Oops In Detail

What is object Orientation ?

Object-oriented or **object-orientation** is a software engineering concept, in which concepts are represented as "objects".
Object orientation eases maintenance by the use of encapsulation and information hiding.

What is Object ?

an **object** is a location in memory having a value and possibly referenced by an identifier. An object can be a variable, function, or data structure.

What is **Object-oriented analysis and design?**

Object-oriented analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.

Object-oriented analysis and design (OOAD) is a popular technical approach to analyzing, designing an application, system
OOAD in modern software engineering is best conducted in an iterative and incremental way. Iteration by iteration, the outputs of OOAD activities, analysis models for OOA and design models for OOD respectively, will be refined and evolve continuously driven by key factors like risks and business value.

What is object oriented design?

Object-oriented design is the process of planning a system of interacting objects for the purpose of solving a software problem. It is one approach to software design.
Object-oriented design is a method of design encompassing the process of object-oriented decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the system under design.

There are two important parts to this definition: object-oriented design
(1) leads to an object-oriented decomposition and
(2) uses different notations to express different models of the
logical (class and object structure) and physical (module and process architecture) design of a system, in addition to the static and dynamic aspects of the system.

An object contains encapsulated data and procedures grouped together to represent an entity. The 'object interface', how the object can be interacted with, is also defined. An object-oriented program is described by the interaction of these objects. Object-oriented design is the discipline of defining the objects and their interactions to solve a problem that was identified and documented during object-oriented analysis.

what is Object-oriented modeling ?

Object-oriented modeling is an approach to modeling an application that is used at the beginning of the software life cycle when using an object-oriented approach to software development.

The software life cycle is typically divided up into stages going from abstract descriptions of the problem to designs then to code and testing and finally to deployment. Modeling is done at the beginning of the process.

Object-oriented modeling is typically done via use cases and abstract definitions of the most important objects.

What is Object-oriented software engineering ?

What is object oriented programming ?
 is a programming paradigm that represents the concept of "objects" that have data fields (attributes that describe the object) and associated procedures known as methods. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.
Object-oriented programming is an approach to designing modular, reusable software systems.

Object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

object-oriented programming has three main things
(1) uses objects, not algorithms, as its fundamental logical building blocks (the “part of” hierarchy we introduced in Chapter l);
(2) each object is an instance of some class; and
(3) classes are related to one another via inheritance relationships .

How are OOA, OOD, and OOP related?

Basically, the products of object oriented analysis serve as the models from which we may start an object-oriented design; the products of object-oriented design can then be used as blueprints for completely implementing a system using object-oriented programming methods.

Algorithmic versus Object-Oriented Decomposition of problem ?

the algorithmic view highlights the ordering of events, and

the object-oriented view emphasizes the agents that either cause action or are the su this approach is better
at helping us organize the inherent complexity of software systems, just as it helped us to
describe the organized complexity of complex systems as diverse as computers, plants, galaxies, and large social institutions. objects upon which these operations act.

Most of problem solving method can categorise in three :

* Top-down structured design
• Data-driven design
• Object-oriented design

Top-down structured design
Each of these variations applies algorithmic decomposition. More software has probably been written using these design methods than with any other. Nevertheless, structured design does not address the issues of data abstraction and information hiding, nor does it provide an adequate means of dealing with concurrency. Structured design does not scale up well for extremely complex systems, and this method is largely inappropriate for use with object-based and object- oriented programming languages.

Data-driven design
the structure of a software system is derived by
mapping system inputs to outputs. As with structured design, data-driven design has been successfully applied to a number of complex domains, particularly information management systems, which involve direct relationships between the inputs and outputs of the system, but require little concern for time-critical events.
Object-oriented design

Its underlying concept is that one should model software systems as collections of cooperating objects, treating individual objects as instances of a class within a hierarchy of classes. Object-oriented analysis and design directly reflects the topology of more recent high-order programming languages such as Smalltalk, Object Pascal, C++, the Common Lisp Object System (CLOS),

The Meaning of Design
design encompasses the disciplined approach we use to
invent a solution for some problem, thus providing a path from requirements to implementation.
design is to construct a system that:
1 Satisfies a given (perhaps informal) functional specification
2 Conforms to limitations of the target medium
3 Meets implicit or explicit requirements on performance and resource usage
4 Satisfies implicit or explicit design criteria on the form of the artifact
5 Satisfies restrictions on the design process itself, such as its length or cost, or the tools
6 available for doing the design"

Kinds of Programming Paradigms

1 Procedure-oriented -> Based on Algorithms

2 Object-oriented -> Classes and objects

3 Logic-oriented -> Goals, often expressed in a predicate calculus

4 Rule-oriented -> If-then rules

5 Constraint-oriented -> Invariant relationships

There is no single programming style that is best for all kinds of applications. For example, rule-oriented programming would be best for the design of a knowledge base, and procedure-oriented programming would be best suited for the design of computation-intense
operations. From our experience, the object-oriented style is best suited to the broadest set of applications; indeed, this programming paradigm often serves as the architectural framework in which we employ other paradigms.

Abstraction

Abstraction is one of the fundamental ways that we as humans cope with complexity.
1
"abstraction arises from a recognition of similarities between certain objects, situations, or processes in the real world, and the decision to concentrate upon these similarities and to ignore for the time being the differences"
2
Shaw defines an abstraction as "a simplified description, or specification, of a system that emphasizes some of the system's details or properties while suppressing others.
3
Gray, and Naumann recommend that ~'a concept qualifies as an abstraction only if it can be described, understood, and analyzed independently of the mechanism that will eventually be used to realize it"

Final
An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.

Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.
Deciding upon the right set of abstractions for a given domain is the central problem in object-oriented design.

The Meaning of Encapsulation

Encapsulation hides the details of the implementation of an object.
The abstraction of an object should precede the decisions about its implementation. Once an implementation is selected, it should be treated as a secret of the abstraction and hidden from most clients.

Abstraction and encapsulation are complementary concepts: abstraction focuses upon the observable behavior of an object, whereas encapsulation focuses upon the implementation that gives rise to this behavior. Encapsulation is most often achieved through information binding, which is the process of hiding all the secrets of an object that do not contribute to its essential characteristics; Encapsulation provides explicit barriers among different abstractions and thus leads to a clear separation of concerns.

To summarize, we define encapsulation as follows:
Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation.

Britton and Parnas call these encapsulated elements the **"secrets" of an abstraction**

Modularity

System details that are likely to change independently should be the secrets of separate modules;

Modularity is the property of a system that has been decomposed into a set of cohesive(by grouping logically related abstractions)   and loosely coupled (by minimizing the dependencies among modules) modules.

Thus, the principles of abstraction, encapsulation, and modularity are An object provides a crisp boundary around a single abstraction, and both encapsulation and modularity provide barriers around this abstraction.

Hierarchy

The Meaning of Hierarchy: A set of abstractions often forms a hierarchy, and by identifying these hierarchies in our ,design, we greatly simplify our understanding of the problem.

We define hierarchy as follows:

Hierarchy is a ranking or ordering of abstractions.

Typing

Meaning of Typing The concept of a type derives primarily from the theories of abstract data types.

"A type is a precise characterization of structural or behavioral properties which a collection of entities all share"

Typing is the enforcement Of the class of an object, such, that objects of different types may not be interchanged, or at the most, they may be interchanged only in very restricted ways.

Strong and Weak Typing

strongly-typed, meaning that type conformance is strictly- enforced: operations cannot be called upon an object unless the exact signature of that operation is defined in the object's class or superclasses. In strongly typed languages, violation of type conformance can be detected at the time of compilation.

on the other hand, is an untyped language: a client can send any message to any class (although a class may not know how respond to the message). Violations of type conformance may not be known until execution, and usually manifest themselves as execution errors.

As Tesler points out, there are a number of important benefits to be derived from using
strongly typed languages:

1 Without type checking, a program in most languages can 'crash' in mysterious ways at runtime.

2 In most systems, the edit-compile-debug cycle is so tedious that early error detection is indispensable.

3 Type declarations help to document programs. Most compilers can generate more efficient object code if types are declared"

Static and Dynamic Binding

Static binding means that the types all variables and expressions are fixed at the time of compilation;

dynamic binding (also called late binding) means that the types of all variables and expressions are not known until runtime.

Concurrency :

Concurrency allows different objects to act at the same time.

Concurrency is the properly that distinguisbes an active object from one that is not active.

Persistence

Persistence saves the state and class of an object across time or space.

Persistence is the property of an object through which its existence transcends time (i.e. the object continues to exist after its creator ceases to exist) and/or space (i. e. the objects location moves from the address space in which it was created

Classes and Objects

From the perspective of human cognition, an object is any of the following:

1 A tangible and/or visible thing
2 Something that may be apprehended intellectually
3 Something toward which thought or action is directed

An object has state, exhibits some well-defined behavior, and has a unique identity.

An object has state, behavior, and identity; the structure and behavior of similar objects are defined in their common class; the terms instance and object are interchangeable.

State : The state of an object encompasses all of the (usually static) properties of the object plus the current (usually dynamic) values of each of these properties.

The state of an object represents the cumulative results of its behavior.

Behavior : Behavior is how an object acts and reacts, in terms of its state changes and message passing.

Identity : Semantics Khoshafian and Copeland offer the following definition: “Identity is that property of an object which distinguishes it from all other objects "

What Is and What Isn't a Class ?

The concepts of a class and an object are tightly interwoven, for we cannot talk about an object without regard for its class. However, there are imiportant differences between these two terms. Whereas an object is a concrete entity that exists in time and space, a class represents only an abstraction, the “essence" of an object, as it were.

we define a class as follows:
A class is a set of objects that share a common structure and a common behavior.

A single object is simply an instance of a class.

Inheritance

Object-oriented programming allows classes to *inherit* commonly used state and behavior from other classes.
In the Java programming language, each class is allowed to have one direct superclass, and each superclass has the potential for an unlimited number of *subclasses*:
inheritance is a feature that represents the "is a" relationship between different classes.

Polymorphism

A concept in type theory, according to which a name (such as a variable declaration) may denote objects of many different classes that are related by some common superclass; thus, any object denoted by this name is able to respond to some common set of operations in different ways.