1.WHAT IS JAVA?

Ans: java is a high level programming language. And platform independent language .It was developed by sun microsystem. Java is a object oriented, secured ,robust language.

Java language can used to develop various applications like

*software or standalone applications

*web applications

*mobile applications

*enterprise applications

*gamming applications

2.WHAT IS JDK?

Ans: It stands for Java Devleopment Kit. It is a developer version that means by using JDK we can develop as well as execute our java programs.

In order to develop and execute it supports various JDK tools which are as follows :

a) javac : java compiler, responsible for compilation.

b) java : Java launcher,responsible for executing java program.

c) jdb : java debugger, for debugging purpose

d) jconsole : java Console, to display the output in the console.

e) javap : java Profiler, To get the details of a class

f) javadoc : Java documentation , for Generating java documentation.

3.DRAW THE JDK DIAGRAM?

4.HOW MANY DATA TYPES WE HAVE?

A data type describes, what type of value the varaiable will hold. In have we have 2 types of data types :

1.reference data gype

2.primitive

5.ASCENDING ORDER OF NUMERIC DATA TYPES?

Ans: Here are the numeric data types in Java, listed in ascending order of their range and precision:

Integer Types

1. byte (8-bit): -128 to 127

2. short (16-bit): -32,768 to 32,767

3. int (32-bit): -2,147,483,648 to 2,147,483,647

4. long (64-bit): -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Floating-Point Types

1. float (32-bit): approximately 3.4e-38 to 3.4e+38 (6-7 decimal places)

2. double (64-bit): approximately 1.8e-308 to 1.8e+308 (15-16 decimal places)

Additional Types

1. char (16-bit): Unicode characters (0 to 65,535)

Note: The range of char is not typically considered numeric, but rather a character code.

Here's a summary:

| Data Type | Bits | Range |

| --- | --- | --- |

| byte | 8 | -128 to 127 |

| short | 16 | -32,768 to 32,767 |

| int | 32 | -2,147,483,648 to 2,147,483,647 |

| long | 64 | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |

| float | 32 | 3.4e-38 to 3.4e+38 |

| double | 64 | 1.8e-308 to 1.8e+308 |

| char | 16 | 0 to 65,535 |

6.WHAT IS TYPE CASTING?

Ans: TYPE CASTING:

CONVERTING A VALUE FROM ONE DATATYPE TO ANOTHER DATATYPE IS KNOWN AS TYPE CASTING.

TYPES:

THERE ARE TWO TYPES OF TYPECASTING:

1.WIDENING TYPECASTING:

Widening type casting in Java is the process of automatically converting a value from a smaller data type to a larger data type. It's also known as implicit conversion or casting down.

2.NARROW TYPECASTING:

Narrowing type casting is the process of reducing a larger data type to a smaller one. Other names for this are casting up or explicit type casting in Java. It does not happen on its own. If we do not do this explicitly, we will get a compile-time error.

7.WHAT IS JVM ARCHITECTURE?

8.HOW MANY CLASS LOADERS WE HAVE

Ans: Class Loaders:

Java's Class Loaders are categorized into different types, each responsible for loading classes from specific locations:

Types of Class Loaders in Java

1. Bootstrap Class Loader (Primordial Class Loader):

- The Bootstrap Class Loader is a machine code responsible for initiating the JVM's operations.
- In Java versions up to 8, it loaded core Java files from rt.jar. However, starting from Java 9, it loads core Java files from the Java Runtime Image (JRT).

- Bootstrap ClassLoader operates independently without any parent ClassLoaders.

2. Platform Class Loader (Extension ClassLoader):

- In Java versions before Java 9, there was an Extension ClassLoader, but from Java 9 onwards, it's referred to as the Platform Class Loader.
- It loads platform-specific extensions from the JDK's module system.
- Platform Class Loader loads files from the Java runtime image or from any other module specified by the system property java.platform or –module-path.

3. System ClassLoader (Application ClassLoader):

- Also known as the Application ClassLoader, it loads classes from the application's classpath.
- It is a child of the Platform Class Loader.
- Classes are loaded from directories specified by the environment variable CLASSPATH, the – class path or -cp command-line option.

9.HOW MANY .DOT EXTENSION WILL CREATE AFTER EXECUTING THE ONE CODE?

Ans: A source code file written in the Java programming language is saved with the extension .java. .Java file extension typically contains Java code that can be compiled into bytecode and executed on the Java Virtual Machine (JVM).

10.CAN I TAKE MULTI CLASSES IN SINGLE JAVA FILE?(YES)

DEFAULT

PRIVATE

There are no restrictions on the number of classes that can be present in one Java program. But each Java program should have only one class declared with public access specifier. There cannot be two public classes in a single Java program

11.HOW MANY TOKENS WE HAVE?

ANS: The five tokens in Java include keywords, identifiers, operators, literals, and separators. Keywords are reserved words with special meanings in the Java language, and identifiers are names given to classes, methods, and variables.

THEY ARE 5 TYPES OF TOKENS

1.keywords

2.identifiers

3.operators

4. literals

5.separators.

12.WHAT ARE ACCESS MODIFIERS WE HAVE?

==========

There are two types of modifiers in Java: access modifiers and non-access modifiers.

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

Private:

The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

Default:

The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.

Public:

The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

There are many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc. Here, we are going to learn the access modifiers only.

Protected

The protected access modifier is accessible within package and outside the package but through inheritance only.

The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class.

It provides more accessibility than the default modifer.

13.HOW MANY ACCESS MODIFIERS TOKENS WE HAVE?

14.EXPLAIN CLASS STRUCTURE ?(STATE AND BEHAVIOR OF PEN CLASS).

==========

A Java class is a blueprint that defines the structure and behavior of objects, and is a fundamental building block in Java programming:

Fields

Also known as attributes or variables, these hold the data or state of the object. For example, a Person object might have a name field, and a Rectangle object might have a width field.

Methods

These are functions that define what the object can do. For example, a Person object might have a greet() method, and a Rectangle object might have a calculateArea() method.

Constructors

These are special methods used to initialize new objects.

Access modifiers

These dictate how the fields and methods can be accessed. Examples include private, public, and protected.

Classes are central to object-oriented programming, and support features such as inheritance, polymorphism, and encapsulation. They promote code organization, reusability, and maintainability.

15.DESCRIPTION OF MAIN METHOD?

==========

The main method in Java is the entry point for a Java program and is responsible for executing it. The main method is a fundamental component of any Java application.

The main method has the following characteristics:

Name:

The method is always named main.

Syntax: The syntax for the main method is public static void main(String[] args).

Access modifier: or public

The method is public so that the JRE can access and execute it.

Static:

The method is static so that the JVM can load the class into memory and call the main method without creating an instance of the class first.

void :-

It is a keyword. It means no return type. Whenever we define any method in java and if we don't want to return any kind of value from that particular method then we should write void before the name of the method.

main() method :

It is a user-defined method because a user is responsible to define some logic inside the main method.

main() method is very important method because every program execution will start from main() method only, as well as the execution of the program ends with main() method only.

String [] args :

String is a predefined class in java available in java.lang package (just like header file) and args is an array variable

Return type:

The method is void because it doesn't return anything.

Here's what happens when a Java program is run:

The JVM loads the program into memory.

The JVM calls the program's main method.

The main method passes in an array of strings containing each successive space separated command line option.

The programmer can use the option strings to control subsequent program logic.

The main method calls various worker methods.

The main method returns to the caller which ultimately exits.

16.WHAT IS MEANT BY "WRITE ONCE ,RUN ANY WHERE" IN THE CONTEXT OF JAVA?

The slogan of java is "WORA" write once run anywhere.

A public class created in one package can be reuse from different packages also by using import statement.

In a single java file, we can declare only one public class that must be our .java file and that class can be reusable to all the packages.

"Write Once, Run Anywhere" (WORA) is a programming principle that allows Java code to be written on one platform and run on any other platform without modification. This is possible because Java is platform independent and compiles code into bytecode that can run on any device with a Java Virtual Machine (JVM).

WORA is a key feature of Java that saves software developers time and effort by eliminating the need to write different versions of software for each platform:

Platform independence

Java code can be written on one platform, such as Windows, and run on another platform, such as macOS or Linux, without any code changes.

Bytecode

Java code is compiled into bytecode, which acts as an intermediate code that doesn't require machine targeting. The JVM interprets or compiles the bytecode into native machine code at runtime

---

17.EXPLAIN HOW JAVA ACHIEVES PLATFORM INDEPENDENCE?

==========

The meaning of Java platform-independent is that the Java compiled code(byte code) can run on all operating systems. A program is written in a language that is human-readable. It may contain words, phrases, etc. which the machine does not understand. For the source code to be understood by the machine, it needs to be in a language understood by machines, typically a machine-level language. So, here comes the role of a compiler. The compiler converts the high-level language (human language) into a format understood by the machines.

Therefore, a compiler is a program that translates the source code for another program from a programming language into executable code. This executable

code may be a sequence of machine instructions that can be executed by the CPU directly, or it may be an intermediate representation that is interpreted by a virtual machine. This intermediate representation in Java is the Java Byte Code.

Step-by-Step Execution of Java Program

---

* Whenever a program is written in JAVA, the java compiles it.

* The result of the JAVA compiler is the .class file or the bytecode and not the machine's native code (unlike the C compiler).

* The bytecode generated is a non-executable code and needs an interpreter to execute on a machine. This interpreter is the JVM and thus the Bytecode is executed by the JVM.

* And finally, the program runs to give the desired output.

C and C++ programs are platform dependent programs that means the .exe file created on one machine will not be executed on the another machine if the system configuration is different.

18.WHAT ROLE DOES THE JVM PLAY IN MAKING JAVA PLATFORM-INDEPENDENT?

==========

The JVM, which executes Java programs, allows platform independence. The JVM interprets or JIT compiles bytecodes into machine code for the operating system. Java uses the JVM to compile code once for all hardware architectures and operating systems. The virtual machine bridges bytecodes to any JVM-compatible system

19.HOW DO BYTE CODE AND THE JAVA COMPILER CONTRIBUTE TO PLATFORM INDEPENDENCE?

Java is platform-independent because it is compiled to a bytecode that can be run on any device that has a Java Virtual Machine (JVM). This means that you

can write a Java program on one platform (such as Windows) and then run it on a different platform (such as macOS or Linux) without making any changes to the code.

20.EXPLAIN THE PROCESS OF EXECUTING A JAVA PROGRAM ON DIFFERENT PLATFORM?

To execute a Java program on a different platform, the Java Virtual Machine (JVM) translates the Java source code into bytecode, which is then translated into machine code for the target platform:

1. **Write the code**: Programmers write Java code in a Java Development Kit (JDK).
2. **Compile the code**: The compiler translates the Java source code into bytecode, which is platform-independent.
3. **Run the code**: The JVM loads the bytecode, allocates memory, and converts the bytecode into machine code for the target platform. The JVM then executes the program and produces output on the console.

Java programs can run on different platforms because there are different versions of the JVM for each platform. As long as the corresponding JVM is installed on the platform, the Java program can run. This is known as the "Write Once Run Anywhere" principle

21.WHAT IS THE DIFFERENCE BETWEEN PLATFORM INDEPENDENCE AND PLATFORM SPECIFIC FEATURES IN JAVA?

Platform independence is the ability of a program to run on multiple platforms without modification, while platform-specific features are characteristics of Java that are specific to a particular platform:

- **Platform independence**

Java is platform independent because it's compiled into bytecode, which is not tied to any specific hardware or operating system. This allows Java programs to be written once and run on any device or operating system that has a Java Virtual Machine (JVM).

- **Platform-specific features**

Object Serialization is a platform-specific feature of Java that extends the core Java Input/Output classes. It supports encoding objects into a stream of bytes, and reconstructing the object graph from the stream.

22.WHAT IS LITERAL IN JAVA?

Literals :

Any constant which we are assiging to variable is called Literal.

In java we have 5 types of Literals :

1) Integral Literal

2) Floating Point Literal

3) Boolean Literal

4) Character Literal

5) String Literal

23.HOW ARE INTEGER LITERALS REPRESENTED IN JAVA?

Integral Literal :

If any numberic literal does not contain decimal or fraction

then it is called Integral Literal.

Example : 12, 89, 45

In integral literal we have 4 data types :

a) byte (8 bits)

b) short (16 bits)

c) int (32 bits)

d) long (64 bits)

An integral literal we can represent in four different forms

1) Decimal Literal (Base 10)

2) Octal Literal (Base 8)

3) Hexadecimal Literal (Base 16)

4) Binary Literal (Base 2) [Available from JDK 1.7v]

24.EXPLAIN THE DIFFERENCE BETWEEN A DECIMAL LITERAL AND AN OCTAL LITERAL IN JAVA?

Decimal Literal :

By default our numeric literals are decimal literal. Here base is 10 so, It accepts 10 digits i.e. from 0-9.

Example :

int x = 20;

int y = 123;

int z = 234;

Octal Literal :

If any Integer literal starts with 0 (Zero) then it will become octal literal. Here base is 8 so it will accept 8 digits i.e 0 to 7.

Example :

int a = 018; //Invalid becuase it contains digit 8 which is out of range

int b = 0777; //Valid

int c = 0123; //Valid

25.WHAT IS A FLOATING-POINT LITERAL , AND HOW IS WRITTEN IN JAVA?

If any numeric literal contains decimal or fraction then it is called floating point literal.

Example : 12.3, 90.7, 56.6

In floating point literal we have 2 data types :

a) float (32 bits)

b) double (64 bits)

example:

public class Test1

{

```
    public static void main(String[] args)
    {
        float b = 15.29F;
        float c = 15.25f;
        float d = (float) 15.30;
        System.out.println(b + " : "+c+ " : "+d);
    }
}
```

26.WHAT IS AN INSTANCE VARIABLE IN JAVA, AND DOES IT DIFFER FROM A LOCAL VARIABLE?

INSTANCE VARIABLE:

It is a class level variable so It has default value.

If a non static variable is defined inside a class but outside of a method then it is called instance variable.

LOCAL VARIABLE:

If we declare a variable inside a method OR block OR Constructor then it is called local/Automatic/Temporary/Stack variable.

Example :

```
public void m1()
{ int x = 100;  //Local Variable  }
```

A local variable must be initialized by the developer before use because local variable does not have default values.

27.EXPLAIN THE CONCEPT OF A STATIC VARIABLE.HOW IT IS DIFFERENT FROM AN INSTANCE VARIABLE ?

A static variable is associated with the class itself rather than with any specific instance of the class. In contrast, an instance variable is associated with a specific instance of a class, and each instance has its own copy of that variable.

28.WHAT IS A PARAMETER VARIABLE IN THE CONTEXT OF METHODS IN JAVA ? HOW ARE PARAMETER VARIABLES USED?

Parameter Variable :

It is a method level variable hence does not have default value.

If a variable is declared inside a method parameter (not inside method body) then it is called Parameter Variable.

End user is responsible to initialize the parameter variable.

29.DEFINE LOCAL VARIABLE IN JAVA .HOW IS THE SCOPE OF LOCAL VARIABLE DETERMINED ,WHAT ARE ITS LIMITATIONS?

LOCAL VARIABLE:

If we declare a variable inside a method OR block OR Constructor then it is called local/Automatic/Temporary/Stack variable.

Example :

```
public void m1()
{ int x = 100;  //Local Variable  }
```

A local variable must be initialized by the developer before use because local variable does not have default values.

SCOPE OF LOCAL VARIABLE:

A local variable has its scope to the function only, in which it is declared. A local variable cannot be used outside the defined function. u202f The scope of global variables is throughout the program and is declared outside the function. They can be used in any function.

30.DISCUSS THE SCOPE OF INSTANCE VARIABLES,STATAIC VARIABLES,PARAMETER VARIABLES,AND LOCAL VARIABLES. HOW DOES THE SCOPE OF EACH TYPE OF VARIABLE?

31.HOW CAN YOU INITIALIZE AN INSTANCE VARIABLE IN JAVA WITH A USER-DEFINED VALUE DURING OBJECT CREATION?

Instance variables can be accessed only by creating objects. We initialize instance variables using constructors while creating an object. We can also use instance blocks to initialize the instance variables.

32.EXPLAIN HOW CONSTRUCTORS CAN BE UTILIZED TO SET USER-DEFINED VALUES FOR INSTANCE VARIABLES IN A JAVA CLASS.

In Java, constructors are used to set the initial state of an object by initializing its instance variables, including setting user-defined values:

- **Constructor invocation:** When the new keyword is used to create a new object of a class, the constructor is automatically called to initialize the object.
- **Constructor name:** The constructor name must match the class name.
- **No return type:** Constructors cannot have a return type.
- **Default constructor:** If no constructor is written, Java provides a default constructor with no parameters. The instance variables are set to their default values, such as 0 for int and double, and null for objects like String.
- **Parameterized constructor:** Constructors can be defined with parameters to initialize objects with specific values.
- **Object instance variables:** If a constructor has an object instance variable, it can copy the referenced object in the parameter using new and the constructor of the referenced object.

33.WHAT IS THE ROLE OF SETTER METHODS,AND HOW DO THEY FACILITATE THE ASSIGNMENT OF USER-DEFINED VALUE OF INSTANCE VARIABLES?

The Role of Setter Methods Setter methods, also known as mutator methods, are Java class methods used to modify the values of an object's fields after its creation. Setter methods are often named with the prefix set followed by the variable name.

34.HOW CAN USER INPUT BE PROCESSED IN

35.what is wrapper class?

Ans: A Wrapper class in Java is a class whose object wraps or contains primitive data types. When we create an object to a wrapper class, it contains a field and in this field, we can store primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

36.which types what is wrapper class, its types ,its work intenally or externally?

Ans:

- **Purpose**

Wrapper classes allow primitive data types to be used as objects, which is useful when working with Collection objects.

- **How it works**

Each primitive data type has a corresponding wrapper class. For example, the wrapper class for the primitive data type int is Integer

Only Objects are moving in the network but not the primary data type so java has introduced Wrapper class concept to convert the primary data types into corresponding wrapper object.

8 types:

Primary Data type		Wrapper Object
byte	:	Byte
short	:	Short
int	:	Integer
long	:	Long
float	:	Float
double	:	Double
char	:	Character
Boolean	:	Boolean

37. why we declare static variables?

Static variables and static methods are two important concepts in Java. Whenever a variable is declared as static, this means there is only one copy of it for the entire class, rather than each instance

having its own copy. A static method means it can be called without creating an instance of the class.

38.where we declare static variables and why?

Static variables are declared in a class, and they are used to store information that is related to the entire class rather than a specific instance of the class.

Here are some reasons why static variables are used:

- **Memory management**

Static variables are used to save memory because there is only one copy of the variable in memory, regardless of how many instances of the class are created.

- **Tracking information**

Static variables are used to keep track of information that is related to the entire class, such as counting the number of objects generated, keeping track of a minimum or maximum value, or keeping track of an average of the values in a collection of objects.

39.diff b/w constructor and method?

What is a Constructor?

In the object-oriented programming world, constructors play an important role. Constructors help in initialising an object. The name of class and constructor always remain the same. Constructor has the group of instructions that are performed at the time of object creation. In general, programmers use constructors to provide initial values to the variables represented in the class.

What is the Method?

In the object-oriented programming world, a method is a grouping of instructions that execute some particular operation and return the

result to the caller. It helps the program to become more manageable. It allows us to reuse the code without writing it again.

40.what will happen if we return keyword or return type to the constructor?

Since constructor can only return the object to class, it's implicitly done by java runtime and we are not supposed to add a return type to it. If we add a return type to a constructor, then it will become a method of the class. This is the way java runtime distinguish between a normal method and a constructor.

41.what is encapsulation?

[Accessing our private data with public methods like setter and getter]

Binding the private data with its associated method in a single unit is called Encapsulation.

Encapsulation ensures that our private data (Object Properties) must be accessible via public methods like setter and getter.

It provides security because our data is private (Data Hiding) and it is only accessible via public methods WITH PROPER DATA VALIDATION.

42.which concept we are using encapsulation? data hiding

Encapsulation is a way to restrict the direct access to some components of an object, so users cannot access state values for all of the variables of a particular object. Encapsulation can be used to hide both data members and data functions or methods associated with an instantiated class or object.

43.what is the diff b/w new operator or new method?

44.where object will be created?

Using the new keyword in java is the most basic way to create an object. This is the most common way to create an object in java. Almost 99% of objects are created in this way. By using this method

we can call any constructor we want to call (no argument or parameterized constructors).

45.what is data hiding?

Data hiding in Java is a concept that involves encapsulating data within a class and restricting access to certain components using access modifiers. It is crucial for enhancing code security, minimizing unintended interference, and promoting modular design by hiding the implementation details of a class.

46. how can we access private variable outside the class?

Make a public accessor for that variable in the same class. Anything outside that class can access that private variable through these public setter/getter(via an object, ofcourse).

47.what are the transfer statement in java?

Transfer Statements in Java

In Java, transfer statements are a set of keywords that allow you to control the flow of execution within a program. They provide mechanisms for altering the default sequence of control flow in loops and conditional blocks. These statements include break, continue, and return. Let's take a closer look at each of them and explore their usage in different scenarios.

1. Java break Statement

The break statement is used to terminate the execution of a loop prematurely. When a break statement is encountered, the control flow immediately exits the loop, and the program continues with the next statement after the loop. This is particularly useful when you want to exit a loop early under certain conditions.

2. Java continue Statement

The continue statement is used to skip the rest of the loop's body for the current iteration and proceed with the next iteration. It is typically

used when you want to skip certain iterations based on a condition without terminating the entire loop.

3. Java return Statement

The return statement is used to exit a method and return a value to the caller. It can also be used to return early from a method before it reaches the end. If a method has a return type other than void, it must return a value of the correct type.

48.what are the conditional statements is their in java?

49.what are the iterative statements are in java?

Loops in java :

A loop is nothing but repeatation of statements based on the

specified condition.

In java we have 4 types of loops :

1) do-while loop

2) while loop

3) for loop

4) for each loop:

It is also known as enhanced for loop, introduced from JDK 1.5

It is used to retrieve the value from the Collection like array.

It will fetch the values one be one from the Collection data so, It is known as for each loop.

50.diff of features of java and oops?

Features of OOP :

1) Class

2) Object

3) Abstraction

4) Encapsulation

5) Inheritance

6) Polymorphism

Features of java:

Object-oriented:

Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behaviour.

Platform Independent:

Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language

Secured:

Java is best known for its security. With Java, we can develop virus-free systems.

Robust:

The English mining of Robust is strong. Java is robust because:

- It uses strong memory management.
- There is a lack of pointers that avoids security problems.

51.what is constructor and method?

what is constructor:

Constructor [Introduction Only]

If the name of the class and name of the method both are exactly same and it does not contain any return type then it is called Constructor.

```
public class Example
{
  public Example() //Constructor
  {
  }
}
```

what is Method?

A method is a block of code which only runs when it is called. You can pass data, known as parameters, into a method. Methods are used to perform certain actions, and they are also known as functions.

52.what is method chaining?

calling one method into another method is ntg but method chaining.

Method chaining in Java is a programming technique that allows developers to call multiple methods on a single object in a single line of code. It's a popular choice for building configurable and expressive objects.

53.what is use of this keyword or what is this keyword?

use of this keyword:

The this keyword refers to the current object in a method or constructor.

The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter).

this keyword:

Whenever instance variable name and parameter variable name both are same then at the time of instance variable initialization our

runtime environment will provide more priority to parameter variable/local variable, parameter variables are hiding the instance variables (Due to variable shadow).

54.what is variable?

In Java, this is a reference variable that refers to the current object on which the method or constructor is being invoked. It can be used to access instance variables and methods of the current object.

55.what is inheritance and its types?

Inheritance:

Inheritance is a mechanism of driving a new class from an existing class. The existing (old) class is known as base class or super class or parent class. The new class is known as a derived class or sub class or child class. It allows us to use the properties and behavior of one class (parent) in another class (child).

Types of Inheritance

Java supports the following four types of inheritance:

- Single Inheritance
- Multi-level Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance