DESIGN PATTERN

RIHAM AHAMED ABDUL RAHEEM
HND COMPUTING IDM

Contents

WHAT IS OOP

OOPs are closely related to the real-world entities, and can be used effectively to represent real-world entities

Object-oriented programming (OOP) is a programming paradigm that organizes software design around data, rather than functions and logic. An object is a data field with its own set of properties and behavior.

An object-oriented program's structure also makes it useful for collaborative development, where projects are organized into groups. Code reusability, scalability, and efficiency are further advantages of OOP. (S.Gillies, 2021)

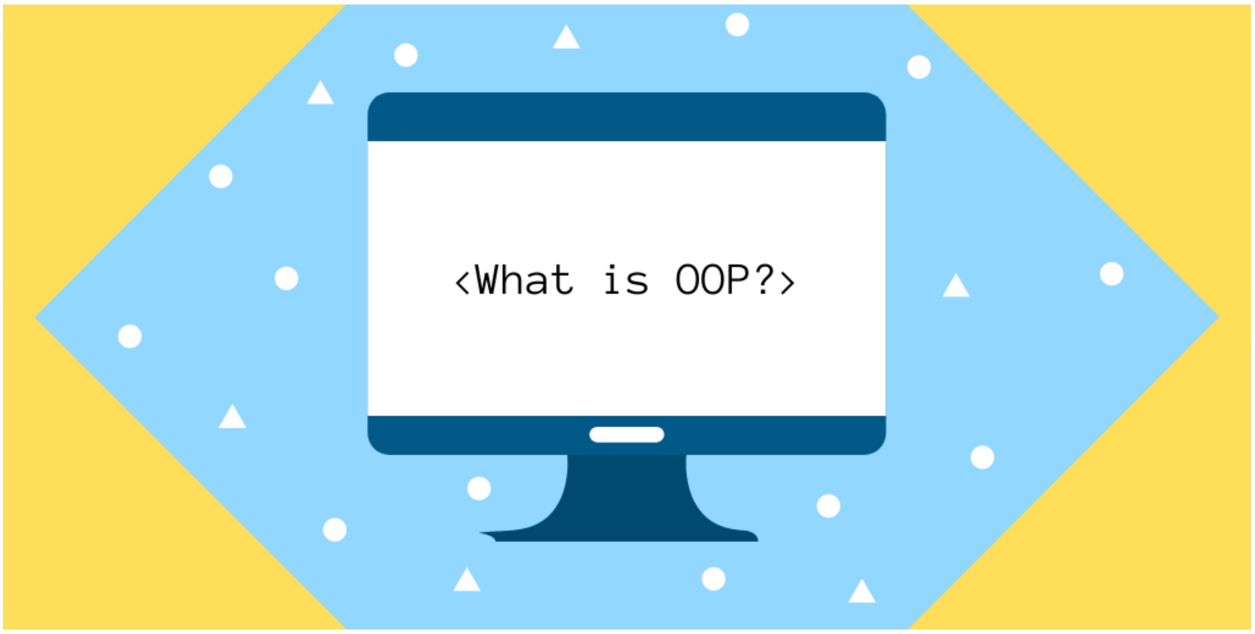

Figure 1: What is OOP?

STRUCTURE OF OBJECT-ORIENTED PROGRAMMING

Object

The description is the sole object defined when a class is created for the first time. Objects are instances of a class that have been constructed with specified data. Real-world items or abstract entities can be represented by objects. (S.Gills, 2021)

Ex: A tree is an Object

Class

Individual objects, properties, and methods are based on classes, which are user-defined data types. (S.Gills, 2021)

Ex: Mango Tree, Apple Tree, Orange Tree etc... are Class of objects mentioned above

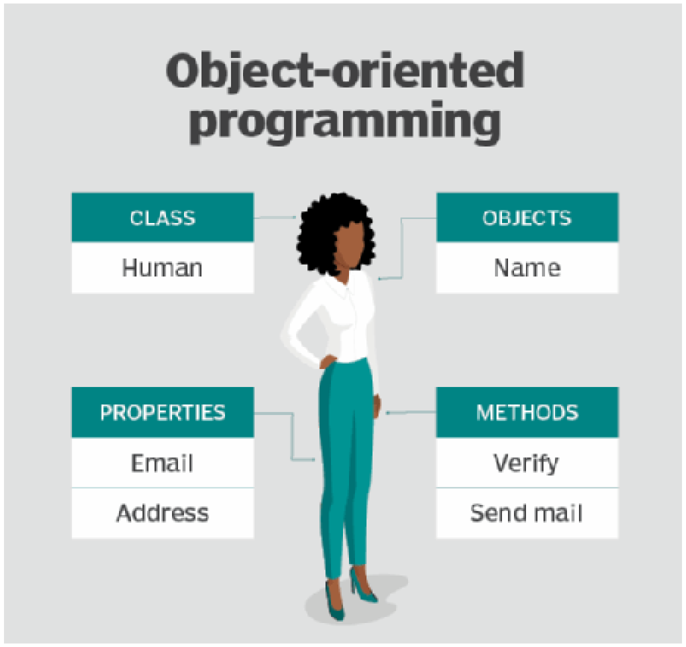

Figure 2: Example of human Object, class, Attributes and methods

Attributes/Properties

The state of an object is represented via attributes, which are defined in the class template. Data will be saved in the attributes field of objects. The class's characteristics are the class's own.

Ex: Tree planting Date, Leafe shape, Scientific name are different attributes of every class tree.

Methods

Methods are functions specified within a class that explain an object's actions. In class declarations, each method begins with a reference to an instance object. In addition, instance methods refer to the subroutines that make up an object. Methods are used by programmers to ensure reusability and to keep functionality contained to a single object at a time. (S.Gills, 2021)

Ex: Every function are called method in the respect of **Protecting Oxygen()** is a one of the method

OBJECT-ORIENTED PROGRAMMING PRINCIPALS OR PILLARS

Encapsulation, Abstraction, Inheritance, and polymorphism are seen as key pillars in object-oriented programming.

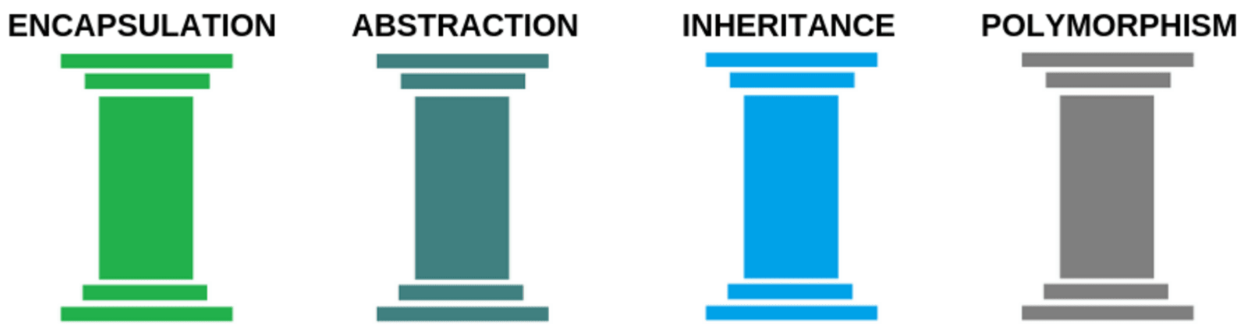

Figure 3: Pillars of OOP

Encapsulation

Encapsulation This concept asserts that an object contains all critical information and only a subset of it is accessible. Each object's implementation and state are kept private within a specified class. This class is not accessible to other objects, and they do not have the authority to alter it. They can only use a limited set of public functions and methods. This data-hiding feature allows for more program flexibility.

Abstraction

Abstraction Internal mechanisms that are relevant for the usage of other objects are only shown by objects, and any extraneous implementation code is hidden. The functionality of the derived class can be expanded. This notion can make it easier for developers to make adjustments or additions over time.

Inheritance

Inheritance Code from other classes can be reused by classes. Developers may establish relationships and subclasses between items, allowing them to reuse similar logic while preserving a distinct hierarchy. This feature of OOP encourages a more complete data examination, shortens development time, and provides greater correctness.

Polymorphism

Polymorphism. Objects are made to share behaviors and come in a variety of shapes and sizes. The software will figure out which meaning or usage is required for each execution of an object from a parent class, avoiding the requirement for duplication of code. After that, a child class is generated, which expands the parent class's capabilities. Different sorts of objects can flow via the same interface thanks to polymorphism. (S.Gills, 2021)

Overloading
In OOP, method overloading is a type of polymorphism. Polymorphism allows objects or methods to behave in a variety of ways depending on how they are utilized. Method overloading is one way in which methods function differently depending on the types and quantity of arguments they receive.

When you have two methods with the same name but distinct signatures, you have overloading (or arguments). We can have two or more methods with the same name in a class. The number and type of parameters supplied as arguments to overloaded methods are used to classify them. The compiler will generate an error if we try to declare more than one method with the same name and the same number of parameters. (Singhal, 2019)

Overriding

Overriding is a feature that allows a subclass or child class to offer a customized implementation of a function that is already given by one of its super-classes or parent classes in any object-oriented programming language. When a subclass's method has the same name, parameters or signature, and return type(or sub-type) as a method in its super-class, the subclass's method is said to override the super-method. class's (geeksforgeeks, 2019)

WHAT IS DESIGN PATTERN

Design Patterns, by definition, are reusable solutions to often encountered issues (in the context of software design). Design patterns began as best practices that could be applied to comparable challenges in various contexts over and over again.

Originally published with c++ and smaltalk code samples, design patterns are very popular in Java and C# (oodesign, 2021)

Design Patterns have two main usages in software development.

- Common platform for developers
- Best Practices

TYPES OF DESIGN PATTERN

The Design Patterns Elements of Reusable Object-Oriented Software, which are divided into three major categories: creational, structural, and behavioral patterns.

Creational design pattern

It's all about class instantiation in these design patterns. This pattern may be further subdivided into patterns for creating classes and patterns for creating objects. While class-creation patterns make good use of inheritance in the instantiation process, object-creation patterns make good use of delegation.

- Abstract Factory
  Creates an instance of several families of classes
- Builder
  Separates object construction from its representation
- Factory Method
  Creates an instance of several derived classes
- Object Pool
  Avoid expensive acquisition and release of resources by recycling objects that are no longer in use

- Prototype

  A fully initialized instance to be copied or cloned
- Singleton

  A class of which only a single instance can exist

Structural design pattern

It's all about Class and Object composition in these design patterns. Inheritance is used to construct interfaces in structural class-creation patterns. Structural object-patterns define how to put objects together to get additional functionality.

- Adapter

  Match interfaces of different classes
- Bridge

  Separates an object's interface from its implementation
- Composite

  A tree structure of simple and composite objects
- Decorator

  Add responsibilities to objects dynamically
- Facade

  A single class that represents an entire subsystem
- Flyweight

  A fine-grained instance used for efficient sharing
- Private Class Data

  Restricts accessor/mutator access
- Proxy

  An object representing another object

Behavioral design pattern

The communication of Class's objects is the focus of these design patterns. Behavioral patterns are those that are primarily concerned with object-to-object communication.

- Chain of responsibility

  A way of passing a request between a chain of objects
- Command

  Encapsulate a command request as an object
- Interpreter

  A way to include language elements in a program
- Iterator

  Sequentially access the elements of a collection
- Mediator

  Defines simplified communication between classes
- Memento

  Capture and restore an object's internal state
- Null Object

  Designed to act as a default value of an object
- Observer

  A way of notifying change to a number of classes
- State

  Alter an object's behavior when its state changes
- Strategy

  Encapsulates an algorithm inside a class
- Template method

  Defer the exact steps of an algorithm to a subclass
- Visitor

  Defines a new operation to a class without change

  (SourceMaking, 2021)

RELATIONSHIP FOR OBJECT ORIENTED PARADIGM OBJECT ORIENTED DESIGN PATTERN

Object oriented paradigm

The object-oriented programming paradigm (OOP) takes a whole new approach to program development. When you use an object-oriented approach, you'll be able to identify the system's objects and how they interact with one another.

The fundamental notion of a new programming methodology gave birth to the object-oriented (OO) paradigm, but interest in design and analytical approaches emerged considerably later. The widespread usage of OO programming languages has led to the emergence of the OO analysis and design paradigm.

Object oriented design pattern

The conceptual model is subsequently evolved into an object-oriented model utilizing object-oriented design after the analysis phase (OOD). The analysis model's technology-independent ideas are mapped onto implementing classes, constraints are established, and interfaces are built in OOD, resulting in a solution domain model. The primary goal of OO design is to create a system's structural architecture.

(Tutorialspoint, 2021)

References

Choudhary, P. K., 2016. *Types Of Relationships In Object Oriented Programming (OOPS).* [Online]
Available at: https://www.c-sharpcorner.com/article/types-of-relationships-in-object-oriented-programming-oops/
[Accessed 08 December 2021].

geeksforgeeks, 2019. *Perl | Method Overriding in OOPs.* [Online]
Available at: https://www.geeksforgeeks.org/perl-method-overriding-in-oops/
[Accessed 08 December 2021].

oodesign, 2021. *Design Patterns.* [Online]
Available at: https://www.oodesign.com/
[Accessed 08 December 2021].

S.Gillies, A., 2021. *What is object-oriented programming?.* [Online]
Available at: https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP
[Accessed 08 December 2021].

S.Gills, A., 2021. *What are the main principles of OOP?.* [Online]
Available at: https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP
[Accessed 08 December 2021].

S.Gills, A., 2021. *What is the structure of object-oriented programming?.* [Online]
Available at: https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP
[Accessed 08 December 2021].

Singhal, G., 2019. *Overload Methods and Invoking Overloaded Methods.* [Online]
Available at: https://www.pluralsight.com/guides/overload-methods-invoking-overload-methods-csharp
[Accessed 08 December 2021].

SourceMaking, 2021. *Design Patterns.* [Online]
Available at: https://sourcemaking.com/design_patterns
[Accessed 08 December 2021].

Tutorialspoint, 2021. *Object-Oriented Paradigm.* [Online]
Available at:
https://www.tutorialspoint.com/software_architecture_design/object_oriented_paradigm.htm
[Accessed 08 December 2021].