STUDENT SEMINARS

Now a day everything is getting advanced every second of time. Everyday starts with something new. Whatever be the field, everything is getting advanced. Lots of researches and studies are carried out various subjects around the world. These updates on various fields cannot be included in the student curriculum. The students have to always keep their eyes on what new things are arriving day by day. This is where the seminars are of great importance. Seminars are capable of keeping the students updated with the technologies. Seminars provide latest information about the things which are happening in science and technology. Students cannot improve their knowledge from textbooks alone. They must be take part in various seminars on latest topics.

Seminars also help them to convey their own ideas to their friends and teachers. They will start thinking about new things which they think could be implemented practically.

Making students take seminars on the subject topics would always help them more to understand the subject. This would give them a chance to collect more information about the seminar topic they are provided with. The result is that they would learn the subject well. Because they have to acquire knowledge about the subject of their own. There are also many advantages apart from acquiring knowledge. By taking seminars in front of their teachers and friends, the students will be able to talk before a crowd later in their life without any difficulty. Also they can learn their mistakes and can improve their seminar presentation skills. It is actually a great chance for the students to improve their skills within their curriculum. They can improve their language. By taking seminars they will become very able to interact with the people which will turn out to be useful in their later life. There is a wide spread impression about seminars.

STUDENT CENTRIC METHODS OF LEARNING

ACADEMIC YEAR 2022-2023

S.No	Date	Nature of Activity	Topic	Name(s) of the lecturer(s) involved	Details of Student (Name, Hall Ticket No etc.,)	No. of students participated
1	8-11-2023	Seminar	Memory Hierarchy	P.Bharathi	Ade.Rani	26
2	8-11-2023	Seminar	Secondary storage devices	P.Bharathi	Godari.Anjali	25

3	4-11-2023	Seminar	Strategies of Business Management	T.Aruna	Bolli Allowokika	35
4	4-11-2023	Seminar	Stages of Growth	T.Aruna	Kotham Jahnavi	45
5	4-11-2023	Seminar	Information Management System	T.Aruna	Katakam.Navya	36
6	30-10-2023	Seminar	Advantages of DDBMS	P.Prathibha	MD.Parveen	10
7	30-10-2023	Seminar	Introduction to DDBMS	P.Prathibha	Thaviti.Rani	9
8	30-10-2023	Seminar	File Organisation	P.Prathibha	Oruganti Akhila	25
9	21-09-2023	Seminar	Linux System	B.Swarnalatha	KOKKONDA RUTHIKA	20
10	23-09-2023	Seminar	Types of Files in Python	B.Swarnalatha	Vanam Akshaya	50
11	23-09-2023	Seminar	Data Acquisition	B.Swarnalatha	Kammala Savitha	38
12	23-09-2023	Seminar	Files in Python	P.Bharathi	Kalmala .Divya	26
13	23-09-2023	Seminar	Data Analysis Sequence	P.Bharathi	Nampally.Soumya	35
14	01-04-2023	Seminar	List	T.Aruna	Daravath Sangeetha	26
15	01-0402023	Seminar	Control Structure	B.Swarnalatha	Lavudya.Ishwariya	21
16	09-02-2023	Seminar	Computer networks	P.Prathibha	Humera Yasmeen	21

17	09-02-2023	Seminar	Difference between structure and Union	T.Aruna	Samanu.Akhila	26
18	09-02-2023	Seminar	Categories of Function	P.Prathibha	Ch.Rajeshwari	28
19	09-02-2023	Seminar	Arrays and its Types	B.Swarnalatha	Mugala .Poojitha	34
20	24-01-2023	Seminar	Structure of C Programming	P.Bharathi	Avanthi	31
21	24-01-2023	Seminar	Programming Languages	P.Bharathi	Ellakula Ashwitha Patel	26
22	24-01-2023	Seminar	Topology	B.Swarnalatha	Nerella Punyasri	26
23	24-01-2023	Seminar	Algorithms	T.Aruna	A.Sravanthi	26
24	24-01-2023	Seminar	Flow Charts	P.Prathibha	Gatike Suvarna	14
25	06-12-2022	Seminar	OOPS Features	P.Bharathi	Tusse Vishesha	18
26	06-12-2022	Seminar	Java Features	B.Swarnalatha	N.Meghana	30
27	06-12-2022	Seminar	Inheritance	T.Aruna	G.Rama Anjali	35
28	06-12-2022	Seminar	Applet	P.Prathibha	Sri Nayana	22
29	06-12-2022	Seminar	Exceptions	T.Aruna	K.Shiva Pavani	26
30	01-12-2022	Seminar	Stack	P.Prathibha	Rajeshwari	26
31	01-12-2022	Seminar	Tree	P.Bharathi	M.Nandini	26
32	01-12-2022	Seminar	Linked List	B.Swarnalatha	T.Shirisha	14

33	01-12-2022	Seminar	Graph	T.Aruna	K.Poojitha	18
34	01-12-2022	Seminar	Heap Sort	P.Prathibha	Md Reshma	30

ACADEMIC YEAR 2021-2022

S.No	Date	Nature of Activity	Topic	Name(s) of the lecturer(s) involved	Details of Student (Name, Hall Ticket No etc.,)	No. of students participated
1	6-09-2021	Seminar	Block Diagram of Computer	Maneesha	ADI VINILA	26
2	6-09-2021	Seminar	Five Basic Operations of Computer System	Maneesha	AJMEERA NAVYA	25
3	13-09-2021	Seminar	Multimedia Systems with features or characteristics	T.Aruna	AMGOTH MAMATHA	35
4	13-09-2021	Seminar	Computer Graphics – 3D Translation Transformation	T.Aruna	ANUMULA RACHANA	45
5	5-10-2021	Seminar	Features of C++	T.Aruna	ARKALA SHIREESHA	36
6	05-10-2021	Seminar	Inheritance in C++	P.Prathibha	BADAVATH SHIRISHA	10
7	01-11-2021	Seminar	Polymorphism	P.Prathibha	BANALA RAVALI	9
8	01-11-2021	Seminar	Anchor Tag and Hyperlink	P.Prathibha	BANDA BHAVIKA	25
9	08-11-2021	Seminar	Linux System	B.Swarnalatha	BEESULA RASHMITHA	20

10	08-11-2021	Seminar	Types of Files in Python	B.Swarnalatha	BHUKYA PRASHANTHI	50
11	25-11-2021	Seminar	Data Acquisition	B.Swarnalatha	BHUKYA SUMALATHA	38

Memory Hierarchy:

Memory Hierarchy Design and its Characteristics

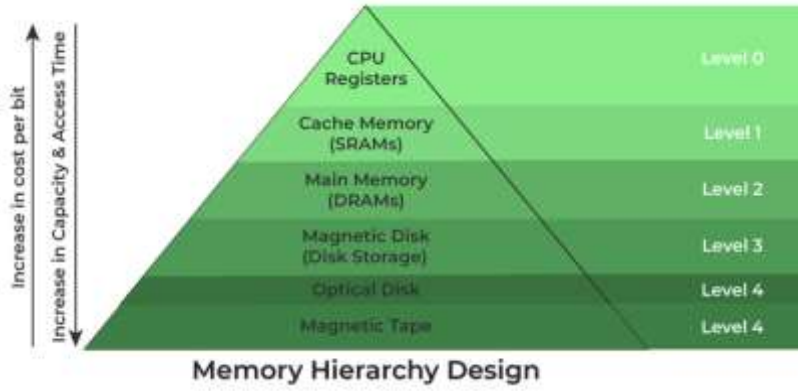
Why Memory Hierarchy is Required in the System?

Memory Hierarchy is one of the most required things in Computer Memory as it helps in optimizing the memory available in the computer. There are multiple levels present in the memory, each one having a different size, different cost, etc. Some types of memory like cache, and main memory are faster as compared to other types of memory but they are having a little less size and are also costly whereas some memory has a little higher storage value, but they are a little slower. Accessing of data is not similar in all types of memory, some have faster access whereas some have slower access.

Types of Memory Hierarchy

This Memory Hierarchy Design is divided into 2 main types:

- **External Memory or Secondary Memory:** Comprising of Magnetic Disk, Optical Disk, and Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via an I/O Module.
- **Internal Memory or Primary Memory:** Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.



Memory Hierarchy Design

Memory Hierarchy Design

1. Registers

Registers are small, high-speed memory units located in the CPU. They are used to store the most frequently used data and instructions. Registers have the fastest access time and the smallest storage capacity, typically ranging from 16 to 64 bits.

2. Cache Memory

Cache memory is a small, fast memory unit located close to the CPU. It stores frequently used data and instructions that have been recently accessed from the main memory. Cache memory is designed to minimize the time it takes to access data by providing the CPU with quick access to frequently used data.

3. Main Memory

Main memory, also known as RAM (Random Access Memory), is the primary memory of a computer system. It has a larger storage capacity than cache memory, but it is slower. Main memory is used to store data and instructions that are currently in use by the CPU.

Types of Main Memory

- **Static RAM:** Static RAM stores the binary information in flip flops and information remains valid until power is supplied. It has a faster access time and is used in implementing cache memory.
- **Dynamic RAM:** It stores the binary information as a charge on the capacitor. It requires refreshing circuitry to maintain the charge on the capacitors after a few milliseconds. It contains more memory cells per unit area as compared to SRAM.

4. Secondary Storage

Secondary storage, such as hard disk drives (HDD) and solid-state drives (SSD), is a non-volatile memory unit that has a larger storage capacity than main memory. It is used to store data and instructions that are not currently in use by the CPU. Secondary storage has the slowest access time and is typically the least expensive type of memory in the memory hierarchy.

5. Magnetic Disk

Magnetic Disks are simply circular plates that are fabricated with either a metal or a plastic or a magnetized material. The Magnetic disks work at a high speed inside the computer and these are frequently used.

6. Magnetic Tape

Magnetic Tape is simply a magnetic recording device that is covered with a plastic film. It is generally used for the backup of data. In the case of a magnetic tape, the access time for a computer is a little slower and therefore, it requires some amount of time for accessing the strip.

Characteristics of Memory Hierarchy

- **Capacity:** It is the global volume of information the memory can store. As we move from top to bottom in the Hierarchy, the capacity increases.
- **Access Time:** It is the time interval between the read/write request and the availability of the data. As we move from top to bottom in the Hierarchy, the access time increases.
- **Performance:** Earlier when the computer system was designed without a Memory Hierarchy design, the speed gap increased between the CPU registers and Main Memory due to a large difference in access time. This results in lower performance of the system and thus, enhancement was required. This enhancement was made in the form of Memory Hierarchy Design because of which the performance of the system increases. One of the most significant ways to increase system performance is minimizing how far down the memory hierarchy one has to go to manipulate data.
- **Cost Per Bit:** As we move from bottom to top in the Hierarchy, the cost per bit increases i.e. Internal Memory is costlier than External Memory.

Advantages of Memory Hierarchy

- It helps in removing some destruction, and managing the memory in a better way.
- It helps in spreading the data all over the computer system.
- It saves the consumer's price and time.

BUSINESS STRATEGY:

WHAT'S A BUSINESS STRATEGY?

Business strategy is the strategic initiatives a company pursues to create value for the organization and its stakeholders and gain a competitive advantage in the market. This strategy is crucial to a company's success and is needed before any goods or services are produced or delivered.

According to Harvard Business School Online's Business Strategy course, an effective strategy is built around three key questions:

1. How can my business create value for customers?
2. How can my business create value for employees?
3. How can my business create value by collaborating with suppliers?

Many promising business initiatives don't come to fruition because the company failed to build its strategy around value creation. Creativity is important in business, but a company won't last without prioritizing value.

The Importance of Business Strategy

A business strategy is foundational to a company's success. It helps leaders set organizational goals and gives companies a competitive edge. It determines various business factors, including:

- Price: How to price goods and services based on customer satisfaction and cost of raw materials
- Suppliers: Whether to source materials sustainably and from which suppliers
- Employee recruitment: How to attract and maintain talent
- Resource allocation: How to allocate resources effectively

Without a clear business strategy, a company can't create value and is unlikely to succeed.

CREATING VALUE

To craft a successful business strategy, it's necessary to obtain a thorough understanding of value creation. In the online course Business Strategy, Harvard Business School Professor Felix Oberholzer-Gee explains that, at its core, value represents a difference. For example, the difference between a customer's willingness to pay for a good or service and its price represents the value the business has created for the customer. This difference can be visualized with a tool known as the value stick.

The value stick has four components, representing the value a strategy can bring different stakeholders.




Advantages of DDBMS

- The database is easier to expand as it is already spread across multiple systems and it is not too complicated to add a system.
- The distributed database can have the data arranged according to different levels of transparency i.e data with different transparency levels can be stored at different locations.
- The database can be stored according to the departmental information in an organisation. In that case, it is easier for a organisational hierarchical access.
- there were a natural catastrophe such as fire or an earthquake all the data would not be destroyed it is stored at different locations.
- It is cheaper to create a network of systems containing a part of the database. This database can also be easily increased or decreased.
- Even if some of the data nodes go offline, the rest of the database can continue its normal functions.

Disadvantages of DDBMS

- The distributed database is quite complex and it is difficult to make sure that a user gets a uniform view of the database because it is spread across multiple locations.
- This database is more expensive as it is complex and hence, difficult to maintain.
- It is difficult to provide security in a distributed database as the database needs to be secured at all the locations it is stored. Moreover, the infrastructure connecting all the nodes in a distributed database also needs to be secured.
- It is difficult to maintain data integrity in the distributed database because of its nature. There can also be data redundancy in the database as it is stored at multiple locations.
- The distributed database is complicated and it is difficult to find people with the necessary experience who can manage and maintain it.

File Handling in Python:

What is a file handling in Python?

A. File handling in Python is the process of reading from and writing to files using the built-in file objects. It allows developers to work with files to store and manipulate data.

Q2. What are the types of file in file handling in Python?

A. The two main types of files in file handling in Python are text files and binary files. Text files store textual data in ASCII or Unicode format, while binary files store non-textual data such as images or audio files.

Q3. How do you write file handling in Python?

A. To write file handling in Python, you can use the built-in open() function to create a file object, and then use the appropriate file methods to perform read/write operations. Once done, you should close the file using the close() method

Q4. What are the file operations in Python?

A. The file operations in Python include opening a file, reading from a file, writing to a file, appending to a file, seeking a specific position in a file, and closing a file. These operations allow developers to manipulate files and their contents in a variety of ways.

Q5. How many types of file operations are there in Python?

A. There are several types of file operations in Python, including read, write, append, seek, flush, and truncate. These operations provide a wide range of functionality for working with files, making it easy to store, retrieve, and manipulate data stored in files.




Programming Languages:

1. Procedural programming languages

A procedural language follows a sequence of statements or commands in order to achieve a desired output. Each series of steps is called a procedure, and a program written in one of these languages will have one or more procedures within it. Common examples of procedural languages include:

- C and C++
- Java
- Pascal
- BASIC

2. Functional programming languages

Rather than focusing on the execution of statements, functional languages focus on the output of mathematical functions and evaluations. Each function–a reusable module of code–performs a specific task and returns a result. The result will vary depending on what data you input into the function. Some popular functional programming languages include:

- Scala
- Erlang
- Haskell
- Elixir
- F#

3. Object-oriented programming languages (OOP)

This type of language treats a program as a group of objects composed of data and program elements, known as attributes and methods. Objects can be reused within a program or in other programs. This makes it a popular language type for complex programs, as code is easier to reuse and scale. Some common object-oriented languages include:

- Java
- Python
- PHP
- C++
- Ruby

Features of Java

Object Oriented:

In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

Platform Independent

Unlike many other programming languages including **C** and **C++**, when Java is compiled, it is not compiled into platform specific machine, rather into platform-independent byte code. This byte code is distributed over the web and interpreted by the **Java Virtual Machine** (JVM) on whichever platform it is being run on.

Simple

Java is designed to be easy to learn. If you understand the basic **concept of OOP Java**, it would be easy to master.Secure

With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

Architecture-neutral

Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.

Portable

Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

Robust

Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.

Multithreaded

With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

Interpreted

Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.

High Performance

With the use of Just-In-Time compilers, Java enables high performance.

Distributed

Java is designed for the distributed environment of the internet.

Dynamic

Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time.

Inheritance:

Why Do We Need Java Inheritance?

- **Code Reusability:** The code written in the Superclass is common to all subclasses. Child classes can directly use the parent class code.
- **Method Overriding:** Method Overriding is achievable only through Inheritance. It is one of the ways by which Java achieves Run Time Polymorphism.
- **Abstraction:** The concept of abstract where we do not have to provide all details is achieved through inheritance. Abstraction only shows the functionality to the user.

Important Terminologies Used in Java Inheritance

- **Class:** Class is a set of objects which shares common characteristics/ behavior and common properties/ attributes. Class is not a real-world entity. It is just a template or blueprint or prototype from which objects are created.
- **Super Class/Parent Class:** The class whose features are inherited is known as a superclass(or a base class or a parent class).
- **Sub Class/Child Class:** The class that inherits the other class is known as a subclass(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.
- **Reusability:** Inheritance supports the concept of "reusability", i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class

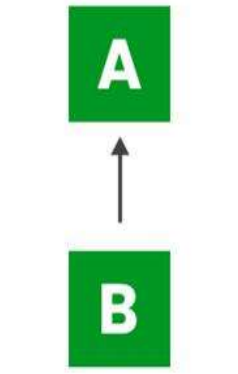
Java Inheritance Types

Below are the different types of inheritance which are supported by Java.

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Multiple Inheritance
5. Hybrid Inheritance

1. Single Inheritance

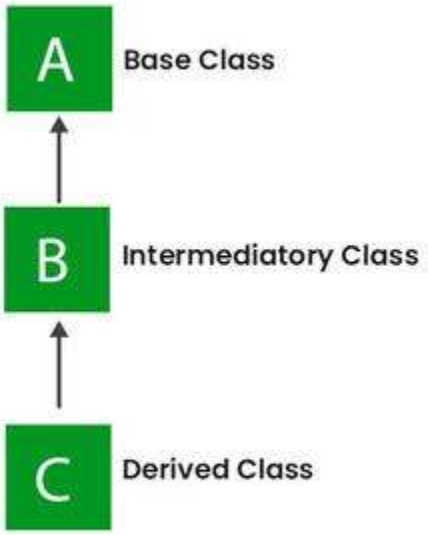
In single inheritance, subclasses inherit the features of one superclass. In the image below, class A serves as a base class for the derived class B.



Single Inheritance

2. Multilevel Inheritance

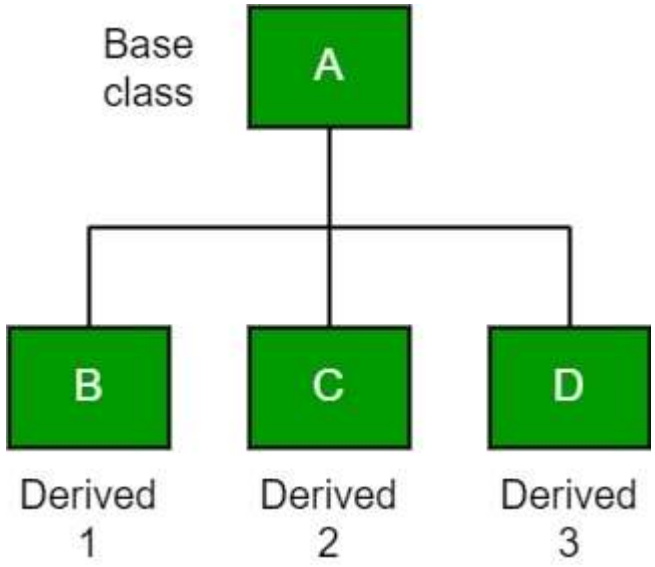
In Multilevel Inheritance, a derived class will be inheriting a base class, and as well as the derived class also acts as the base class for other classes. In the below image, class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C. In Java, a class cannot directly access the grandparent's members.



Multilevel Inheritance

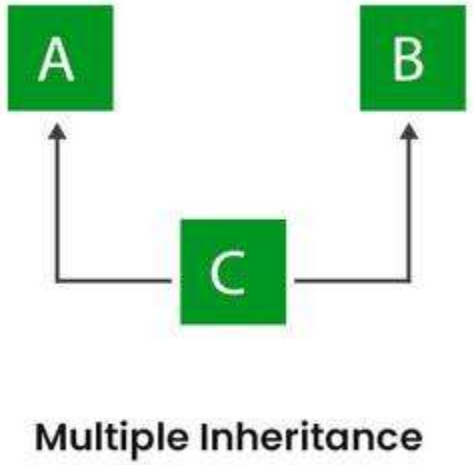
3. Hierarchical Inheritance

In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass. In the below image, class A serves as a base class for the derived classes B, C, and D.



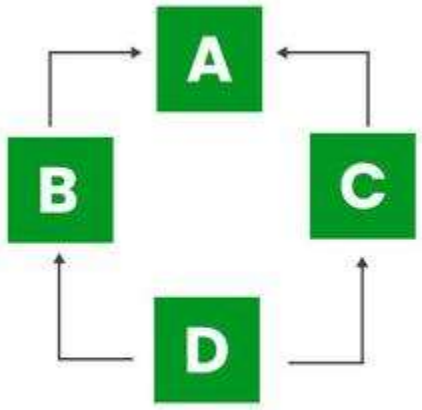
4. Multiple Inheritance (Through Interfaces)

In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes. Please note that Java does **not** support multiple inheritances with classes. In Java, we can achieve multiple inheritances only through Interfaces. In the image below, Class C is derived from interfaces A and B.



5. Hybrid Inheritance

It is a mix of two or more of the above types of inheritance. Since Java doesn't support multiple inheritances with classes, hybrid inheritance involving multiple inheritance is also not possible with classes. In Java, we can achieve hybrid inheritance only through Interfaces if we want to involve multiple inheritance to implement Hybrid inheritance.
However, it is important to note that Hybrid inheritance does not necessarily require the use of Multiple Inheritance exclusively. It can be achieved through a combination of Multilevel Inheritance and Hierarchical Inheritance with classes, Hierarchical and Single Inheritance with classes. Therefore, it is indeed possible to implement Hybrid inheritance using classes alone, without relying on multiple inheritance type.



Hybrid Inheritance

Advantages Of Inheritance in Java:

1. Code Reusability: Inheritance allows for code reuse and reduces the amount of code that needs to be written. The subclass can reuse the properties and methods of the superclass, reducing duplication of code.
2. Abstraction: Inheritance allows for the creation of abstract classes that define a common interface for a group of related classes. This promotes abstraction and encapsulation, making the code easier to maintain and extend.
3. Class Hierarchy: Inheritance allows for the creation of a class hierarchy, which can be used to model real-world objects and their relationships.
4. Polymorphism: Inheritance allows for polymorphism, which is the ability of an object to take on multiple forms. Subclasses can override the methods of the superclass, which allows them to change their behavior in different ways.

Disadvantages of Inheritance in Java:

1. Complexity: Inheritance can make the code more complex and harder to understand. This is especially true if the inheritance hierarchy is deep or if multiple inheritances is used.
2. Tight Coupling: Inheritance creates a tight coupling between the superclass and subclass, making it difficult to make changes to the superclass without affecting the subclass.

Applet

What is Applet?

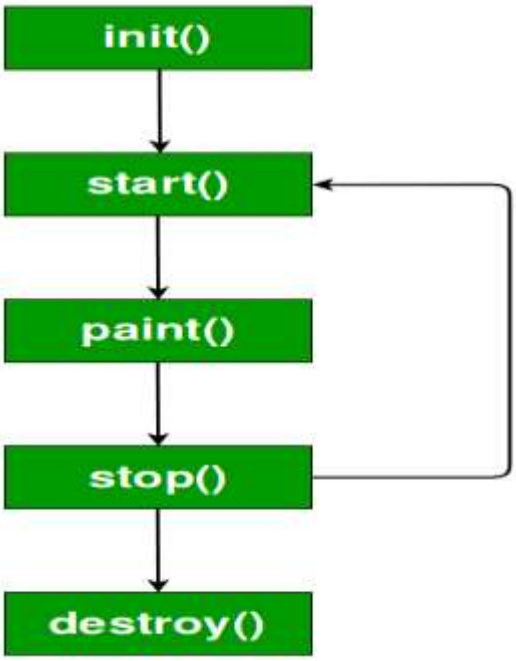
An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server.

Applets are used to make the website more dynamic and entertaining.

Important points :

1. All applets are sub-classes (either directly or indirectly) of *java.applet.Applet* class.
2. Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
3. In general, execution of an applet does not begin at main() method.
4. Output of an applet window is not performed by *System.out.println()*. Rather it is handled with various AWT methods, such as *drawString()*.

Life cycle of an applet :



It is important to understand the order in which the various methods shown in the above image are called. When an applet begins, the following methods are called, in this sequence:

1. init( )
2. start( )
3. paint( )

When an applet is terminated, the following sequence of method calls takes place:

1. stop( )
2. destroy( )

Let's look more closely at these methods.

1. init( ) : The **init( )** method is the first method to be called. This is where you should initialize variables. This method is called **only once** during the run time of your applet.

2. start( ) : The **start( )** method is called after **init( )**. It is also called to restart an applet after it has been stopped. Note that **init( )** is called once i.e. when the first time an applet is loaded whereas **start( )** is called each time an applet's HTML document is displayed onscreen. So, if a user leaves a web page and comes back, the applet resumes execution at **start( )**.

3. paint( ) : The **paint( )** method is called each time an AWT-based applet's output must be redrawn. This situation can occur for several reasons. For example, the window in which the applet is running may be overwritten by another window and then uncovered. Or the applet window may be minimized and then restored.

paint( ) is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, **paint( )** is called.

The **paint( )** method has one parameter of type Graphics. This parameter will contain the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.

Note: This is the only method among all the method mention above, which is parameterized. It's prototype is

public void paint(Graphics g)

where g is an object reference of class Graphic.

Exceptions

Exception Handling in Java is one of the effective means to handle runtime errors so that the regular flow of the application can be preserved. Java Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc

In Java, Exception is an unwanted or unexpected event, which occurs during the execution of a program, i.e. at run time, that disrupts the normal flow of the program's instructions. Exceptions can be caught and handled by the program. When an exception occurs within a method, it creates an object. This object is called the exception object. It contains information about the exception, such as the name and description of the exception and the state of the program when the exception occurred.

Major reasons why an exception Occurs

- Invalid user input
- Device failure
- Loss of network connection
- Physical limitations (out-of-disk memory)
- Code errors
- Opening an unavailable file

Errors represent irrecoverable conditions such as Java virtual machine (JVM) running out of memory, memory leaks, stack overflow errors, library incompatibility, infinite recursion, etc. Errors are usually beyond the control of the programmer, and we should not try to handle errors.

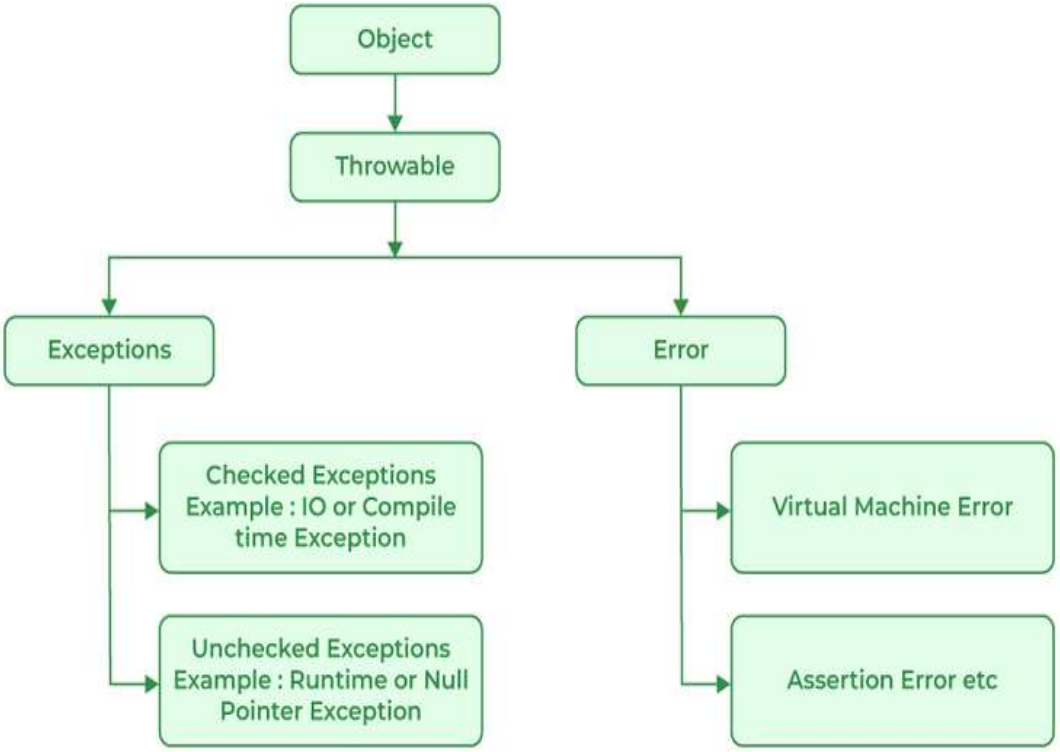
Difference between Error and Exception

Let us discuss the most important part which is the **differences between Error and Exception** that is as follows:

- **Error:** An Error indicates a serious problem that a reasonable application should not try to catch.
- **Exception:** Exception indicates conditions that a reasonable application might try to catch.

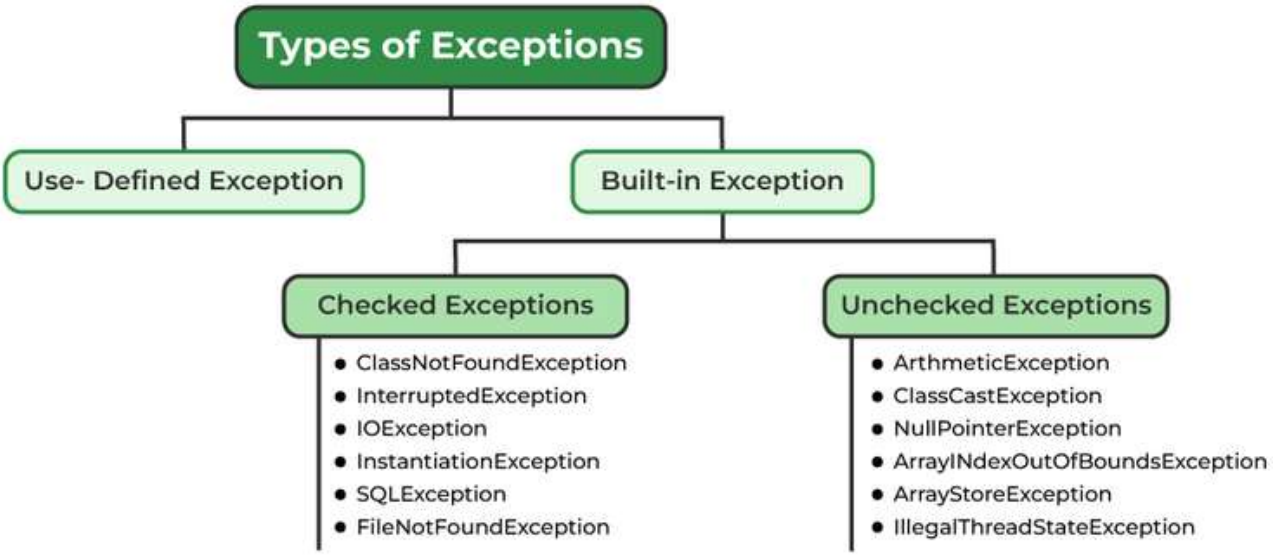
Exception Hierarchy

All exception and error types are subclasses of the class **Throwable**, which is the base class of the hierarchy. One branch is headed by **Exception**. This class is used for exceptional conditions that user programs should catch. NullPointerException is an example of such an exception. Another branch, **Error** is used by the Java run-time system(JVM) to indicate errors having to do with the run-time environment itself(JRE). StackOverflowError is an example of such an error.



Types of Exceptions

Java defines several types of exceptions that relate to its various class libraries. Java also allows users to define their own exceptions.



Exceptions can be categorized in two ways:

1. **Built-in Exceptions**
   - Checked Exception
   - Unchecked Exception
2. **User-Defined Exceptions**

Let us discuss the above-defined listed exception that is as follows:

1. Built-in Exceptions

Built-in exceptions are the exceptions that are available in Java libraries. These exceptions are suitable to explain certain error situations.

- **Checked Exceptions:** Checked exceptions are called compile-time exceptions because these exceptions are checked at compile-time by the compiler.

- **Unchecked Exceptions:** The unchecked exceptions are just opposite to the checked exceptions. The compiler will not check these exceptions at compile time. In simple words, if a program throws an unchecked exception, and even if we didn't handle or declare it, the program would not give a compilation error.

Note: *For checked vs unchecked exception, see* *Checked vs Unchecked Exceptions*

2. User-Defined Exceptions:

Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, users can also create exceptions, which are called 'user-defined Exceptions'.

The ***advantages of Exception Handling in Java*** are as follows:

1. Provision to Complete Program Execution
2. Easy Identification of Program Code and Error-Handling Code
3. Propagation of Errors
4. Meaningful Error Reporting
5. Identifying Error Types

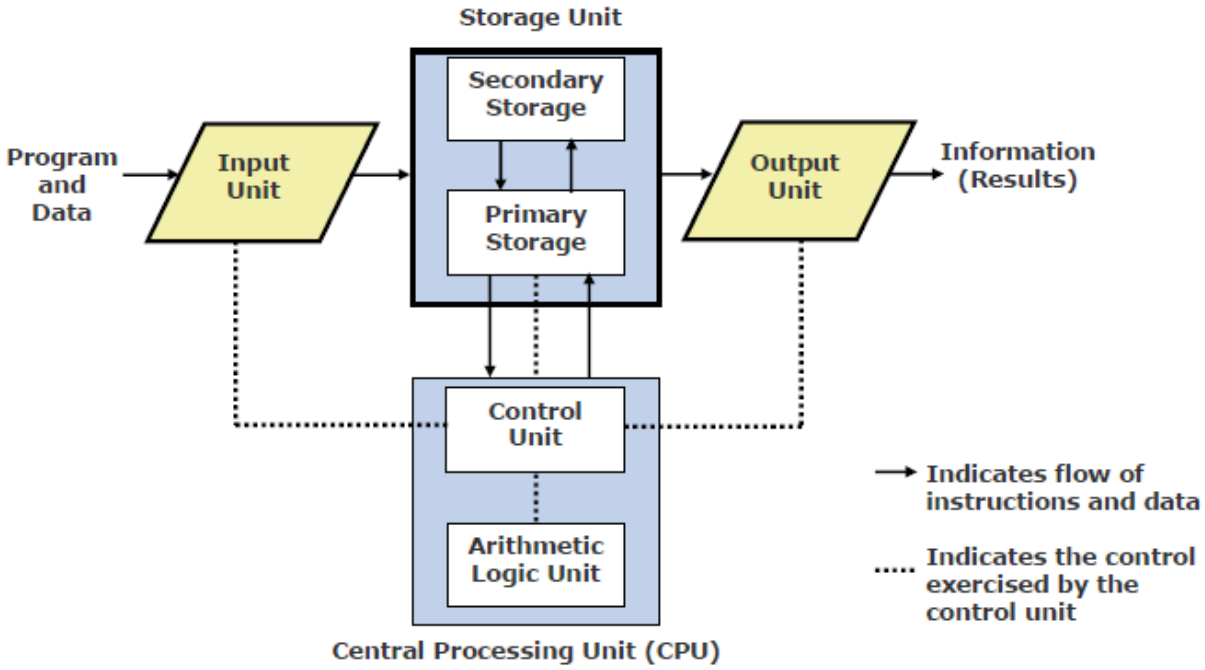
Stack:

What is Stack?

Stack is a linear data structure that follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out). LIFO implies that the element that is inserted last, comes out first and FILO implies that the element that is inserted first, comes out last.

There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO(Last In First Out)

Block Diagram of a Computer



Input Unit-

An input unit of a computer system performs the following functions-

1. It accepts (or read) data and instructions from the outside world (for example- in English)

2. It converts these data and instructions in computer acceptable form (i.e. into Binary/Coded Form)

3. It supplies the converted data and instructions to the computer system for further processing.

Output Unit-

An output unit of a computer system performs the following functions-

1. It accepts the results produced by the computer, which are in coded form and hence, cannot be easily understood by the user.

2. It converts these coded results to human acceptable (readable) form.

3. It supplies the converted results to outside world.

Storage Unit-

The storage unit of a computer system holds (or stores) the following-

1. Data and instructions required for processing (received from input devices)

2. Intermediate results of processing.

3. Final results of processing, before they are released to an output device.

There are two types of storage in a computer system-

Primary Storage

- Instructions of programs running in the computer system, are stored in the primary memory
- It is used to hold data, intermediate results, and results of ongoing processing of job(s).
- It is fast in operation.

- It is small in storage capacity.
- It is expensive.
- It is volatile in nature (i.e. if power turns off then data stored in it is lost)
- Example - Random Access Memory (RAM)

Secondary Storage

- It is used to hold stored program instructions.
- It is used to hold data and information of stored jobs.
- It is large in storage capacity.
- Due to large storage capacity, It is slower in operation than primary storage.
- It is much cheaper than primary storage.
- It is non-volatile in nature (i.e. if power turns off then data stored in it will not lost)
- Example - Hard Disk, Pen Drive, etc.

Central Processing Unit (CPU)

- It is the brain of a computer system.
- It is responsible for controlling the operations of all other units of a computer system.
- It consist of an Arithmetic Logic Unit and Control Unit.

Figure-2: Components of a Central Processing Unit (CPU)

Arithmetic Logic Unit (ALU)

Arithmetic Logic Unit of a computer system is a place where the actual execution of instructions takes place during processing operation. As the name suggests, it is responsible for performing all arithmetic operations (such as addition, subtraction, multiplication. division etc.) and logical operations (such as less than, greater than, equal to, etc.).

Control Unit

Control Unit of a computer system manages and coordinates the operations of all other components of the computer system.

Note-

One more part is there inside the Central Processing Unit called as **CPU Registers.**

Registers have very small storage capacity with very high execution speed. They are used by the processor to store small amount of intermediate data that are needed during processing of instructions. There are many registers available in the CPU, each has some name and is used for specific purpose. Some of them are-

- Accumulator (AC)
- Memory Address Register (MAR)
- Memory Data Register (MDR)
- Program Counter (PC)
- Instruction Register (IR)
- General Purpose Registers, etc.

Five Basic Operations of Computer System

1. Inputting - It is the process of entering data and instructions into the computer system.

2. Storing - It is the process of saving data and instructions in the computer system so that they are readily available for initial or intermediary processing whenever required.

3. Processing - It is the process of performing arithmetic operations (such as addition, subtraction, multiplication, division, etc.) or logical operations (such as comparisons like less than, greater than, less than or equal to, greater than or equal to, equal to, not equal to, etc.) on data so that it can be converted into useful information.

4. Outputting - It is the process of producing useful information or results for the user such as a visual display on the screen or a printed report.

5. Controlling - It is the process of directing the method and sequence in which all of the above operations are performed in the computer system.

Multimedia Systems with features or characteristics

What are Multimedia Systems:

A multimedia system is responsible for developing a multimedia application. A multimedia application is a bundle of different kinds of data. A multimedia computer system is one that can create, integrate, store, retrieve delete two or more types of media materials in digital form, such as audio, image, video, and text information.

Following are some major characteristics or features of a Multimedia System:

Very High Processing Power:

To deal with large amount of data, very high processing power is used.

File System:

File system must be efficient to meet the requirements of continuous media. These media files requires very high-disk bandwidth rates. Disks usually have low transfer rates and high latency rates. To satisfy the requirements for multimedia data, disk schedulers must reduce the latency time to ensure high bandwidth.

File formats that support multimedia:

Multimedia data consists of a variety of media formats or file representation including ,JPEG, MPEG, AVI, MID, WAV, DOC, GIF,PNG, etc. AVI files can contain both audio and video data in a file container that allows synchronous audio-with-video playback. Like the DVD video format, AVI files support multiple streaming audio and video. Because of restrictions on the conversion from one format to the other, the use of the data in a specific format has been limited as well.

Input/Output:

In multimedia applications, the input and output should be continuous and fast. Real-time recording as well as playback of data are common in most of the multimedia applications which need efficient I/O.

Operating System:

The operating system must provide a fast response time for interactive applications. High throughput for batch applications, and real-time scheduling,

Storage and Memory:

Multimedia systems require storage for large capacity objects such as video, audio, animation and images. Depending on the compression scheme and reliability video and audio require large amount of memory.

Network Support:
It includes internet, intranet, LAN, WAN, ATM, Mobile telephony and others. In recent years, there has been a tremendous growth of multimedia applications on the internet like streaming video, IP telephony, interactive games, teleconferencing, virtual world, distance learning and so on. These multimedia networking applications are referred as continuous-media applications and require high communication latency. Communication Latency is the time it takes for a data packet to be received by the remote computer.

Software Tools:
For the development of multimedia applications, the various software tools like programming languages, graphics software's, multimedia editing software's scripting languages: authoring tools, design software's etc are required. In addition to these the device drivers are required for interfacing the multimedia peripherals.

Computer Graphics – 3D Translation Transformation

3-D Transformation: In very general terms a 3D model is a mathematical representation of a physical entity that occupies space. In more practical terms, a 3D model is made of a description of its shape and a description of its color appearance.3-D Transformation is the process of manipulating the view of a three-D object with respect to its original position by modifying its physical attributes through various methods of transformation like Translation, Scaling, Rotation, Shear, etc.

Properties of 3-D Transformation:

- Lines are preserved,
- Parallelism is preserved,
- Proportional distances are preserved.

One main categorization of a 3D object's representation can be done by considering whether the surface or the volume of the object is represented:

Boundary-based: the surface of the 3D object is represented. This representation is also called b-rep. Polygon meshes, implicit surfaces, and parametric surfaces, which we will describe in the following, are common representations of this type.

Volume-based: the volume of the 3D object is represented. Voxels and Constructive Solid Geometry (CSG) Are commonly used to represent volumetric data.

Types of Transformations:

1. Translation
2. Scaling
3. Rotation
4. Shear
5. Reflection

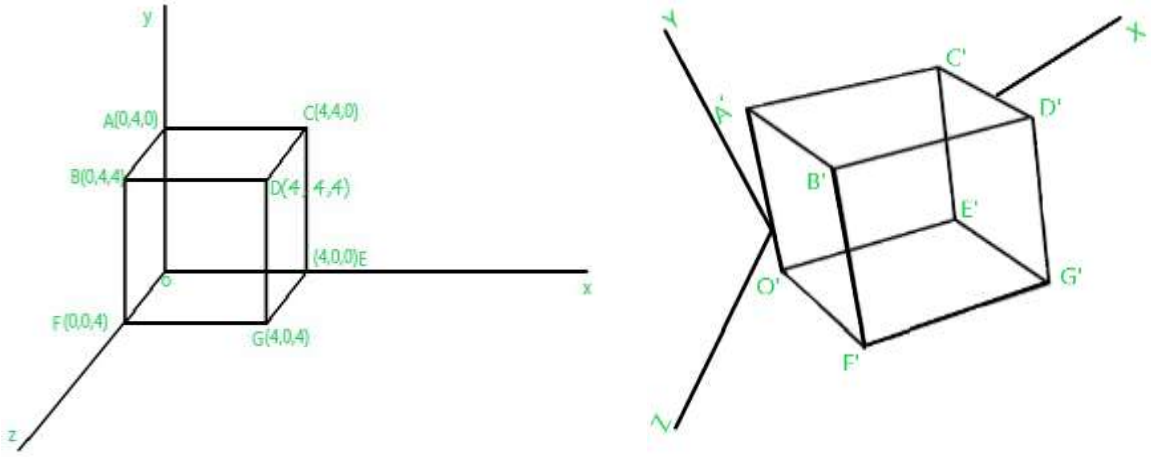
Translation: It is the process of changing the relative location of a 3-D object with respect to the original position by changing its coordinates. Translation transformation matrix in the 3-D image is shown as –

$$\mathbf{T}[\mathbf{x},\mathbf{y},\mathbf{z}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ D_x & D_y & D_z & 1 \end{bmatrix}$$

Where D_x, D_y, D_z are the Translation distances, let a point in 3D space is P(x, y, z) over which we want to apply Translation Transformation operation and we are given with translation distance [D_x, D_y, D_z] So, new position of the point after applying translation operation would be –

Problem: Perform translation transformation on the following figure where the given translation distances are $D_x = 2$, $D_y = 4$, $D_z = 6$.

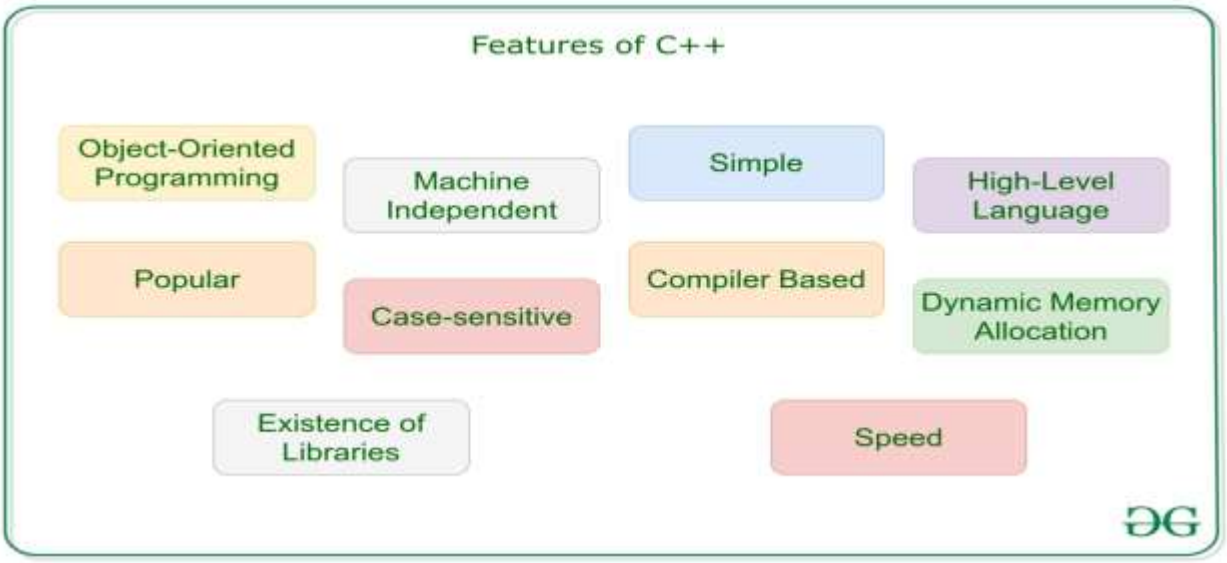
Solution: On applying Translation Transformation we get corresponding points –



Features of C++

C++ is a general-purpose programming language that was developed as an enhancement of the C language to include an object-oriented paradigm. It is an imperative and **compiled** language. C++ has a number of features, including:

- Object-Oriented Programming
- Machine Independent
- Simple
- High-Level Language
- Popular
- Case-sensitive
- Compiler Based
- Dynamic Memory Allocation
- Memory Management
- Multi-threading



1. Object-Oriented Programming

C++ is an Object-Oriented Programming Language, unlike C which is a procedural programming language. This is the most important feature of **C++**. It can create/destroy objects while programming. Also, It can create blueprints with which objects can be created. We have discussed the Object-Orient Programming Concepts in C++ in this article.
Concepts of Object-oriented programming Language:

- Class
- Objects
- Encapsulation
- Polymorphism
- Inheritance
- Abstraction

2. Machine Independent

A C++ executable is not platform-independent (compiled programs on Linux won't run on Windows), however, they are machine-independent. Let us understand this feature of C++ with the help of an example. Suppose you have written a piece of code that can run on Linux/Windows/Mac OSx which makes the C++ Machine Independent but the executable file of the C++ cannot run on different operating systems.

3. Simple

It is a simple language in the sense that programs can be broken down into logical units and parts, has rich library support and has a variety of data types. Also, the Auto Keyword of C++ makes life easier.

Auto Keyword

The idea of the auto keyword was to form the C++ compiler to deduce the data type while compiling instead of making you declare the data type every freaking time. Do keep in mind that you cannot declare something without an initializer. There must be some way for the compiler to deduce your type.

4. High-Level Language

C++ is a High-Level Language, unlike C which is a Mid-Level Programming Language. It makes life easier to work in C++ as it is a high-level language it is closely associated with the human-comprehensible English language.

5. Popular

C++ can be the base language for many other programming languages that supports the feature of object-oriented programming. **Bjarne Stroustrup** found Simula 67, the first object-oriented language ever, lacking simulations, and decided to develop C++.

6. Case-sensitive

It is clear that C++ is a case-sensitive programming language. For example, **cin** is used to take input from the input stream. But if the **"Cin"** won't work. Other languages like HTML and MySQL are not case-sensitive languages.

7. Compiler Based

C++ is a compiler-based language, unlike Python. That is C++ programs used to be compiled and their executable file is used to run them. C++ is a relatively faster language than Java and Python.

8. Dynamic Memory Allocation

When the program executes in C++ then the variables are allocated the dynamical heap space. Inside the functions, the variables are allocated in the stack space. Many times, We are not aware in

advance how much memory is needed to store particular information in a defined variable and the size of required memory can be determined at run time.

9. Memory Management

- C++ allows us to allocate the memory of a variable or an array in run time. This is known as Dynamic Memory Allocation.

- In other programming languages such as Java and Python, the compiler automatically manages the memories allocated to variables. But this is not the case in C++.
- In C++, the memory must be de-allocated dynamically allocated memory manually after it is of no use.
- The allocation and deallocation of the memory can be done using the new and delete operators respectively.

10. Multi-threading

- Multithreading is a specialized form of multitasking and multitasking is a feature that allows your system to execute two or more programs concurrently. In general, there are two sorts of multitasking: process-based and thread-based.
- Process-based multitasking handles the concurrent execution of programs. Thread-based multitasking deals with the multiprogramming of pieces of an equivalent program.
- A multithreaded program contains two or more parts that will run concurrently. Each part of such a program is named a thread, and every thread defines a separate path of execution.
- C++ doesn't contain any built-in support for multithreaded applications. Instead, it relies entirely upon the OS to supply this feature.

Inheritance in C++

The capability of a class to derive properties and characteristics from another class is called **Inheritance**. Inheritance is one of the most important features of Object-Oriented Programming.

Inheritance is a feature or a process in which, new classes are created from the existing classes. The new class created is called "derived class" or "child class" and the existing class is known as the "base class" or "parent class". The derived class now is said to be inherited from the base class.

When we say derived class inherits the base class, it means, the derived class inherits all the properties of the base class, without changing the properties of base class and may add new features to its own. These new features in the derived class

will not affect the base class. The derived class is the specialized class for the base class.

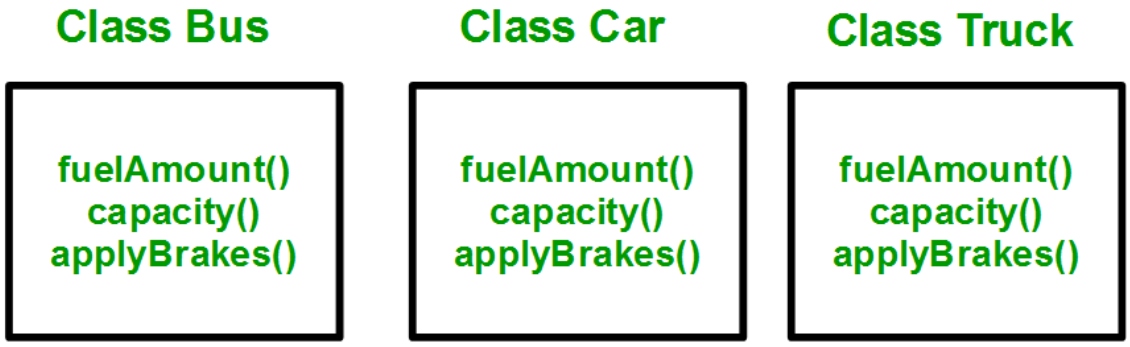
- **Sub Class:** The class that inherits properties from another class is called Subclass or Derived Class.
- **Super Class:** The class whose properties are inherited by a subclass is called Base Class or Superclass.

The article is divided into the following subtopics:

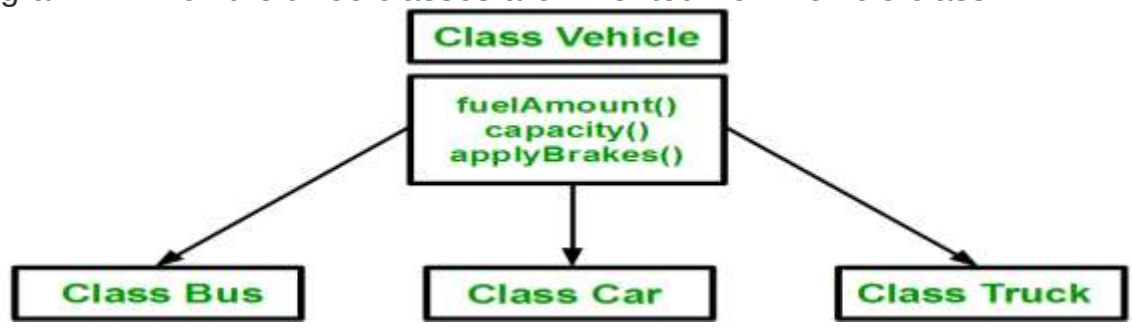
- Why and when to use inheritance?
- Modes of Inheritance
- Types of Inheritance

Why and when to use inheritance?

Consider a group of vehicles. You need to create classes for Bus, Car, and Truck. The methods fuelAmount(), capacity(), applyBrakes() will be the same for all three classes. If we create these classes avoiding inheritance then we have to write all of these functions in each of the three classes as shown below figure:



You can clearly see that the above process results in duplication of the same code 3 times. This increases the chances of error and data redundancy. To avoid this type of situation, inheritance is used. If we create a class Vehicle and write these three functions in it and inherit the rest of the classes from the vehicle class, then we can simply avoid the duplication of data and increase re-usability. Look at the below diagram in which the three classes are inherited from vehicle class:



Using inheritance, we have to write the functions only one time instead of three times as we have inherited the rest of the three classes from the base class (Vehicle).
Implementing inheritance in C++: For creating a sub-class that is inherited from the base class we have to follow the below syntax.
Derived Classes: A Derived class is defined as the class derived from the base class.
Syntax:

```
class  <derived_class_name> : <access-specifier> <base_class_name>
```

```
{
        //body
}
```

Where
class — keyword to create a new class
derived_class_name — name of the new class, which will inherit the base class
access-specifier — either of private, public or protected. If neither is specified, PRIVATE is taken as default
base-class-name — name of the base class

Note: A derived class doesn't inherit ***access*** to private data members. However, it does inherit a full parent object, which contains any private members which that class declares.

Example:

```
1. class ABC : private XYZ          //private derivation
        {           }
2. class ABC : public XYZ           //public derivation
        {           }
3. class ABC : protected XYZ        //protected derivation
        {           }
4. class ABC: XYZ                   //private derivation by default
{       }
```

Modes of Inheritance: There are 3 modes of inheritance.

1. **Public Mode**: If we derive a subclass from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in the derived class.
2. **Protected Mode**: If we derive a subclass from a Protected base class. Then both public members and protected members of the base class will become protected in the derived class.
3. **Private Mode**: If we derive a subclass from a Private base class. Then both public members and protected members of the base class will become Private in the derived class.

.

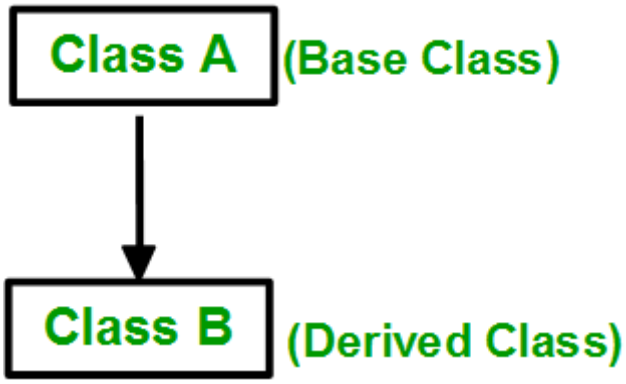
Base class member access specifier	Type of Inheritence		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Not accessible (Hidden)	Not accessible (Hidden)	Not accessible (Hidden)

Types Of Inheritance:-

1. Single inheritance
2. Multilevel inheritance
3. Multiple inheritance
4. Hierarchical inheritance
5. Hybrid inheritance

Types of Inheritance in C++

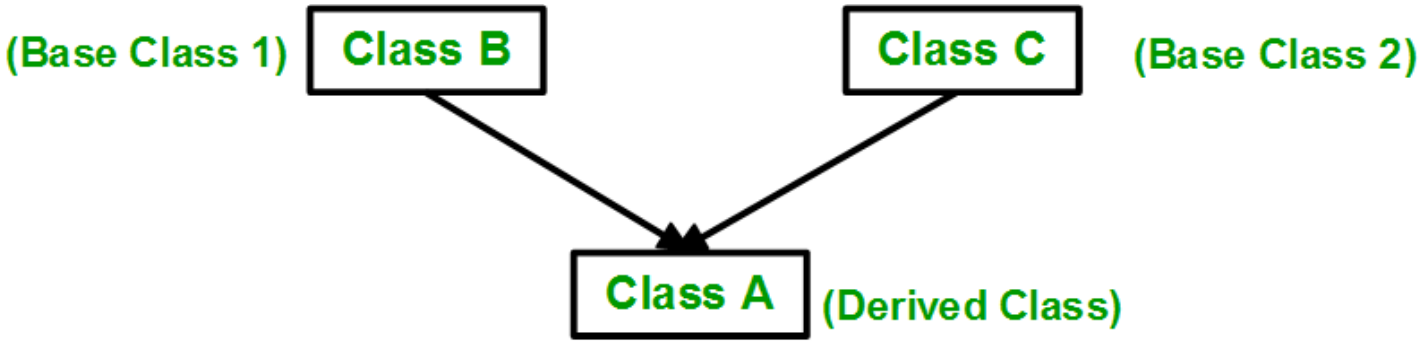
1. Single Inheritance: In single inheritance, a class is allowed to inherit from only one class. i.e. one subclass is inherited by one base class only.



Syntax:

```
class subclass_name : access_mode base_class
{
  // body of subclass
};
OR
class A
{
... .. ...
};
class B: public A
{
... .. ...
};
```

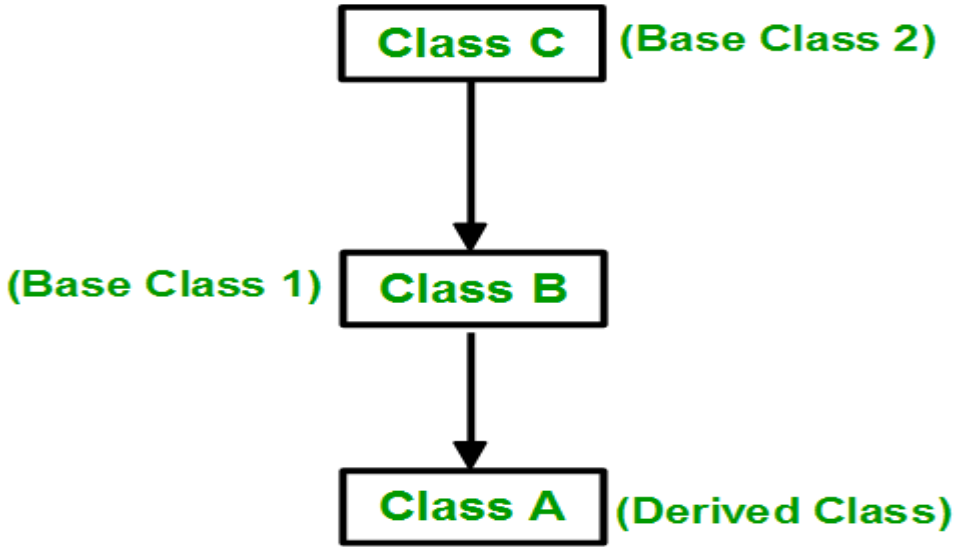
2. Multiple Inheritance: Multiple Inheritance is a feature of C++ where a class can inherit from more than one class. i.e one **subclass** is inherited from more than one **base class**.



Syntax:

```
class subclass_name : access_mode base_class1, access_mode base_class2,
....
{
  // body of subclass
};
class B
{
... .. ...
};
class C
{
... .. ...
};
class A: public B, public C
{
... ... ...
};
```

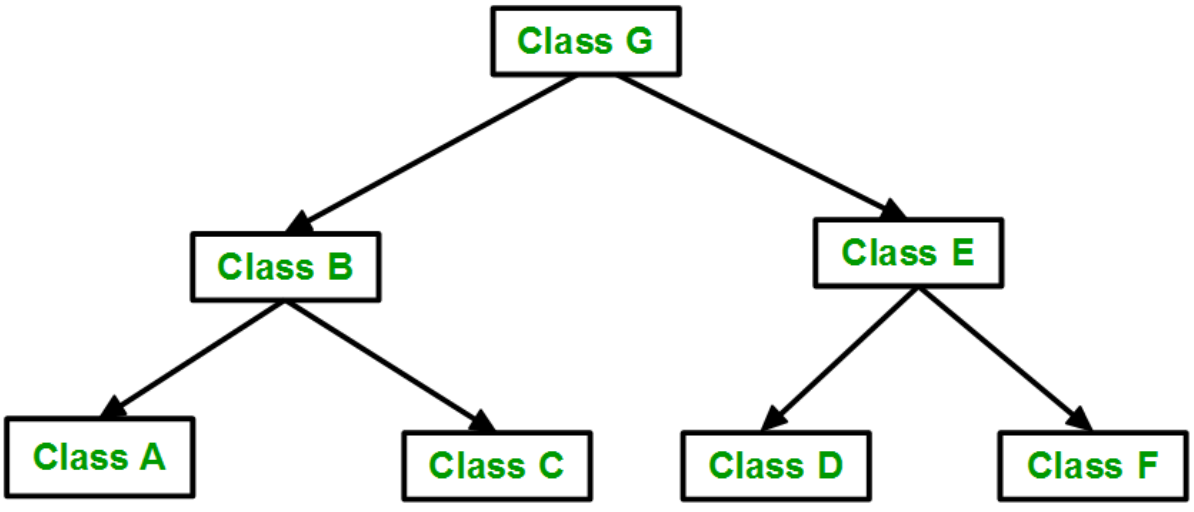
3. Multilevel Inheritance: In this type of inheritance, a derived class is created from another derived class.



Syntax:-

```
class C
{
... .. ...
};
class B:public C
{
... .. ...
};
class A: public B
{
... ... ...
};
```

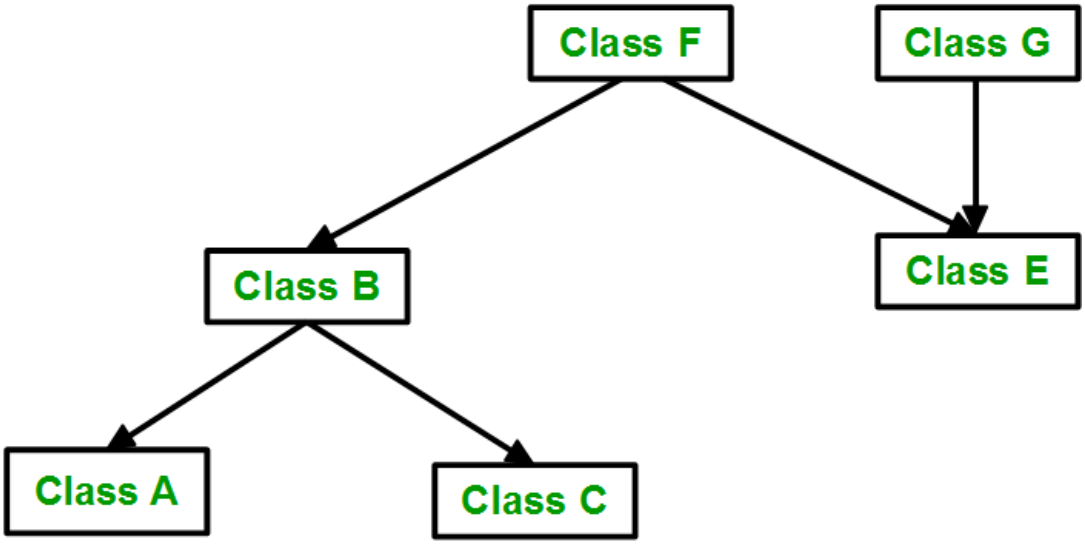
4. Hierarchical Inheritance: In this type of inheritance, more than one subclass is inherited from a single base class. i.e. more than one derived class is created from a single base class.



Syntax:-

```
class A
{
    // body of the class A.
}
class B : public A
{
    // body of class B.
}
class C : public A
{
    // body of class C.
}
class D : public A
{
    // body of class D.
}
```

5. Hybrid (Virtual) Inheritance: Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.
Below image shows the combination of hierarchical and multiple inheritances:



6. A special case of hybrid inheritance: Multipath inheritance:
A derived class with two base classes and these two base classes have one common base class is called multipath inheritance. Ambiguity can arise in this type of inheritance.


GPS Map Camera
Hanamkonda, Telangana, India
2G2J+65R, Waddepally, Wadepally, Hanamkonda,
Telangana 506002, India
Lat 18.000927°
Long 79.530456°
13/10/22 11:19 AM GMT +05:30
Google