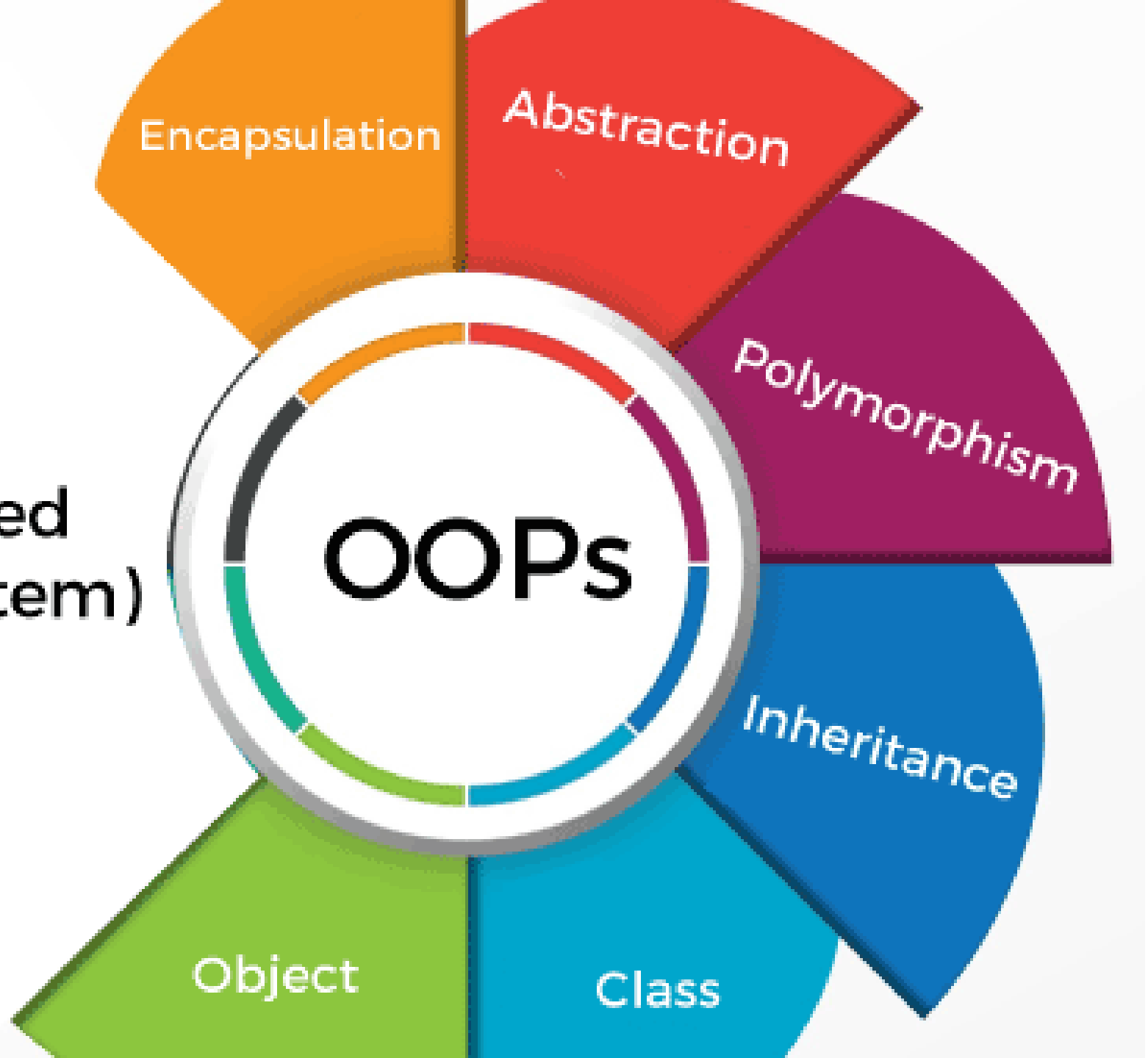
( Object - Oriented Programming System)



Unit 9: Object Oriented Programming

>>>Review of Fundamentals of Procedural Programming,

>>>Class and Objects,

>>>Data Abstraction,

>>>Information Hiding & Encapsulation,

>>>Constructors, destructors, and object creation,

>>>Name space and references ,

>>>Class Methods , Methods Overloading ,

>>>Inheritance ,

>>>Polymorphism ,

>>>Abstract Classes,

>>>Abstract Methods ,

>>>Exceptions , Exception Handling.

# Introduction of Object Oriented Programming

- As the name suggests, Object-Oriented Programming or OOPs refers to languages that use objects in programming.
- Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming.
- The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

# Introduction of Object Oriented Programming

- OOPs Concepts:

- Class
- Objects
- Data Abstraction
- Encapsulation
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing

# OBJECT

- It is a basic unit of Object-Oriented Programming and represents the real-life entities. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. An object has an identity, state, and behavior.
- Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

- For example "Dog" is a real-life Object, which has some characteristics like color, Breed, Bark, Sleep, and Eats.

# Class

- A class is a user-defined data type. It consists of data members and member functions, which can be accessed and used by creating an instance of that class. It represents the set of properties or methods that are common to all objects of one type. A class is like a blueprint for an object.

- For Example: Consider the Class of Cars. There may be many cars with different names and brands but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range, etc. So here, the Car is the class, and wheels, speed limits, and mileage are their properties.

# CLASS

- Collection of objects is called class. It is a logical entity.
- A class can also be defined as a blueprint from which you can create an individual object.
- Class doesn't consume any space.

# Class

- A class describes the state or behavior of an object. A class can have attributes and methods:

- Attributes: An Attribute is a public variable inside the class/object. For example, Length is an attribute of int data type. In other words, **class attributes** are the variable within a class. You can access attributes by creating an object of a class and by using dot syntax(.).

- Methods: A method is a group/block of code that takes input from the user, process it, and provides the output. The method runs only when it is called.

# Data Abstraction

- Data abstraction is one of the most essential and important features of object-oriented programming.
- Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.
- Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car, but he does not know about how on pressing the accelerator the speed is increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc in the car. This is what abstraction is

# Data Abstraction

- Hiding internal details and showing functionality is known as abstraction.
- For example phone call, we don't know the internal processing.
- In Java, we use abstract class and interface to achieve abstraction.

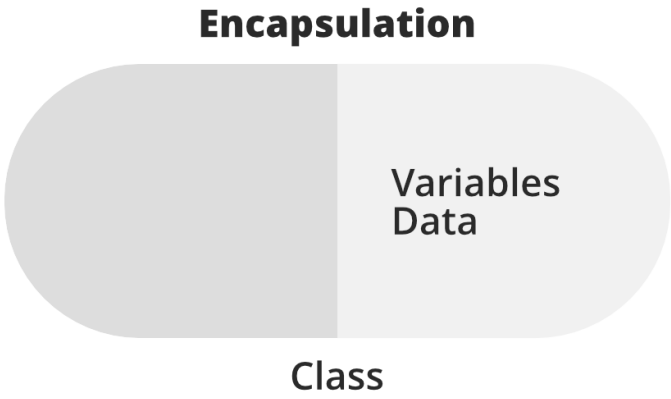
# ENCAPSULATION

- ✓ Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

- ✓ A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here

# ENCAPSULATION

- Encapsulation is defined as the wrapping up of data under a single unit.
- It is the mechanism that binds together code and the data it manipulates.
- In Encapsulation, the variables or data of a class are hidden from any other class and can be accessed only through any member function of their class in which they are declared. As in encapsulation, the data in a class is hidden from other classes, so it is also known as data-hiding.



# ENCAPSULATION

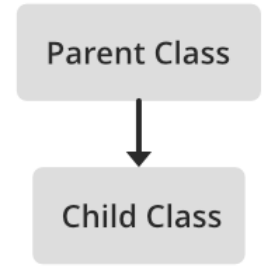
- Consider a real-life example of encapsulation, in a company, there are different sections like the accounts section, finance section, sales section, etc.
- The finance section handles all the financial transactions and keeps records of all the data related to finance.
- Similarly, the sales section handles all the sales-related activities and keeps records of all the sales. Now there may arise a situation when for some reason an official from the finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of the sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data. This is what encapsulation is. Here the data of the sales section and the employees that can manipulate them are wrapped under a single name “sales section”.

# INHERITANCE

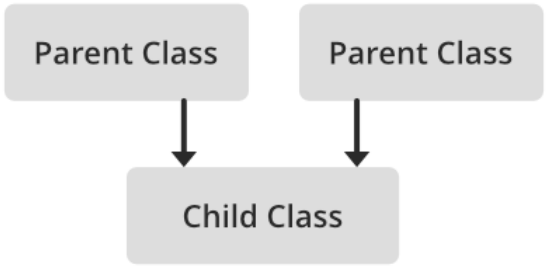
- Inheritance is an important pillar of OOP(Object-Oriented Programming).
- The capability of a class to derive properties and characteristics from another class is called Inheritance.
- When we write a class, we inherit properties from other classes.
- So when we create a class, we do not need to write all the properties and functions again and again, as these can be inherited from another class that possesses it. Inheritance allows the user to reuse the code whenever possible and reduce its redundancy.

# INHERITANCE

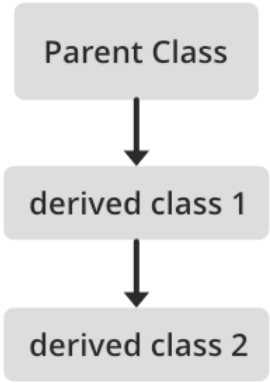
- When one object acquires all the properties and behaviors of a parent object, it is known as inheritance.
- It provides code reusability.
- It is used to achieve runtime polymorphism



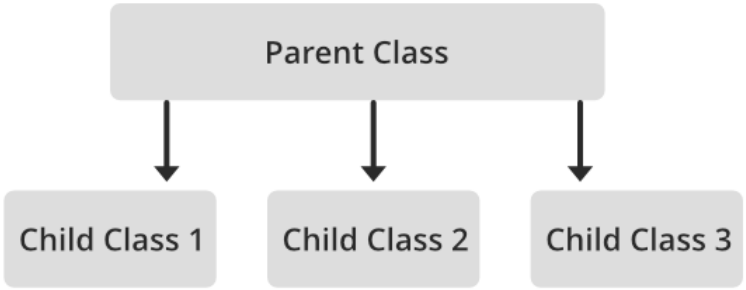

**Single inheritance**



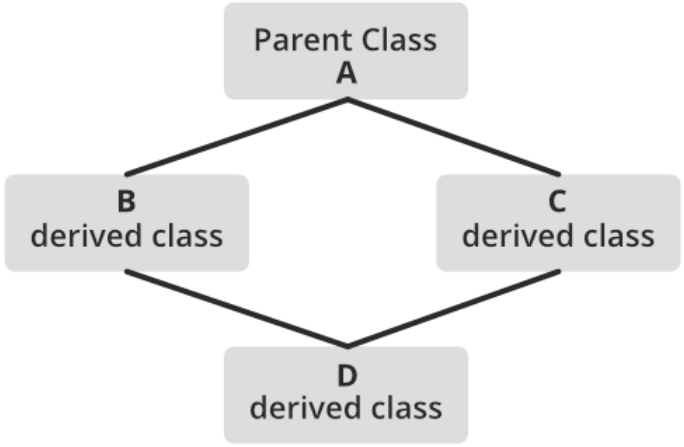

**Multiple inheritance**




**Multilevel inheritance**




**Hierarchical inheritance**




**Hybrid Class**

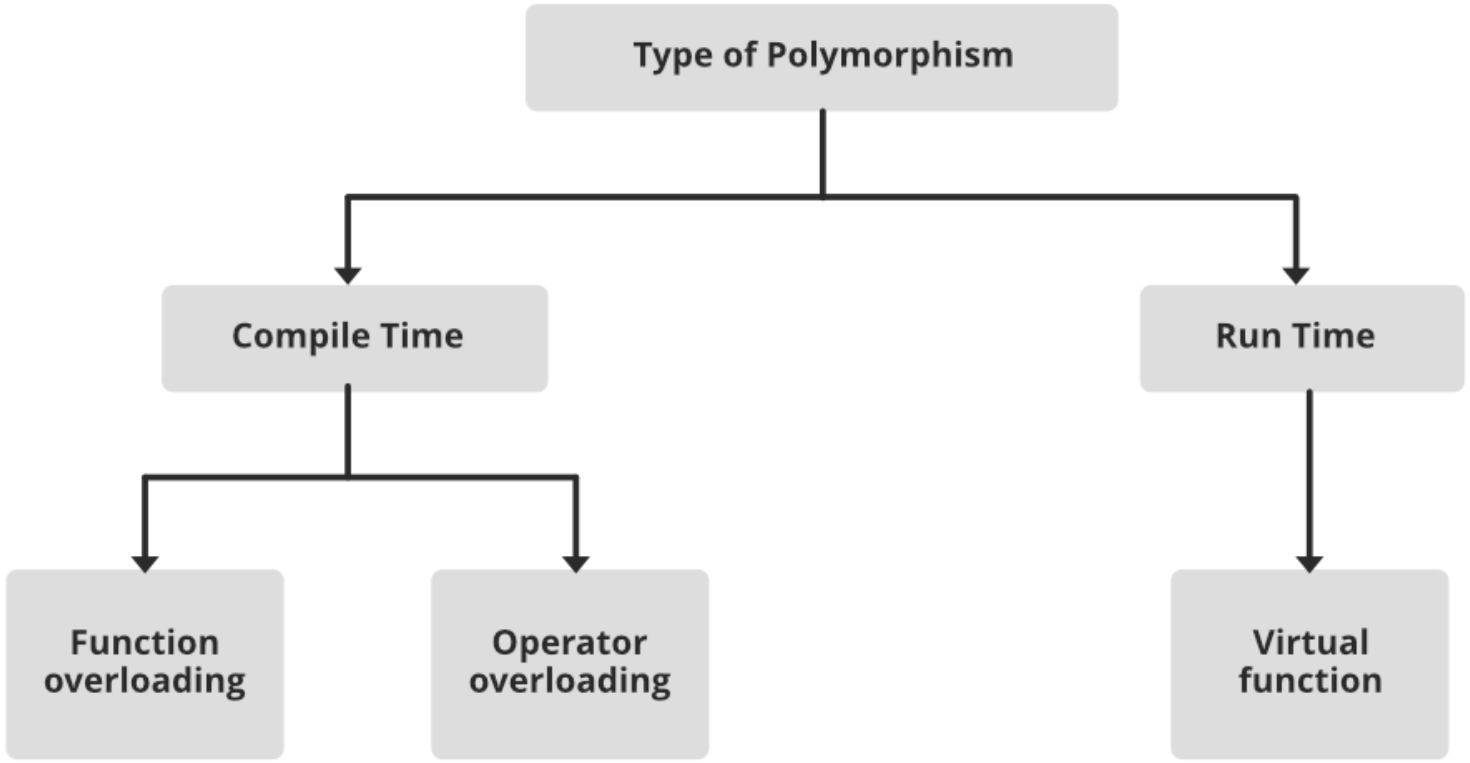
# POLYMORPHISM

- The word polymorphism means having many forms.
- In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.
- For example, A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations. This is called polymorphism.

# POLYMORPHISM

- If one task is performed in different ways, it is known as polymorphism.
- For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.
- In Java, we use method overloading and method overriding to achieve polymorphism.
- Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

# POLYMORPHISM



# Dynamic Binding

- In dynamic binding, the code to be executed in response to the function call is decided at runtime.
- Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run time.
- Dynamic Method Binding One of the main advantages of inheritance is that some derived class D has all the members of its base class B. Once D is not hiding any of the public members of B, then an object of D can represent B in any context where a B could be used. This feature is known as subtype polymorphism.

# Message Passing

- It is a form of communication used in object-oriented programming as well as parallel programming.
- Objects communicate with one another by sending and receiving information to each other.
- A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results. Message passing involves specifying the name of the object, the name of the function, and the information to be sent.

# COUPLING

- Coupling refers to the knowledge or information or dependency of another class. It arises when classes are aware of each other.
- If a class has the details information of another class, there is strong coupling.
- In Java, we use private, protected, and public modifiers to display the visibility level of a class, method, and field.

# Association

- Association represents the relationship between the objects. Here, one object can be associated with one object or many objects. There can be four types of association between the objects:

- One to One
- One to Many
- Many to One, and
- Many to Many

- For example, One country can have one prime minister (one to one), and a prime minister can have many ministers (one to many). Also, many MP's can have one prime minister (many to one), and many ministers can have many departments (many to many).

# AGGREGATION

- Aggregation is a way to achieve Association.
- Aggregation represents the relationship where one object contains other objects as a part of its state.
- It represents the weak relationship between objects.
- It is also termed as a has-a relationship in Java.
- Like, inheritance represents the is-a relationship.
- It is another way to reuse objects.

# COMPOSITION

- The composition is also a way to achieve Association.
- The composition represents the relationship where one object contains other objects as a part of its state.
- There is a strong relationship between the containing object and the dependent object.
- It is the state where containing objects do not have an independent existence.
- If you delete the parent object, all the child objects will be deleted automatically.

# EXCEPTION HANDLING

- Exceptions: Exceptions are runtime anomalies or unusual conditions that a program may encounter while executing .

- Anomalies might include conditions such ass division by zero, accessing an array outside of its bounds or running out of memory or disk space. When a program encounters an exception condition, it must be identified and handled.

- Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch, and throw.

# EXCEPTION HANDLING

- **Types of exceptions:There are two kinds of exceptions**

**1.Synchronous exceptions**
**2.Asynchronous exceptions**

**1. Synchronous exceptions: Errors such as “Out-of-range index” and “over flow” are synchronous exceptions**

**2. Asynchronous exceptions: The errors that are generated by any event beyond the control of the program are called asynchronous exceptions**

# Exception Handling Mechanism

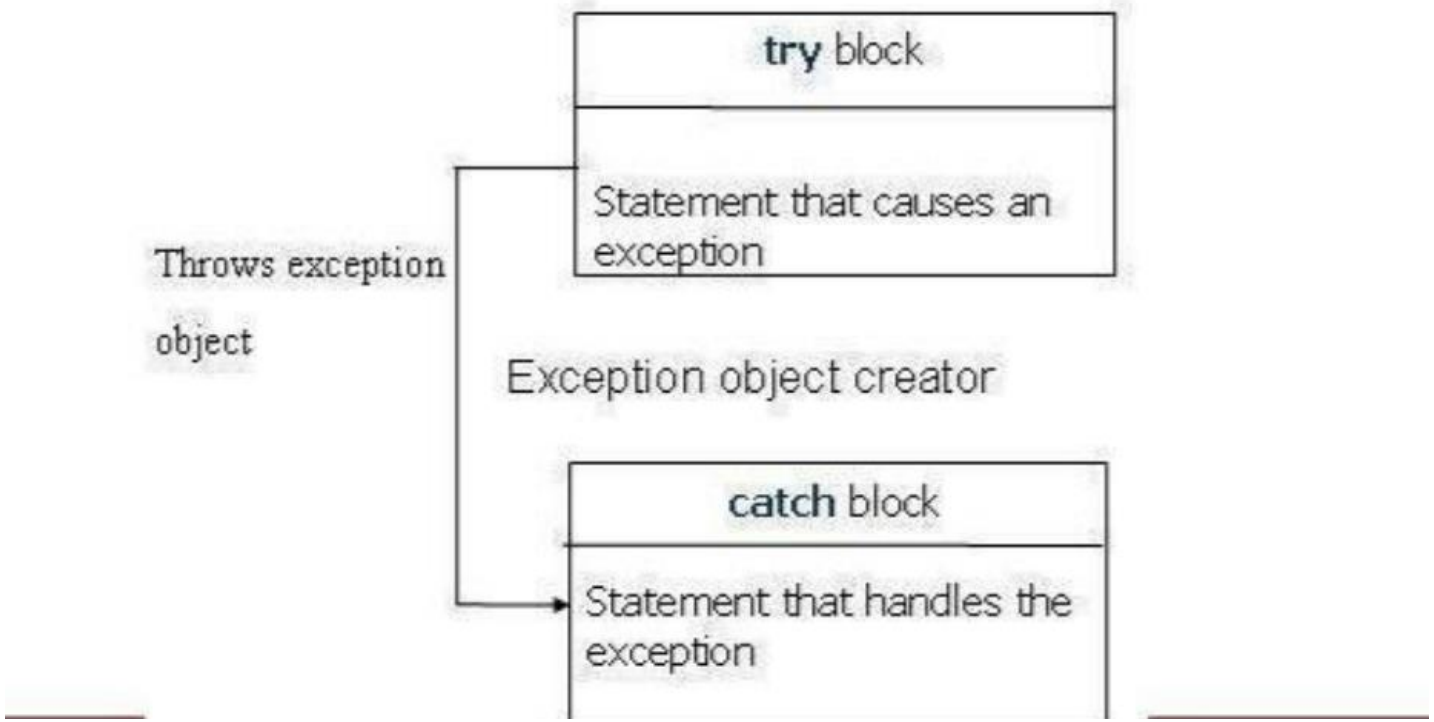
- An exception is said to be thrown at the place where some error or abnormal condition is detected. The throwing will cause the normal program flow to be aborted, in a raised exception.
- An exception is thrown programmatic, the programmer specifies the conditions of a throw.
- In handled exceptions, execution of the program will resume at a designated block of code, called a catch block, which encloses the point of throwing in terms of program execution. The catch block can be, and usually is, located in a different function than the point of throwing.
- C++ exception handling is built upon three keywords: try, catch, and throw.

# Exception Handling Mechanism

- Try is used to preface a block of statements which may generate exceptions. This block of statements is known as try block.

- When an exception is detected it is thrown by using throw statement in the try block. Catch block catches the exception thrown by throw statement in the try block and handles it appropriately.

# EXCEPTION HANDLING MECHANISM



## Overview of C language:

1.C language is known as structure oriented language or procedure oriented language
2.Employs top-down programming approach where a problem is viewed as a sequence of tasks to be performed.
3.All program code of c can be executed in C++ but converse many not be possible
4. Function overloading and operator overloading are not possible.
5. Local variables can be declared only at the beginning of the block.
6. Program controls are through jumps and calls to subroutines.
7.Polymorphism, encapsulation and inheritance are not possible.
For solving the problems, the problem is divided into a number of modules. Each module is a subprogram.
8. Data abstraction property is not supported by procedure oriented language.
9. Data in procedure oriented language is open and can be accessed by any function.

**Overview of C++ language:**

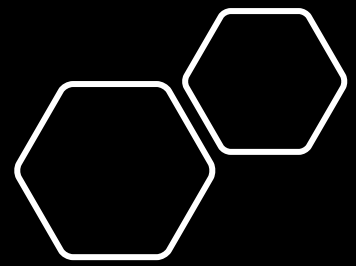
1. C++ can be considered as an incremental version of c language which consists all programming language constructs with newly added features of object oriented programming.
2.c++ is structure(procedure) oriented and object oriented programming language.
3.The file extension of C++ program is ".CPP"
4. Function overloading and operator overloading are possible.
5. Variables can be declared in inline i.e when required
6. In c++ more emphasis is give on data rather than procedures
7.Polymorphism, encapsulation and inheritance are possible.
8. Data abstraction property is supported by c++.
9. Data access is limited. It can be accessed by providing various visibility modes both for data and member functions. there by providing data security by data hiding
10.Dymanic binding is supported by C++
11..It supports all features of c language
12.It can be called as an incremental version of c language

| | Procedure Oriented Programming | Object Oriented Programming |
|---|---|---|
| 1 | program is divided into small parts called **functions**. | program is divided into parts called **objects**. |
| 2 | Importance is not given to **data** but to functions as well as **sequence** of actions to be done. | Importance is given to the data rather than procedures or functions because it works as a **real world**. |
| 3 | follows **Top Down approach**. | OOP follows **Bottom Up approach**. |
| 4 | It does not have any access specifier. | OOP has access specifiers named Public, Private, Protected, etc. |
| 5 | Data can move freely from function to function in the system. | objects can move and communicate with each other through member functions. |

| | | |
|---|---|---|
| 6 | To add new data and function in POP is not so easy. | OOP provides an easy way to add new data and function. |
| 7 | Most function uses Global data for sharing that can be accessed freely from function to function in the system. | In OOP, data can not move easily from function to function,it can be kept public or private so we can control the access of data. |
| 8 | It does not have any proper way for hiding data so it is **less secure**. | OOP provides Data Hiding so provides **more security**. |
| 9 | Overloading is not possible. | In OOP, overloading is possible in the form of Function Overloading and Operator Overloading. |
| 10 | Example of Procedure Oriented Programming are : C, VB, FORTRAN, Pascal. | Example of Object Oriented Programming are : C++, JAVA, VB.NET, C#.NET. |

| Sr. No | Procedure Oriented Programming | Object Oriented Programming |
|---|---|---|
| | POP is divided into small parts called as functions | In OOP, program is divided into parts called objects |
| | In POP, Importance is not given to data but to functions as well as sequence of actions to be done. | In OOP, Importance is given to the data rather than procedures or functions because it works as a real world. |
| | POP follows top-down approach. | OOP follows bottom-up approach. |
| | POP does not have any access specifier. | OOP has access specifiers named Public, Private, Protected, etc. |
| | In POP, data can move freely from function to function in the system. | In OOP, objects can move and communicate with each other through member functions. |
| | To add new data and function in POP is not so easy. | OOP provides an easy way to add new data and function. |
| | In POP, most function uses global data for sharing that can be accessed freely from function to function in the system. | In OOP, data cannot move easily from function to function, it can be kept public or private so we can control the access of data. |
| | POP does not have any proper way for hiding data so it is less secure. | OOP provides data hiding so provides more security. |
| | In POP, overloading is not possible. | In OOP, overloading is possible in the form of Function Overloading and Operator Overloading. |
| | Example of POP are: C, VB, FOR-TRAN, Pascal. | Example of OOP are: C++, JAVA, VB.NET, C#.NET. |

ffjfj



**Q. Who invented OOP?**

**a) Andrea Ferro**
**b) Adele Goldberg**
**c) Alan Kay**
**d) Dennis Ritchie**

Alan Kay invented OOP, Andrea Ferro was a part of SmallTalk Development. Dennis invented C++ and Adele Goldberg was in team to develop SmallTalk but Alan actually had got rewarded for OOP.

Q. Which is not a feature of OOP in general definitions?

a) Efficient Code
b) Code reusability
c) Modularity
d) Duplicate/Redundant data

Q. Which feature of OOP indicates code reusability?

a) Abstraction
b) Polymorphism
c) Encapsulation
d) Inheritance

Inheritance indicates the code reusability. Encapsulation and abstraction are meant to hide/group data into one element. Polymorphism is to indicate different tasks performed by a single entity.

**Q. What is encapsulation in OOP?**

**a) It is a way of combining various data members and member functions that operate on those data members into a single unit**
**b) It is a way of combining various data members and member functions into a single unit which can operate on any data**
**c) It is a way of combining various data members into a single unit**
**d) It is a way of combining various member functions into a single unit**

Q. Which of the following is not true about polymorphism?

a) Helps in redefining the same functionality
b) Increases overhead of function definition always
c) It is feature of OOP
d) Ease in readability of program

Q. What is an abstraction in object-oriented programming?

a) Hiding the implementation and showing only the features
b) Hiding the important data
c) Hiding the implementation
d) Showing the important data

Q. Which feature of OOP reduces the use of nested classes?

a) Inheritance
b) Binding
c) Abstraction
d) Encapsulation

Q. Encapsulation and abstraction differ as _____

a) Hiding and hiding respectively
b) Binding and Hiding respectively
c) Hiding and Binding respectively
d) Can be used any way

Q. Which of the following can be overloaded ?

A. Variable
B. Objects
C. Class
D. Function

Q. The major goal of object - oriented programming is

A. Speed
B. Reuse
C. User - interface
D. Top - down program development

Q. Which of the following most accurately describes "multiple inheritance"?

A. When two classes inherit from each other.
B. When a child class has two or more parent classes.
C. When a base child has two or more derived classes.
D. When a child class has both an "is a" and "has a" relationship with its parent class.

Q. Assertion (A): C++ is an object-oriented programming language.

Reason (R): C++ supports class, inheritance, templates, and exception handling.

A. Both (A) and (R) are true and (R) is correct explanation of (A).
B. Both (A) and (R) are true, but (R) is not the correct explanation of (A).
C. (A) is true, but (R) is false.
D. (A) is false, but (R) is true.

Q. An object-oriented programming concept that refers to the ability of a variable, function, or object to take on multiple forms is :

A. Inheritance
B. Hierarchy
C. Polymorphism
D. State Transition

Q. If a class C is derived from class B, which is derived from class A, all through public inheritance, then a class C member function can access

A. Only protected and public data of C and B
B. Only protected and public data of C
C. All data of C and private data of A and B
D. Public and protected data of A and B and all data of C

1. public inheritance:

In this, public members of the base class remain public in the derived class and protected members of base class remains protected in base class.

2. private inheritance:

in this, each protected and public member of base class remains private in the derived class.

3. protected inheritance:

In this, each private and public member of base class remains protected in the derived class.

| Access | public | protected | private |
|---|---|---|---|
| Same class | YES | YES | YES |
| Derived Class | YES | YES | NO |
| Outside class | YES | NO | NO |

| | |
|---|---|
| Encapsulation | Reduce complexity + increase reusability |
| Abstraction | Reduce complexity + isolate impact of changes |
| Inheritance | Eliminate redundant code |
| Polymorphism | Refactor ugly switch/case statements |