Unit I : Introduction to OOSE

Principles of OOPs

Object-oriented programming is based on the following principles:

- Object:

Object is an instance of a class.

It is an entity with a well defined boundary and identity that encapsulates state and behavior. **State is represented by attributes and relationships, behavior is represented by operations, methods, and state machines**. An object is an instance of a class.

An object is an individual [thing] with a state and relationships to other objects. The state of an object identifies the values for that object of properties of the classifier of the object.

• Any real-world thing can be treated as anobject in the OO
Example A Fruit ,ACar,A Person, A House

- Class:

A class represents a template (specification) for several objects and describes how these objects arestructured internally.

• A collection of similar types of objects is considered as a class. The objects that have similarbehaviour and information structure can be grouped under a class.

• A class serves as a blueprint for its objects. That is, once a class has been defined, any number ofobjects belonging to that class be created.

• The objects of a class are also known as the instances or the variables of that class and the process ofcreating objects from a class is known as instantiation.

• A class is defined as, "a user-defined datatype, which contains the entire set of asimilar data and the functions that theobjects possess". OR

• Class is defined as, "an abstract data typecharacterized by a set of properties

(attributes and functions) common to itsobjects".

Example. No of persons(employee) working in company

A person is a object and employee is a class which shares similar bheavior and structure

- Encapsulation- The implementation and state of each object are privately held inside a definedboundary, or class. Other objects do not have access to this class or the authority to make changes but are only able to call a list of public functions, or methods. This characteristic of data hiding provides greater program security and avoids unintended data corruption.

  Another example of encapsulation is a capsule. Basically, capsule encapsulate several combinations of medicine.

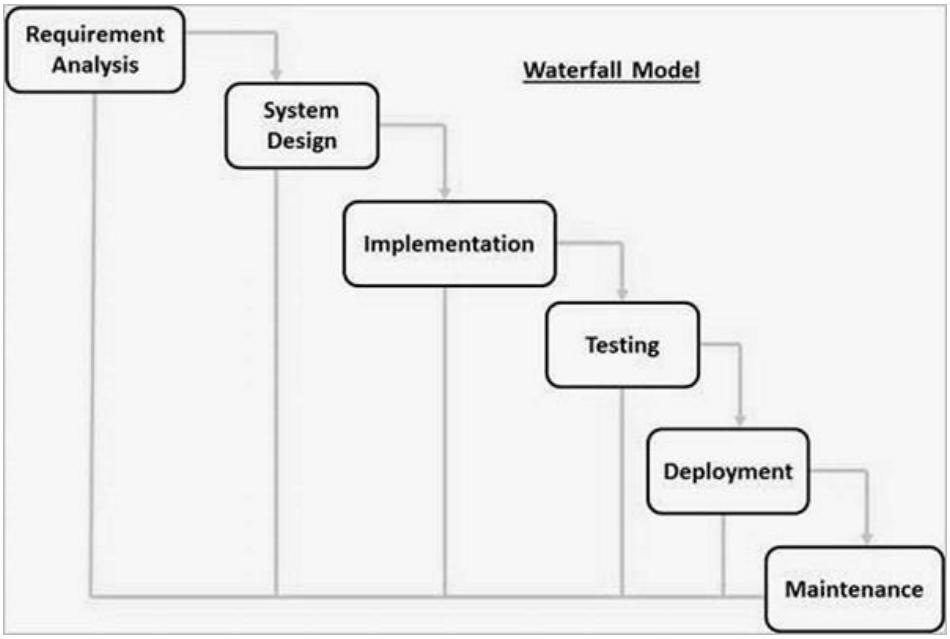
- Abstraction- Objects only reveal internal mechanisms that are relevant for the use of other objects, hiding any unnecessary implementation code. This concept helps developers make changes and additions over time more easily.
- example of Abstraction is ATM Machine
-
- Inheritance- Relationships and subclasses between objects can be assigned, allowing developers to reuse a common logic while still maintaining a unique

hierarchy. This property of OOP forces a more thorough data analysis, reduces development time and ensures a higher level of accuracy.

- Example .parent-child relationship.
- Polymorphism- Objects are allowed to take on more than one form depending on the context. The program will determine which meaning is necessary for each execution of that object, cutting down on the need to duplicate code.

  Real life example of polymorphism: A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behaviour in different situations. This is called polymorphism.

SDLC - Waterfall Model



The sequential phases in Waterfall model are −

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

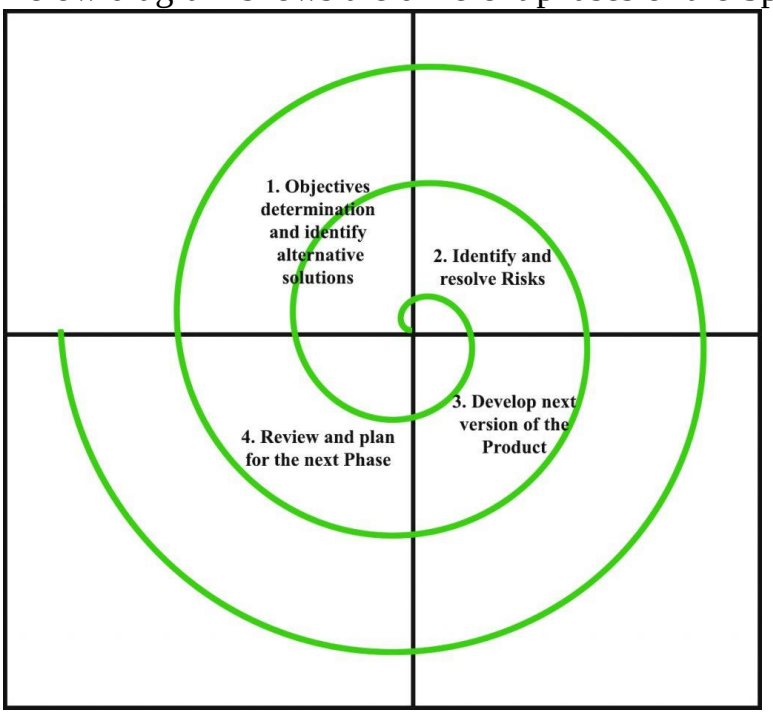
Spiral model

Spiral Model is a combination of a waterfall model and iterative model. Each phase in spiral model begins with a design goal and ends with the client reviewing the progress. The spiral model was first mentioned by Barry Boehm in his 1986 paper.

The development team in Spiral-SDLC model starts with a small set of requirement and goes through each development phase for those set of requirements. The software engineering team adds functionality for the additional requirement in every-increasing spirals until the application is ready for the production phase.

In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. **Each loop of the spiral is called a Phase of the software development process.**

Below diagram shows the different phases of the Spiral Model:



Each phase of Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
2. **Identify and resolve Risks:** During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution.
3. **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

What is Structural Analysis?

The Structural Analysis/Structured Design (SASD) approach is the traditional approach of software development. It is based upon the waterfall model. The development phases of a SASD system are given bellow:

- Feasibility Study
- Requirement Analysis and Specification
- System Design
- Implementation
- Post-implementation Review

What is Object Oriented Analysis (OOA)?

Object Oriented Analysis (OOA) is process of discovery where a development team understands and models the requirements of the system. In OOA requirements are organized as objects. It integrates all the process and data. But in others or traditional structural analysis both process and data are considered independently/separately. They use flow chart/structure charts for process and ER diagrams for data.

But In OOA some advance models are used. The common models used in OOA are: Use cases, Object models. Use cases describe pictures or overview for standard domain functions that the system must achieved. Object models describe the names, class relations, operations, and properties of the main objects. User-interface prototypes can also be created for better understanding.

Object Oriented Analysis (OOA) begins by looking at the problem domain (the area of expertise or application that needs to analyze in order to solve a problem). Its aim is to produce a conceptual model of the information that exists in the area being analyzed. For the analysis there are a variety of sources. It can be a formal

document, a written requirements statement, interviews with stakeholders/other interested parties, other methods, etc. The final result of object oriented analysis will appear in the form of a conceptual model that describes what the system is functionally required to do.

An object is the basic unit of object-orientation approach. An object is a real-world element in an
object–oriented environment that may have a physical or a conceptual existence.
• An object is a chunk of structured data in a running software system. Each object has:
1. **Identity** that distinguishes it from other objects i Identity n the system.
2. **State** that determines the characteristic properties State of an object as well as the values of the
properties that the object holds.
3. **Behavior** that represents externally visible activitBehaviories performed by an object in terms of changesin its state.
• Object can be defined technically as "acombination of data and their associated functions." OR
• An object is "an abstraction of something ofinterest to the program, normally something in
the real world".
• An object-oriented model consists of a numberof objects; these are clearly delimited parts of
themodeled system.
• Any real-world thing can be treated as anobject in the OO
Example A Fruit ,ACar,A Person, A House

Operations are the methods that abstractservices of objects. An operation is a functionthat may be applied to or by object in a class.
• In the Fig. 1.5, an object Yash is representedfrom outside. We are able to see operations,only the properties are hidden in an object
oriented system.

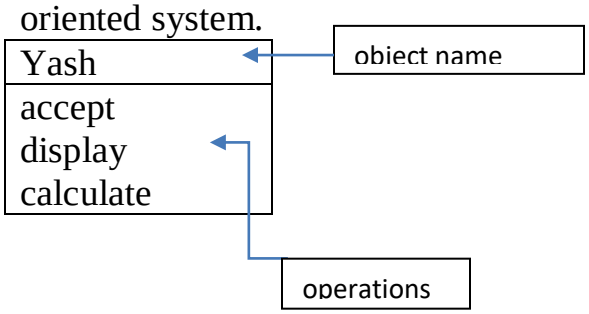


Fig. 1.5: Object Yash and its Operations

•

The three analysis techniques that are used in conjunction with each other for object-oriented analysis are object modelling, dynamic modelling, and functional modelling.

Object Modelling

Object modelling develops the static structure of the software system in terms of objects. It identifies the objects, the classes into which the objects can be grouped into and the relationships between the objects. It also identifies the main attributes and operations that characterize each class.

The process of object modelling can be visualized in the following steps −

- Identify objects and group into classes
- Identify the relationships among classes
- Create user object model diagram
- Define user object attributes
- Define the operations that should be performed on the classes
- Review glossary

Dynamic Modelling

After the static behavior of the system is analyzed, its behavior with respect to time and external changes needs to be examined. This is the purpose of dynamic modelling.

Dynamic Modelling can be defined as “a way of describing how an individual object responds to events, either internal events triggered by other objects, or external events triggered by the outside world”.

The process of dynamic modelling can be visualized in the following steps −

- Identify states of each object
- Identify events and analyze the applicability of actions
- Construct dynamic model diagram, comprising of state transition diagrams
- Express each state in terms of object attributes
- Validate the state–transition diagrams drawn

Functional Modelling

Functional Modelling is the final component of object-oriented analysis. The functional model shows the processes that are performed within an object and how the data changes as it moves between methods. It specifies the meaning of the operations of object modelling and the actions of dynamic modelling. The functional model corresponds to the data flow diagram of traditional structured analysis.

The process of functional modelling can be visualized in the following steps −

- Identify all the inputs and outputs
- Construct data flow diagrams showing functional dependencies
- State the purpose of each function
- Identify constraints
- Specify optimization criteria

Advantages of OOA

The OOA provides better performance. Some common advantages of OOA are given bellow:

- Its focuses on data rather than the procedures as in Structured Analysis
- The objectives of encapsulation and data hiding help the developer to develop the systems that cannot be tampered by other parts of the system
- It allows effective software complexity management by the virtue of modularity
- It can be upgraded from small to large system easily

Structured Analysis VS Object Oriented Analysis

Some differences between Structured Analysis and Object Oriented Analysis are given bellow:

- Structured Analysis treats processes and data as separate components. Where, OO Analysis Combines processes and data into single component which called object
- Structured Analysis does not support re-usability of code. So, the development time and cost is naturally high. But OO analysis supports code re-usability which reduce the development time and cost.

- In Structured Analysis, the focus is only on process and procedures. Modeling techniques used in it are DFD(Data Flow Diagram), Flowcharts etc. This approach is old and is not preferred.
- Whereas in Object Oriented Analysis, the focus is more on capturing the real world objects in the current scenario that are of importance to the system. It stresses more on data structure and less on procedural structure. Without actually identifying objects, what are you going to interactwith, and whose state will you change.In this approach, objects are identified, their relationships among each other, possible states that each object can be in, and finally how all objects collaborate with each other to achieve a broader system goal are identified.

Object Oriented construction :

- Identify the implementation environment
  - Identifying and investigating the consequences that the implementation environment will have on the design
- Incorporate these conclusions and develop a first approach to a design model
- Describes how the objects interact in each specific use case

Identifying the Elements of an Object Model

Identifying Objects and classes :

External entities (e.g., other systems, devices, people) that produce or consume information to be used by a computer-based system.

Things (e.g, reports, displays, letters, signals) that are part of the information domain for the problem.

Occurrences or *events* (e.g., a property transfer or the completion of a series of robot movements) that occur within the context of system operation.

Roles (e.g., manager, engineer, salesperson) played by people who interact with the system.

Organizational units (e.g., division, group, team) that are relevant to an application.

Places (e.g., manufacturing floor or loading dock) that establish the context of the problem and the overall function of the system.

Structures (e.g., sensors, four-wheeled vehicles, or computers) that define a class of objects or in the extreme, related classes of objects.

Specifying Attributes

 Attributes describe an object that has been selected for inclusion in the analysis model. In essence, it is the attributes that define the object—that clarify what is meant by the object in the context of the problem space.

 To store info objects use attributes. To each entity object we can tie several attributes.

Guidelines for defining attributes

i. Keep the class simple,state only enough attributes to define object state
ii. Attributes to be fully described in the problem domain
iii. Omit derived attributes
iv. You can add more attributes in subsequent iterations

Defining Operations

o Operations change one or more attribute values that are contained within an object.

o Three categories of objects:

(1) operations that manipulate data in some way (e.g., adding, deleting, reformatting, selecting)

(2) operations that perform a computation

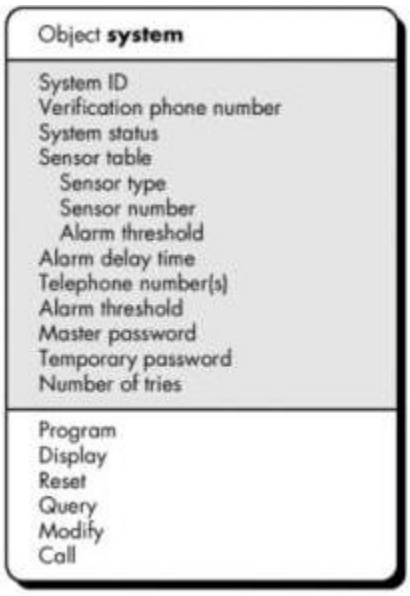
(3) operations that monitor an object for the occurrence of a controlling event.

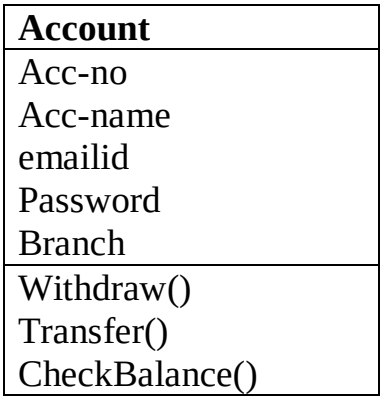
Finalizing the Object Definition

- Definition of operations is the last step in completing the specification of an object.
- Generic life history of an object can be defined by recognizing that the object must be created, modified, manipulated or read in other ways, and possibly deleted.
- Some of the operations can be ascertained from likely communication between objects.

o e.g.

- **sensor event** will send a message to **system** to ***display*** the **event location** and **number**
- **control panel** will send **system** a ***reset*** message to update **system status**
- **audible alarm** will send a ***query*** message
- **control panel** will send a ***modify*** message to change one or more attributes without reconfiguring the entire system object;
- **sensor event** will also send a message to ***call*** the **phone number(s)** contained in the object.
- Final Result




Account
Acc-no
Acc-name
emailid
Password
Branch
Withdraw()
Transfer()
CheckBalance()