

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
		# Installation of JAVA (JDK) 18 ✓		
		# Installation of Eclipse . IDE ✓		
		<pre> graph TD     A[JAVA] --&gt; B[Core Java (S/w)]     A --&gt; C[Advanced Java (web.)]     A --&gt; D[Android Java (App)] </pre>		



classmate  
Date \_\_\_\_\_  
Page \_\_\_\_\_

### Q. Why learn Java ? / Features of Java .

- Java is a simple and easy to learn.
- It is open source language and multithreaded.
- It is platform independent language.
- It is secure and Embedded.
- It gives high performance.
- Compile and interpreted (JVM).
- Robust - garbage collector.
- It having large library and Framework.
- It is guaranteed to be Write Once, Run Anywhere.

### Q. What is Java ? When it is discovered ?

- Java is class based, high level, object oriented, programming language developed by 'James Goseling' in 1995.
- The first version of Java (Java JDK-1.0) was released on 23<sup>rd</sup> January of 1996 by the 'Sun Microsystem'.
- Latest version of Java (JDK 18.0) is released on 22<sup>nd</sup> March of 2022 by 'Oracle'.



## \* Variables and Datatypes \*

- Variables —
- ① variables are nothing but piece of memory used to store information.
  - ② One variable can store one information at the time.
  - ③ Variables are also used to reuse information.
  - ④ To utilize variables in Java programming language we need to follow some steps :-

- ↳ 1) variable declaration (Allocating / reserving memory)
- ↳ 2) variable initialization (Assigning or inserting value)
- ↳ 3) Usage.

Note — According to all programming languages directly dealing with information is not a good practice hence to overcome this variables are introduced.

- Datatypes —
- ① Data types are used to represent type of data or information which we are going to use in Java program.
  - ② In Java programming it is mandatory to declare datatype before declaring variable.
  - ③ In Java, Datatypes are divided in two types:
    - ↳ 1) Primitive Data type
    - ↳ 2) Non-primitive datatype



## (A) Primitive Data Types

- There are 8 types of primitive datatypes
- All the primitive datatypes are keywords.
- Memory size of primitive datatypes are fixed.
- They are divided as below:

// syntax : datatype variablename ;

### 1. Numeric + Non-decimal : →

Data Type

Size

① byte 1 byte = 8 bits (-128 to 127)

② short 2 byte = 16 bits

③ int 4 byte = 32 bits

④ long 8 byte = 64 bits

### 2. Numeric + Decimal : →

⑤ Float 4 byte = 32 bits

⑥ double 8 byte = 64 bits

### 3. Character : →

⑦ char 2 byte = 16 bit

### 4. Conditional : →

⑧ boolean 1 bit = true / false



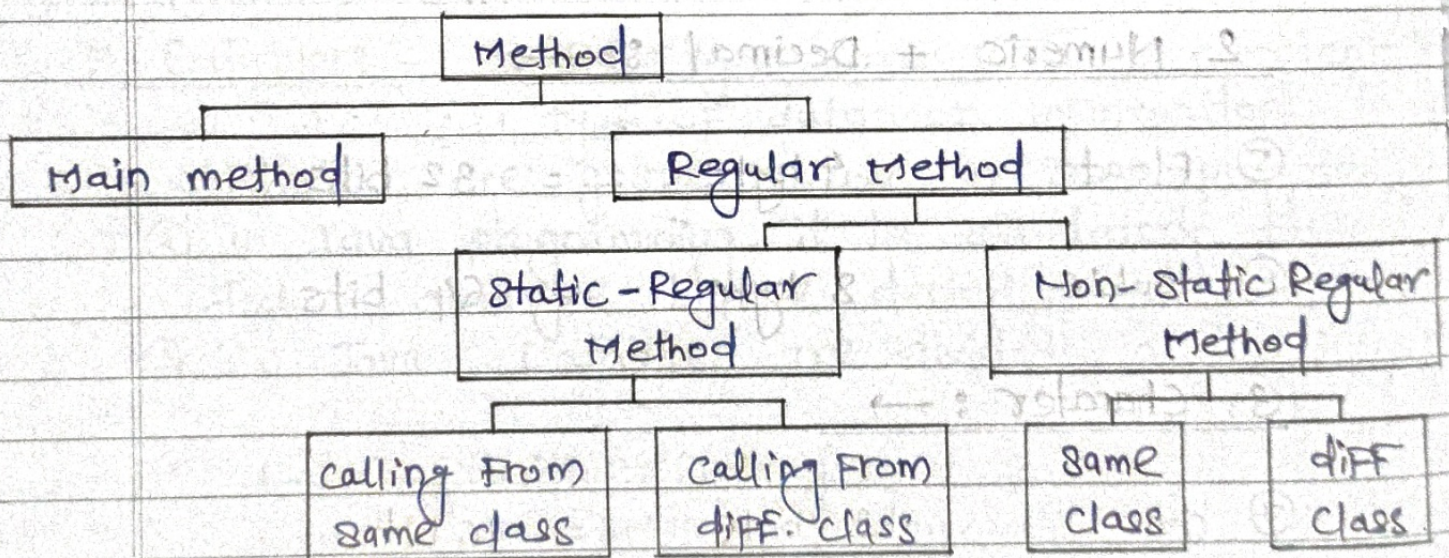
## (B) Non-Primitive Data Types

- Non-primitive data types are identifiers.
- Memory size of non-primitive datatype is not defined / it is user defined.
- Identifiers starts with capital letters.

e.g., String, Class, Array, Interface, etc.

### \* Methods \*

- 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.
- We use methods to reuse the code, to define the code once, and use it many times.





### (A) Main Method : →

- In any Java program, the main () method is the starting point where JVM starts program execution.
- So, JVM needs to call the main () method.
- Without main method we can not run any Java program.

### (B) Regular Method : →

#### • Static Regular method -

- Static regular method is a method that belongs to a class, but it does not belong to an instance of that class.
- This method can be called without the instance or object of that class.
- In this method, the method can only access only static data members and static methods of another class.
- The static method cannot be overridden because of early binding / compile time.

#### • Non-Static Regular Method -

- Every method in a Java defaults to a non-static method without a static keyword preceding it.
- In the non-static method, the method can access static data members and static as well as non-static members and methods from same or different class.



→ The non-static method can be overridden due to runtime / late binding.

### Note -

1) At the time of program execution main method is going to get executed automatically, whereas regular methods are not going to get executed automatically.

2) At the time of program execution priority goes to main method only.

3) To call a regular method from main method unless we need to make call method from main method, until unless if the method call is not made regular method will not get executed.

4) Regular methods can be called multiple times.

- Calling by method name — `// regular(); // method();`
- calling by class name — `// classname.methodname();`
- static regular from same class — `// methodname();`
- non-static from same class —  
`// classname Rvname = new classname ();`
- static regular from diff. class — `// classname.methodname();`
- non static from diff class —  
`// classname Rvname = new classname ();`



## \* Java Operators \*

### □ Arithmetic Operators — (+, -, \*, /, %)

Addition (+) subtraction (-) multiplication (\*)  
division (/) modulus (%)

### □ Rational operators — (>, <, >=, <=, ==, !=)

greater than (>) smaller than (<)

greater than Or equal to (>=)

smaller than or equal to (<=)

equal to (==) not equal to (!=)

### □ Logical Operators — (&&, ||)

And (&&) Or (||)

### □ Increment Operator — (++)

### □ Decrement Operator — (--)

### □ Assign Operator — (=)



## \* Java Loops \*

1. For Loop

2. While loop

3. do while loop

4. For each loop

### 1. For Loop -

↳ For is the most commonly used in loop Java.

↳ IF we know iteration in advance for loop is the best choice.

↳ In Java, For loop is used to iterate a part of program several times.

↳ IF the number of iteration is fixed it is recommended to use for loop.

Syntax :

```
for (initialization; condition; inc/dec)
{ // code to be executed
}
```

### 2. While Loop -

↳ In Java, while loop is used to iterate a part of programs for several times.

↳ IF the number of iteration is not fixed, it is recommended to use while loop.

Syntax :

```
initialization; while (condition)
{ // code to be executed
  inc/dec
}
```



### 3. Do while loop -

- ↳ In Java, do while loop is used to iterate a part of program several times.
- ↳ Use it if number of iteration is not fixed and you must have to execute the loop at least one time.

Syntax:

```
Initialization;  
do { // code to be executed  
    // inc/dec;  
    while (condition);  
}
```

### \* Control Statement \*

- ↳ Java compiler executes the code from top to bottom.
- ↳ The statements in the code are executed according to the order in which they appear.
- ↳ However, Java provides the statement that can be used to control the flow of java code. Such statements are called control flow statements.
- ↳ It is one of the fundamental features of Java, which provide a smooth flow of program.
- ↳ There are five types of statements.



\*Control Statements\*

1. Simple IF statement

2. IF-else statement

3. IF else IF ladder statement

4. Nested if statement

5. Switch statement

1 IF-statement -

↳ It is the most basic statement among all control flow statement in Java.

Syntax :

```

if (condition)
{
    // code
    // executed when code is true.
}

```

2. IF-else statement -

if-else

↳ The <sup>↑</sup> statement is an extension to the if statement, which uses another block of code i.e. else block.

↳ The else block is executed if the condition of the if block is false.

Syntax :

```

if (condition)
{
    // code to be executed
}
else
{
    // code to be executed
}

```



## \* Types of Variables \*

classmate

Date \_\_\_\_\_  
Page \_\_\_\_\_

### 1. Local Variable

Non-static local variable

### 2. Global Variable

static/class global variable

non-static/instance global variable

#### 1. Local Variable -

A variable declared inside the body of the method is called local variable.

- ↳ They can be used only by statements that are inside that function or block of code.
- ↳ Local variables are declared in methods, constructors or blocks.
- ↳ Always non-static.
- ↳ Access modifiers cannot be used for local variables.
- ↳ Local variables are visible only within the declared method, constructor or block.
- ↳ There is no default value for local variables.

#### 2. Global Variable -

Creating method outside of block/method is known as global variables.

- ↳ Scope of global variable remains through the class and they are permanent.
- ↳ There are two types of global variables.



## Global Variables

static global variable

Non-static global variable

### (A) Static global variable :- / class

- Declaring the keyword using static keyword is known as class / static variable because to access to static variables class name is used.

- To access static variable from different class we need to call like,

Syntax :-

classname.variablename ;

### (B) Non-static global variable :- / Instance

- All the non-static variables are known as instance variable because to access non-static variable instance (object) need to be created.

- To access non-static variable we need to create an object.

Syntax :-

objectname.variablename ;



## \* Constructors in Java

classmate

Date

Page

→ In Java, constructor is a special method that is used to initialize objects/variables.

→ The constructor is called when an object of a class is created.

→ At the time of constructor declaration we have to follow some steps

① constructor name should be same as class name

② you should not declare any return type for the constructor (like void).

③ Any no. of constructor can be declared in a java class but constructor name should be same as class name, but in constructor (arguments/parameters) should be different.

### # Use of constructor

→ To initialize the data member or variable

→ To copy/load non-static members of class into object → when we create object of class.

### # Types of constructors

(A) Default constructor

(B) User defined constructor.



## \* Java Keywords And Identifiers

classmate

Date

Page

→ Keywords are predefined, reserved words used in Java programming that have special meanings to the compiler.

eg.,

Abstract, assert, boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, enum.

Extends, final, finally, float, for, goto, if, implements, import, instanceof

int, interface, long, native, new, package, private, protected, public, return.

short, static, strictfp, super, switch, synchronized, this, throw, throws, transient, try, void, volatile, while.

## # Java Identifiers

→ Identifiers are the name given to variables, classes, methods, etc.



## # Rules For Naming an Identifier

- ↳ Identifiers cannot be a keyword
- ↳ Identifiers are case-sensitive
- ↳ It can have a sequence of letters and digits.
- ↳ The first letter of an identifier can not be a digit.
- ↳ Whitespaces are not allowed. Similarly, you cannot use symbols such as @, # and so on.



# O-O-P-S Concept

classmate

Date \_\_\_\_\_

Page \_\_\_\_\_

OOPS - Object oriented Programming System.

Q. What is oop?

→ oop stand for object oriented programming language/system.

The main purpose of oop is to deal with real world entity using programming language.

Q. OOPS Features?

→ ① Class - template / logical entity  
No space required

② Object - Having own state and behaviour  
- Instance of class.  
- main purpose data processing

③ Inheritance

④ Polymorphism

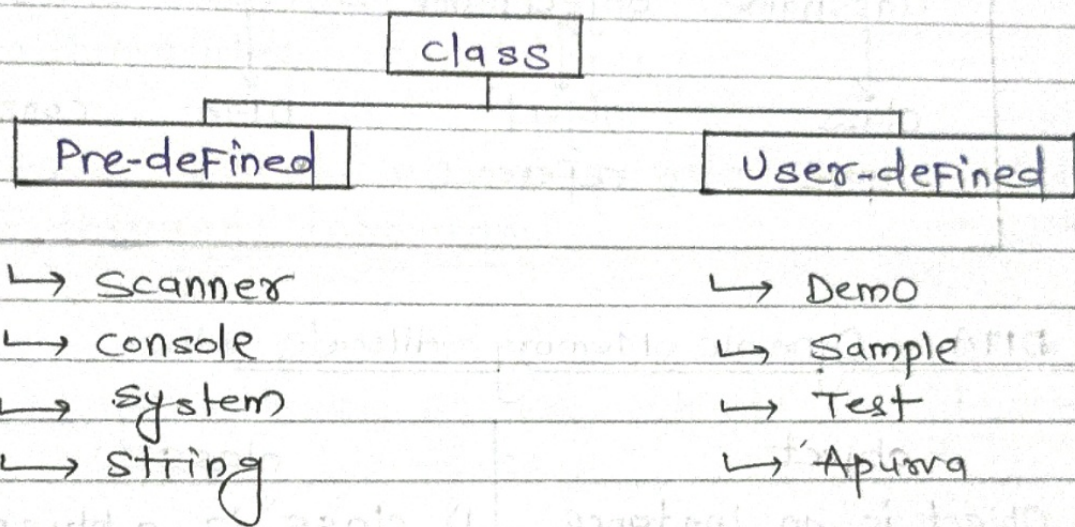
⑤ Encapsulation

⑥ Abstraction



## Q. What is class in Java ?

- class is a collection of objects and it doesn't take any space on memory,
- Class is also called as blueprint or logical entity.
- Class is differentiated in two parts.



① User defined class :- A class which is created by Java programmer is called user-defined class.

Syntax! —

```
class classname  
{  
    // data  
    // methods  
}
```

\* class is a collection of data and methods.



## Q What is object in Java ?

→ Object is an instance of class that executes that class.

→ Once the object is created, it takes up space like other variable in memory.

Syntax:-

classname	objectname	=	New	class name()
↓	↓		↓	↓
class	object		DMA	constructor
name	reference			

## DMA - Dynamic Memory Allocate

object	class
1) Object is an instance of class.	1) class is a blueprint or template from which object are created
2) Object is a real world entity such as pen, laptop, mob, bed, mouse keyboard, etc.	2) class is a group of similar object.
3) Object is a physical entity.	3) class is a logical entity
4) Object created many times as per requirement	4) class is declared once
5) object allocates memory when it is created	5) class doesn't allocated memory when it is created
6) Object is created through new keyword mainly.	6) class is declared using class keyword.



## Q. What is Inheritance in Java ?

→ When we construct a new class from existing class in such a way that the new class access all the features and properties of existing class is called inheritance.

Note — In Java extends keyword is used to perform inheritance.

— It provides reusability.

— We cannot access private members of class through inheritance.

— A subclass contains all the features of super class. so we should create the object of subclass.

— Method overriding only possible through inheritance.

Syntax :—

```
class A
{
    // code to be executed (+, -)
}
class B class extends class A
{
    // code to be executed (+, -)
    // (x, /, %)
}
```



Date \_\_\_\_\_  
Page \_\_\_\_\_

✧ There are mainly Four types of Inheritance —

- (A) Single Level Inheritance
- (B) Multilevel Inheritance
- (C) Multiple Inheritance / Hybrid Inheritance
- (D) Hierarchical Inheritance

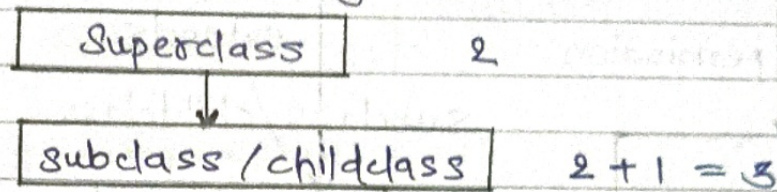
✧ Inheritance —

- It is one of the oops principal where one class acquires the properties of another class with the help of 'extends' keyword is called Inheritance.
- The class from where properties are acquiring is called super class.
- The class in which properties are inherited or delivered is called sub-class.
- Inheritance takes place between two or more than two classes.



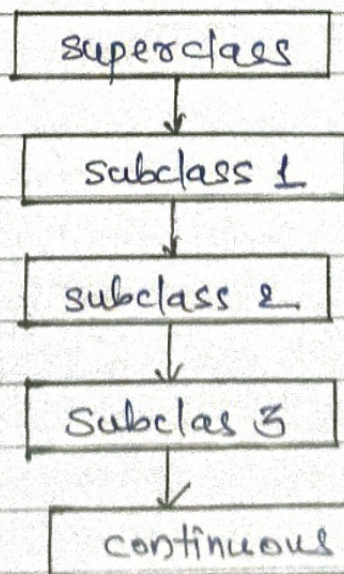
### (A) Single Level Inheritance —

- It is an operation where inheritance takes place between two classes.
- To perform single level inheritance only two classes are mandatory.



### (B) Multi Level Inheritance —

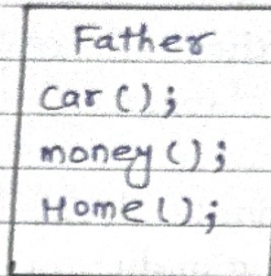
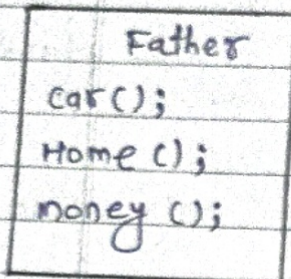
- Multilevel inheritance takes place between three or more than 3 classes.
- In multilevel inheritance 1 subclass acquires properties of another super class and that class acquires properties of another super class and phenomenon continues.





Example of Single Level Inheritance —

Superclass / base class

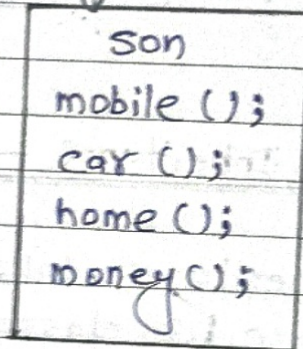
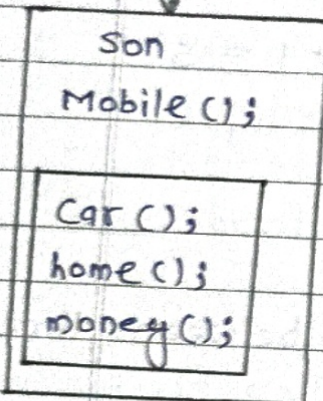


- the class from where property are acquiring or inheriting is called super class

permission

extends

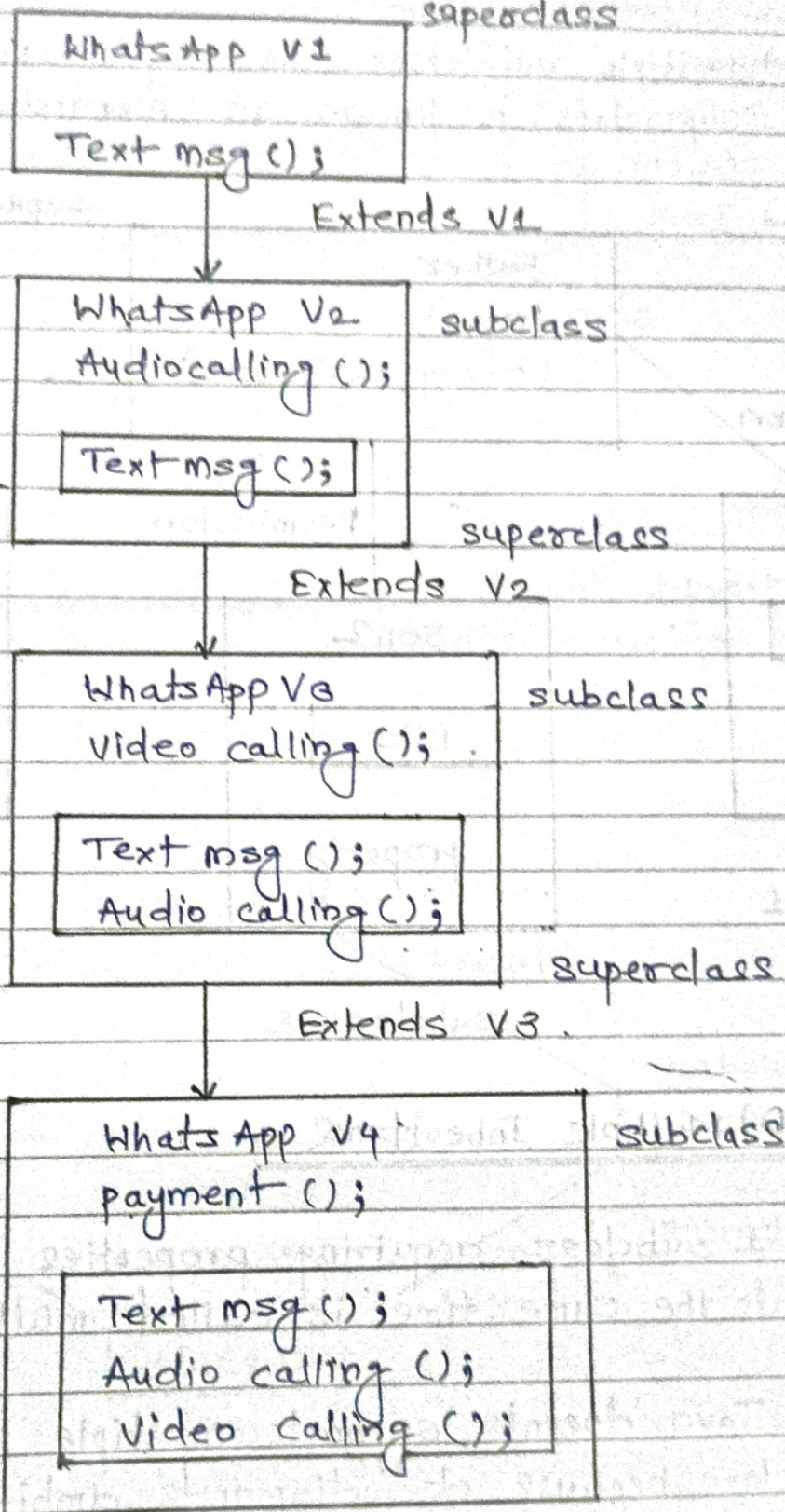
Subclass / child class



- the class in which properties are acquired or inherited is called sub-class



## Example of multilevel inheritance

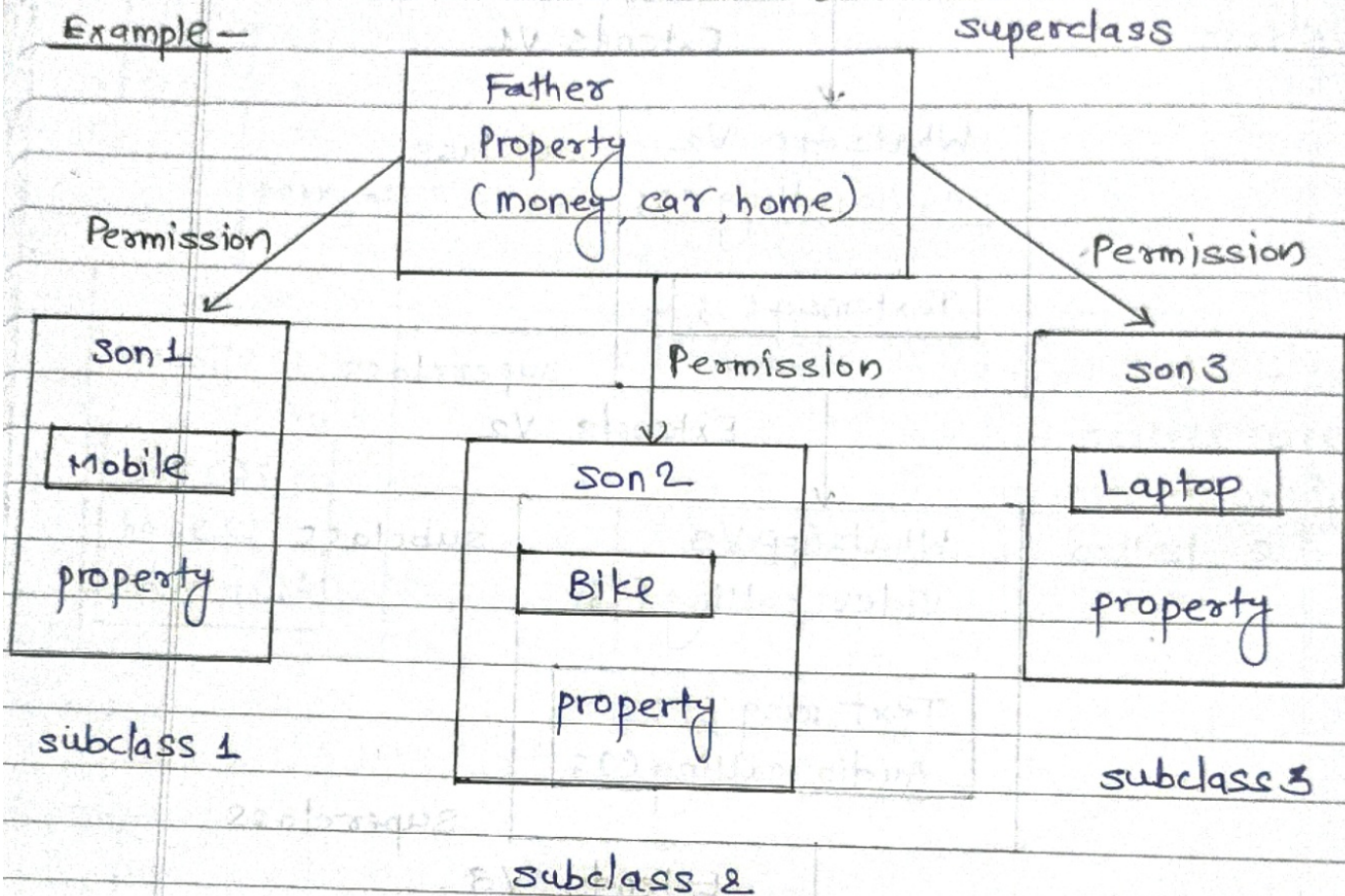




### (C) Hierarchical Inheritance —

→ Multiple subclasses can acquire properties of one superclass is known as hierarchical inheritance

Example —

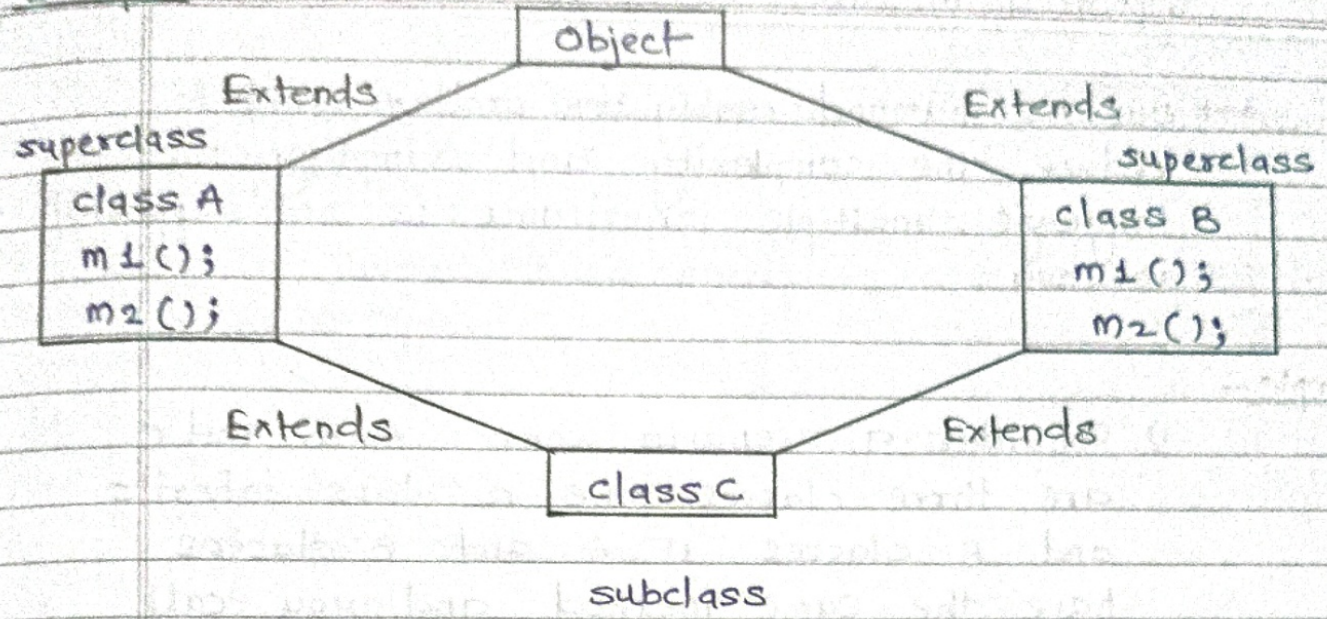


### (D) Multiple Inheritance —

- 1 subclass acquiring properties of 2 superclass at the same time is called multiple inheritance.
- Java doesn't support multiple inheritance using class because of diamond ambiguity problem.
- By using interface we can achieve multiple inheritance.

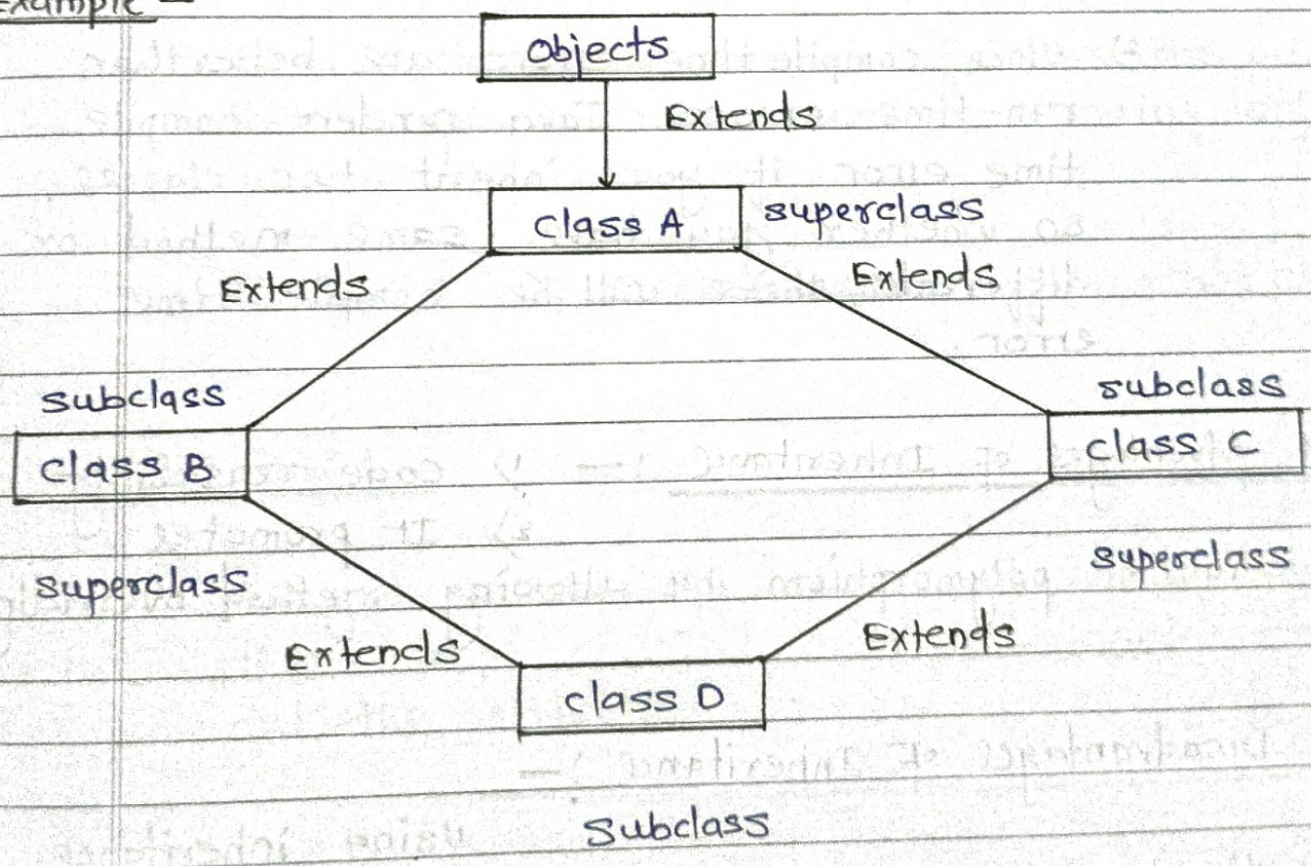


Example -



(E) Hybrid Inheritance — Java doesn't support Hybrid Inheritance using class.

Example -





## Imp q. why multiple inheritance is not supported in Java?

→ Due to diamond ambiguity problem and to reduce the complexity and simplify the language, multiple inheritance is not supported in Java.

### Example -

1) Consider a scenario where A, B and C are three classes the C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call method of A or B class.

2) Since compile time errors are better than run-time errors. Java renders compile time error if you inherit two classes so whether you have same method or different, there will be compile time error.

\* Advantages of Inheritance :- 1) code reusability  
2) It promotes runtime polymorphism by allowing method overriding

\* Disadvantages of Inheritance :-

the two classes (superclass and subclass) gets tightly coupled using inheritance.



## \* Java Specifiers / Modifiers \*

classmate

Date \_\_\_\_\_

Page \_\_\_\_\_

→ Access specifiers are used to represent scope of members of class.

→ In Java, Access specifiers are classified into Four types

1. Private
2. Default
3. protected
4. public.

### (A) Private —

→ If u declare any member of class as private then scope of that member remains only within the class.

→ It cannot be accessed from the other classes.

### (B) Public —

→ If you declare any member of class as public then scope of that member remains throughout the project.



### (c) Protected —

→ IF you declare any member of class as protected then scope of that member remains only within the package that class which is present outside the package can access it by one condition  
i.e. inheritance operation.

### (D) Default —

→ IF you declare any member of class as default then scope of that member remains only within the package.

→ It cant be access from other packages.

→ There is no keyword to represent default access specifiers.

Access Modifiers	Private	Public	Protected	default
Within the class	Yes	Yes	Yes	Yes
Within the package	No	Yes	Yes	Yes
Outside package by subclass	No	Yes	Yes (using inheritance)	No
Outside package	No	Yes	No	No



polymorphism - many forms

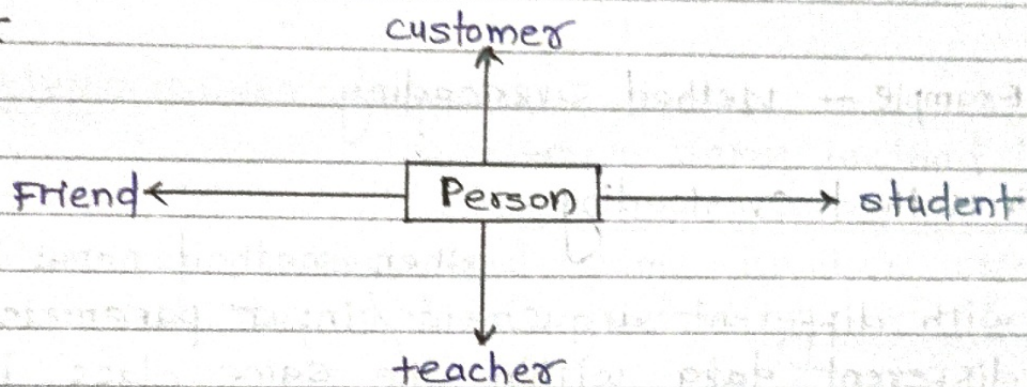
Q. What is Polymorphism in Java ?

→ It is one of the oops principal where one object showing different behaviour at different stages is known as polymorphism.

Poly - many / morphism - Forms

→ Polymorphism is the greek word whose meaning is 'same object having different behaviour'.

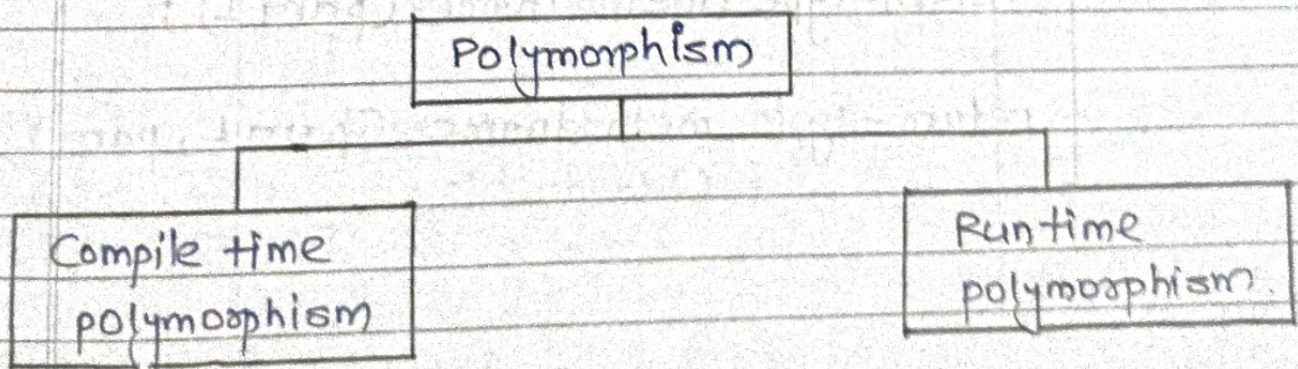
Example -



eg,

- ① void person (customer)
- ② void person (student)
- ③ void person (Friend)
- ④ void person (teacher)

→ There are two types of polymorphism.





#### (4) Compile Time Polymorphism

→ A polymorphism which exists at the time of compilation is called compile time or early binding or static polymorphism.

or

→ In compile time method declaration and definition are binded during the compilation time based on argument and parameters is known as compile time polymorphism.

Example — Method Overloading

#### \* Method Overloading —

When method name is same with different argument/input parameters with different data within the same class is called Method overloading.

or

Whenever a class contains more than one method with same name and different types of parameters called method overloading.

Syntax —

```
return-type methodname (para 1);
```

```
return-type methodname (para 1, para 2);
```



## (B) Runtime Polymorphism

→ A polymorphism which exists at the time of execution of program is called runtime polymorphism / late binding / dynamic polymorphism.  
or

→ Method declaration and definition are binded during the runtime or execution time based on argument. is known as runtime polymorphism

### Example - Method Overriding

#### \* Method Overriding -

When same method is present in parent class as well as in child class with same name and same number of argument is called as method overriding.

or

Whenever we writing method in super and subclass in such a way that method name and parameter must be same called method overriding.

Syntax -

```
class A {  
    void show () {}  
}  
class B Extends class A.  
    void show () {}  
}
```



Q. What is difference between method and constructor?

Method	Constructor
1) A method is a collection of statements that return a value upon its execution.	1) A constructor is a block of code that initializes the newly created object.
2) Method will take the parameter and also it return the values.	2) Constructor will take the parameter but it will not return the values.
3) A method names can be anything.	3) A constructor name should be same as the class name.
4) Method must be have return type.	4) A constructor doesn't have return type.
5) IF you want to call method that time we have to create an object and through object reference name we can call the method.	5) Whereas constructor will automatically invoked at the time of creating the object.



Q. What is the difference between method overloading and method overriding?

Method Overloading	Method Overriding
1) When the method name is same with different argument / input parameter with different data type within the same class.	1) When the same method is present in parent class as well as in child class with the same name and same no. of argument.
2) It is performed within the same class.	2) It is performed in more than one class.
3) It is example of compile time polymorphism.	3) It is example of runtime polymorphism.
4) static method can be overloaded.	4) static method cannot be override.
5) Parameter must be different.	5) Parameter must be same.



## Q. What is Abstraction in Java ?

- Abstraction is the process of hiding the implementation details and showing only functionality to users.
- Only the highlighted set of services provided to the user.
- We can achieve security and enhancement in the abstraction operation.
- There are two types to implement an abstract class.

Example - 1) Abstract class  
2) Interface

### \* Abstract class -

A class which contains a abstract keyword in its declaration is called abstract class.

Note :- 1) We cannot create object for abstract class.

- 2) It may or may not contain abstract methods.
- 3) To use an abstract class, you have to inherit it from sub classes.
- 4) If a class contain partial implementation then we should declare a class as an abstract.



### \* Concrete class -

A class which provides definitions for all the incomplete methods which are present in abstract class with the help of extends keyword is called concrete class.

### Q. Real time example of Abstraction.

Consider a real time example of man driving a car. The man only knows that pressing accelerators will increase the speed of car, or applying breaks will stop the car, but he does not know how on pressing the accelerator the speed is actually increasing, he doesn't know about the inner mechanism of the car or the implementation of the accelerator, breaks, etc in the car.

There are two ways in which the object can be initialized while inheriting the properties of parent and child classes. They are.

child c = new child (); The use of this initialization is, to access all the members present in both parent and child classes, as we are inheriting the properties.

Parent P = new child (); This type of initialization is used to access only the members present in the parent class and the methods which are overridden in the child class. This is because the parent class is upcasted to the child class.



## Q. What is Interface in Java ?

Date \_\_\_\_\_  
Page \_\_\_\_\_

- The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.
- In other words you can say that interfaces have abstract methods and variables. It can not have a method body.
- Java interface also represents the IS-A relationship.

### Features of Interface -

- ① Variables declared inside the interface are by default static and final.
- ② Methods declared inside the interface are by default public and abstract.
- ③ constructor concept is not applicable for interface.
- ④ Object of interface can not be created.
- ⑤ Interface supports multiple inheritance.
- ⑥ We can achieve 100% abstraction in interface (1-7)



## Java 8 and 9 interface improvement

- Since Java 8, we can have default and static methods in an interface.
- Since Java 9, we can have private methods in an interface.



## Interface

## Abstract Class

① When should we go for interface — when we don't know about implementation but we have requirement specification that time we should go for interface.

② Interface supports multiple inheritance.

③ Constructor concept is not applicable for interface.

④ Method declared inside the interface are by default public & abstract (upto version 1.7)

⑤ Every variable present inside the interface is always static and final whether we are declared or not.

⑥ Interface can implement using implements key word.

① When should we go for interface — when we are talking about implementation but not completely implementation that time we should go for an abstract class.

② Abstract class doesn't support multiple inheritance.

③ Constructor concept is applicable for abstract class.

⑤ No restriction.

⑤ The variable present inside the abstract class need not to be static and final.

⑥ Abstract class can be extended using extends keyword.



⑦ We cant declare variable with private and protected in interface	⑦ No restriction
⑧ We cant declare method as private & protected in interface	⑧ No restriction
⑨ We can achieve 100% abstraction (upto 1.7)	⑨ We can achieve (0-99%)
⑩ Interface can declared using interface keyword	⑩ Abstraction can declared using abstract keyword.



## Q. Difference between Early Binding and Late Binding

Early Binding	Late Binding
<p>① Early binding is the process of linking a function with an object during the compilation process.</p>	<p>① Late binding is a run-time polymorphism with method overriding.</p>
<p>② static binding is another name for early binding.</p>	<p>② Dynamic binding is another name for late binding.</p>
<p>③ Early binding happens at compile-time.</p>	<p>③ Late binding happens at run-time.</p>
<p>④ Execution speed is faster in early binding.</p>	<p>④ Execution speed is lower in late binding.</p>
<p>⑤ The class information is used by Early binding to resolve method calls.</p>	<p>⑤ The object is used by late binding to resolve methods calls.</p>



## Q. What is Array ?

- Array is a collection of similar type of element.
- We can store only Fixed set of element.
- Array is index based and first element of the array is 0th index.

### Advantages of array -

- ① We can represent huge member of Array values using single variable.
- ② Random access.
- ③ Code optimization.

### Disadvantages of array -

- ① Fixed in size (once we create array there is no chance of increasing size)

→ To overcome this problem we use collection/Array list.

- ② Array can hold only similar type of data.

→ To overcome this problem we use object array



Q. What happens if we declare an array without assigning the size?

It is not possible to declare an array without size. When we declare an array without assigning the size, it throws the compile-time error. For eg., `height = new int []`.

Q. Can we declare array size as negative?

No, the array size cannot be negative. If we declare an array with a negative size, it throws Negative Array Size Exception at run-time.

Q. When Array Index Out of Bounds Exception occurs?

The Array Index Out of Bounds Exception occurs when the program tries to access the index of an array. The exception also occurs when the index is higher than the size of the array or the index is negative.



# Introduction to Array

classmate

Date \_\_\_\_\_

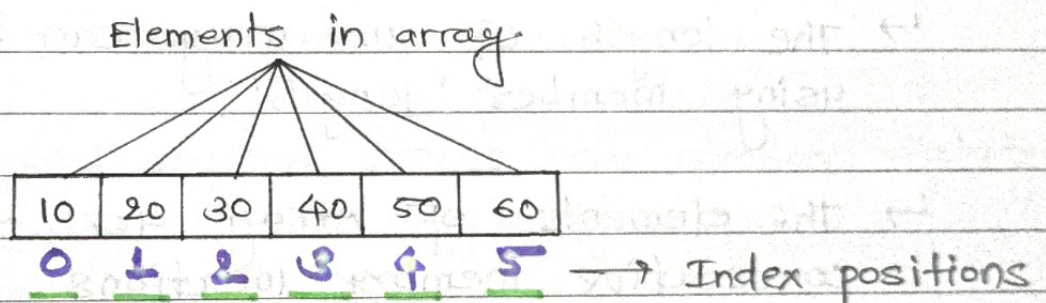
Page \_\_\_\_\_

## Q. What is Array?

- An array is an object that holds a fixed number of values of homogeneous or similar datatype.
- Or we can say Array is a Data structure where we store similar elements.
- The length of an array is assigned when the array is created and after creation its length is fixed.

For example,

```
int a[] = new int [6];
```



- It will create an array of length 6 and index value always start from 0.



### Q. What are the Features of an Array?

- ↳ A java array variable can be declared like other variables with [] after the data type
- ↳ The variables in the array are ordered and each have an index beginning from 0.
- ↳ In Java, Arrays are objects, and thus they occupy memory in 'Heap Area.'
- ↳ The direct superclass of an Array is Object.
- ↳ Arrays are always created at the runtime
- ↳ The length of an array can be found by using member 'length'.
- ↳ The elements of Array are stored in consecutive memory locations.

### Q. What are the advantages of Array?

- ↳ Arrays are used to store multiple data items of same type by using only single name
- ↳ We can access any element randomly by using indexes provided by arrays.
- ↳ Arrays can be used to implement other data structure like linked lists, stacks, queues, trees, graphs, etc.



→ Primitive type to wrapper classes object conversion will not happen so it is fast.

Q. What are the disadvantages of an Array?

- 1) Fixed size :- We need to mention the size of the array, thus they have fixed size. When array is created size can not be changed.
- 2) Memory Wastage :- There is a lot of chance of memory wastage. Suppose we create an array of length 100 but only 10 elements are inserted, thus 90 blocks are empty and thus memory wasted.
- 3) Strongly Typed :- Array stores only similar datatype, thus strongly typed.
- 4) Reduce Performance :- The elements of array are stored in consecutive memory locations, thus to delete an element in an array we need to traverse throughout the array so this will reduce performance.
- 5) No methods :- Array does not have array or remove method.



Q. What is string?

- String is an object that represents the sequence of characters.
- String is a class present inside "java.lang" package.
- String is used to store collection of characters.
- At the time of string declaration, initialization, object creation takes place.
- String objects are immutable in nature / can't be change.
- String class is final class can't be inherited to other classes.
- Object creation of string can be done in 2 ways:

- 1) By string literal
- 2) By using new keyword.

Q. Why string class is final in Java?

- The reason behind the string class being final is because no one can override the methods of the string class. So that it can provide the same features to the same new string objects as well as to the old ones.



## Q. Why strings are immutable in Java?

### 1. String literal :-

As java uses the concept of string literal. Suppose there are 5 reference variables, all refer to one object "Sachin". If one reference variable changes the value of the object, it will be affected by all the reference variable. That's why strings are immutable in java.

### 2. Security :-

If we don't make the string immutable, it will pose a serious security threat to the application.

For example, database usernames, passwords are passed as strings to receive database connections. The socket programming host and port descriptions are also passed as strings. The string is immutable, so its value cannot be changed. If the string doesn't remain immutable, any hacker can cause the security issue in the application by changing the reference value.

### 3. Code optimization and performance :-



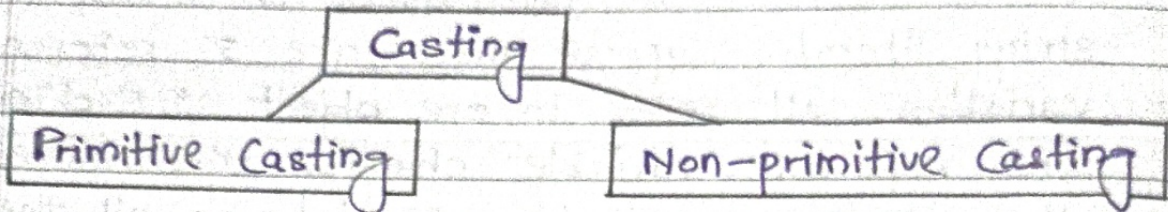
Q. What is casting in Java ?

classmate

Date \_\_\_\_\_

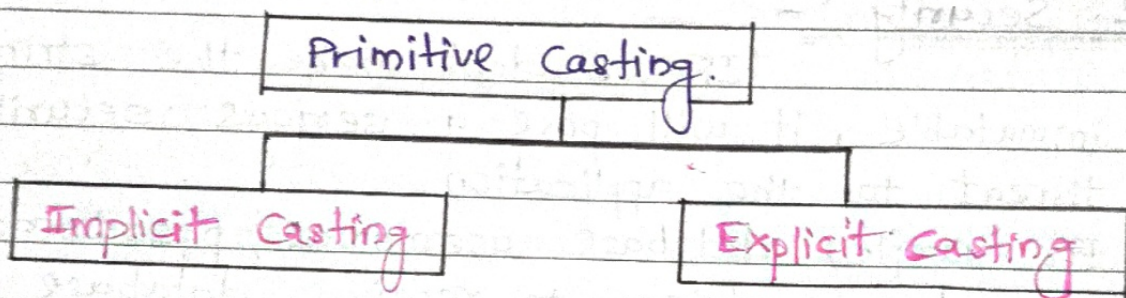
Page \_\_\_\_\_

→ converting one type of information into another type is called casting.



(A) Primitive Casting -

Converting one data type into another datatype is called primitive casting.



1) Implicit Casting -

Converting lower data type information into higher data type information is called implicit casting.

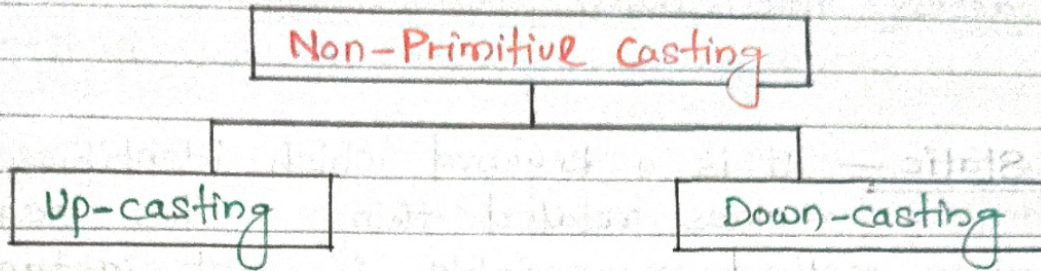
2) Explicit Casting -

Converting higher data type information into lower data type information is called explicit casting.



## (B) Non-Primitive Casting -

Converting one type of class into another type of class is called non-primitive casting.



### 1) Up-casting -

Upcasting is a type casting of child object into a parent object.

Because object of child class is assigned to reference variable of parent class.

### 2) Down casting -

\* Is not directly possible in Java \*



Page \_\_\_\_\_

\* Public — It is an access modifier, which defines who can access this method. Public means that this method will be accessible by any class (IF other classes are able to access this class.)

\* Static — It is a keyword which identifies the class related things. This means the given method or variable is not instance related but class related. It can be accessed without creating the instance of a class.

\* Void — It is used to define the Return type of the method. It defines what the method can return. Void means the method will not return any value.

\* Main — It is the name of the method. This method name is searched by JVM as a starting point of program for an application with a particular signature only.

\* String[] args — It means any array of sequence of characters (strings) that are passed to the "main" function.



# FINAL

## \* Definition -

Final is the keyword and access modifier which is used to apply restrictions on a class, method or variables.

## \* Applicable to -

Final keyword is used with the classes, methods and variables.

## \* Functionality -

1) once declared, Final variables becomes constant and cannot be modified.

2) Final method cannot be overridden by subclass.

3) Final class cannot be inherited.

## \* Execution -

Final method is executed only when we call it.



## FINALLY

### \* Definition -

Finally is the block in Java Exception Handling that executes the important code whether the exception occurs or not.

### \* Applicable to -

Finally block is always related to the try and catch block in exception handling.

### \* Functionality -

1) Finally block runs the code even if exception occurs or not.

2) Finally block cleans up all the resources used in try block.

### \* Execution -

1) Finally block is executed as soon as the try-catch block is executed.

2) Its execution is not dependent on the exception.



## FINALIZED

### \* Definition —

Finalize is the method in Java which is used to process of clean up just before the object is garbage collected.

### \* Applicable to —

Finalize () method is used with the objects.

### \* Functionality —

Finalize method performs the cleaning activities with respect to the object before its destruction.

### \* Execution —

Finalize method is executed just before the object is destroyed.