some facilities rather than building them again and again. This is done with the use of a class.

CODE MAINTENANCE

This feature is more of a necessity for any programming languages; it helps users from doing re-work in many ways.



DESIGN BENEFITS

Object-Oriented Programs forces the designers to have a long and extensive design phase.

FEATURES OF OOP

PROBLEMS SOLVING

Decomposing a complex problem into smaller chunks or discrete components is a good practice.

DATA REDUNDANCY

Data redundancy is one of the greatest advantages of OOP.

SECURITY

We can controll accses data and methods.

EASY TROUBLESHOOTING

Working with OOP language, you will know where to look for. This is the advantage of using encapsulation in OOP

04.

SOLID AND DESIGN PATTERNS

SOLID

SINGLE RESPONSIBILITY

A class should have one and only one reason to change, meaning that a class should have only one job.

OPEN/CLOSED

Objects or entities should be open for extension but closed for modification.

LISKOV SUBSTITUTION

Let q(x) be a property provable about objects of x of type T. Then q(y) should be provable for objects y of type S where S is a subtype of T.

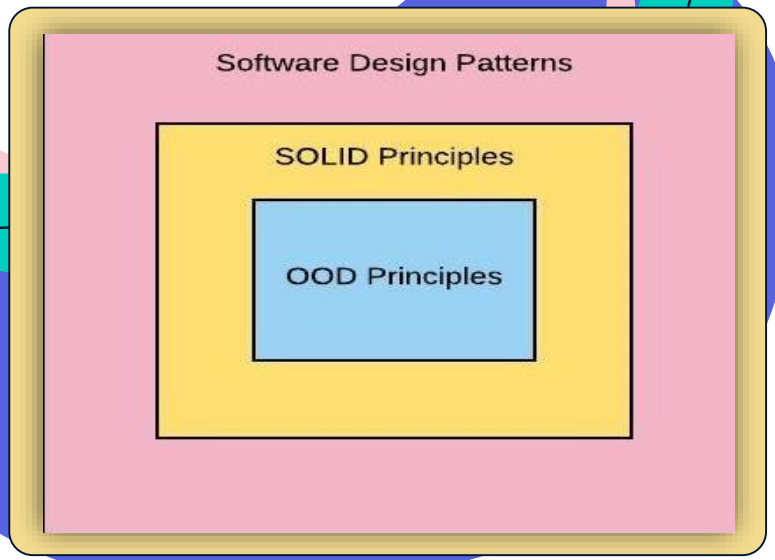
INTERFACE SEGREGATION

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

Entities must depend on abstractions, not on concretions. It states that the high-level module must not depend on the low-level module, but they should depend on abstractions.

DESIGN PATTERNS



Design patterns are typical solutions to common problemsin software design. Each pattern is like a blueprintthat you can customize to solve a particulardesign problem in your code.

THANKS!

FASA UNIVERSITY

MOHSEN SALARI

9711312

software engineering

RESOURCES

RESOURCES

- https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design
- https://www.educba.com/advantages-of-oop/
- https://kajalrawal.medium.com/solid-design-principles-82fc4bdebb8