CLASS 1 JAVA DATED 15-09-2022

- JVM
- JRE
- JDK
- JIT

*****JVM (JAVA VIRTUAL MACHINE)

<table>
<tr><td rowspan="3">.class file BYTE CODE -></td><td>CLASS LOADER</td><td>CLASS Verifier</td><td rowspan="3">LOW LEVEL LANGUAGE</td></tr>
<tr><td rowspan="2">JIT COMPILER</td><td>MEMORY-BLOCK</td></tr>
<tr><td>GARBAGE COLLECTOR</td></tr>
</table>

* JVM stands Java Virtual Machine. It is called virtual because it doesn't exist alone. It needs a body for implementation. The components of JVM are:

1. Class loader : class loader loads class file into JVM.
2. Class verifier : class verifier reads class file line by line it generates low level code.
3. JIT compiler: JIT(just in time) compiler is used inside JVM to help class verifier so that over all efficiency of JVM can be improved.
4. Memory Block: memory block is responsible for all memory related tasks.
5. Garbage collector: garbage collector is responsible to read unwanted members from memory so that smooth execution can be maintained.

JVM works as a interpreter

***JRE (JAVA RUNTIME ENVIRONMENT)**

JVM

.Inbuilt class files and packages
.inbuilt library files
.other support files

JRE provides implementation to JVM , so JVM physically present inside JRE. JRE combines JVM along with inbuilt class files, library files and packages. By using JRE we can only run an application but we cannot do development part.

*** JDK(Java Development Kit):**

JDK
JRE
JVM

DEVELOPMENT TOOLS
text editor
jarac
debugger
other development

JDK stands Java Development Kit. JDK combines JRE along with development tools which are needed by a programmer to develop an application. By using JDK we can develop an application as well as we can run the application. JDK is platform dependent and so we have different JDK for different platform.

***JIT COMPILER**

JIT stands Just In Time Compiler. It is the part of JVM which is added to help class verifier so that overall efficiency can be improved at runtime.

***KEYWORDS:**

Keywords are predefined words which are used to perform special tasks. JAVA keywords are compiler aware words and compiler knows which keyword will perform which task. JAVA is a case sensitive language and all the JAVA keywords are written in lower case letters. In JAVA there are total 49 keywords for example:

if, else, do, while, for, continue, break, switch, case, default, byte, short, int, long, float, double, boolean, char, public, static, void, class etc.

CLASS 2 16-09-2022

*** IDENTIFIERS :** In java identifiers are the names for a class , name for a method, name of a variable, name of a package, name of a interface or name of any class members.

*** Rules for an Identifier:**

1. An identifier can be written by using alphabets, capital letter a to z or lower case a to z, numbers 0 to 9 or special characters(only $ and _).
2. Any identifiers must not start with numbers.
3. Any space is not allowed for identifier.
4. Any keywords must not be used as an identifier.

Examples:
correct : FirstProgram123, FirstProgram$123, First_Program, _FirstProgram, _123FirstProgram, public123, Public

incorrect :345FirstProgram, FirstProgram@123, public

***CONVENTION**

Conventions in JAVA are a set of guidelines for good programming practice.

1. Conventions for a Class name :
   - If a class name has only one word then the first character should be in upper case. Example : “Test” rather then “test”
   - If a class name has multiple words then the first character of each word should be in upper case. Example : class FactorialProgram, SumOfFactorial.

2.Conventions for a Method name:
   - if a method name will have only one word then it should be written in lowercase letter. Example: test(), sum(), factorial().
   - If a method has multiple words then the first word should be in lower case and from second word the first character of each word should be written in upper case. Example : findSum(), findFactorial(), findSumOfFactorial().

3. Conventions for a Variable name:
   - If a variable has only one word then it should be written in lower case and if it has multiple words then there should be “_” between each words. Example : (String[] “args”)
     int “age” = 24;
     int “age_of_employ” = 24
     int[] “age” = {20, 18, 19, 27}

CLASS 3 19-09-2022
THEME DATA TYPES

***LITERALS:**

Different types of value that is used in writing a program is called Literals.

Two Types of Literals:
1. Single Valued Literals
   - Numeric Type literals
   - Character Type Literals
   - Boolean Type Literals
2. Multi Valued Literals:
   - String
   - Array
   - List
   - Set
   - Queue
   - Map
   - Class = Multi valued Data

***DATA TYPES:**

In JAVA different types of data are categorized based on their type and based on their size. Based on their type it is dived as:

1. Primitive Type data
2. Non-Primitive Type data

1. **Primitive Type data:** In JAVA the single value types of data is called Primitive Types of data. In JAVA their are total 8 primitive types of data and all are in the form of keywords.

2.**Non-Primitive Type Data:** The Multi-Value types of data in JAVA is called Non-Primitive Types of Data. Non-Primitive Types of Data is also called User Defined Data in JAVA.

DATA TYPE:

1. Primitive Type

   - Numeric Type
     1. Integer Type : byte, short, int, long
     2. Decimal Type : float, double
   - Character Type : char
   - Boolean Type : boolean

2. Non- Primitive Type
   - ➔ String
   - ➔ Array
   - ➔ List
   - ➔ Set
   - ➔ Queue
   - ➔ Map
   - ➔ Class : Non Primitive Data Example: (Employ, Car Fruit, Etc)

1 bit = 2 values
2 bit = 4 values
3 bit = 8 values
1 byte = 8 bits
2 bytes = 16 bits
1 kb = 1024 bytes
1 mb = 1024 kb
1 GB = 1024 mb
1 TB = 1024 GB
1 PB = 1024 TB

```
class Program1
{
        Public Static Void main(String[] args)
        {
        System.out.println(    )
        }
}
```

CLASS 4 20-09-2022

Formula to find out Range = -2^{n-1} up-to $+2^{n-1}-1$

DATA TYPE	DEFAULT VALUES	SIZE	RANGE
boolean	False	1 bit	0 to 1
byte	0	71 byte	-128 to -127 2^7 upto 2^7-1
short	0	2 byte	-32,768 upto 32,767 -2^{15} upto $2^{15}-1$
Int	0	4 byte	-2^{31} upto $2^{31}-1$
long	0l or 0L	8 byte	-2^{63} upto $2^{63}-1$
float	0.0f or 0.0F	4 byte	-2^{31} upto $2^{31}-1$
double	0.0	8 byte	-2^{63} upto $2^{63}-1$
char	‘\u0000’	2 byte	0 to 65,535

***Char**

1. ASCII Value : Every character is assigned with an unique positive integer value which is called ASCII value.

Example : ‘A to Z’ = 65 to 90
‘a to z’ = 97 to 122
‘0 to 9’ = 48 to 57

example in code :

```
class Program2
{
        public static void main(String[] args)
        {
                System.out.println(‘@’)
                System.out.println(‘@’+)
                System.out.println(‘d’+‘0’)
                System.out.println(‘A’+‘B’)
        }
}
```

2. Unicode Value: Every character is assigned with a code in Unicode form. Example : ‘\u0012’

Number System:
1 Binary Form : 0 ,1
2 Octal Form : 0, 1, 2, 3, 4, 5, 6, 7
3 Decimal Form : 0, 1, 2, 3, 4 , 5, 6, 7, 8, 9
4 Hexadecimal form : 0, 1, 2, 3, 4 , 5, 6, 7, 8, 9, a-10, b-11, c-12, d-13, e-14, f-15

***VARIABLE :**

Variables in JAVA is a block of memory or a container which is used to store a value. A Variable can change its value any time in the program and if it is changed then old value will be replaced by new value in memory block.

1. Declaration of a Variable : Providing data type to a variable is called Variable Declaration.
>Syntax:-
data_type Variable_name; example

- byte x;
- int y, z;
- double a, b, c;
- etc

2. Initialization of a Variable : Assigning value to a already declared variable is called Initialization of a Variable.
>Syntax:-
Variable_name = Value; example

- x = 100;

- y = 123;
- z = 2345;

- a = 12.5;
- b = 18.5;
- c = 20.8;

etc

3. Declaration and Initialization :

>Syntax:-
data_type Variable_name = Value; example

- **int**(primitive data of int type) **a**(variable name) =**234**(int type value)**;**
  (declaration) **(initialization)**

byte a = 120;
short b = 1234;
int c = 35689;
long d = 358961125l;
float e = 5.389f;
double f = 32.568972;
char g = '@';(char is always in single quotes**)**
String h = "Today is a good day";(string is always in double quotes**)**
boolean i = true;

HOME WORK:
USE ALL THE OPERATION MENTIONED ABOVE IN DIFFERENT SITUATIONS

CLASS 5 21-09-2022
TYPE OF VARIABLE

*** A Variable is of types**

1. **Local Variable:** A Variable which is not directly inside class block a Local Variable.
   -In other words if a Variable is inside method block, constructor block or inside any block after class block then such variable is called Local Variable.

```
class Program1
{
        Public static void main(String[] args)
        {
                int a = 12; local variable
                int a = 13; local variable
        }
}
```

2. **Global Variable :** If a variable is directly inside class block then such Variable is called Global Variable. Global Variable are of two types **Static Variable and Non-Static(Instance Variable)**

```
class Program1
{
                int a = 12; global variable
                int a = 13; global variable

}
```

Example

```
class Program1
{
        int a = 12; global Variable
        int b = 20; global Variable

        Public static void main(String[] args)
        {
                int a = 12; local variable
                {
                        int e = 50; local variable
                }
                int a = 13; local variable
        }
        int f = 60; global Variable
        {
                int g = 80; local variable
        }
        int h = 80; global Variable
}
```

- **Static Variable: -** If a Global Variable is declared with a keyword “static” then such variable is called Static Variable.
- **Non-Static Variable** : - If a Global Variable is not declared with a keyword “static” then such variable is called Non-Static Variable.

  - EXAMPLE
    - Class Employ
    - {
    - static String CompName= “xyz”; **Static Variable**
    - string EmpName= “xyz”; **Non-Static Variable**
    - }

NOTE : -

1. We can access a Static Variable any where directly.
2. We can not access Non-Static Variables directly. To access Non-Static Variables we need object reference.

***CHARACTERISTICS OF A VARIABLE**

1. A Global Values will always have a default value when value is not provided by programmer.

Data type	**Default value**
Int	0
Double	0.0
String	Null
Non-primitie data	Null

2. A Local Variable will not have any default value. Initialization of a local varibale is mandatory in order to use it.

3. Same variable can not be declared more than once inside same block.
Example:

```
class Demo1
{
        public static void main(String[] args)
        {
        int i= 40;
                system.out.println(i);
        }
        {
        int i= 50;
                system.out.println(i);
        }
}
```

4. To change a value of a variable only variable name shoud be used. ex. int a = 10; then a = 40;

5. A Variable is only visible or accessible insite the same block in which it is declared. Outside the block a variable is not visible and it cannot be accessed.
example

```
class A
{
        int a = 12;
        Public Static void main(String[] args)
        {
                int b = 20;
                => a , b accessible
                {
                        int c = 30;
                        => a, b, c accesible
                }
                => only a and b accessible
}
=only a accessible
```

6. A variable is created when controls enters the block and it is deleted by garbage collector when control comes out of the block.
example

```
class A
{
        Public Static void main(String[] args)
        {
                int a = 10;
                S.O.P(a);
                {
                        int b= 30;
                        S.O.P(a);
                        S.O.P(b);
                }
                S.O.P(a);
                S.O.P(b); ERROR WILL SHOW
        }
}
S.O.P(a); ERROR WILL SHOW
```

CLASS 6 22-09-2022

*OPERATOR:

Operator are predefined symbols which are used to perform specific task.

*** OPERANDS** Different values needed by an operator is called operands.

***Types of Operators:**

1. **Unary Operator : only one operand is needed**
2. **Binary Operator : two operands are needed**
3. **Ternary Operator : three operands are needed**

***Arithmetic Operator:-**
Arthemetic operators are binary operator which are used to perform different Arthemetic Operations. They are :-
+ = used for adding the numbers and concatenation
-
*****
/ = gives quotient Value
% = gives remainder Value

example:
int x = 48;
int y = 10;
sop(x/y);
o/p = 4
int/int=int

example:
double x = 48;
int y = 10;
sop(x/y);
o/p=4.8
double/int, int/double, double/double= double

***Concatenation:-**
Concatination is a process to attach or joint different types of data with string. “+” operator is used as concatination operator.
Example:

Int x = 30;
sop(x)
sop(“The Value of x is:” +x);
o/p: The Value of x is : 30
int y = 40;
int z = x + y;
sop(“The sum of x and y is :” +z);
o/p: The sum of x and y is : 70
sop(“the sum of”+x+”and”+y+”is: “+2);

o/p: The sum of 30 and 40 is : 70

Questions to do:

1. **sop(“the sum is:”+60+40);**
   =the sum is : 6040
2. **sop(40+60+”is the sum”);**
   =100 is the sum
3. **sop(“the sum is:”+(40+60));**
   =the sum is : 100
4. **sop(40+80+”is the sum”+60+70);**
   =120 is the sum 6070
5. **sop(“The Value is:”+80-50);**
   =ERROR
6. **sop(50-30+”is the value”);**
   =20 is the value
7. **sop(“the value is :” + 20*4);**
   = The value is: 80
8. **sop(“the value is :” + 20/4);**
   = The value is : 5

CLASS 7 23-09-2022

***ASSIGNMENT OPERATOR:** Equals to(=) is used as an assignment operator. When there will be assignment then control reads statement from right to left. First right side value is calculated and then it is assigned to left side variable.

***COMPOUND ASSIGNMENT OPERATOR:** The combination of arithmetic operator and assignment operator is called compound assignment operator. They are :

1. + =
2. - =
3. * =
4. / =
5. % =

- a - = 10, // a = a - 10
- a * = 5, // a = a * 5
- a / = 6, // a = a / 6
- a % = 5, // a = a % 5

***RELATIONAL OPERATOR:** The Relational Operator is used to compare to numbers and gives boolean result(in true or false). They are :

1. >
2. >=
3. <
4. <=
5. == (comparison operator)
6. \ =
7. ! =

example:
int a=12;
int b=08;
int c=1900;
SOP(20>15)**;**
sop(a>b*2)
sop(a%4==0)
sop((a+b)/5==0)
sop(c%100!=0)

***INCREMENT/DECREMENT OPERATOR:** Is used to Increase or Decrease the value of a variable by 1.

- **Post Increament Operator** will first display the same value and after that the value will be incremented by 1 at memory.
  - **a + + ;**
    first display the same value of **a** after that increase the value of **a** by 1 at memory.

- **Post Decrement Operator**
  - **a - -;**
    first display the same value of **a** after that decrease the value of **a** by 1 at memory.

- **Pre Increment Operator** is used to first increase the value of variable by one at memory and after thatit will display the increamented value
  - **+ + a;**
    First increase the value of **a** variable at memory after that display the updated value of **a**.

- **Pre Decrement Operator**
  - **- - a;**
    First dicrease the value of **a** by 1 at memory after that dispaly the update value of **a.**

PRE EXAMPLE

- Int a =10;
- sop(a++);
- sop(a);
- sop(a- -);
- sop(a);

POST EXAMPLE

- Int a =10;
- sop(++a);
- sop(a);
- sop(- -a);
- sop(a);

EXAMPLE

- int a=24;
- sop(a++);
- sop(a++);
- sop(a--);
- sop(a++);

- sop(a++);
- sop(++a);
- sop (a);

EXAMPLE

```
int a=12;
a++;
int b=a++;
int c=++b;
sop(a);
sop(b);
sop(c);
```

EXAMPLE

```
int a=16;

a++;
int b=++a;
++b;
int c=b++;
int d=a++ + --b;

System.out.println(a);
System.out.println(b);
System.out.println(c);
System.out.println(d);
```

CLASS 8 26-09-2022

***TYPE CASTING : -**

Type casting is a process to convert one type of data into another type. It is of two types:

1. Primitive Type Casting

2. Non-Primitive Type Casting

1. Primitive Type Casting: Converting one type of primitive data into another type of primitive data is called Primitive type casting. It is of two types :

***Widening**

***Narrowing**

byte<short<char<int<long<float<double

2. Non-Primitive Type Casting: Converting a non-primitive data into another non-primitive data is called Non-Primitive type casting. It is of two types :

***Upcasting**

***Downcasting**

***WIDENING**: Is a process to convert a smaller type of data into bigger type. In widening there will be no data loss and so it is done by compiler automatically. This also called implicit conversion.
For Example :

byte a=100;
int b=a;
sop(a);100
sop(b);100

int x = 138;
double y=x;
sop(x);138
sop(y);138.0

int a= '@';
double b=a;
sop(a);@
sop(b);64

int x = 'd';
double y=x;
sop(x);d
sop(y);100

***NARROWING:** Narrowing is a process to convert bigger type of data into smaller types. In Narrowing the will be data loss so narrowing is not done by compiler automatically. It is achieved programatically by using **type cast** operator. This is also called Explicit Conversion.
Example:

double a=12.568;
int b=(int) a; "int" here in brackets is a type cast operator.
Sop(a);12.568
sop(b); 12

int x=126;
char y=(char) x;
sop(x); 126
sop(y); ~

***LOGICAL OPERATOR :** Logical Operator takes boolean operants and gives boolean results.

They are:

- **Logical AND(&&)**
- **Logical OR(||)**
- **Logical NOT(!)**

***Local AND:** Logical AND takes two bollean operands and if both are true then it gives result True otherwise it will be False.
Example:

OP1	OP2	&&
True	True	True
True	False	False
False	True	False
False	False	false

Logical AND(&&) : Fast and efficient because it can skip 2nd operand execution if 1st operand is false.

Logical AND(&): Slow because it checks both operands in every case.

Example

```
class Program1
{
    public static void main(String[] args)
    {
        int x=15;
        int y=30;

        System.out.println(x++>15 && y++>20);
        System.out.println(x);
        System.out.println(y);

    }
}
```

***Logical OR(||):-** Logical OR takes two boolean operands and it gives result true if atleast one operands is true.

OP1	OP2	||
True	True	True
True	False	True
False	True	True
False	False	False

Logical OR(||): It will not execute 2nd operand if 1st Operand is True. So it is fast and efficient.

Example

```
class Program1
{
	public static void main(String[] args)
	{
		int a=15;
		int b=30;

		System.out.println(++a>15 || b++>20);
		System.out.println(a);
		System.out.println(b);

	}
}
```

Logical OR(|): It will execute both in any case, so it is slow.

Examples:

```
class Program1
{
	public static void main(String[] args)
	{
		int a=15;
		int b=30;

		System.out.println(++a>15 | b++>20);
		System.out.println(a);
		System.out.println(b);

	}
}
```

Example

```
class Program1
{
	public static void main(String[] args)
	{
		int a=20;
		int b=25;
		 int c =15;
		 int d=10;

		System.out.println(a>10 || b > 15 | c>30 || d<10);
		System.out.println(a>10 && b > 15 | c>30 || d<10);

	}
}

```

Example

```
class Program1
{
	public static void main(String[] args)
	{
		int x=15;
		int y=30;
		 int z =18;
		 int p=10;

		System.out.println(x++>15 && y++>20 || z++>15 || p++>8);
		System.out.println(x);
		System.out.println(y);
		System.out.println(z);
		System.out.println(p);
	}
}
```

***Logical NOT(!):** Logical NOT takes one boolean operands and it will change its boolean state.

OP1	!
True	False
False	True

Example

```
Int x=15;
int y=30;

System.out.println(!true); false
System.out.println(!(x+y)%5==0); false
System.out.println(!(y%6!=0));  true
```

CLASS 9 27-09-2022

*USER INPUT

To take user input three step are to be followed:

1st Step. We have to import Scanner class of util package by using keyword “import”. Import statement should be the very first statement before class.

- import java.util.Scanner;(it imports only one class Scanner of util package)
- import java.util.*;(this imports all the classes of util packages)

lang package is by default available to every program

Lang	Util
Sql	Io
etc	

2nd Step. We have to create **object** of **Scanner** class inside the block where we want to take user input.

- Scanner sc = new Scanner(System.in);
  this is object creation of Scanner class.

3rd Step. To take different types of input from user we have to use different method available in Scanner class.

Data type	**Method of Scanner class**
byte	nextByte()
Short	nextShort()
Int	nextInt()
Long	nextLong()
Float	nextFloat()
Double	nextDouble()
Boolean	nextBoolean()
String(only First word)	next()
String(complete string)	nextLine()
Char	next().charAt(index)

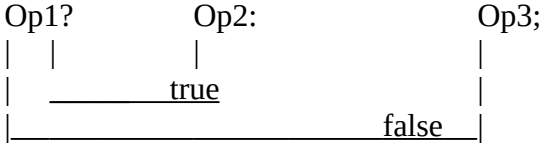
Example:

```
import java.util.Scanner;
class Program
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println(“Enter the first number”);
        int a=sc.nextInt();
        System.out.println(“Enter the second number”);
        int b=sc.nextInt();
        int c=a+b;
        System.out.println(“The sum is: ” +c);
    }
}
```

CLASS 10 28-09-2022

*CONDITIONAL OPERATOR:

Conditional operator takes three operands to perform the operation.



In conditional operator the first operand must be boolean type. If the boolean result from operand one is true then control goes to operand two and operand two get executed. If operand one is false then control goes to operand three and operand three will get executed. Syntax: “?”.

Example :

- s.o.p(true? “Monday” : “Tuesday”);
- o/p = Monday
-
- s.o.p(false? 10*3+40: 20*5-10);
- o/p = 90
-
- s.o.p(24>16 ? 24+16 : 24-16);
- o/p = 40
-
- s.o.p(24<16 ? 24+16 : 24-16);
-
- int b= false? 10 * 3+40: 20*5-10;
- s.o.p(b)
- int x= 24<16? 24+16 : 24 – 16;

Q1. Take user input and print the no is even number or odd number by using conditional operator?

Q2. Take user input and print if this number is diveded by both 7 or 5 not?

Q3. Take user input and print if the last digit of the number is 3 or not 3?

29-09-2022
CLASS-11

***FLOW CONTROL STATEMENT:-**

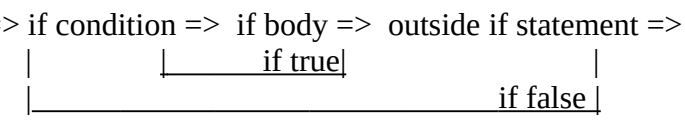
Flow Control statement in JAVA is used to change or alter the flow of execution of the program.
They are: -

- if statement
- if else statement
- else if statement(ladder if statement)
- nested if statement

***if Statement:** "if" is a block of statement which is used to check a condition and when condition is true then "if's" block executes otherwise "if's" block will not execute.
Syntax:-

```
if(condition)
{
    //if body;(will only get executed if condition is true)(this gets skipped if the condition is false)
}
outside if statement(will get executed if condition is false)
```

FLOW CHART:



example

- sop("main Method starts");
- if(true)
- {
- sop("This is if body");
- sop("insid if body");
- }
- sop("This if outside statement")
-
- O/p main method starts
- this if body
- inside if body
- this is outside statement
-
- sop("main Method starts");
- if(false)
- {
- sop("This is if body");
- sop("inside if body");
- }
- sop("This if outside statement")
-
- O/p main method starts
- this is outside statement

03-10-2022
CLASS 12

***IF ELSE STATEMENT :** In if else statement a condition is checked and if the condition is true then if block executes otherwise else block executes.
Syntax:-

```
if(condition)
{
        if block;
}
else
{
        else block;
}
outside if else statement
```

Example:

```
class Test
{

        public static void main(String[] args)
        {
                sop("This is main method");
                if(true)
                {
                        Sop("Inside if body");
                }
                else
                {
                        sop("This is else block");
                }
        Sop("Program ends")
        }

}
```

Qeustion : WAJP to check if a number is even or odd?
Questions: WAJP to check if a number is devided by 8 or not?
Question: WAJP to check if the last two digit of a number is 34 or not.
Question: WAJP to check if a number is divisible by either 7 or 8 or not .
Question: WAJP to check if the number is divided by both 12 and 16 or not.

***<u>ELSE IF STATEMENT:-</u>**

Else if statement is used to check multiple condition and when a condition is found true then a particular if block executes and after that control comes out of the else if block.
Syntax:

```
If(Condition)
{
if body1;
}
else if(condition2)
{
if body2;
}
else if(condition3)
{
if body3;
}
else if(condition4)
{
if body4;
}
else if(condition5)
{
if body5;
}
*
*
*
else
{
else body;
}
outside else if body;
```

Else if statement is also called **LATTER IF STATEMENT** where condition is checked one by one. If all the condition will be false then only else block will be executed.

Example:

```
Scanner sc=new Scanner(System.in);
sop(“Enter the marks: ”);
int mark=sc.nextInt();
if(mark>=90)
{
        sop(“You are a++ grade”);
}
else if(mark<90 && marks>=75)
{
        sop(“You are A+ Grade”);
}
else if(marks>=60)
```

```
{
sop(“You are first class”);
}
else if(marks>=40)
{
        sop(“you are second class”);
}
else
{
        sop(“you are fail ”);
}
sop(“Program ends”)
```

Q1. Take a number in input and print if the number is positive, negative or zero?

```
Scanner sc=new Scanner(System.in)
int a=sc.newInt();
if(a<0)
{
        sop(a is -negative);
}
else if(a>0)
{
        sop(a is positive);
}
else if(a==0)
{
        sop(a is zero);
}
```

Q2. Take 3 number from user and print which is the the biggest?

```
Scanner sc=new Scanner(System.in)
int a=sc.newInt();
int b=sc.newInt();
int c=sc.newInt();
if(a>b&&a>c)
{
        sop(a is bigger);
}
else if(b>c)
{
        sop(b is bigger);
}
else
{
        sop(c is bigger);
}
```

Q3. Take Cost price and Selling price from user and print % of profit or % of loss value or no profit no loss?

```
{
Scanner sc=new Scanner(System.in)
double cp=sc.nextDouble();
double sp=sc.nextDouble();
}
if(sp>cp)
{
        sop("there is profit");
        double profit = sp-cp;
        double perPro=(profit*100)/cp;
        sop("The percentage is: "+ perPro + "%");
}
else if (cp>sp)
{
        sop("Ther is a less");
        double loss=cp-sp;
        double lossPer=(loss*100)/cp;
        sop("The percentage loss is: " +lossPer+ "%");
}
else
{
        sop("There is no profit no less.");
}
}
```

***NESTED IF STATEMENT:**

When one if statement is inside another if statement then it is called nested if.
Syntax:

```
if(condition)
{
        if (condition)
        {
                *
                        *
                        *
                *
        }
        else
        {
                else body;

        }
}
else
{
        else body;
```

-
- }

Question1 If No is 4 digit or bigger then that, then check if it is even or odd?

```
{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
        if(n>999)
        {
                if(n%2==0)
                {
                        sop("The number is Even");
                }
                else
                {
                        sop("The no is odd");
                }
        }
        else
        {
                sop("The is less then 4 digit number");
        }
}
```

do the assignment of if else exercise program from instagram back date complete till october06

CLASS -13
06-10-2022

***Switch Case Statement :** Switch case statement is used to match the Switch value with case value. If it matches with a case value then a particullar case statement executes.

Syntax:-

(**keyword**)switch(value)
{
case(**keyword**) value1:
    statement1;
    [break]; (**this is optional keyword**)

case value2:
    statement2;
    [break];

case value3:
    statement3;
    [break];

*
*
*
default:(**keyword**)
    default statement;
}

Points:-

- Switch can take a value or expression which should be only “int” type or “char” type or “string” type value.
- Switch value is matched with th different case value inside switch block. If switch value matches with the case value, then that case statement will execute and switch block can be terminated by using “break” keyword.
- Switch type value must match with case type value .
- Every case should have unique value.
- If the switch value will not match with any case value then default statement will execute.

Questions:

1. Take user input as two number a and b with the operation to perform(+,-,%,*,/).

2. Enter a character and print if it is a vowel both lowercase and upper case by just using switch.

07-10-2022
CLASS 14
LOOP

***LOOP*:-**

Loop is a block of statement which excutes repeatedly untill the loop condition will fail. Loop is of two types:

1. Entry Control Loop **2. Exit Control Loop**

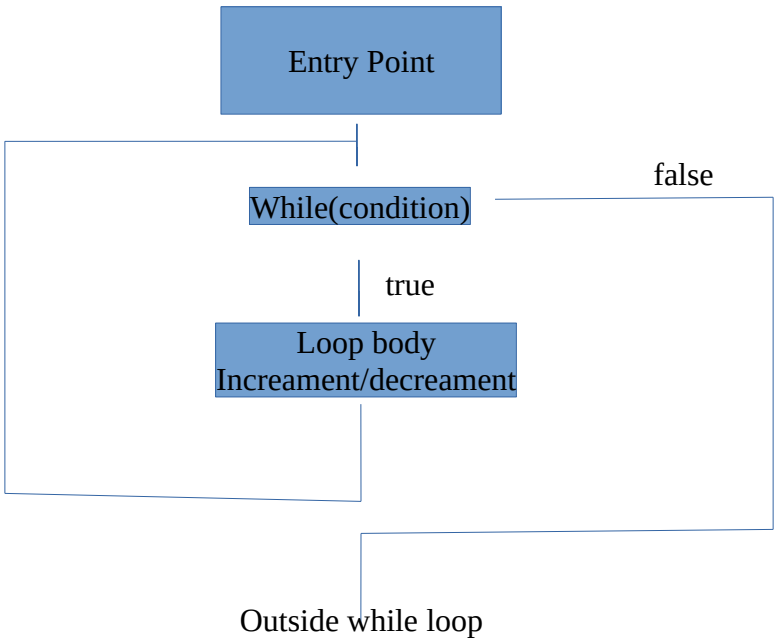
1. Entry Control Loop: If the loop condition is at entry point and after loop condtion there is loop body then such loop is called Entry Control Loop. Examples : **while loop, for loop.**

2.Exit Control Loop: If the loop condition is at exit point and before loop condition there is loop body then it is called Exit Control Loop Examples: **do while loop.**

- **While Loop:** While loop is an **Entry Control Loop** where first loop condition is checked and the after that loop body executes repeatedly untill loop condition will fail.

  Syntax :
  while (condtion)
  {
  while body;
  increament/decreament;
  }
  outside while loop

flow chart



Examples:

```
class Program1
{
        public static void main(String[] args)
        {
                int i=1
                while(i>=10)
                {
                        System.out.println(“Hello! World and i  is ” +i);
                        i + +;
                }
                System.out.println(“Program is completed”);
        }

}
```

Q1. WAJP to print all the factors of a numbers.
Q2. WAJP to count all the factors of a number.
Q3. WAJP to take user input and print that no is a prime no or not
Q4 WAJP to print sum of 100 natural number.
Q5 WAJP to print sum of squares of 100 natural number.
Q6 WAJP to print the sum of cube of 100 natural number.
Q7 WAJP to print the sum of all even number upto 100.
Q8 WAJP to print the sum of all number upto 100 which is divided by 7.
Q9 WAJP to print the sum of all number upto 100 which ends with 7.
Q10 WAJP to print the sum of all number upto 500 which are divisible by 7 and also ends with 7.
Q11 WAJP to print the sum of all the factors of a number
Q12 WAJP to print sum of all the digits of a number.
Q13 WAJP to cont the no of digits of a number.
Q14 WAJP to print factorial of 10 by using while loop.
Q15 WAJP to print the power of a number.
Take two user input and print power.

Q16 WAJP to take a user input and generate table(pahada) for that number.
Q17 WAJP TO PRINT THE TABLE UPTO THE NUMBER WHICH IS GIVEN BY USER AND ADD A DIVIDIED AT THE END OF EVERY TABLE.
Q18 WAJP to print if the number is perfect number or not a perfect number.
Fifth photo

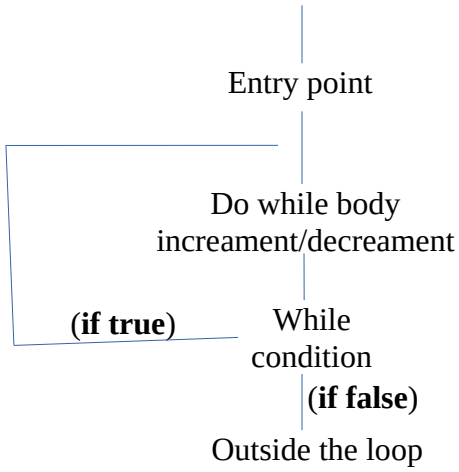
11-10-2022
CLASS 14

***DO WHILE LOOP**

Do while loop is an exit control loop where first loop body executes and after that loop condition is checked. In do while loop, loop body executes minimum one time whatever maybe the condition.
Syntax:

```
do
{
do-while body;
increament/deceament
}while(condtion);
```

Flow Chart:



HOME WORK : DO ALL WHILE LOOP QUESTION BY DO WHILE LOOP.

Q1. wajp to take user input and if the input is positive number then repeat the operation and ask the user input again and again until the number is zero or negative is entered then stop the loop and print the sum of all positive number entered.

13-10-2022
CLASS 15

***FOR LOOP**

For loop is an entry control loop which executes repeatedly untill the condition at for will be false.

Sytnax:

```
for(initialization;condition;inreament/decreament)
{
    for loop body;
}
```

Example:

```
for(int i=1;i<=5;i++)
{
    sop(“the i is: “ +i);
}
```

o/p; the i is:1
o/p; the i is:2
o/p; the i is:3
o/p; the i is:4
o/p; the i is:5

***BREAK KEYWORD**

Break is a keyword in java which is used to terminate the loop or switch block.

When JVM comes on break keyword then from then point loop will be terminated.

***CONTINUE KEYWORD**

Continue is a keyword which is used to skip a particullar execution of the loop.

14-10-2022
CLASS -16

Point to note for “for Loop”:

1. At **for loop** we can initialize more than one variables.

2. We can provide increament and decreament to more then one variable at for loop.

3. We cannot give condition for more than one variable at for loop.

Example:
1.
```
for(int i=1, int j=10; i<=10; i++)
{
sop(i+ “ “+j);
}
```
o/p
1 10
2 10
3 10
4 10
5 10
.
.
.
10 10

2.
```
for(int i=1, int j=10; i<=10; i++, j--)
{
sop(i+ “ “+j);
}
```
o/p
1 10
2 9
3 8
4 7
.
.
.
.
10 1

Q1 WAJP TO FIND LCM OF THREE NUMBERS.
LCM IS EITHER EQUAL TO BIGGEST NUMBER OR BIGGER THEN THAT.
See logic at first photo

```
public static void main(String[] args)
{
int a =15;
int b=20;
int c=40;
int big=(a>b?a:b)>c?(a>b?a:b):c      ;
for (int i=big;   ;i++)
{
    if (i%a==0&&i%b==0&&i%c==0)
    {
        sop(“lcm is: ” +i );
        break;
    }
}
}
```

Q2 WAJP TO FIND HCF OF THREE NUMBERS.
HCF IS EITHER EQUAL TO SMALLEST NUMBER OF SMALLER THEN THAT.

```
public static void main(String[] args)
{
int a =16;
int b=20;
int c=12;
int small=(a<b?a:b)<c?(a<b?a:b):c;
for (int i=small;   ;i--)
{
    if (a%i==0&&b%i==0&&c%i==0)
    {
        sop(“HCF is: ” +i);
        break;
    }
}
}
```

***STRONG NUMBER:**
If the sum of factorial of each digit of a number is equal to number itself then such number is called **strong number.**
Example:
142= 1!+4!+2!=1+24+6=31(not a strong number)
145= 1!+4!+5!=1+24+120=145(strong number)

***FIBONACCI NUMBER/SERIES:**

NOTES:-(WRITE YOUSELF FIND THE RELATED PHOTO)

15-10-2022
CLASS -17

***METHOD**

A method in JAVA is a block of statement which is used to perform different types of unique tasks.

#Points.

1. A method is a class member so it should be designed directly inside class block. A method should not be designed inside any other block.
2. Inside a class block we can design multiple method as much as we need.
3. A method will execute only when it is called.
4. A method is called by its method signature. Example : test(); , demo(); etc.
5. When a program executes then JVM always calls the main method.
6. A method will execute when it is called from main method.
7. A method can be called multiple times again and again.

example

```
Class Program1
{
    public staticvoid test (name) (  ) (parameter)
    {
        Sop("This is test method");
    }

    public static void main (name) (String[] args) (parmeter)
    {
        sop("This is main method");
        demo();
        test();
        sop("This is main method ending");
        test();
    }

    public static void demo (name) (  ) (parameter)
    {
        sop(" This is demo method");
        test();
        sop("demo method ends")
    }
}
```

o/p:

```
this is main method
this is demo method
this is test method
demo method dens
this is test method
main method ends
this is test method
```

CLASS -18
17-10-2022

***Structure of a method:**

public(**access modifier**) static(**non-access modifier**) void(**return type**) main(**method name**) (String[] args)(**arguments/parameter**)

1. **Access Modifier**: Access Modifier is used to provide accesibility or visibility to a method. It can be **public, private, protected and default.** If a method is **public** then such method can be accessed from anywhere. If a method is **private** then such type of method can be accessed only inside the same file.

2. **Non-Access Modifier:** Non-access modifier is used to provide the behaviour to a method. JAVA provides different keywords which are used as a non-access modifier for a method. Example:
Static, final, abstract, synchronized, strictFP(strict floating point), transient, volatile etc
A method can have more then one non-access modifier.

3. **Return Type** : A method can return any type of value(primitive or non primitive type value). If a method does not return anything then its return type should be void.

4. **Method name :** A method can be given name as per the convention for a method.

5. **Arguments or Parameter :** A method can accept different arguments as its parameter.

***Syntax of a method:**

----------Method Declaration----------
access modifier non-access modifier return type method name(arguments/parameter)
---------(**Method Signature**)----------

```
{
    //method body;
}
```

1. **Method Signature:** Method name along with its parameter is called method signature. A method is called by its method signature.

2. **Method Declaration:** Access Modifier, Non-access Modifier, Return type along with Method signature is called Method Declaration.

3. **Method Definition:** Method declaration along with method body is called Method Definition.

***Main Advantage of Method:-**

The main advantage of method is code reusability once we have designed a method then we can reuse it again and again inside any other files.

```
class Program1
{
        public static void main(String[] args)
        {
                //factorial program (cannot be used in other files)
        }
        public static void FindFactorial()
        {
                 (can be used in other files)
        }
}
```

***Types of Method:**

There are two types of method:

1. No-Argument Method
2. Parameterized Method

1. **No-Argument Method :** If a method does not take any argument at its parameter then such method is called No-Argument Method.

Questions
ALL BY USER INPUT
1. Design a method which will print the no is prime or not.
2. Design a method and print the no of digit of the no.
3. Design a method and print no of factors of a number.
4. Design a method and print if the no is perfect number or not.

CLASS 19
18-10-2022

***PARAMETERIZED METHOD:**

If a method takes different argument at its parameter then ssuch method is called parameterized method.

Syntax:
public static void test(data type 1 variable 1, data type 2 variable 2,.....)
{

}

Example:
public static void test(int x, double y, string z)
(formal argument)
{

}
call test(20, 18.5, “Mohan”)
(actual argument)

***Points:**
1. Formal Arguments:- The argument which is passed at method declaration is called formal argumment. At formal argument we declare different types of data.

2. Actual Aguments:- The argument which is passed at method call is called actual argument.

3. Compile time binging:- At the time of compilation compiler checks the actual argument matches with formal argument or not. If it matces then compilation will be successfull and binding will be completed.
If it doesn’t match then compiler gives an error and binding will be un-successful.

Example:
1.

```
public static synchronized final void main(String[] args)
{
        System.out.println(“This is main method of example1”)     ;
        findFactorial(10);
        findFactorial(12);
        findPower(12, 6);
        findPower(9, 5);
}
public static void findFactorial(int n)
{
        int fact=1;
        for(int i=1; i<=n; i++)
        {
                fact=fact*i;
        }
        System.out.println(n+”!= “+fact);
}
```

```
public static void findPower(int a, int b)
{
        int pow=1;
        for(int i=1;i<=b;i++)
        {
                pow=pow*a;
        }
        System.out.println(a+" to power "+b+"is: " +pow);
}
```

2 with user input

```
import java.util.Scanner;
class example2
{
        public static synchronized final void main(String[] args)
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("This is main method of example2")     ;
                System.out.println("Enter the base value: ");
                int x=sc.nextInt();
                System.out.println("Enter the base value: ");
                int y=sc.nextInt();
                findPower(x, y);
        }
        public static void findPower(int a, int b)
        {
                int pow=1;
                for(int i=1;i<=b;i++)
                {
                        pow=pow*a;
                }
                System.out.println(a+" to power "+b+"is: " +pow);
        }
}
```

Q1. Take user input and pass it at method parameter and print if the number is a prime number? Also try telling the user the nearest prime number if the number is not a prime number.

*<u>RETURN STATEMENT:</u>

A method in JAVA can return any type of value primitive or non-primitve. Its return type will be void only if the method does not return any thing.

Points:-

1. If a method has to return a value then we must mention the data type at method declaration.
Example:

```
public static int countDigit(int n)
{
    int count=0;
    while(n>0)
}
```

2. A method returns a value by using a keywords return. Return statement must be the last statement of the method.

3. A method can only return one value we cannot return multiple values from a method.

4. After the return statement we must not write any other statement other wise compiler will gives an error **unreachable statemtent.**

Example

```
public static void main(String[] args)
{
int x=CountDigit(58376);
sop(x+10)
}

public static int CountDigit(int n)
{
        int count = 0;
        while(n>0)
        {
                count ++;
                n=n/10;
        }
        sop("This digit is: " count++);

return count;
}
```

Questions

1. Return digit count.
2. Return sum of digit.
3. Return sum of factors of a number.
4. Return sum of all the number which is an even number upto 100.

5. If a method returns as value then we can store the result in same type of data at the time of method calls.

6. If a method has return type then we can write such method inside S.O.P statement.
Return statementis also called Control Transfer Statement.

Example

sop(test()) **allowed**
sop(demo()) **error void type not allowed here**

```
public static int test();
{
    return 10;
}

public static void demo();
{

}
```

Question:

5. WAJP to design a method to check a number is prime or not with return type (return boolean value).
6. WAJP to check a number is strong or not by designing a method by return type(return boolean value).
7. WAJP TO CHECK WHETHER THE NMBER IS ARMSTRONG NUMBER OR NOT.

20/10/2022
CLASS 20

***RECURSION**

Recursion in a process where a method calls itself. When a method calss itself then such method is called recursive method where it executes again and again repeatedly. We can achieve recursion in three way.

- By calling method from inside method body.
- By calling method from parameter as an argument.
- By calling method from return statement.

1. Recursion from method's body: When a method calles itself from its method body then it starts executing again and again repeatedly and if it is not controlled then an exception occurs called Stack Over Flow error.

```
class Example1
{
	public static void main(String[] args)
	{
		sop("This is main method");
		test(3);
		sop("main method ends");
	}

	public static void test(int n)
	{
		if (n>1)
		{
			sop(n);
			test(n-1);
		}
		else
		{
			sop(n)
		}
	}
}
```

Question: Print 10 natural numbers but not by loop

2. Recurssion from method parameter as an argument: When a method calls itself from parameter as an argument.Then recussion is achieved from method parameter.

Points:-

- To achieve this recursion, method must be parameterized method.
- Method must have return type to acheive this recursion.

```
Class example1
{
	public static void main(String[] args)
	{
		//for two number
		sop("The sum is : "+sum(20, 70));
		//for more then two number
		sop("The sum is : "+sum(sum(20, 70), 80));
		//for more then three number
		sop("The sum is : "+sum(sum(20, 70), sum(80, 100)))
	}
```

```
    public static int sum(int a, int b)
    {
        return a+b;
    }

}
```

Questions: Find LCM, Big number and HCF with this method.
Hint for finding big

```
class program1
{
        public static void main(String [] args)
        {
                sop("The big is: " +findBig(80, 60)); (if there are only two numbers)
                sop("The big is: " +findBig(findBig(80, 60), 130); (if there are three numbers)
                sop("The big is: " +findBig(findBig(80, 60), findBig(70, 50)); (if there are four numbers)
                sop("The big is: " +findBig(findbig(30, findBig(80, 60), findBig(70, 50))); (if there are five numbers)
        }

        public static int big(int a, int b)
        {
                return a>b?a:b;
        }
```

3. Recursion From Return Statement.

When a method calls itself from return statement then it is recursion from return statement.

Example1

```

        public static string test(int n)
        {
                return n+" "+test(n-1);
        }
```

Example2

```
public static void man(String[] args)
{
        sop("main method starts");
        sop(test(100));
        sop("main method ends");
}
public static int test(int n)
{
        if(n>1)
        {
                return n+test(n-1);
        }
        else
        {
                return n;
        }
}
```

07/11/2022

***OBJECT ORIENTED PROGRAMMING(OOPS)**

Object oriented programming is a mechanism where program is designed with the help of class and object to solve real-time problems.

The four main pillars of **Object Oriented Programming** are:-

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction

Question. Is JAVA a 100% Object Oriented Language or not?

Answer : No JAVA is not a 100% Object Oriented Language because JAVA provides two types of data **Primitive and Non Primitive.**

In JAVA non-primitive data always refers to an object but primitive type of data is not an object type of data thats why JAVA is not 100% Object oriented language

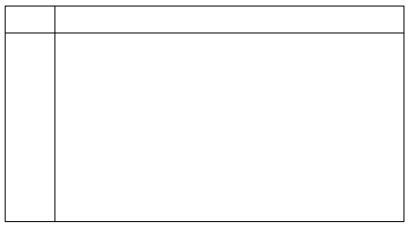
- Int x = 10;

Here int is not an object type data and 10 is not an object.

- Class Employee
- {
- }
- employ e = object;
- employ c = new Employ();

Here, employ is a non-primitive data which is an object type data and Employ() is an object

BLUE PRINT(CLASS)



ACTUAL EXISTANCE OF THE OBJECT(OBJECT)

***CLASS**

Class is a blueprint, master-copy or prototype from which an object is created, class is a logical entity which does not require any memory space. In JAVA every class represents a non-primitive type data and its default value is none/null.

A Class in JAVA provides a road map from which multiple objects can be created with different details.

***OBJECTS**

An object in JAVA is a physical entity which is created as per the class is designed. Object in JAVA is created at runtime dynamically inside Heap Area.

Whenever an object is created then memory will be allocated in heap area which is part of RAM.

08/11/2022

***CHARACTERISTICS OF CLASS OR OBJECTS:-**

A class in JAVA has two characteristics **States and Behavior.**

***STATES**

States of a class are the properties or details about the class. States of a class forms Global Variable of a class. Example : empName, age, Salary etc.

***BEHAVIOR**

The action performed by object is called behavior of a class. Behavior of a class form methods of a class. Example : work(), eat(), run() etc.

***Design a class/blue-print for employ**

```
class employ
{
  string empName;
  int age;
  double salary;
public void work()
{
  Sop("employ is working");
}
public void eat()
{
  Sop("employ is eating");
}
}
```

***Design a blueprint for**

2. Tree
    1. States
        1. Name
        2. Height
        3. Color
    2. Behavior
        1. grow()
        2. giveOxygen()
        3. giveFruit()
3. Browser
    1. States
        1. name
        2. version
        3. size
    2. Behavior
        1. click()
        2. search()
        3. back()
4. Vehicle
    1. States
        1. name
        2. price
        3. color
    2. Behavior

1. start()
2. stop()
3. drive()
4. accelerate()

***<u>GLOBAL VARIABLE:</u>**

A variable which Is directly inside class block is called **Global Variable.** It is of two types.

- ➔ Static Variable
- ➔ Non-Static Variable also known as **Instance Variable(Instance stands for object)**

***<u>STATIC VARIABLE:</u>**

Available which is declared with keyword static is called a static variable if a variable is static then it will be same for all the objects.

***<u>NON STATIC VARIABLE:</u>**

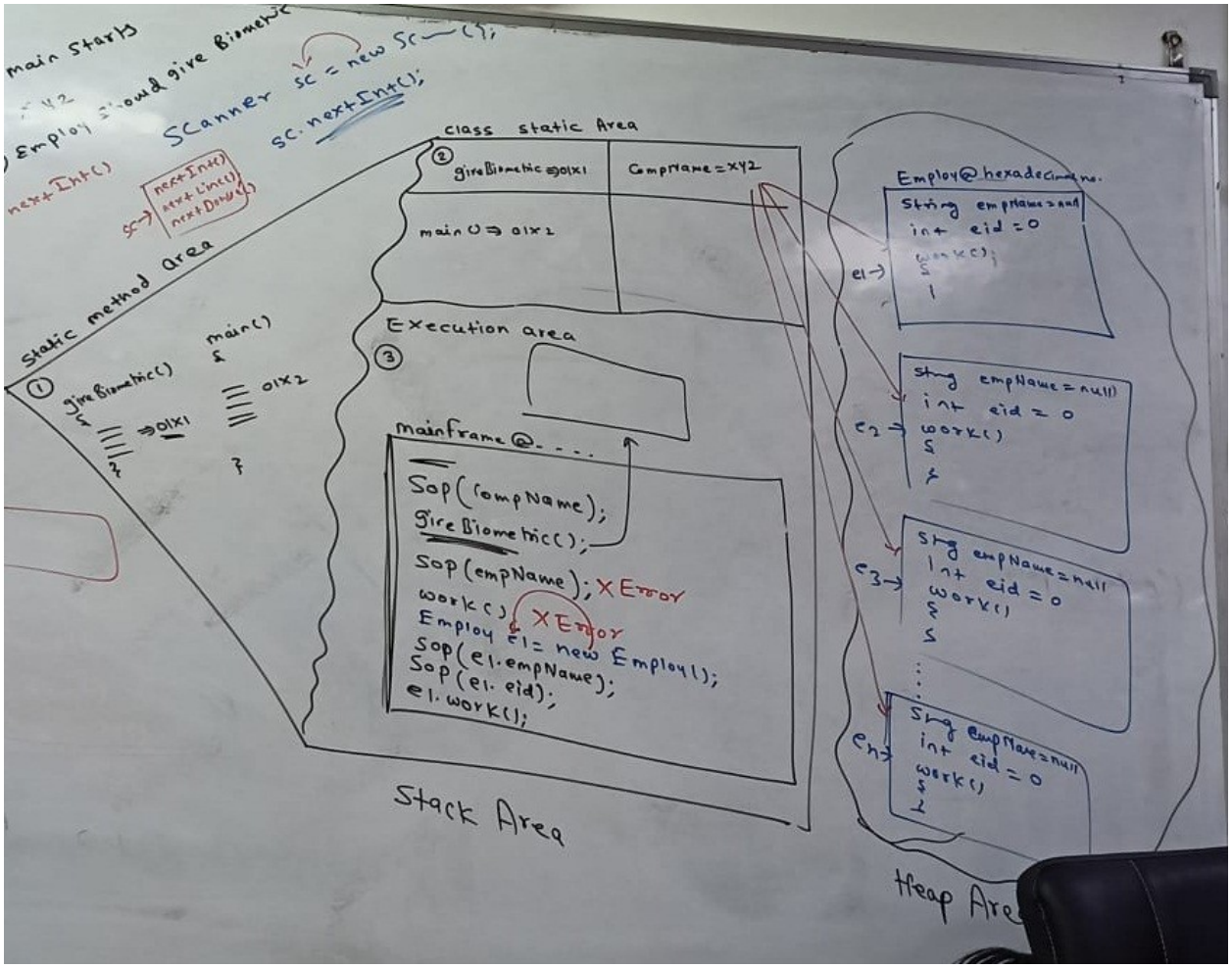
If a global variable is not declared with the keyword static then such variable is called Non-Static Variable. A Non-Static Variable will be different for different object.

***Important Note:**

- A static variable can be accessed directly inside main method.
- A non static variable can't be accessed directly inside main method. A non-static variable can be accessed inside main method only by using object reference.

```
Class employ
{
        Static String compName= "xyz";
        String empName;
        Int eid;

        Public static void giveBiometric()
        {
                Sop("Employ should give biometric");
        }
        Public void work()
        {
                Sop("Employ is working");
        }
        Public static void main(String[] args)
        {
                Sop("main starts");
                Sop(compName);
                giveBiometric();
                Employ e1=new employ();
                Sop(e1.empName);
                sop(e1.eid);
                e1.work();
        }

}
```

O/P
Main starts
Xyz
Employee should give biometric


main Starts
Employ e1 = new Employ();
Scanner sc = new Sc();
sc.nextInt();
nextInt()
sc → nextInt() nextLine() nextDouble()
Static method area
① giveBiometric() { } ⇒ 01x1
main() { } 01x2
class static Area
② giveBiometric ⇒ 01x1
CompName = XY2
main() ⇒ 01x2
Execution area
③
mainframe @....
Sop(compName);
giveBiometric();
Sop(empName); XError
work(); XError
Employ e1 = new Employ();
Sop(e1.empName);
Sop(e1.eid);
e1.work();
Stack Area
Employ@hexadecimal no.
String empName = null
int eid = 0
work()
e1 →
e2 →
e3 →
en →
Heap Area

09/11/2022

*DIFFERENCE BETWEEN STACK AREA AND HEAP AREA

Stack Area

- All local members and static member are loaded in **Stack Area.**
- Main method is executed inside stack area.
- Stack Area member will have short life.
- Stack Area members provide faster execution.

Heap Area

- Object is created at runtime dynamically inside **Heap Area** and all the non-static members are loaded inside object.
- Object is created
- Heap area member will have long life.
- Heap area members are slow in execution.

*OBJECT CREATION IN JAVA:

Object in JAVA is created with the help of “new” keyword and **constructor** of the class.
To create object in JAVA there must be blueprint or class available for which object can be created.

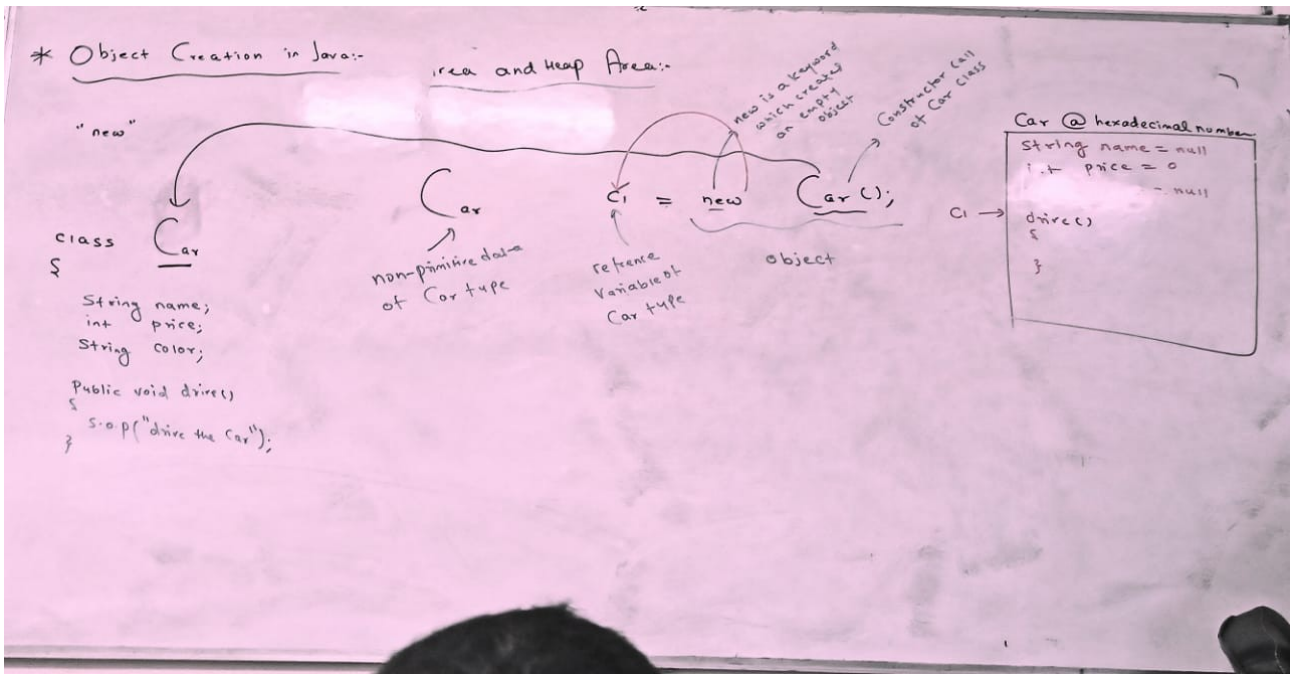
```
Class Car
{
        String name;
        int price;
        String color;
        Public void drive()
        {
                Sop(“drive the car”);
        }
}
```

Car c1 = new Car();

C1= **Car@hexadecimalvalue**

String name = null;
Int price = 0;
String color = null
drive()
{

}



***NEW KEYWORD:**

New is a keyword in JAVA which is used to create object. It creates an empty object and this object is assigned with an address in hexadecimal form, whenever **new** keyword is used then it creates a new object.

***CONSTRUCTOR:**

Constructor in JAVA is used to load all non-static members inside object. The name of constructor must be same as the class name.

***INITIALIZATION OF AN OBJECT:**

We can initialize the object in following three ways:

1. By using reference of the object.
2. By using non static method.
3. By using constructor of the class.

(1). By Using Reference Of The Object: We can initialize the object by using its refence in following ways:

```
class Car
{
        String name;
        double price:
        String color;
        public static void main(String[] args)
        {
                Car c1=new Car();
                c1.name="Maruti";
                c1.price=1234.01;
                c1.color="white";
```

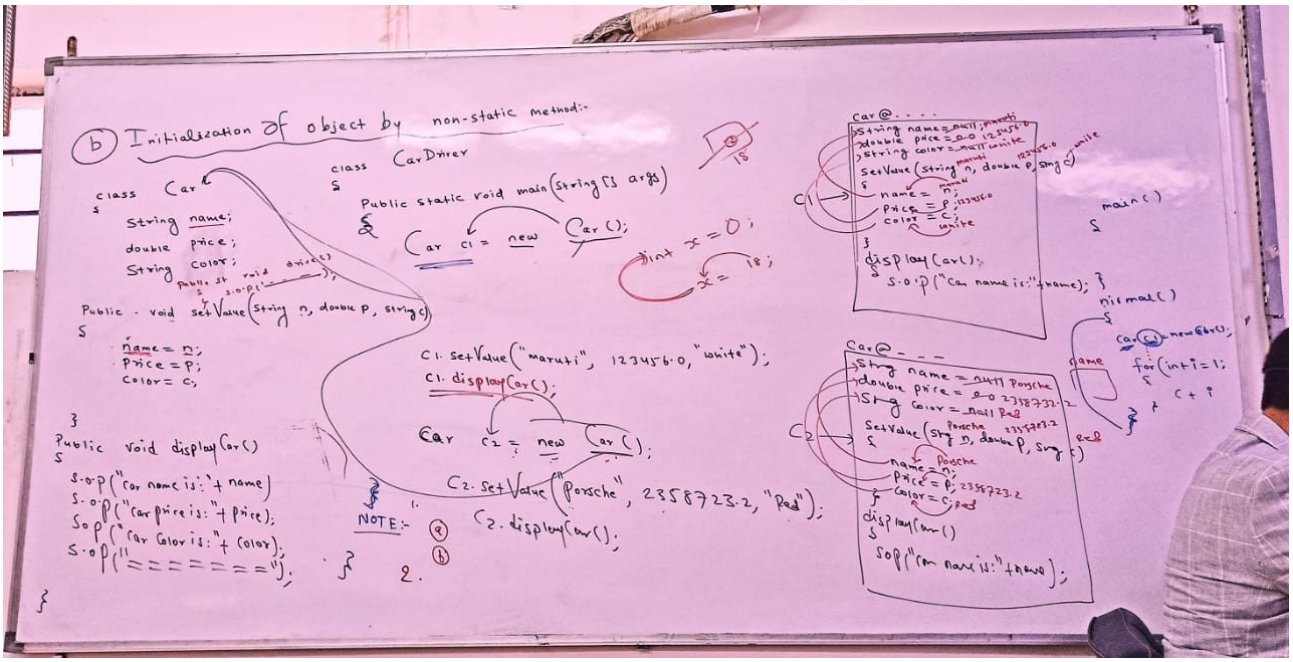
C1
name="Maruti";
Price=1234.01;
color="white";

```
                Car c2=new Car();     C2
                c2.name=”Audi”;       name=”Audi”;
                c2.price=4321.09;     Price=4321.09;
                c2.color=”Yellow”     color=”Yellow”;
        }
}
```

10-11-2022

(2). INITIALIZATION OF OBJECT BY NON-STATIC METHOD:-

When an object is created then all the non static members are loaded in the object. A non static method can be used to initialize all the members which are loaded inside object.



Note:

1. Inside a static method:

1.1 We can access all static member directly.

1.2 We cannot access non static member directly inside a static method we can accesss non static members inside a static method only an object refence.

2. Inside a non-static method we can access static members as well as non static members directly.

Question : Create objects by non static method for Browser, Tree and Vehicle.

11/11/2022

Initialization by Constructor

We can intialize the object with the help of constructor. When an object is created than the constructor loads all the non-static members inside object. So constructor can be used to load non-static members inside object as well.

```
package oops;
public class Laptop {
	String name;
	int price;
	int ram;
	Laptop(String n, int p, int r)
	{
		name = n;
		price = p;
		ram = r;
	}
	public void displayLaptop()
	{
		System.out.println("Name is : "+name);
		System.out.println("Price is : "+price);
		System.out.println("Ram is : "+ram);
	}
}

public class LaptopDriver {
	public static void main(String[] args) {
		Laptop l1= new Laptop("Dell", 5800, 9);
		l1.displayLaptop();
		Laptop l2= new Laptop("Lenovo", 5800, 9);
		l2.displayLaptop();
	}
}
```

***STATIC KEYWORD:**

Static is a keyword which is used for memory managment purposes in java. If a memebr is declared with the keyword static then such member is called **Static member.** A static member is loaded in class static area and only one copy of this member is loaded and shared with all the object in heap area.

Inside same class we can access a static member directly but inside different class we can access static member by using class name as refernce.

As static member are shared with object as well, local variable can access static memebr on object reference as well.

```
*********paste code here******************
```

Note : Static can be a Variable, Method, Block/Static Initializer Block and Class.

***STATIC VARIABLE**

If a Global Vairable in JAVA is declared with the keyword static then it is called a static variable. It is loaded in class satatic area and shared with all the object. Inside same class we can access it

directly but in different class we can access it directly but in different class we can access it on class name reference.

```
class Program1
{
	static int i=12;;
	public  static void main(String[] args)
	{
		System.out.println(i);
	}

}
class Program2
{

	public  static void main(String[] args)
	{
		System.out.println("i from program2: "+Program1.i);
		System.out.println(Math.PI);
		System.out.println(Math.E);
		System.out.println(Math.pow(12, 4));
		System.out.println(Math.sqrt(120));
		System.out.println(Math.min(12, 20));
		System.out.println(Math.max(12,20));
		System.out.println(Math.log(120));
		System.out.println(Math.exp(3));
		System.out.println(Math.sin(120));
		System.out.println(Math.sinh(120));
		System.out.println(Math.ceil(12.6));
		System.out.println(Math.floor(12.6));
		System.out.println(Math.round(12.6));
	}

}
```

Points : Local variable gets priorty inmethod if the name of the variable is same.

*********paste code here**********

Question. Is it possible that a Local variable and global variable can have same name?

Answer Yes a local variable and global variable can have same name.

Question. If a local variable and Global variable exsist with the same name which will get preference?

Answer If a local variable and global variable exist with same name the prefence is given to a local variable and global variable become hiden which is called variable hiding.

STATIC METHOD

If a method in JAVA is declared with the keyword static then it is called a **Static method**. A static method is loaded in class static area.
Inside same class we can access a static method directly but inside another class we can access a static method by class name reference.

15/11/2022

STATIC BLOCK

If a block in JAVA is declared with the keyword static then such block is called Static Block.

Syntax:
Static
{
	//static block
}

Example

```
public class Program9 {
	static int i=14;
	static int j=test();
	static
	{
		i=demo();
		System.out.println(i+"    "+j);
	}
	public static void main(String[] args) {
		int p=test();
		System.out.println(i+"   "+j+"  "+p);
	}
	public static int test()
	{
		System.out.println("test method: "+i+"   "+j);
		i=30;
		return i+12;
	}
	public static int demo()
	{
		System.out.println("demo method: "+i+"   "+j);
		int j=36;
		return 100;
	}


}
```

A static block is loaded in class static area before main method start the execution. Static block executes during it loads in class static area.

NON-STATIC BLOCK

If a block in JAVA is not declared with the keyword static then it is called non-static block.
Syntax:
{
//non-static block
}
Example

```
public class Program2 {
	static int count=0;
	{
		count++;
		System.out.println("This is non-static block-1");
	}
	public static void main(String[] args) {
		System.out.println("This is main method");
		Program2 p1=new Program2();
	}
	{
		System.out.println("This is non-static block-2: "+count);
		if(count<=20)
		{
			Program2 p=new Program2();
		}
	}
}
```

CONSTRUCTOR

Constructor is a block of statement which is used to load all non static members inside the object. Along-with loading all non static members it is used to initialize the non static members inside object.

Points:

- The name of constructor must be same as the class name.
- A constructor can have access modifier.
- A constructor cannot have non-access modifier. So a constructor can't be static, final, abstract or synchronized.
- A constructor cannot have return type.
- A constructor can accept different arguments at its parameter.
- A constructor call always takes place together with new keyword.
  Employ e1 = new Employ()

16/11/2022

TYPES OF CONSTRUCTOR

1. Default Constructor
2. User Defined constructor
2.1. No Argument Constructor
2.2. Parameterized Constructor

1. Default Constructor

```
class Employ
{
    String name ;
    int salary;

    public void work()
    {

    }
}
```

Javac Employ.java after compilation

```
class Employ
{
String name ;
int salary;
Employ()<=is added by compiler as default constructor
{

}
public void work()
{

}
}
```

```
class EmployDriver
{
    public static void main(String[] args)
    {
        Employ e1 - new Employ();
    }
}
```

A constructor which is added by compiler at compling time when there is no constructor designed by programmer then it is called **Default Constructor.** It is added only when there is no user defined constructor in the program. Default constructor help in object creation when there is no constructor designed by programmer.

NO ARGUMENT CONSTRUCTOR

If a user defined constructor takes no argument at its parameter, then it is called **No Argument constructor.**

*******paste photo of code**********

```
class Employ
{
String name;
int eid;
double salary;
Employ()
{
```

```
System.out.println("Employ object is created");
}

}
class EmployDriver
{
public public static void main      (String[] args) {
Employ E1 = new Employ();
Employ E2 = new Employ();
Employ E3 = new Employ();
Employ E4 = new Employ();
}
}
```

PARAMETER CONSTRUCTOR

If a user defined construtor takes different arguments at its parameter then such cosntructor is called **Parameterized Construtor.** Parameterized constructor is useds to load and initialize all non-static member inside object.

```
class Employ
{
   String name;
   int eid;
   double salary;
   Employ(String name, int eid, double salary)
   {
      this.name = name;
      this.eid = eid;
      this.salary=salary;
   }
   public void displayEmploy()
   {
      System.out.println("The name : " +this.name);
      System.out.println("The eid : " +this.eid);
      System.out.println("The salary : " +this.salary);
   }
}


class EmployDriver
{
   public static void main(String[] args)
   {
      Employ E1 = new Employ("Mohan", 101, 23992.2);
      E1.displayEmploy();
   }
}
```

17/11/2022

^This Keyword^

This is a keywords which is used to refer global non static members of the same class. This keyword holds the reference of current object under execution

//this keyword example

```
class Employ
{
    String name;
    int eid;
    double salary;
    Employ(String name, int eid, double salary)
    {
        this.name = name;
        this.eid = eid;
        this.salary = salary;
    }
    public void displayEmploy();
    {
        System.out.println("name is: "+this.name);
        System.out.println("eid is: "+this.eid);
        System.out.println("salary is: "+this.salary);
    }
}

class EmployDriver
{
    public static void main(String[] args)
    {
        Employ e1 = new Employ();
    }
}
```

Constructor OverLoading

Constructor Over Loading where we can have multiple constructors inside same class with different parameters. We can achieve constructor over loading in three different ways.

1. By changing the number of parameters at constructor.
Ex.

```
Dog(String name)
{
    this.name=name;
}
Dog(String name, Double height)
{
    this.name= name;
    this.height= height;
}
```

2. By changing the data type of parameter at constructor.
Ex.

```
Dog(String name)
{
    this.name=name;
}
Dog(double height)
{
    this.height=height;
}
```

3. By changing the sequence of parameters at constructor.
Ex.

```
Dog(String name, double height)
{
    this.name=name;
    this.height=height;
}
Dog(double height, String name)
{
    this.height=height;
    this.name=name;
}
```

Copy Constructor

++++++see photo++++++++++ of logic++++++++++++

Copy Constructor is a special type of constructor which is used to copy details of first constructor inside another constructor.

Ex.

```
//Copy Constructor example
class Employ
{
    String name;
    int eid;
    double salary;
    Employ(String name, int eid, double salary)
    {
        this.name = name;
        this.eid = eid;
        this.salary = salary;
    }
    //copy constructor logic
    Employ(Employ e)
    {
        this.name = e.name;
        this.eid = e.eid;
        this.salary = e.salary;
    }
    public void displayEmploy();
    {
```

```
        System.out.println("name is: "+this.name);
        System.out.println("eid is: "+this.eid);
        System.out.println("salary is: "+this.salary);
    }
}

class EmployDriver
{
    public static void main(String[] args)
    {
        Employ e1 = new Employ();
        Employ e2 = new Employ(e1);       //copy constructor object call
    }
}
```

18/11/2022

this() statement

This call statement is used to call constructors of the same class.

Syntax:

this(varName1, varName2, ...);

Constructor Chaining

Constructor chaining is a process to call one constructor from inside another constructor. Construtor chaining is achieved with the help of **this() statement.** Inside constructor the constructor call must be the first statement. So, **this()** call must be the first statement inside the constructor.

Example:

```java
class Employ{

	String name;
	int eid;
	double salary;
	int age;

	Employ(){
		System.out.println("Employ object is created");
	}
	Employ(String name){
		this();
		this.name=name;
	}
	Employ(String name, int eid){
		this(name);
		this.eid=eid;
	}
	Employ(String name, int eid, double salary){
```

```
            this(name, eid);
            this.salary=salary;
      }
      Employ(String name, int eid, double salary, int age){
            this(name, eid, salary);
            this.age=age;
      }
}

class EmployDriver {

      public static void main(String[] args) {
            Employ e1=new Employ("Mohan",101,58326.5,23)
      }
}
```

IMPORTANT QUESTIONS

Question. What is Constructor?
Question. What is differnce between method and constructor?
Question. What is default constructor?
Question. Constructor overloading
Question. Copy Constructor
Question. This keywords
Question. this() Statement
Question. Constructor Chaining

METHOD OVERLOADING

If in a class there are multiple methods with same name and different parameters. Then it is called **method overloading**. In method overloading **Non Access Modifier** and **Return type** does not play any role.

Different ways to achieve Method Overloading:

1. By changing the number of parameter.

Ex.
Public static void test(int a)
{

}
public static void test(int a, int b)
{

}

2. By changing the type of parameter.

Ex.
Public static void test(int a)
{

}
public static void test(double a)
{

}

3. By changing the sequence of parameter.

Ex.
Public static void test(int a, double b)
{

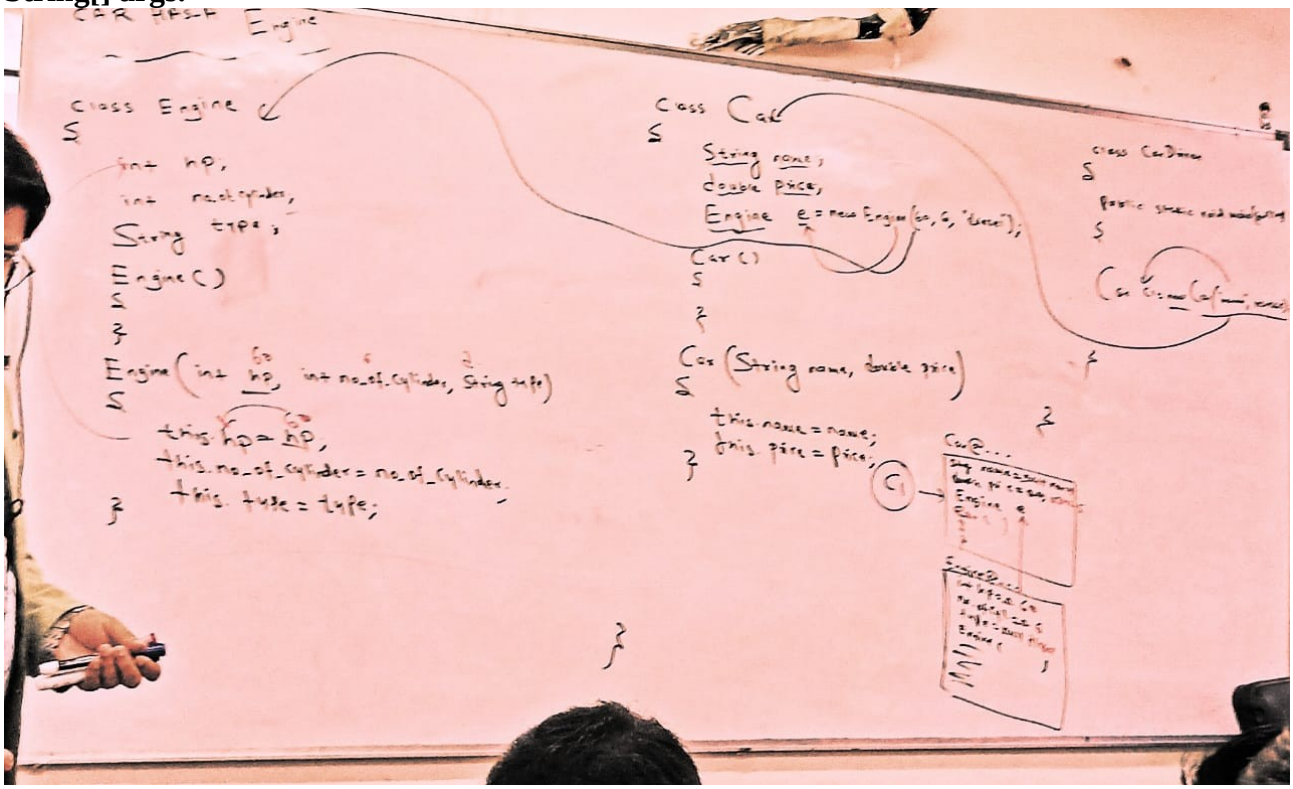
}
public static void test(double a, int b)
{

}

21/11/2022

Important
Questions. Can we overload main method in JAVA or not ?

Answer. Yes, We can have multiple main methods in JAVA with different parameters.
***** At the time of execution only one main method will execute where parameter is **String[] args.** Other main method will only execution when it is called inside main method with parameter **String[] args.**



Relationships

The connection between two classes or between two objects is called **Relationship**. It is of two types :

1. HAS-A Relationship
2. IS-A Relationship(Inheritance)

^HAS-A Relationship^

HAS-A Relationship is used to represent a relationship which may be completely dependent or partially dependent. HAS-A Relationship is of two types:

1. Composition

If a HAS-A Releationship is a type of relationship where one object is completely dependent on another object. The dependent object cannot exist without main object.

Example:

Car	HAS-A	Engine
Mobile	HAS-A	Screen
Laptop	HAS-A	Processor
Tree	HAS-A	Fruit

Examples:

```
class Engine
{
int hp;
int strokes;
String type;
Engine()
{

}
Engine(int hp, int strokes, String type)
{
this.hp=hp;
this.strokes=strokes;
this.type=type;
}
}

class Car
{
String name;
int price;
Engine e=new Engine(60, 6, "Diesel");
Car()
{
```

Two objects created
Car@....
String name = null;//**maruti**
Int price = 0;//**123456**
Engine e;
Car()
{

}
Car(String name, int price)
{
This.name=name;
This.price=price;
}

Engine@...
Hp=0//**60**
Strokes=0//**6**
type=null//**diesel**
Engine()
{

}
Engine(int hp, int strokes, string type)
{
This.hp=hp;
This.strokes=strokes;
This.type=type;
}

```
}
Car(String name, int price)
{
this.name=name;
this.price=price;
}

}


class CarDriver
{
public static void main(String[] args)
{
Car c1 = new Car("Maruti", 123456);//compasition HAS A Relationship
System.out.println("Car Object address: "+c1);
System.out.println("Car engine object address: "+c1.e);
System.out.println("Car name: "+c1.name);
System.out.println("Car price: "+c1.price);
System.out.println("Car engine hp: "+c1.e.hp);
System.out.println("Car engine strokes: "+c1.e.strokes);
System.out.println("Car engine type: "+c1.e.type);

}
}
```

2nd EXAMPLE

```
class Engine
{
int hp;
int strokes;
String type;
Engine()
{

}
Engine(int hp, int strokes, String type)
{
this.hp=hp;
this.strokes=strokes;
this.type=type;
}
public void displayEngine()
{
System.out.println("Car hp is: "+hp);
System.out.println("Car strokes is: "+strokes);
System.out.println("Car type is: "+type);
}
}
```

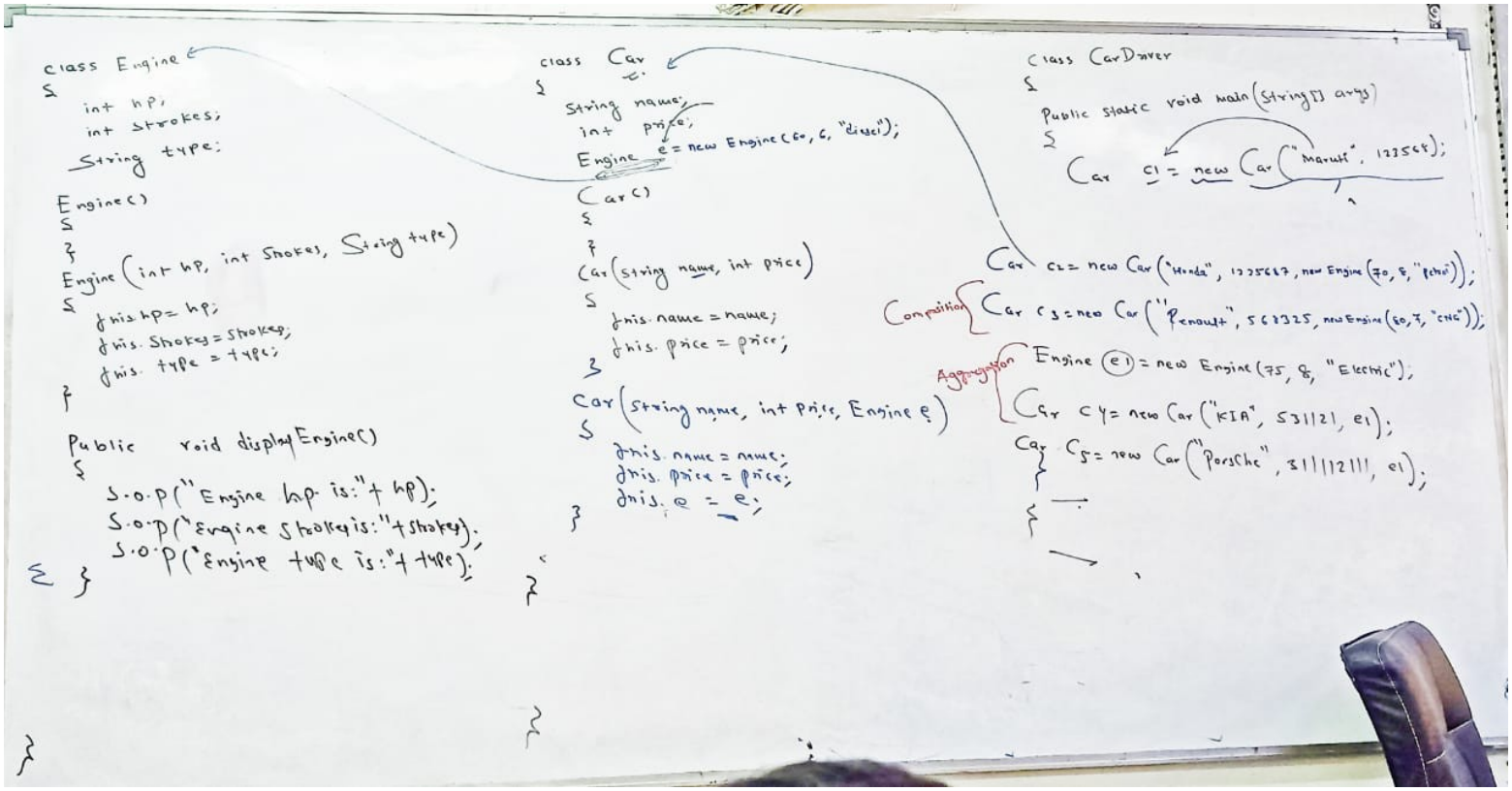
```
class Car
{
String name;
int price;
Engine e=new Engine(60, 6, "Diesel");
Car()
{

}
Car(String name, int price)
{
this.name=name;
this.price=price;
}
public void displayCar()
{
System.out.println("Car name is: "+name);
System.out.println("Car price is: "+name);
System.out.println("Car engine hp is: "+e.hp);
e.displayEngine();
}

}
class CarDriver
{
public static void main(String[] args)
{
Car c1 = new Car("Maruti", 123456);//compasition HAS A Relationship
System.out.println("Car Engine Details");
c1.e.displayEngine();
System.out.println("Car Engine Details");
c1.displayCar();

}
}
```

2. Aggregation

If HAS-A Relationship is a week relationship where one object can exist without another object then such relationship is called **Aggregation.**

Example:k

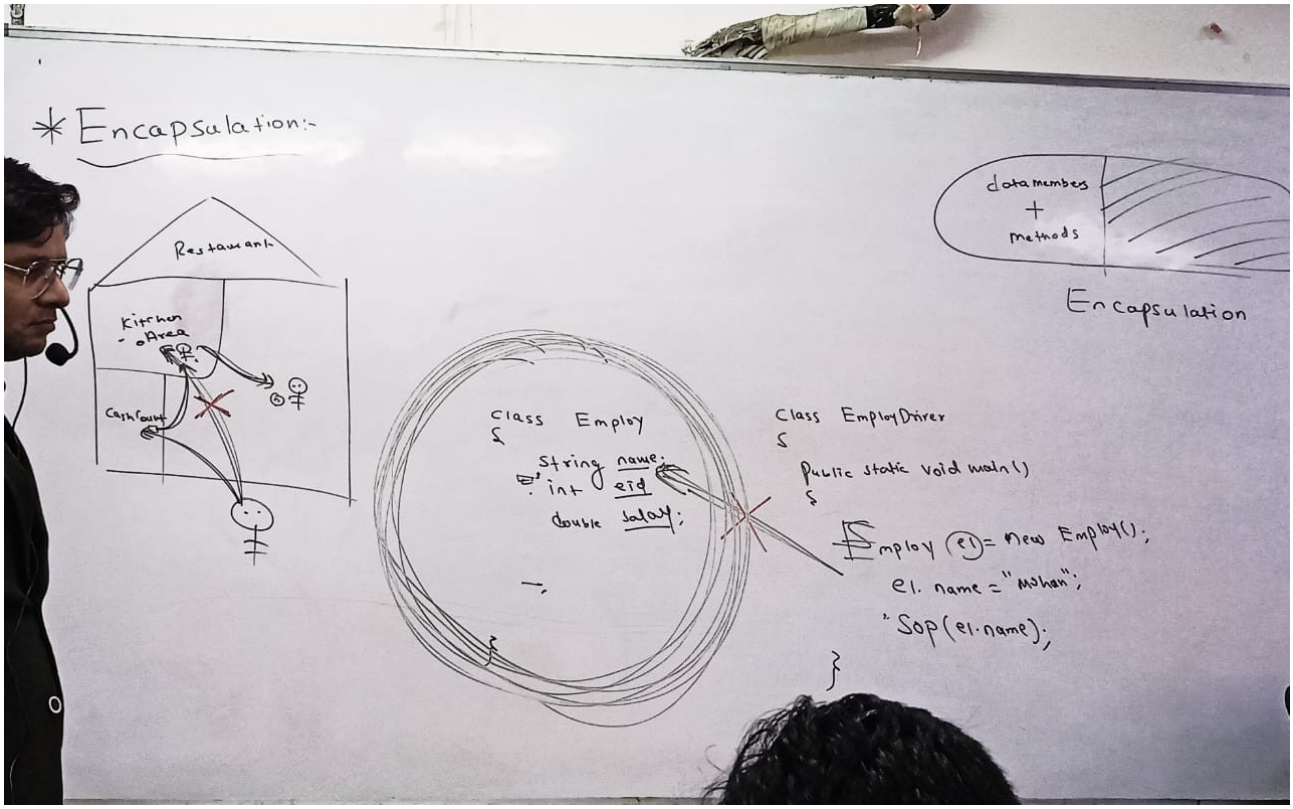
Student	HAS-A	Address
Mobile	HAS-A	Cover
Car	HAS-A	Music Player



24/11/2022

ENCAPSULATION

Encapsulation is a process to restrict direct accessiblity and provide indirect and validated accessibility.
In other words incapsulation is a process to rapper/rapup data members inside a single class so that other class should not be able to access it directly.



<How to achieve Encapsulation>

Step1. To achive encapsulation we have to make all data members private. A private member is visible or accessible only inside same class but not visible or accessible from any other class.

```
class Employ
{
	private String name;
	private  int eid;
	private double salary;

	public static void main(String[] args)
	{
		System.out.println(name);
	}
}
```

Step2. To provide the accessibility of the private members there should be public type of **getter() method and setter() method.**

2.1 Design of **getter() method:**

A **getter() method** is a public type method which is used to get or display or read private member of a class. Inside **getter() method** validation is performed and if validation is successfull then accessibility will be provided.
Every private members should have their own **getter() method**

```
class Employ
{
        private String name;
        private   int eid;
        private double salary;

        public String getName()
        {
         //validation
         return name;
        }

        public int getEid()
        {
        //validation
        return eid;
        }

        public double getSalary()
        {
        //validation
        return salary;
        }

}
```

2.2 Design of **setter() method:**
A **setter() method** is a public type method which is used to set or update or write the details of private memebrs. For different private members there should be different **setter() method.**

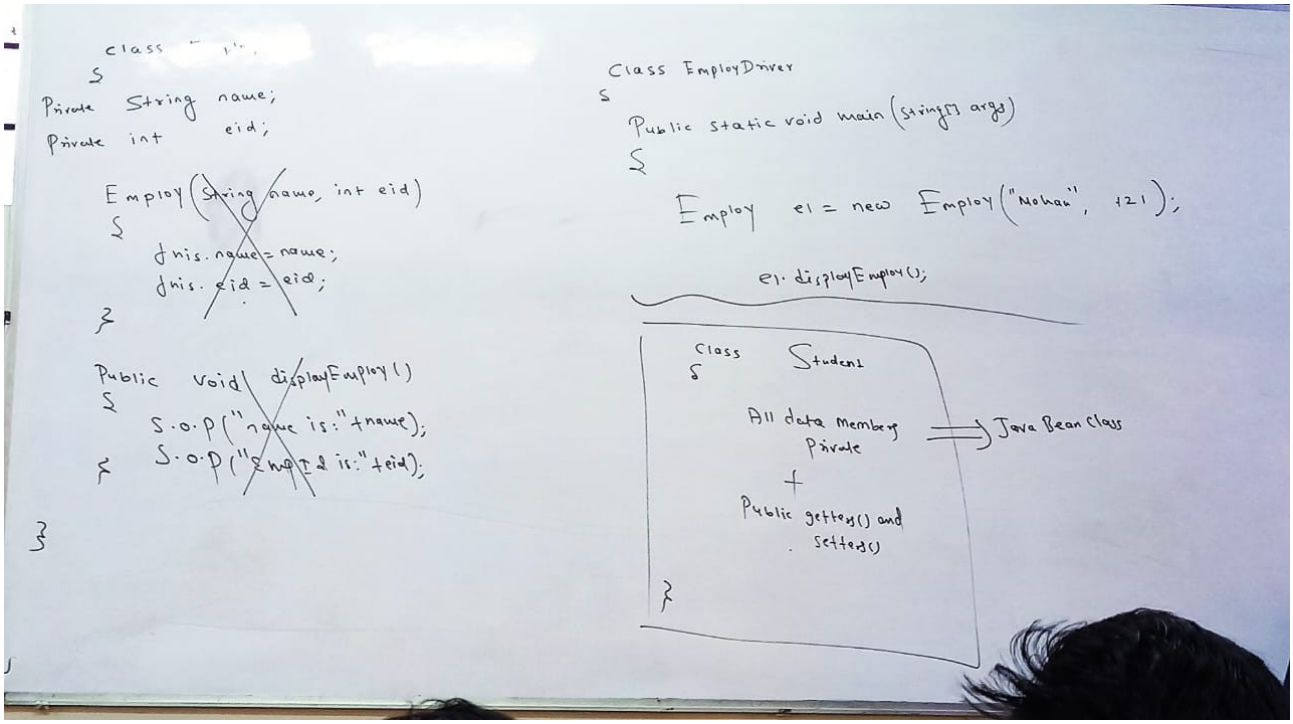
```
class Employ
{
        private String name;
        private     int eid;
        private double salary;

        public String getName()
        {
         //validation
         return name;
        }

        public int getEid()
        {
        //validation
        return eid;
        }

        public double getSalary()
        {
        //validation
        return salary;
        }

```

```
	public void setName(String name)//SETTER
	{
		//validation
		this.name==name;
	}
}

class EmployDriver
{
	public static void main()
	{
		Employ e1= new Employ;
		e1.name= "Mohan;
		e1.setName("Mohan");//SETTER
		System.out.println(e1.getName());
	}
}
```

Questions:
Achieve Encapsulation for BANK, FACEBOOK AND CLUB

25/11/2022

Step3. An encapsulated class should not have constructor and display method.



^JAVA BEAN CLASS^

If in a class all data members are private and there are public getter () and setter () method designed for every private members then it is called **JAVA Bean Class.**

Class Strudent
{
All data members private
+
Public getter() and setter() method

}

The object created from **Java Bean Class** is called **Bean Object.**

Student s1= new Student();// s1 is bean object

Data Hiding

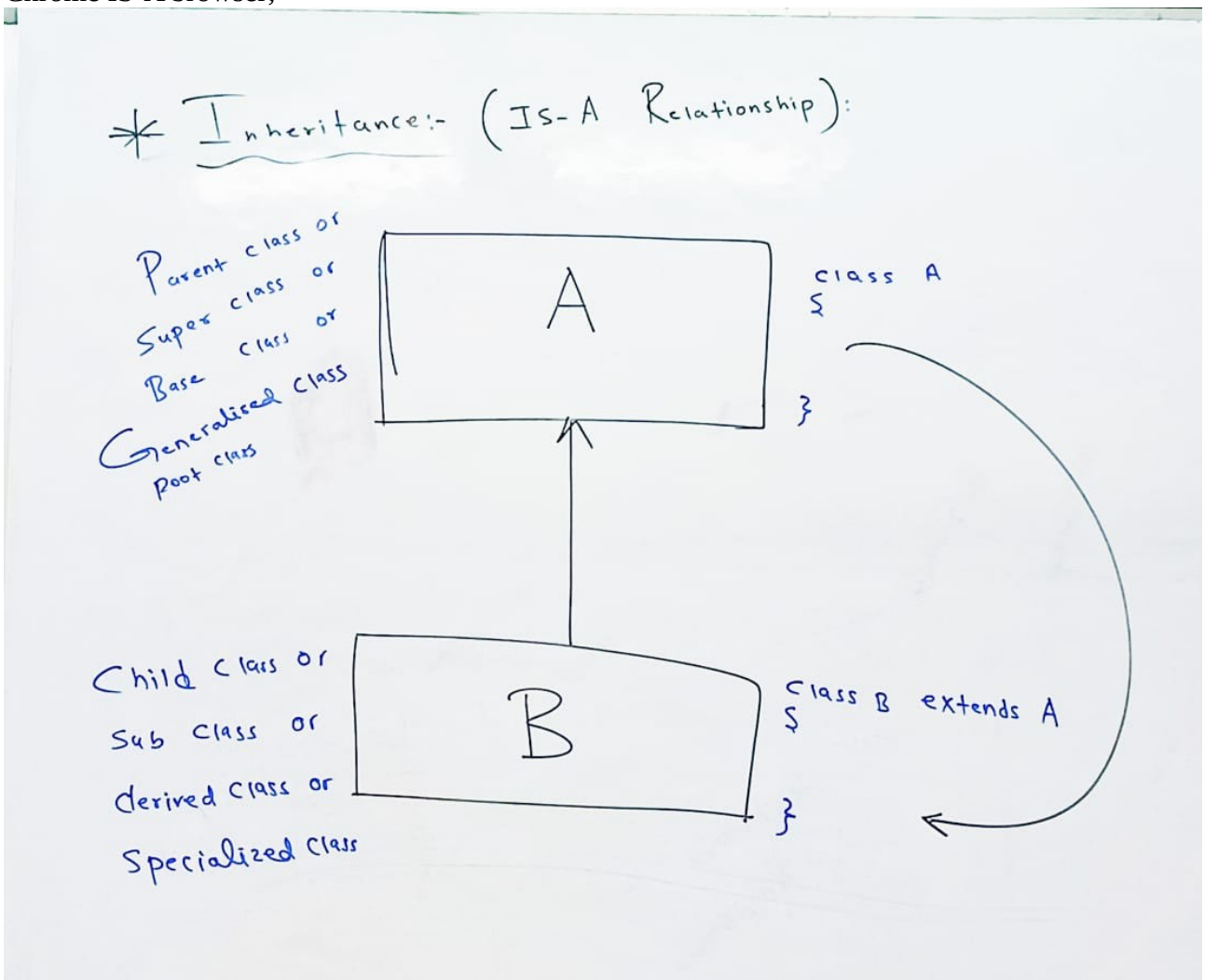
Data Hiding is a proccess to stop direct accessibility of a class data member and provide indirect and validated accessibility through **getter() and setter() method.** By using data hiding we can:-

1. Only set the private data = if only **setter()** is designed.
2. Only get the private data = if only **getter()** is designed.
3. Get and set both = if both **getter() and setter()** are designed.
4. Neither get nor set = if **getter() and setter()** are not designed.

Important:
<u>*INHERITANCE(IS-A RELATIONSHIP)*</u>

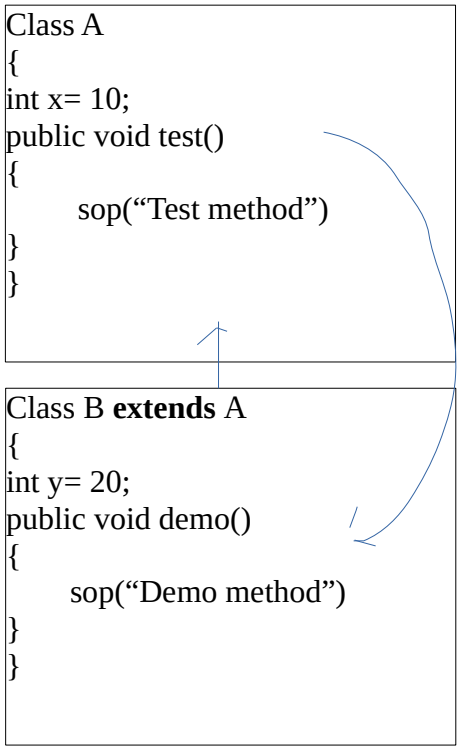
A. What is inheritance?
B. Types of inheritance.
C. Why multiple inheritance not supported in java with class?
D. Super Keywords
E. Super() Statement
F. upCasting and downCasting
G. Method overriding

Inheritance is used to represent parent child relationship where a child class inherits all the properties from a parent class. **Inheritance** in java represents **IS-A Relationship. Inheritance is Unidirectional** so, only a child can inherit all the properties from a parent. But, parent will not have the properties of child.
In Java **inheritance** is achieved with the help of **keyword "extends"** where a child class always extends to a parent class to inherit the properties.
Ex:
Mango IS-A Fruit;
Roti IS-A Food;
MarkerPen IS-A pen;
Gun IS-A weapon;
Chrome IS-A browser;



Types of Inheritance

1. Single Level Inheritance:- If there is one parent class and one child class then such inheritance is called **Single Level Inheritance.**
We can access global members directly by inheritance. Consctructor will not be inherited.



Total member = 2
2 = declared memebr

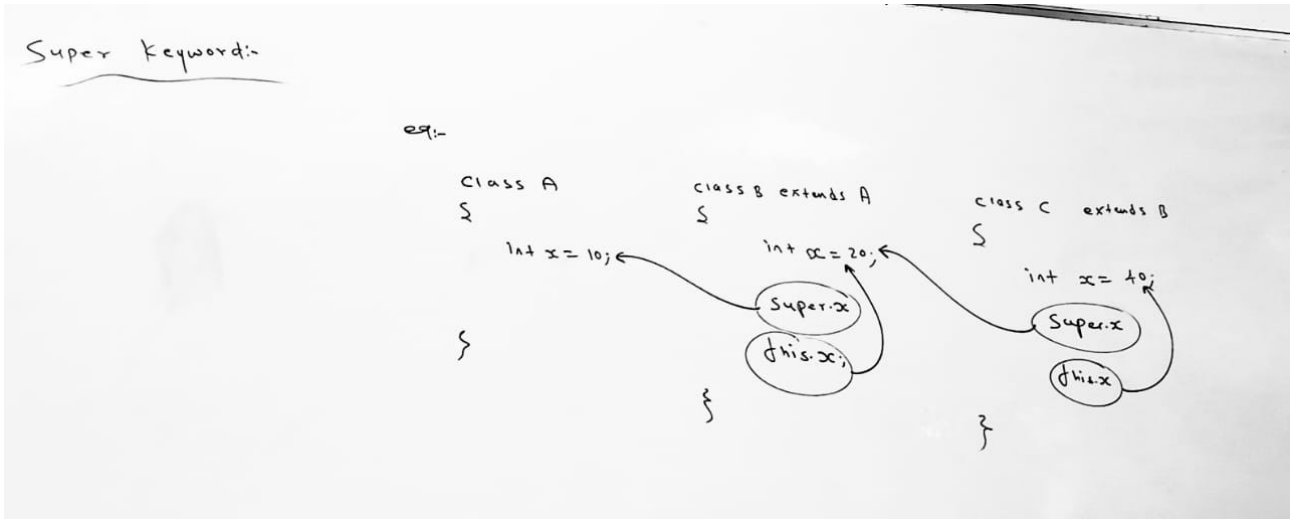
Total member = 2
2 = declared memebr

Super keyword is used to access the variable of parent keyword. **super**//keyword**.a**//variable name

29/11/2022

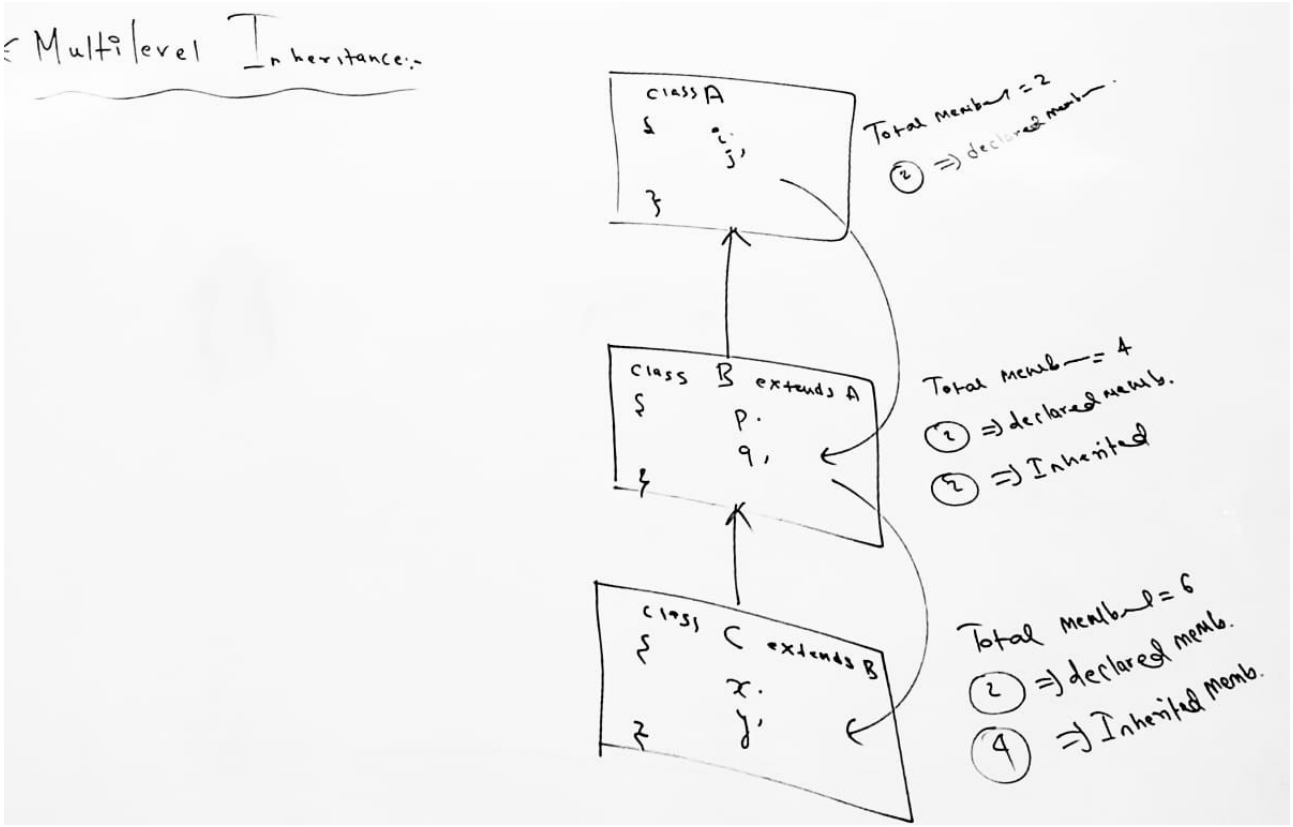
SUPER KEYWORD

Super is a keyword which is used to refer non-static member of immediate parent class.



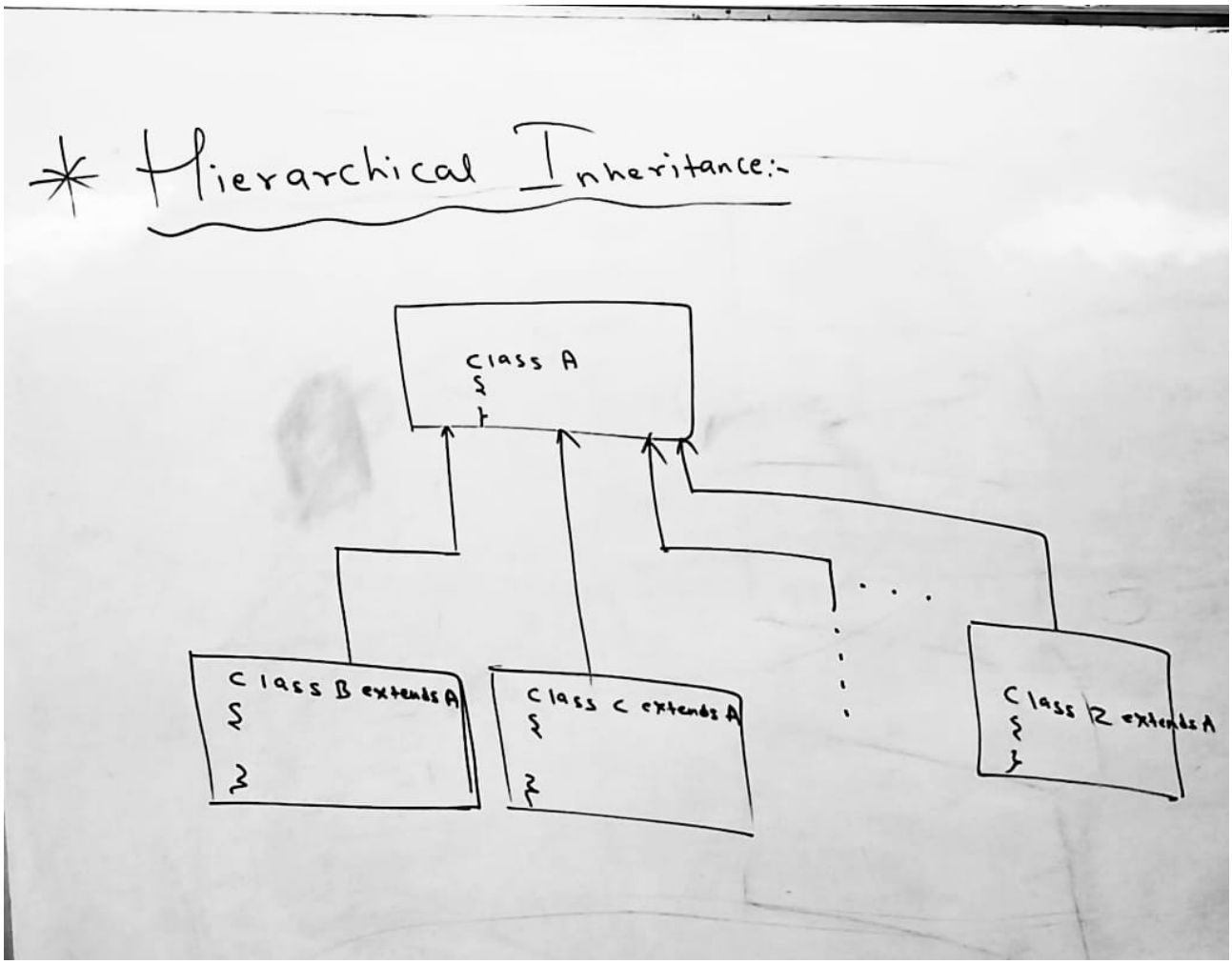
MULTI LEVEL INHERITANCE

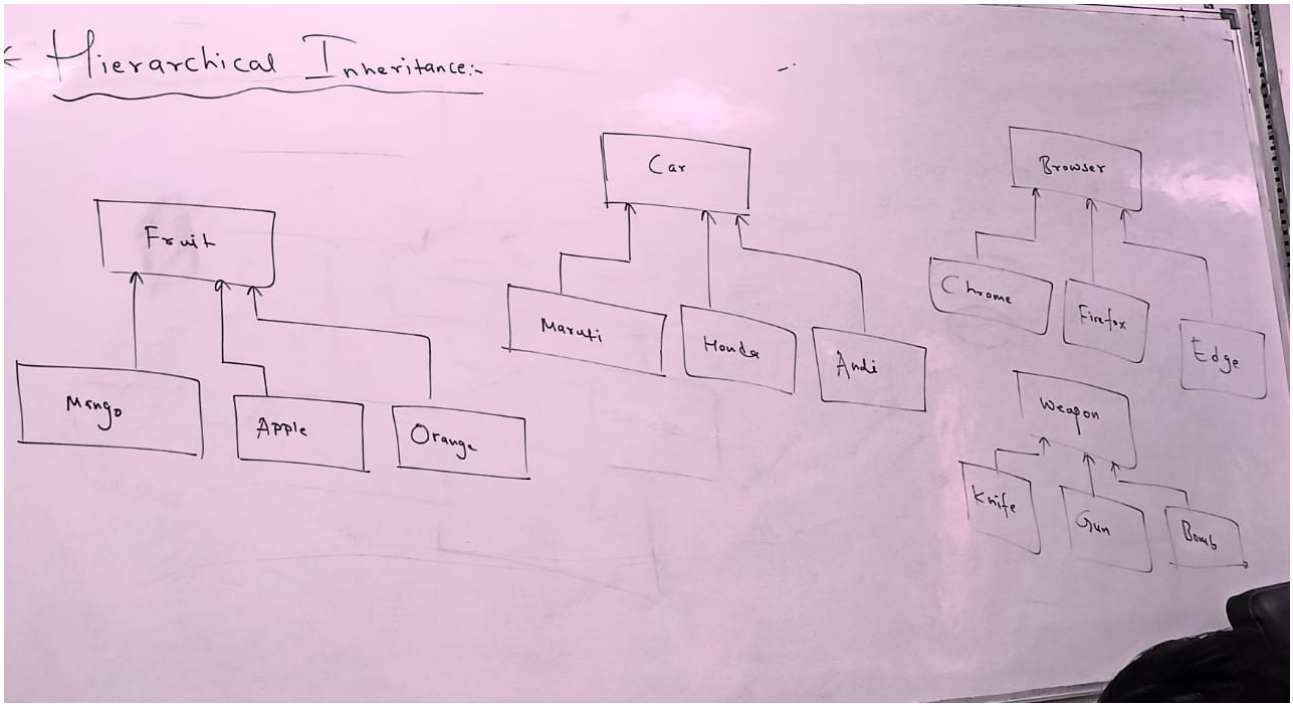
If there is a parent class and child class and child class is parent class for another child class, then such hierarchy is called **Multi-Level Inheritance.**



HIERARCHICAL INHERITANCE

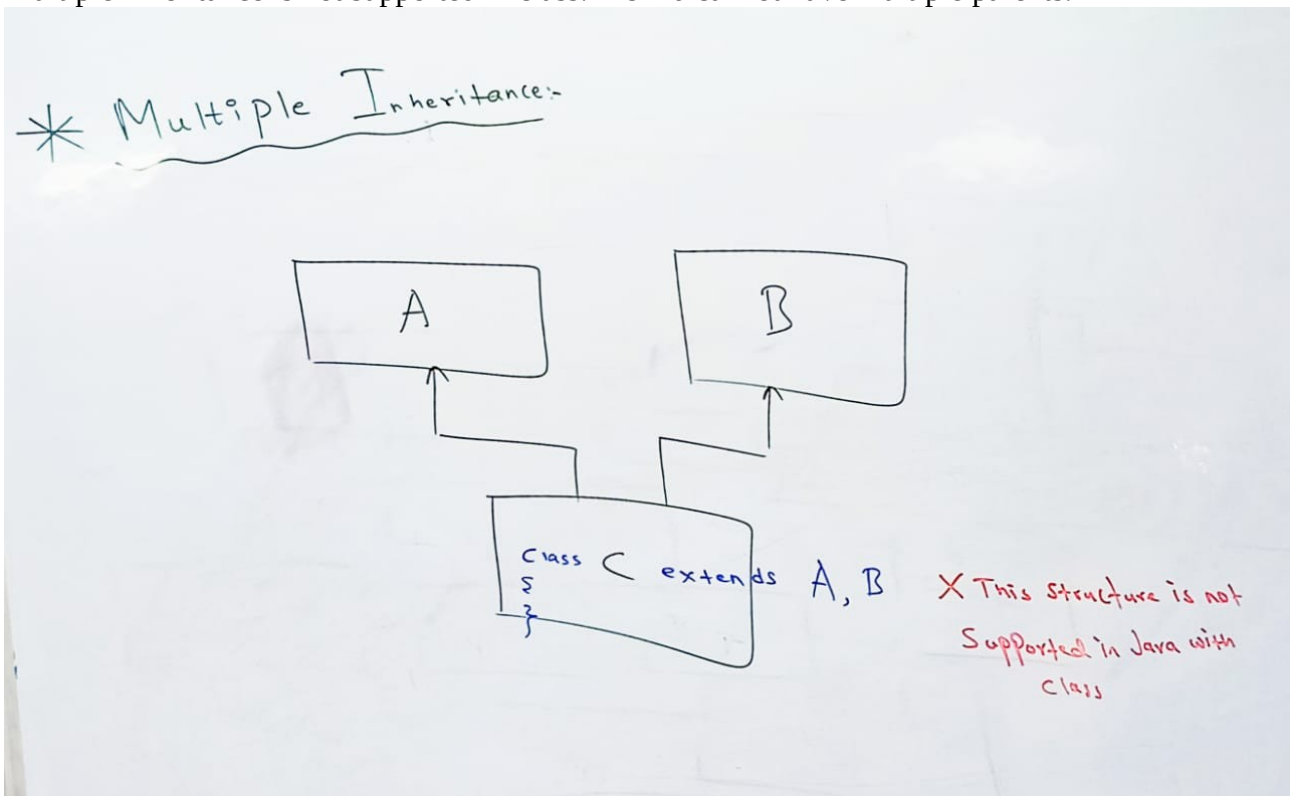
If there is a parent class and multiple child class then such inheritance is called **Hierarchical Inheritance.**





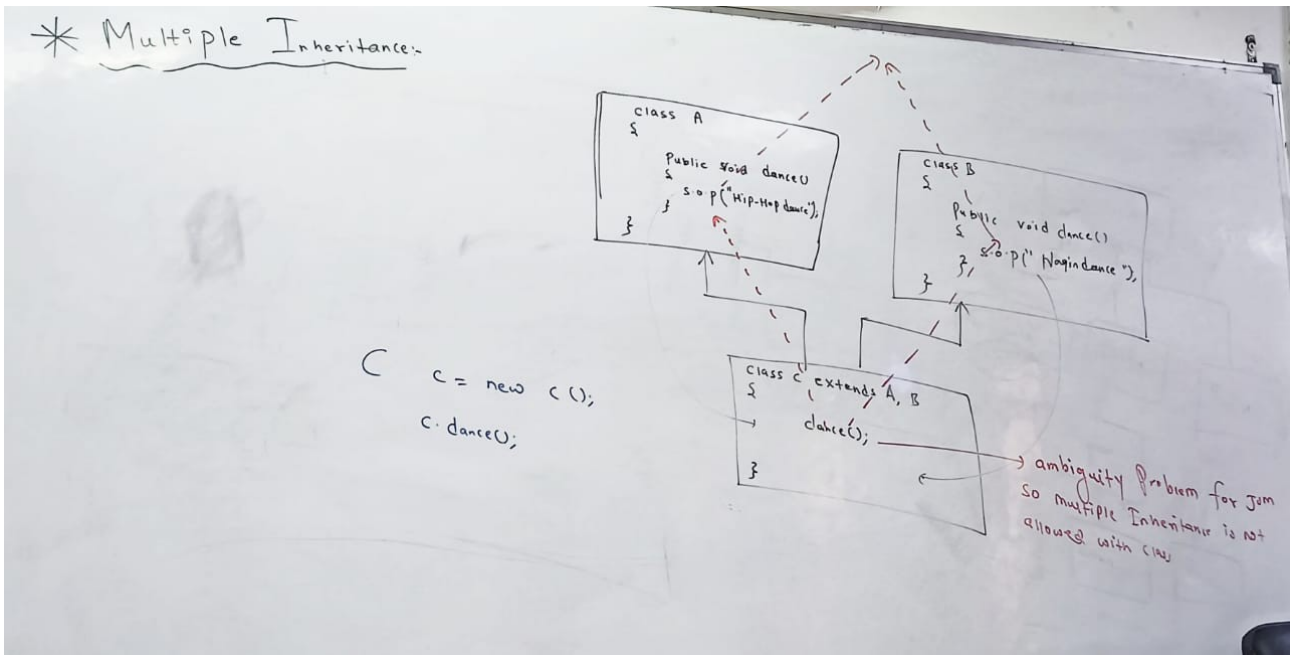
MULTIPLE INHERITANCE

If there is multiple parent class and single child class then it is called **Multiple Inheritance.**
Multiple Inheritance is not supported in JAVA with class.
Multiple inheritance is not supported in class. A child cannot have multiple parents.



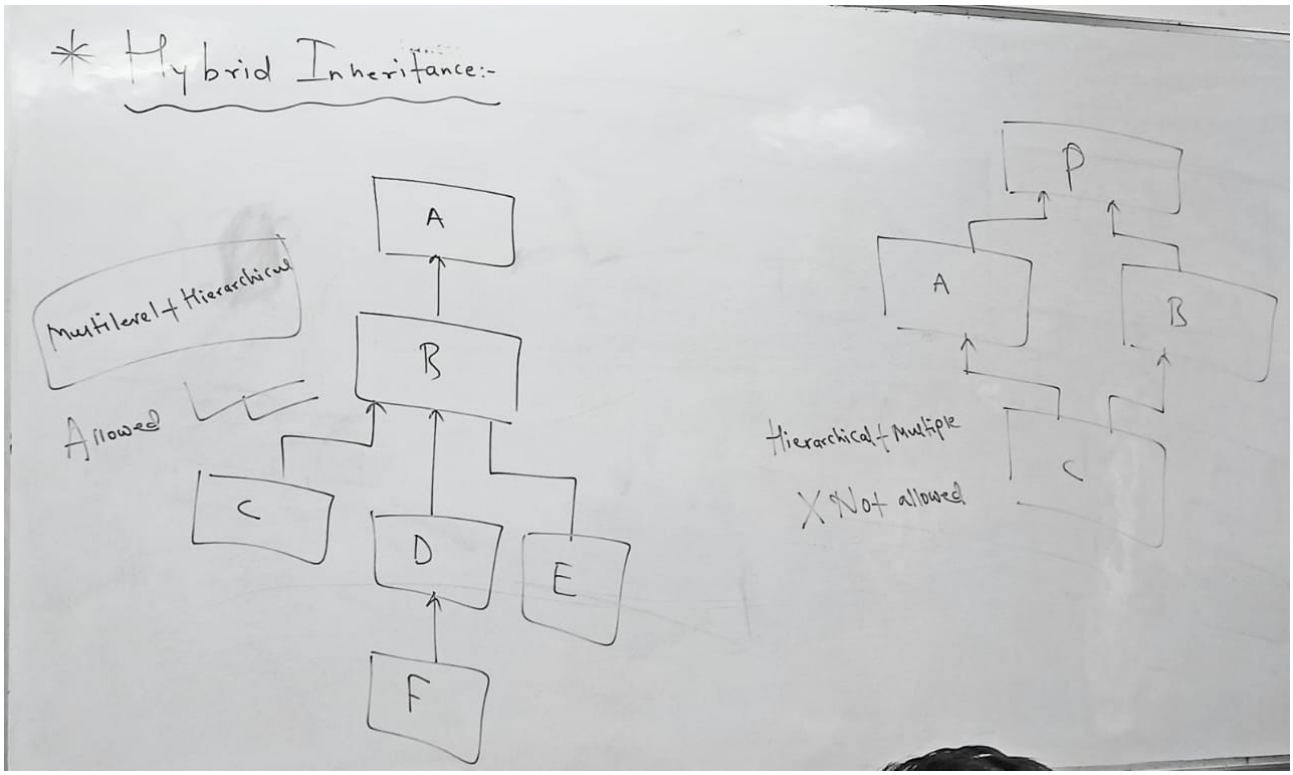
Ques. Why Multiple Inheritance is not supported in JAVA with class.

Ans. In **Multiple Inheritance** if Multiple Parent has methods with the same name then when this method is called on child object reference then it createsconfusion for JVM that which parent class method should be implemented. This problem of JVM is called **ambiguity problem** for JVM, because of this **ambiguity problem for JVM multiple inheritance** is not allowed in JAVA with class.
This problem of **Multiple Inheritance** is also called diamond shape problem in JAVA. Diamond shaped problem in JAVA is resolved with the help of **Interface.**



HYBRID INHERITANCE

The combination of two or more then two inheritance is called **Hybrid Inheritance.**



Note:
1. A child class cannot inherit the private member of parent class.

30/11/2022

Very Important
NON-PRIMITIVE TYPE CASTING

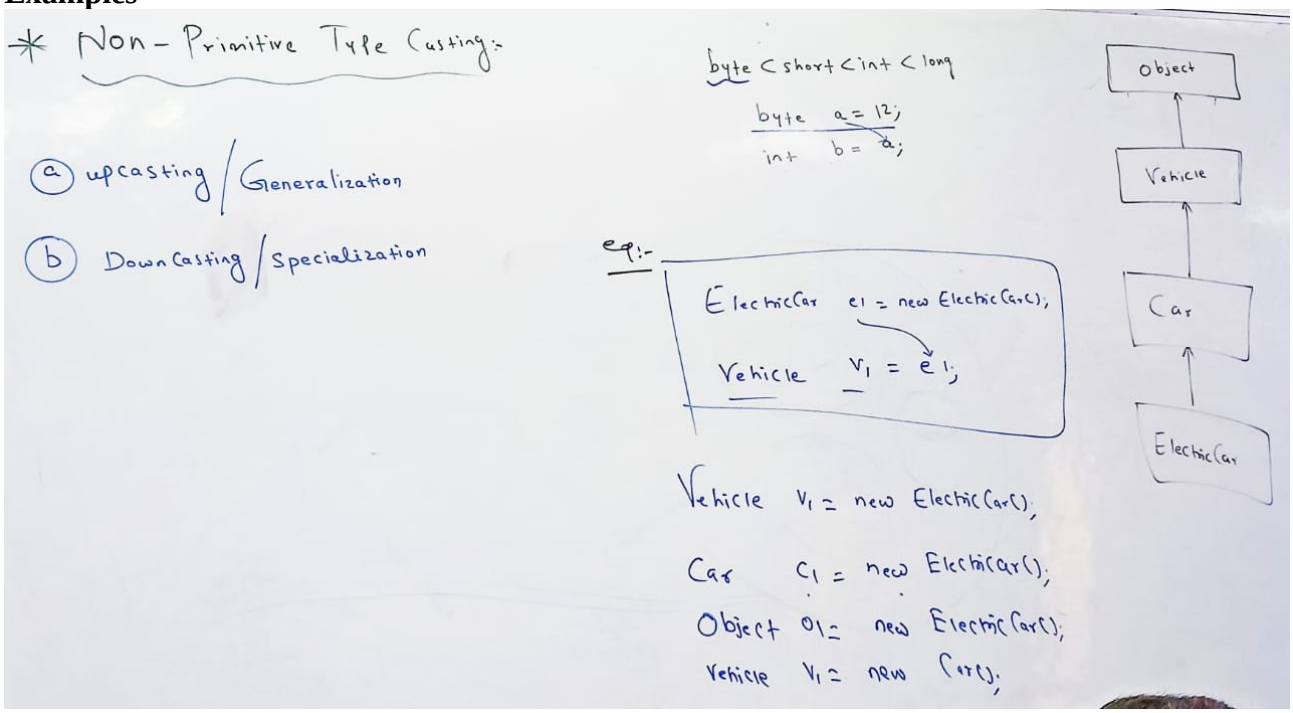
Converting a non-primitive data into another non primitive data is called **Non-Primitive Type Casting.** To achieve non primitive type casting there must be **Parent child relation ship or IS-A relationship.** It is of two types:

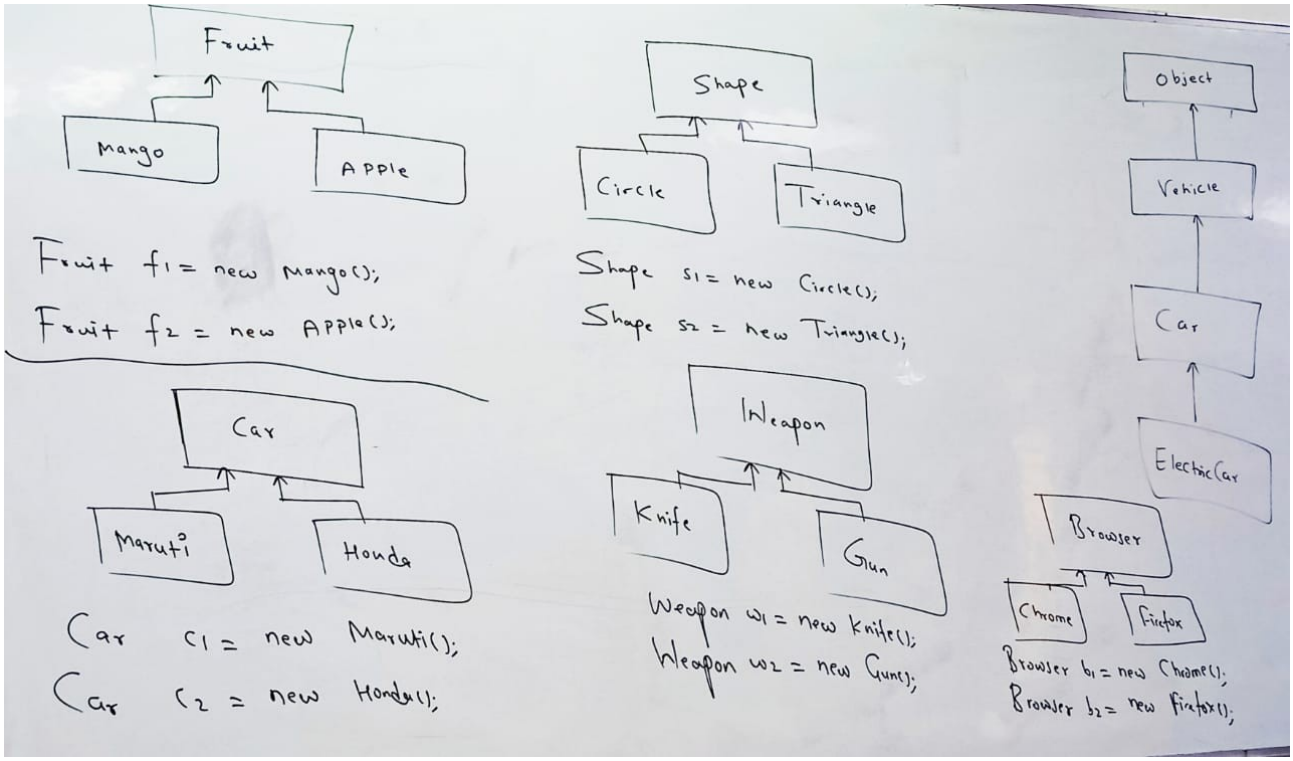
1. UpCasting or Generalization
2. DownCasting or Specialization

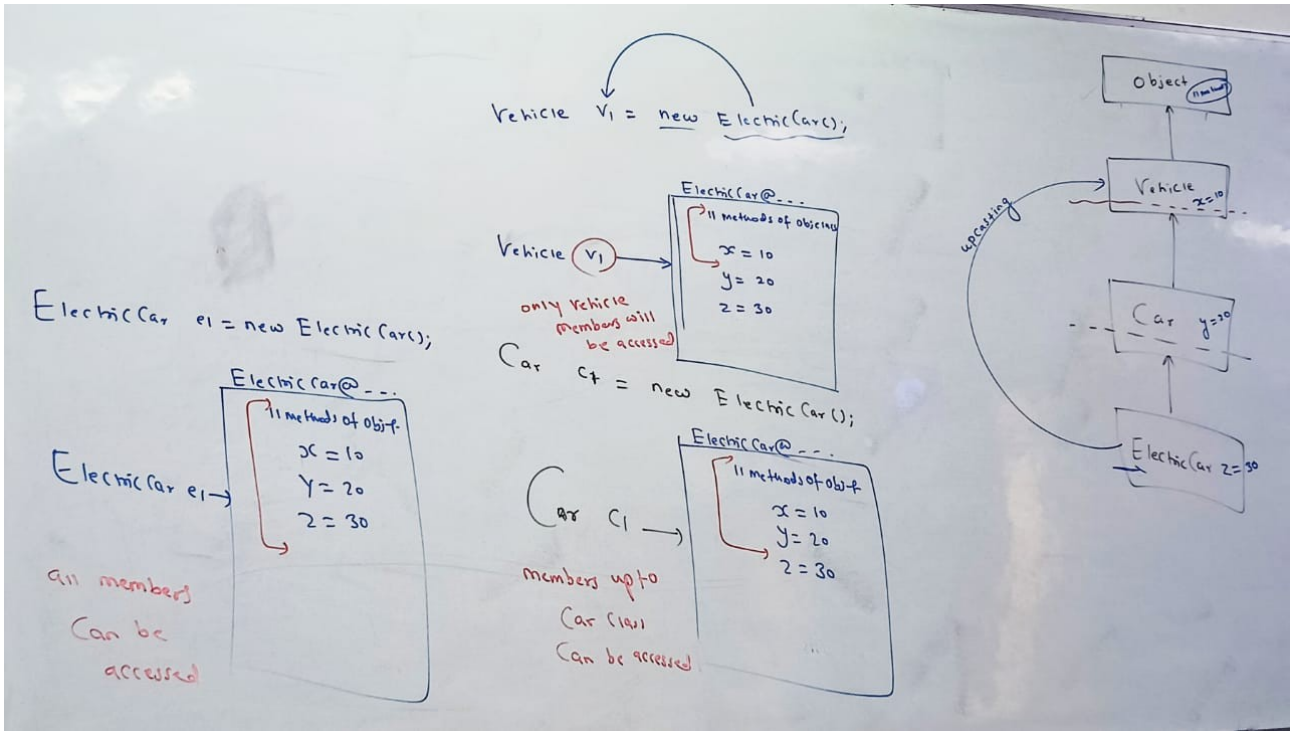
^UpCasting or Generalization^

UpCasting is a proccess to convert or store a child type object on parent reference.
In other words if a child object is stored on parent type reference then it is called upcasting. On upcasting we can access only parent member and child members can not be accessed.

Examples

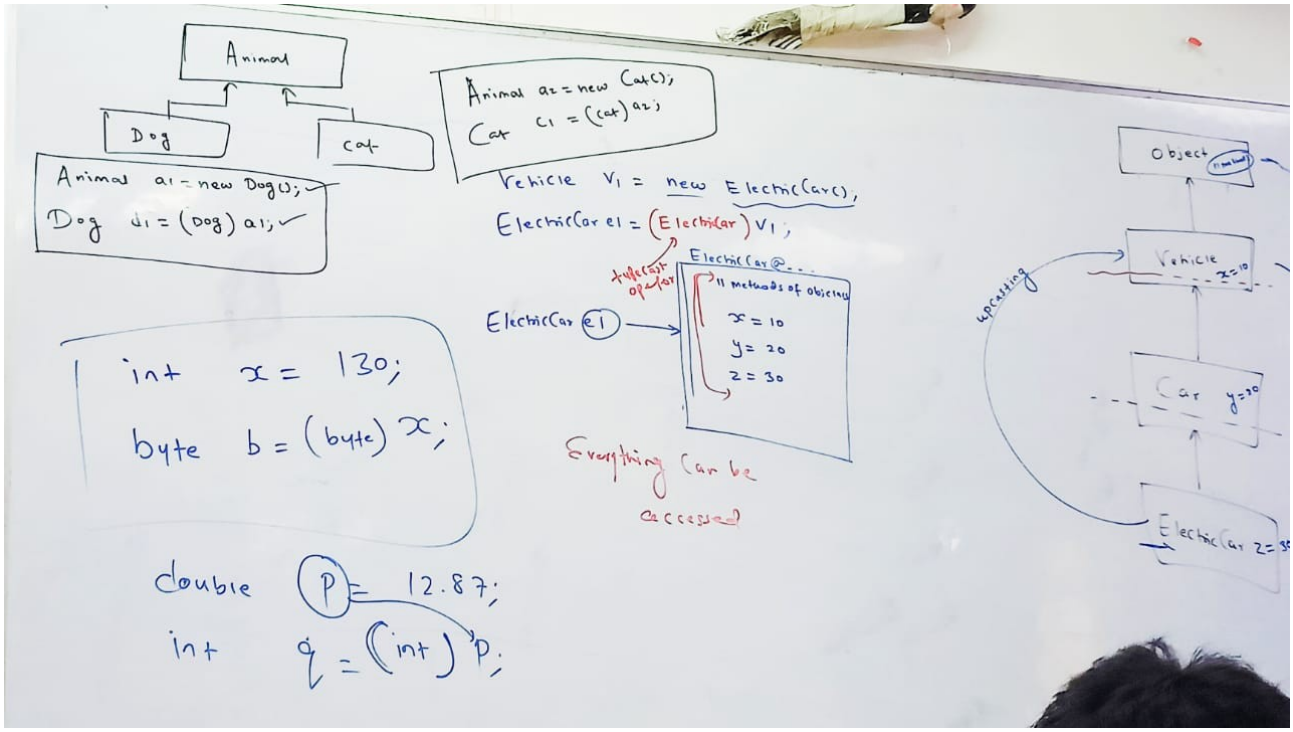






^DownCasting or Specialization^

Converting parent type refernce into child type is called **DownCasting or Specialization.** On downcasting we can access parent members as well as child members. To achieve downcasting there should be upcasting.

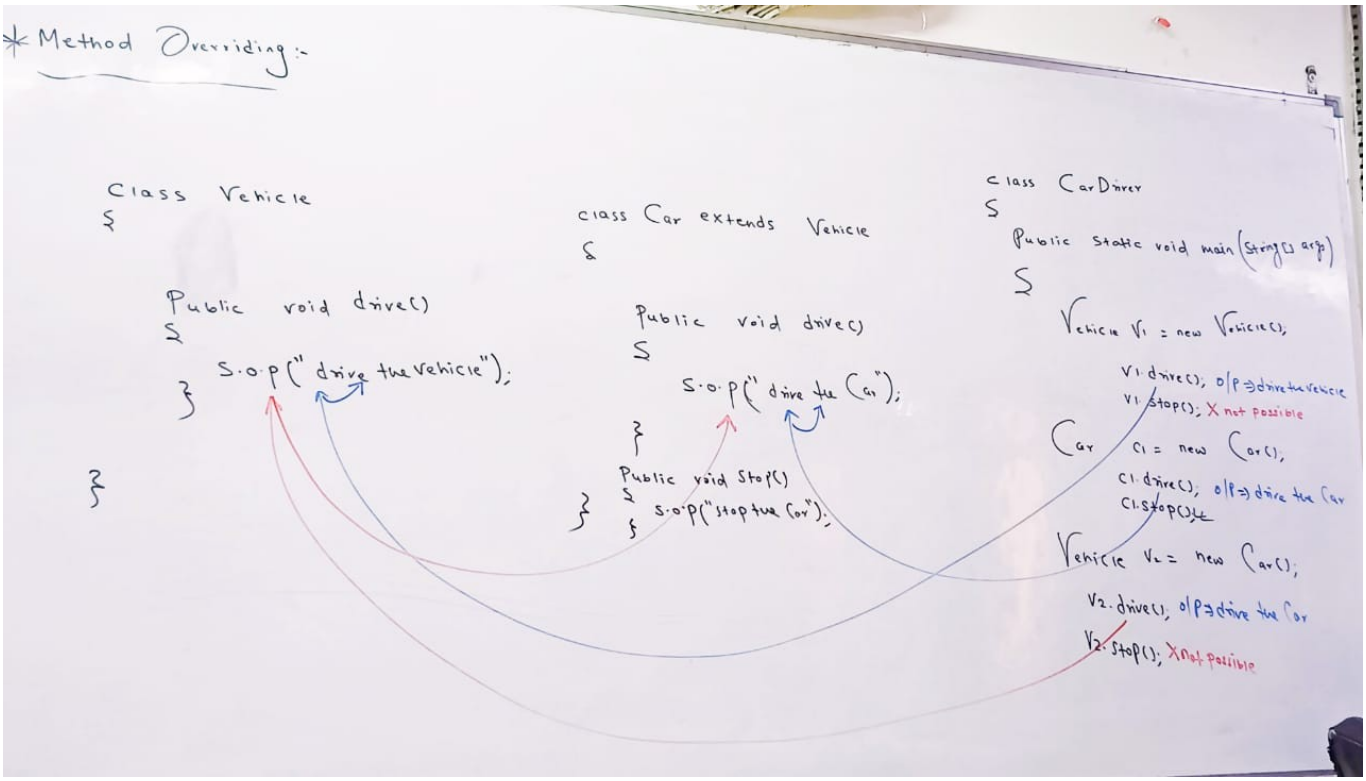

Animal
Dog
cat
Animal a1 = new Dog();
Dog d1 = (Dog) a1;
Animal a2 = new Cat();
Cat c1 = (cat) a2;
Vehicle V1 = new ElectricCar();
ElectricCar e1 = (ElectricCar) V1;
typecast operator
ElectricCar@...
ElectricCar e1
// methods of object
x = 10
y = 20
z = 30
Everything can be accessed
int x = 130;
byte b = (byte) x;
double p = 12.87;
int q = (int) p;
Object
Vehicle x=10
Car y=20
ElectricCar z=30
upcasting

01-12-2022

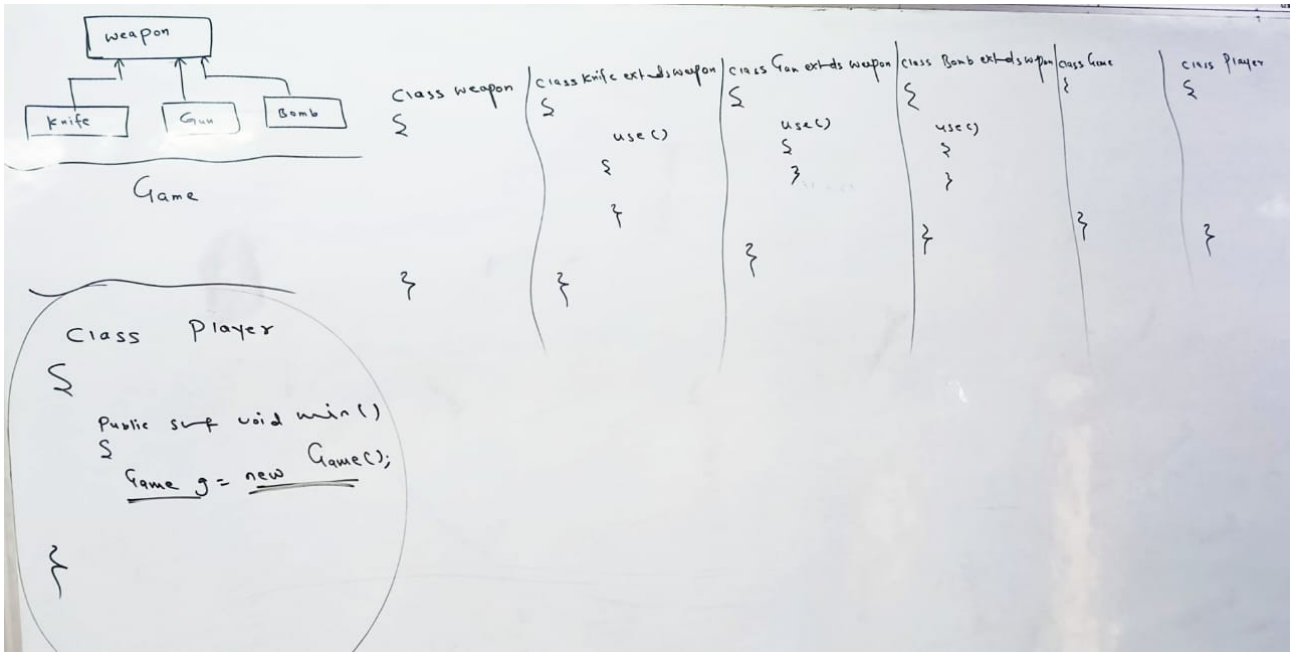
Over riding with upcasting is Very Important
Method OverRiding

When in parent class & child class there are non static methods with same parameter and same return type then it is called **Method Overriding.**
In **Method overriding** execution of method depends on the type of object created.
The output will change at the runtime as we give refernce of the parent class to child class objects.

Method OverRiding concept is used to change the parent implementation from child class.



Example Program
Weapon
_ Knife
_ Gun
_ Bomb



dsfgs

02-12-2022

***Points about return type from method:**

1. A method can return primitive type data or non primitive type data

```
primitive return
public static int test()// primitive return
{
	int x= 10;
	return x;
}

method call:-
	int x//10 = test();
	double y//10.0 = test();//UpCasting
__________or__________
public static double test()// primitive return
{
	int x= 10;
	return x;
}

method call:-
	double x//10.0= test();
	int y//10 = (int)test();//DownCasting

___________or_____________________________________________
Non primitive return
public static Employ demo()//Non primitive return
{
	Employ e1 = new Employ();
	return e1;
}

method call:-
	Employ e= demo();
	Object o = demo();//upcasting
```

```
__________or__________
public static Object demo()//Non primitive return
{
	Employ e1 = new Employ();
	return e1;
}

method call:-
	Object o= demo();
	Employ e =(Employ)demo();//DownCasting
```

paste photo +_+_+_+_+_+-

D.I.Y CREATER HIRARCHY LIKE GAME PROGRAM

Q1. Vending Machine

coffee
Juice
wine

shopkeeper**OBJECT**

will allow user to press button enter the amount

user

will click the amount

Q2. Browser

Chrome
Firefox
Edge

Computer **Object**

will allow user to choose the browser

user

will chose the task and browser

Return value and reference must be same.

Object e= test()

public static Object test()
{
return new employ();
}

Polymorphism

If a member behaves differently with the same name then it is called **Polymorphism.**
Same / One Variable = Different Value
Same / One Method = Different Implementation
Same / One Object = Different Behaviour
Meaning: poly = **many**, morphism = **forms**.

Example
giveGift()
- -parent/brother/sister
- -friend
- -gf/bf

Polymorphism is of two types

1. Compile Time Polymorphism / Static Polymorphism
2. Run Time Polymorphism / Dynamic Polymorphism

^Compile Time Polymorphism / Static Polymorphism^

If the implementation of the memeber is decided by compiler and same gets executed then is called **Compile Time Polymorphism.** What compiler Binds or sees = **same gets executed.**
We can achieve **Compile Time Polymorphism** by:

1. Method Overloading
2. Constructor Overloading
3. Method Shadowing
4. Variable Shadowing
5. Operator Overloading (Operator overloading is not supported in JAVA).

^Run Time Polymorphism / Static Polymorphism^

When the binding done by compiler at compile-time can be changed at run-time by JVM and implementation is provided by JVM then it is called **Run Time Polymorphism.** What compiler Binds or sees = **same may not get executed.**
Different ways to achieve **Run Time Polymorphism :**

1. Method OverRiding
2. UpCasting

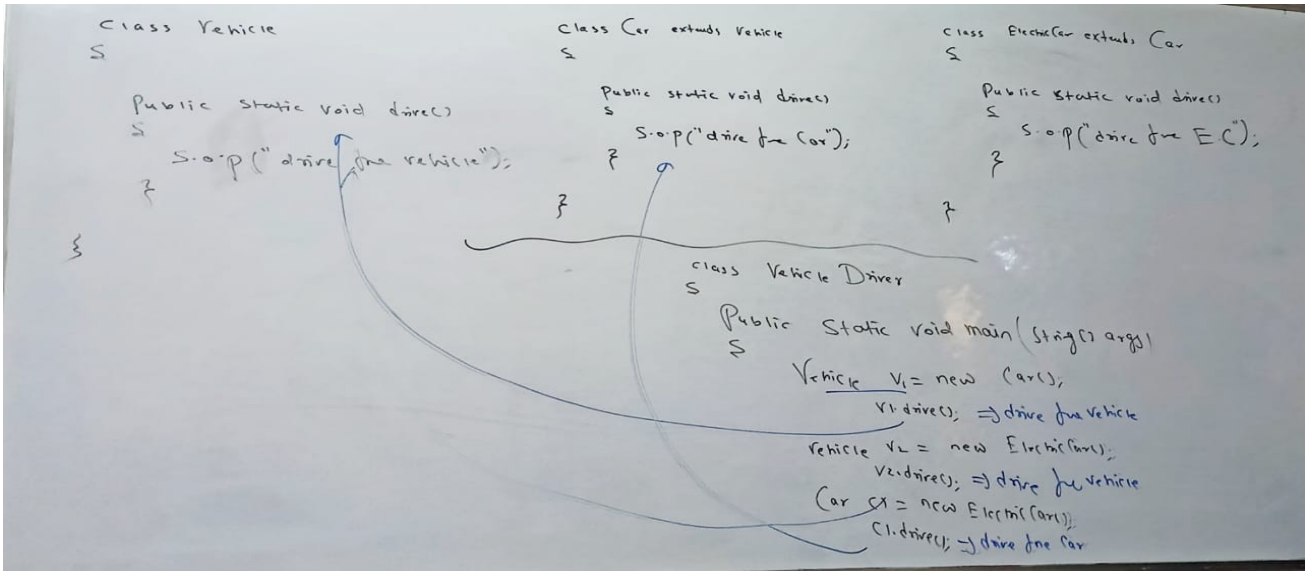
06/12/2022

Question1. Why JAVA does not support operator overloading?

Answer. In JAVA at the place of operator overloading there is a more powerful concept called **Method Overloading.** If Operator Overloading was allowed then it will only increase the complexity of the code, So Operator Overloading is not allowed in JAVA.

Method Shadowing

If in parent class and in child class there are static methods with the same name, same return type and same parameter then it is called **Method Shadowing.** In **method shadowing** execution of the method depends on the type of reference provided and it does not depend on the type of object created (**unlike method overriding).**



Variable Shadowing

If in parent class and in child class there are variables with the same name either static or non static then it is called **Variable Shadowing.** In **Variable Shadowing** implementation directly depends on type of reference provided and it does not depend on the type of object created.

Question. Can we Override main method in JAVA or not?
Answer. Method over riding concept is aplicable only for non static methods in parent class and in child class. As the main method of JAVA is a static method. So method overriding concept is not applicable for main method.

Note :
1 Compile time polymorphism : Exectuion depends on the reference provided and it does'nt depend on the type of object created.
2. Run time Polymorphism : Execution depends on the type of object created and it doesn't depend on the type of reference provided.

*Abstraction *

Abstraction in JAVA is a mechanism to hide the implementation details and showing only functionality. In other words abstraction is a mechanism to hide **how it does** from user and show **what it does** to user.

Abstraction in JAVA can be achieved in two ways:

1. By using Abstract class(0% to 100% abstraction).
2. By using interface(100% abstraction).

07-12-2022

Abstract Class
If a class in JAVA is delcared with the keyword **Abstract** then such class is called an Abstract Class
syntax:

```
abstract class vehicle
{
        //abstract class
}
```

Points :
1. If a class is abstract then we cannot create any object of that class.
Vehice v1 = ~~new Vehicle();~~
2. We cannot create object but we can use reference of an abstract class.
3. Inside an abstract class we can have abstract methods as well as concrete method.

3.1 *Abstract Method*

If a method in JAVA is declared with the keyword **abstract** then such method is called Abstract Method. An Abstract method is only allowed inside an abstract class. An abstract method do not spacify any method body. A static method cannot be abstract. Only a non static method will be abstract because abstaction is achieved through non static methods.

Example:

```
abstract class Vehicle
{

public abstract void start();
public abstract void drive();

}
```

3.2 *Concrete Method*

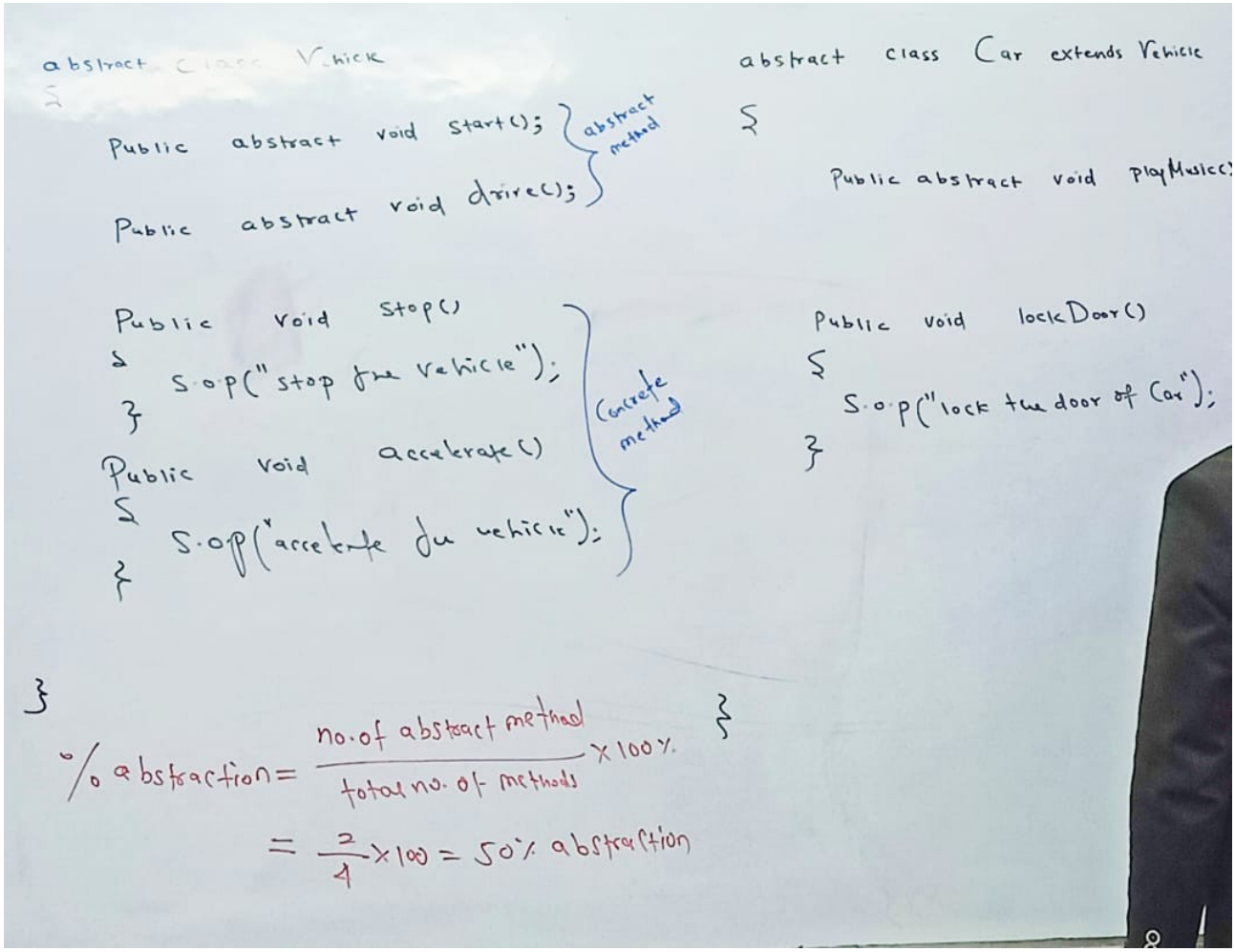
If a method in JAVA is not declared with the keyword abstract then it is called concrete method. A concrete method will always have the method body.
Example

```
public void stop();
```

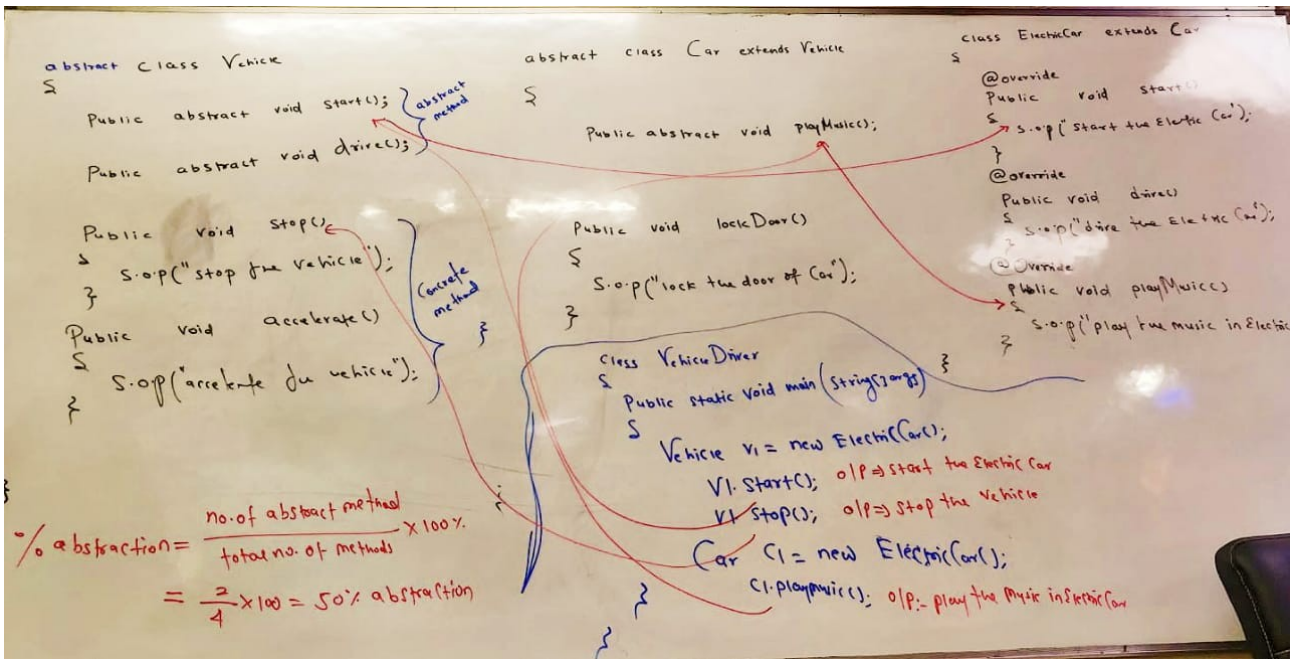
```
{
        sop(“stop the vehicle”);
}
public void accelerate()
{
        sop(“accelerate the vehicle”);
}
```

Percentage of Abstraction : Number of abstract method/total number of methods * 100%

4. A abtract class can extend another abstract class.



5. In the end there should be a child class of abstract class which will extend abstract class. To provide method body to all the abstract methods with the help of concept method over-riding. This class must be a normal class for which object will be also created.



Interface

Interface is a mechanism in JAVA to achieve complete abstraction of 100% abstraction.

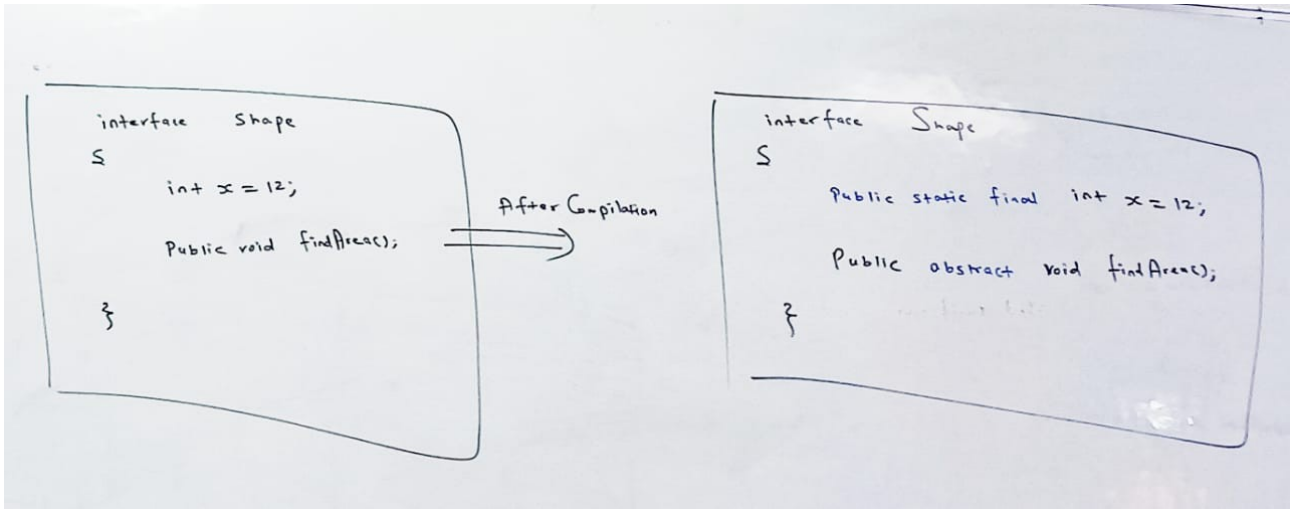
Ques. How to have interface?
Ans. We can have interface by using keyword **Interface.**

Syntax:

```
interface Fruit
{
        //interface
}
```

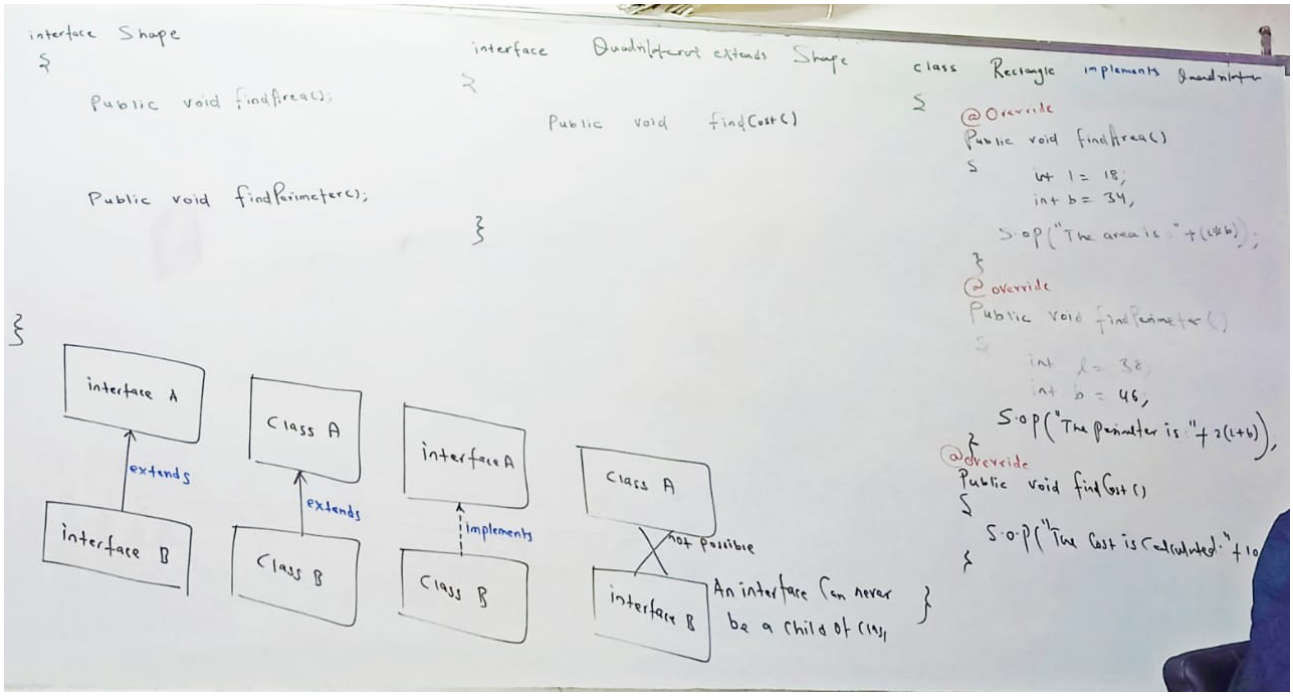
Interface is also a blueprint like class but it have different properties then class.

Points: -
1.Interface in JAVA is a blueprint same as class but it has different characteristics then a class.
2.A class is declared with the keyword class but an interface is declared with the keyword Interface.
3.We cannot create an object of an interface.
4.Inside an interface we cannot have constructor.
5.Inside an interface a variable is always by default public static and final type variable.
6.Inside interface a non static method is always by default abstract.

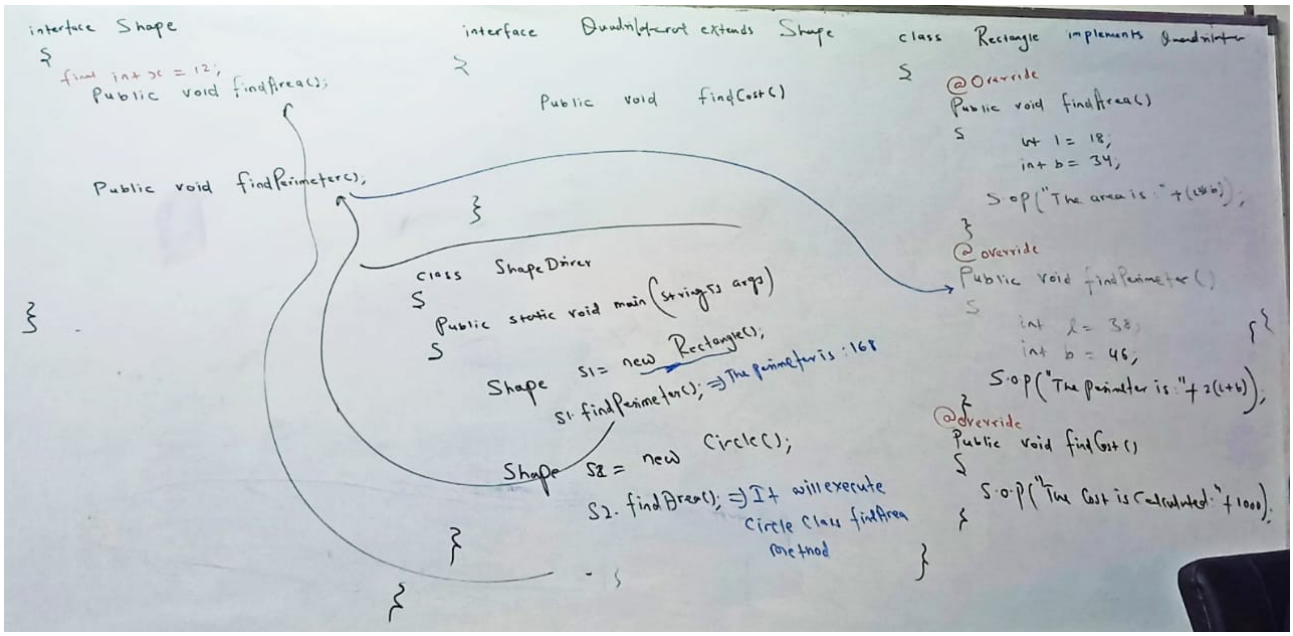


<u>08-12-2022</u>

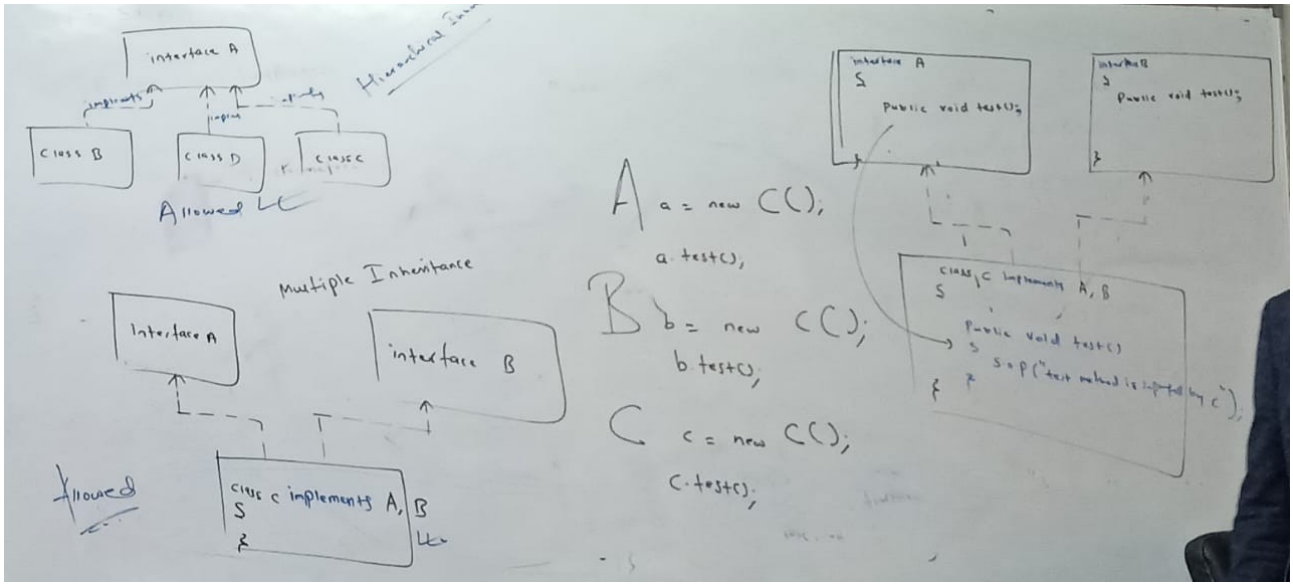
7. Since JAVA 8.0 an interface can have static and default method.
8. Since JAVA 9.0 and interface can have private methods. (non static methods cannot be private). Only a static method can be private. The non static method cannon be private because it has to be visible to other class so that it can be overridden.
9. An interface represents IS-A relationship where one interface can extend another interface.
10. An interface cannot be a child of a class.
11. In the end there should be a class which implements an interface to provide body to all the abstract methods of interface. A class always implements when interface.



12. One interface can be implemented by many child class.

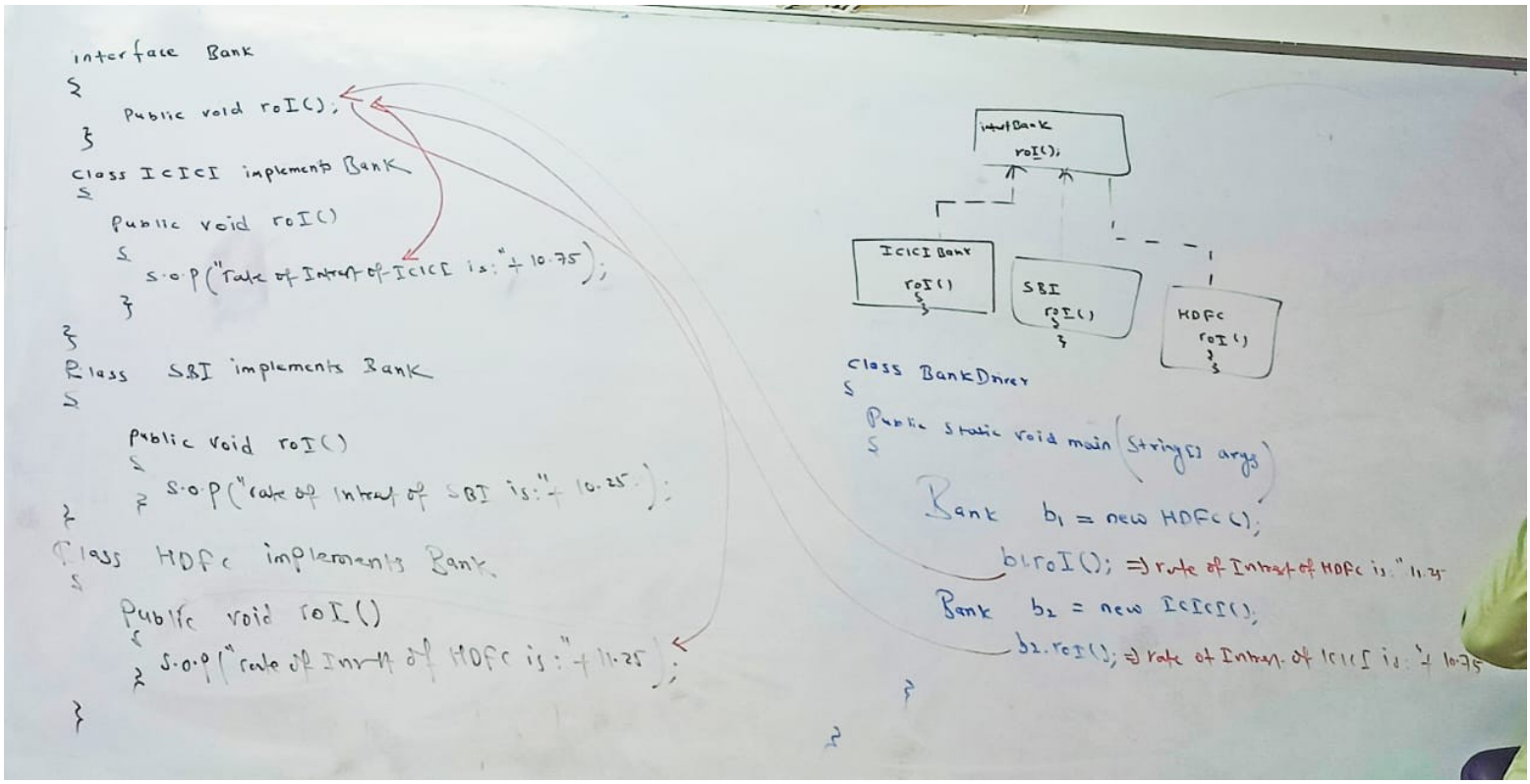


Question. How multiple inheritance is supported in JAVA in case of interface ?



Answer. In case of interface , interface can have only abstract type method. A single class can implement multiple interface so that the implementation provided by class will be common tto all the interfaces and in this manner ambiguity problem is resolved and interface allows multiple inheritance.

09-12-2022



***Some Special types of Interface:-**

1. Functional Interface: If an interface has only one abstract method then it is called functional interface. It can have many static methods but abstract method should be only one. **Example: Runable interface :** It is an example of functional interface which has only one abstract method called run.

```
Interface Runnable
{
public void run();
}
```

This interface is used to create threads.

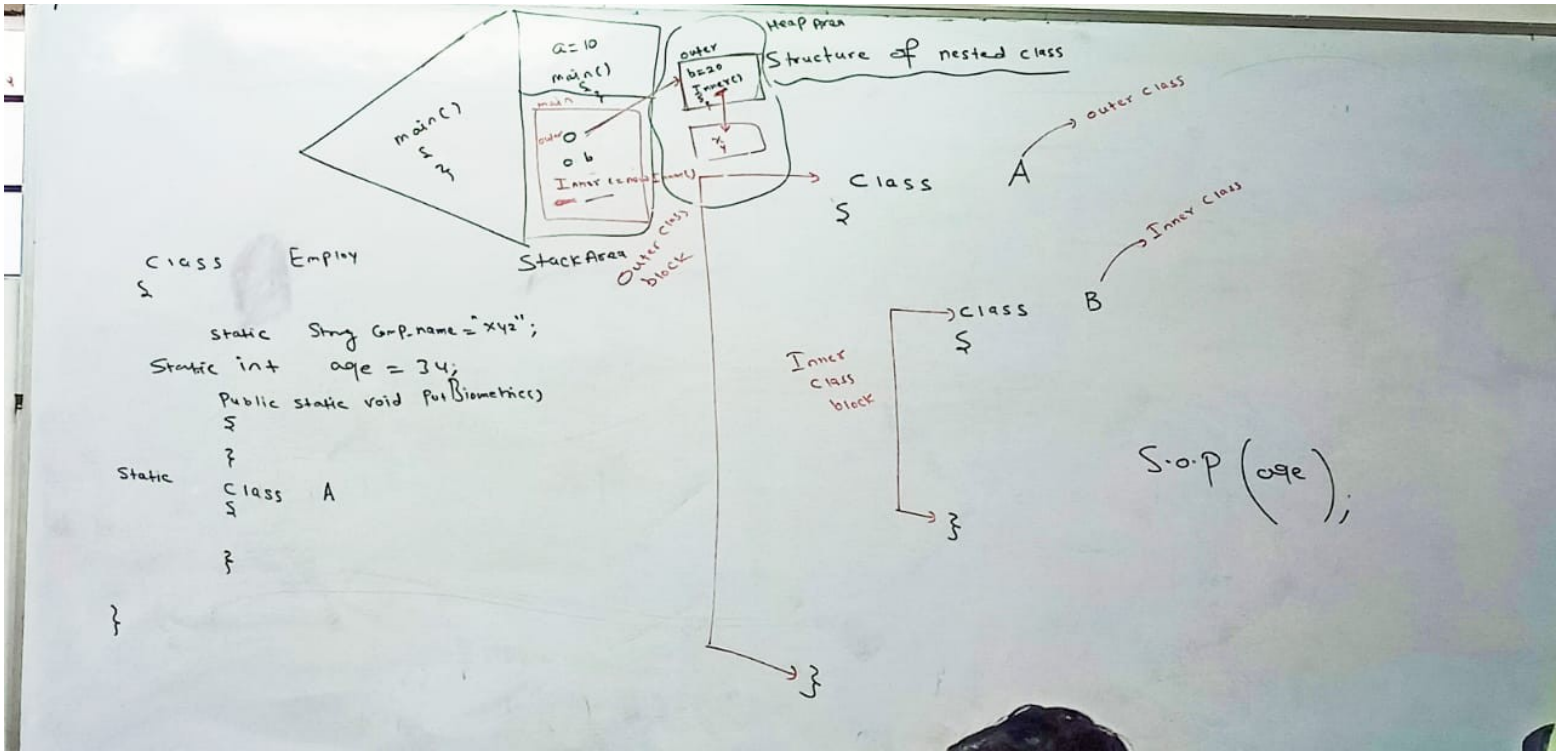
Questions. How many ways are there to create a thread in java?
Find answer online.

2. Marker/Tagged Interface : If an interface has no member then it is called Marker or Tagged Interface.
In other words an empty interface is called Marker or Tagged Inteface. **Example : Clone-able and Serializ-able Interface** are the example of Marker Interface it is used to pass some important pass some imortant message to JVM.

Static Class

If a class in JAVA is declared as static then such class is called Static Class. Only an Inner/Nested Class can be static.

Structure of Nested Class:




```
package static_class;

public class Outer {

static int a = 10;
int b=20;

static class Inner
{
static int x = 30;
int y = 40;
}
```

```
package static_class;

public class Driver {

public static void main(String[] args) {
Outer o = new Outer();
System.out.println(o.a);
System.out.println(o.b);

Outer.Inner i= new Outer.Inner();
System.out.println(i.x);
System.out.println(i.y);
System.out.println(Outer.Inner.x);
}
}
```

>>nested inner static class can be used in object on class reference
ex class1.class2 C1 = new class1.class2

>>A static class can only be an Inner class. To access the inner class members which is static class programmer have to give reference of outer class to access it.

10-12-2022

STRING

A string in JAVA is a group of character or sequence of characters.
Example
1. “Today is monday”
2. { ‘a’ , ‘n’, ‘a’, ‘n’,’d’}

A string in JAVA is a predefined class. As every class in JAVA is referred as non-primitive data. So a string is a non-primitive data type in JAVA.
Ex.
class Car//**this is a Non primitive data**
{

}
class Mobile//**this is a Non primitive data**
{

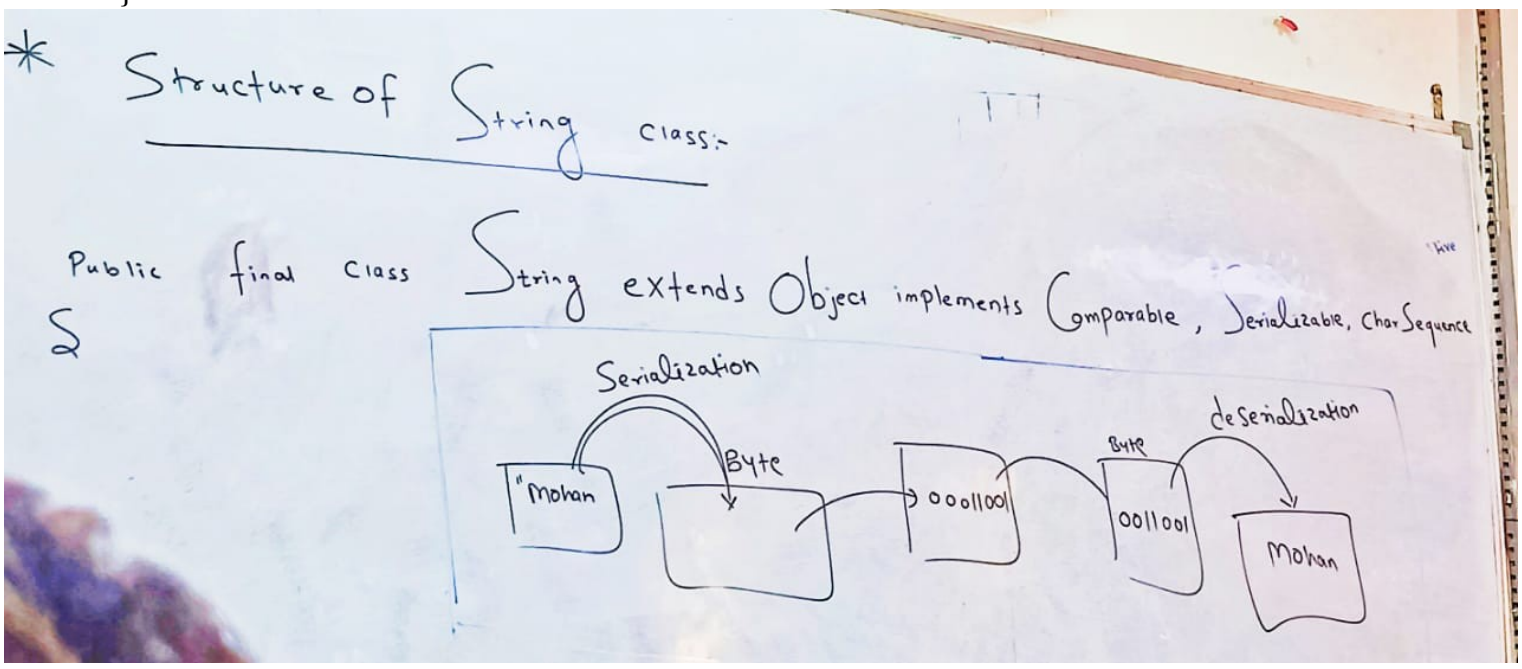
}
class String //**this is a Non primitive data**
{

}

Every non-primitive data type has default value null. So the default value of a string is also null.

^Structure of string Class.^
public final class String extends Object implements comparable, Serializable, Char Sequence
{

}



Home work : find out about Final, Comparable, serializable, Char sequence, JVM reference, charArray.
Q1. Why is String class final?
Q2. What is String Constant pool in java?
Q3. Why string is immutable in JAVA?
'==' operator compares the reference address of the object in non-primitive data.

Q1 Why String class in java is made final?

Answer. String is a very special type of data so reason behind making it final is that any other class should not extend it and so its behaviours will not be modified by overriding its methods.

Q2. How to create a string. ?
Answer A string in java can be created by using three classes:
1. String Class
2. StringBuilder Class
3. StringBuffer class.

1. String Creation by using String Class.

String in JAVA can be created by using a string class in three different ways.

1.1 By using String literal.
1.2 By using new Keyword
1.3 By using Character Sequence.

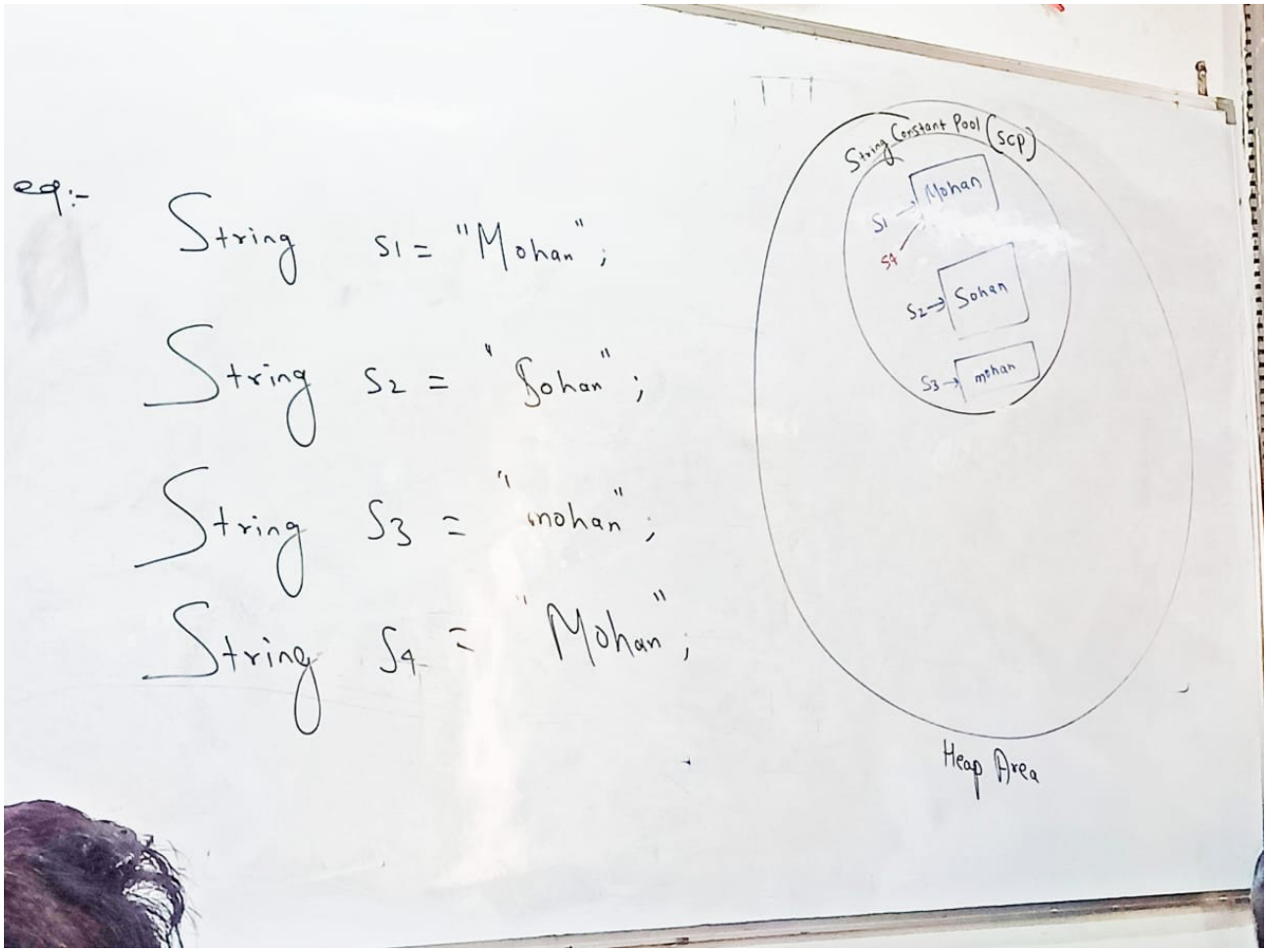
1.1 By using String literal.

When a String is created by using string literals then a string object is created by using that literals.

Example :
```
String s1 = "Mohan";
String s2 = "Sohan";
String s3 = "mohan";
String s4 = "Mohan";
```
Literals are differet values for ex: "mohan", "@", "18.5", "5".

When a string is created by using string literal then it gets memory inside an area which is called **String Constant pool(Scp).**

String Constant Pool: String constant pool is a dedicated area inside heap memory which is used for String object creation by using literals. When a string is created by using string literals then memory allocation takes place in String Constant pool.



In a string constant pool duplicate members will never get created. In this area multiple objects will not be created for the same string. If two string has same elements then it will refer to the same object. The object which is created by using string literals inside String Constant pool will having **refernce provided by programmer or if there is no reference then** JVM maintaines the reference by itself.

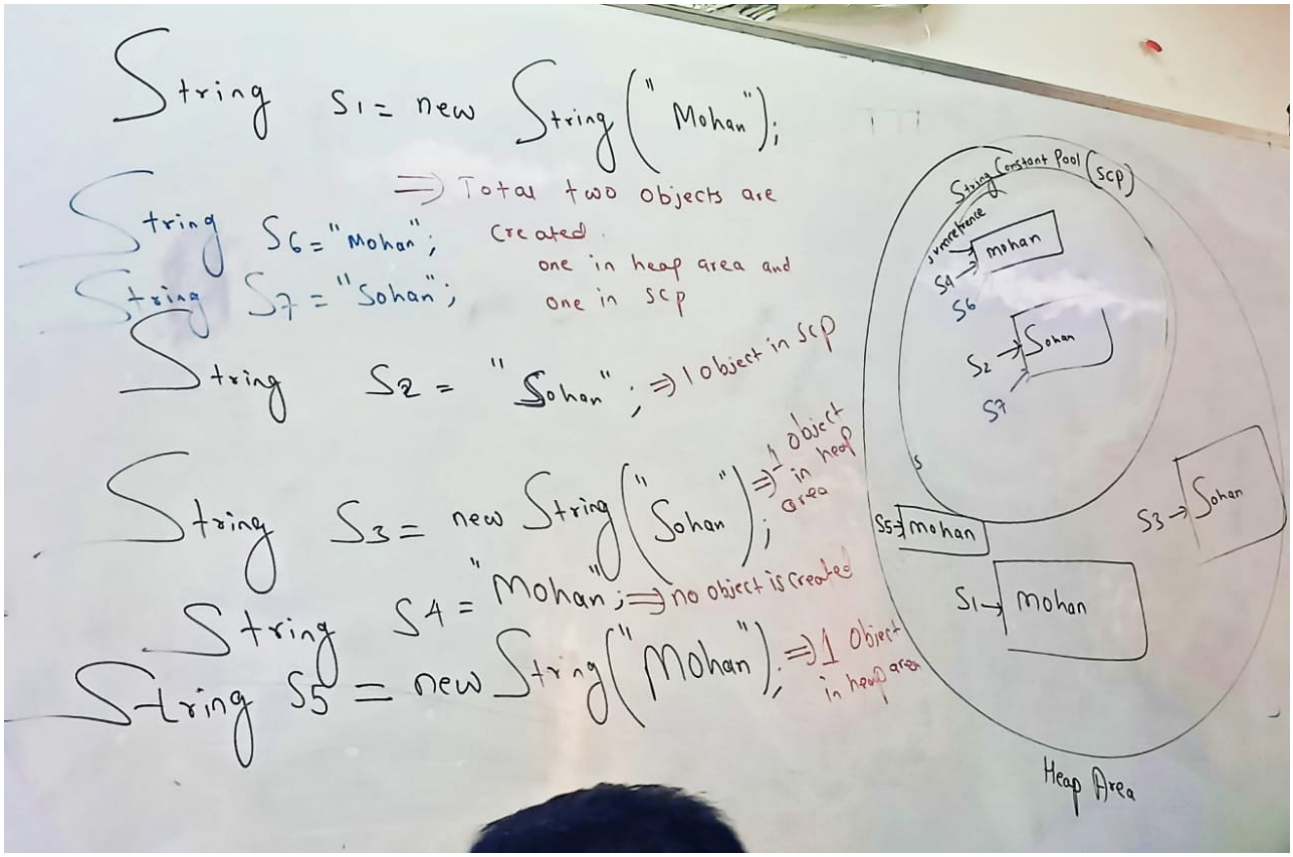
The Object which is created inside string constant pool will be not applicable for garbage collection. The object/content inside **SCP** will not be deleted only the reference will be null because it is the part of heap area which is expandable and it gets increased as needed by the program it will only be deleted after the program itself gets closed or ended.

SCP was part of method area which is in stack area untill JAVA 1.6 version. **SCP** become part of heap area since JAVA 1.7 version which is a expandable area.

'==' operator compares the reference address of the object in non-primitive data. To compare the content one must use equal method of object class.

1.2 By using new Keyword

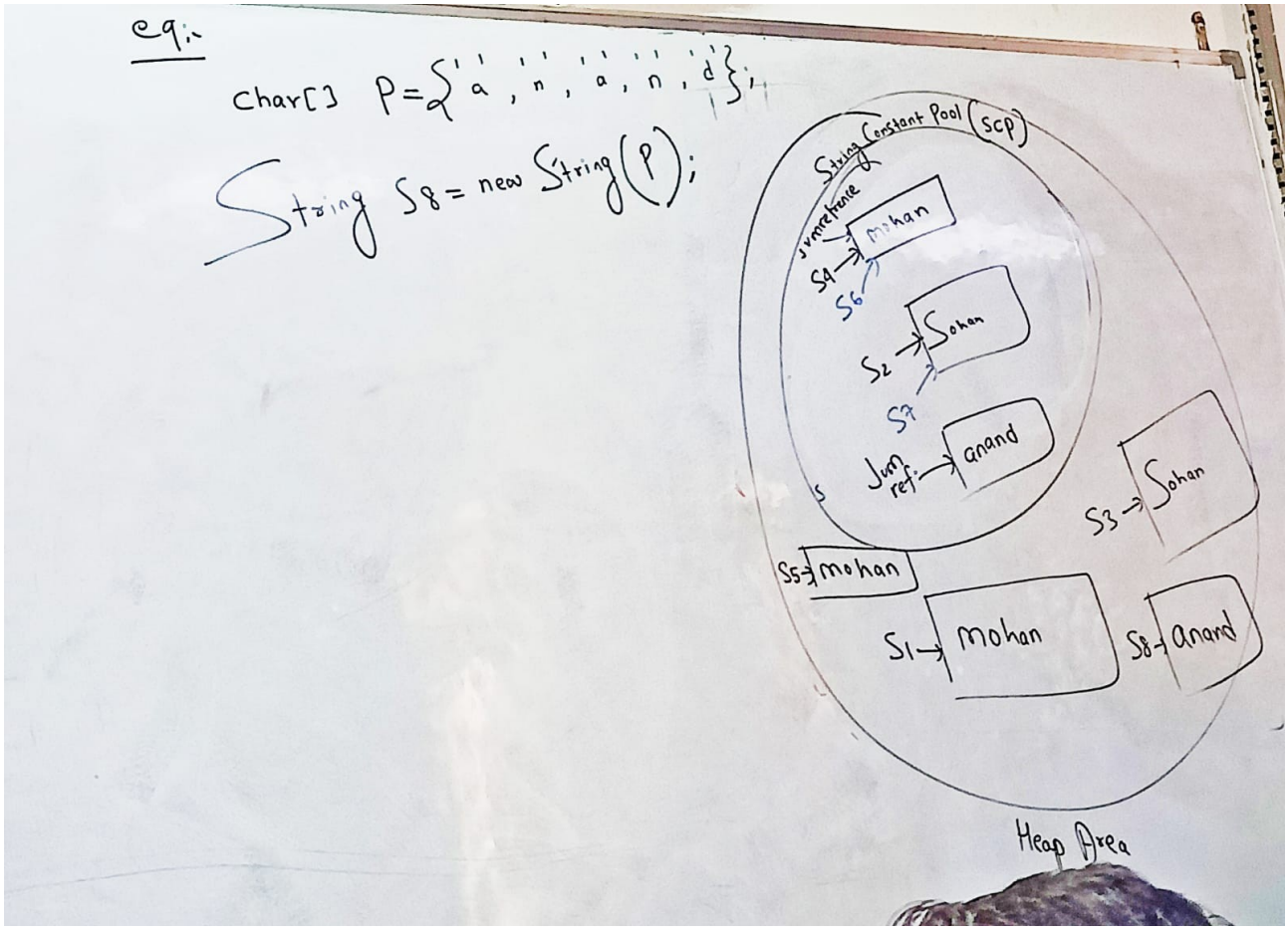
A string can be created by using new keyword. When a string is created by using new keyword then it gets memory in heap area. After allocating memory in heap area it checks whether the same string is available in **SCP** or not. If the same string is not available in **SCP** then it allocates memory for same string in **SCP** as well and it is reffered by JVM Internal Reference.
If in **SCP** same string is available in advance the memory allocation will not happen in **SCP.**



1.3 By using Character Sequence.

String can be created by the sequence of characters or by the array of characters, In this way **character array** is passed as string parameter. Object gets created in this way same as it gets created by using new keyword.

Ex.
```
char[] P = {'a','n','a','n'};
String s8 = new String (P);
```



'==' with String

'==' is a asignment operator which is used to compare two primitive type data or two non primitive type data.

In-case of primitive data '==' compares values of the primitive data.

In-case of non-primitive data '==' compares the address or reference of two non-primitive data. If address will be same then it returns true, otherwise it returns false in-case address is different. '==' cannot compare the internal content of the two non-primitive data.
To compare the internal content of two non-primitive data we have **equals method.**

Methods of String Class

1. lenght(): Length method is used to get the length of a provided string.

```
class Demo
{
	public static void main(String[] args) {
		String s1 = "MOhan is Happy";
		String s2 = "Monday";
		System.out.println("s1 Length : "+s1.length());
		System.out.println("s2 Length : "+s2.length());
	}
}
```

2. charAt(index) : this method is used to get the character from the string at provided index. Index value must be anything 1 less then the length of the string. So mohan = 0 to 4 index not 5 as it starts from 0 not 1.

M	O	H	A	N
0	1	2	3	4

```
class Demo
{

	public static void main(String[] args) {

		String s1 = "Mohan";
		System.out.println(s1.charAt(4));
	}
}
```

Example:

```
class copyCode
{
	public static void main(String[] args) {
		String s1 = "today is monday";
		System.out.println(s1.charAt(4));
		System.out.println("=========for loop ===========");
		for(int i = 0;i<s1.length();i++)
		{
			System.out.println(s1.charAt(i));
		}
		System.out.println("==========even place characeter");
		for(int i=0;i<s1.length();i++)
		{
			if(i%2==0)
			{
				System.out.println(s1.charAt(i));
			}
		}
	}
}
```

example 2

```
package string;
public class Program8 {
	public static void main(String[] args) {
	String s1 = "Mohan i5s H34pp4y";
	int count = 0;
	for(int i = 0;i<s1.length();i++)
	{
	 if(s1.charAt(i)>='0' && s1.charAt(i)<='9') 0 and 9 is used for asci value
		{
			System.out.println(s1.charAt(i));
			count++;
		}
	}
	System.out.println("total number character are :" +count );
	}
}
```

3. toCharArray();

By using this method we convert a string in the form of character array.
Example

```
package string;
public class Program10 {
            public static void main(String[] args) {

                  String s1 = "Mohan i5s H34pp4y";
                  int count = 0;
                  char[] p=s1.toCharArray();

                  for(char c:p)
                  {
                        if(c!=' ')
                        {
                        count++;
                        }
                  }
                  System.out.println("length of string is :"+count);
      }
}
```

Q1 **find the length of the string without using lenth method?**
BY FOR EACH LOOP

```
package string;
public class Program9 {

            public static void main(String[] args) {

                  String s1 = "Mohan i5s H34pp4y";
                  int count = 0;
                  char[] p=s1.toCharArray();

                  for(char c:p)
                  {
                        System.out.println(c);
                        count++;
                  }
                  System.out.println("length of string is :"+count);
      }
}
```

4. Trim()
Trim method is used to remove the space from starting and from the end position in the provided string.
Example 1

```
package string;
public class Program11 {

      public static void main(String[] args) {

            String s1 = "  Mohan i5s H34pp4y   ";
            System.out.println(s1);
            s1.trim();
            System.out.println(s1.trim());
            System.out.println("=================");
            String s2 = s1.trim();
```

```
            System.out.println(s1);//string never change cause its immutable
            System.out.println(s2);
      }
}
```

Q1 Why string in java is immutable?

Ans. In java immutable means one which cannot change. In Java a string is very special type of data and one string literal can be referred by multiple reference at a time. If one reference will change the string content then it will be changed for all which can create a big problem for all the other reference.

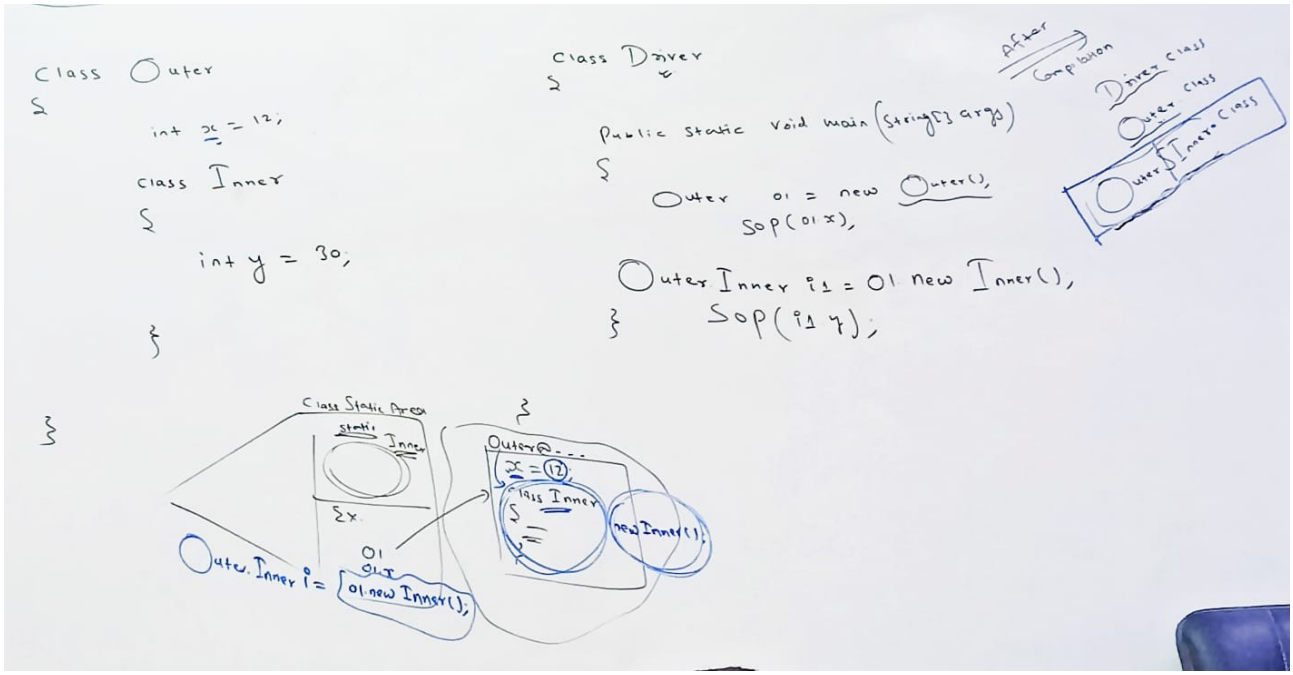
Reason behind making a string immutable is that string cannot change its orignal content once it is created. So that all the other reference will not be affected.

String immutablity says that once string is created then any changes is not possible on the created string. If we will try to change the created string then it will not change the original string but a new string object will be created with the changed details.

12/12/2022

***Non-Static Inner Class:-**

If an inner class is not declared with the keywords static then such class is called non-static inner class. Non static inner class gets loaded inside outer class object when the object of outer class is created.




```java
package nonstatic_class;

public class Outer {

int x= 12;
class Inner
{

int y = 30;
public void test()
{
System.out.println("test method : "+(x+y));
}
}

}
```

```java
package nonstatic_class;

public class Driver {

public static void main(String[] args) {
Outer O1 = new Outer();
Outer.Inner I1 = O1.new Inner();
System.out.println(O1.x);
System.out.println(I1.y);
I1.test();
}
}
```

Points :

1. When outer class gets compiled then always class file for inner class gets generated on the reference of as : O**uter$Inner.class**

2. Outer class can access only outer class member .

3. Inner Class can access outer class members as well as inner class members.

Final Keyword

Final is a keyword in JAVA which is used to restrict modification of a class member. Final can be:
1. A Variable
2. A Method
3. A Class

1. Final Variable : If a variable in java is declared with the final keyword then such variable is called a **Final Variable.** Once a final variable is assigned then we cannot re-asign it again.
Ex
Final int X = 35;
X = 60; //**Error can't reassign x.**

If a final variable is only declared then it is called **Blank Final Variable**. A blank final variable can be assigned only once. Blank final variable can be only local variable. Global varibale cannot be a blank final variable.
Ex.
Final int X;//Upper case convention for a final variable.
X = 60;//**can only be assigned once**

A final type varibale can be inherited from parent class to child class. Final type variable should be uppercase

12/12/2022

Final Method
If a method in Java is declared with keyword final then such method is called final method.

```
public final static void test()
{
        sop("this is a final method");
}
```

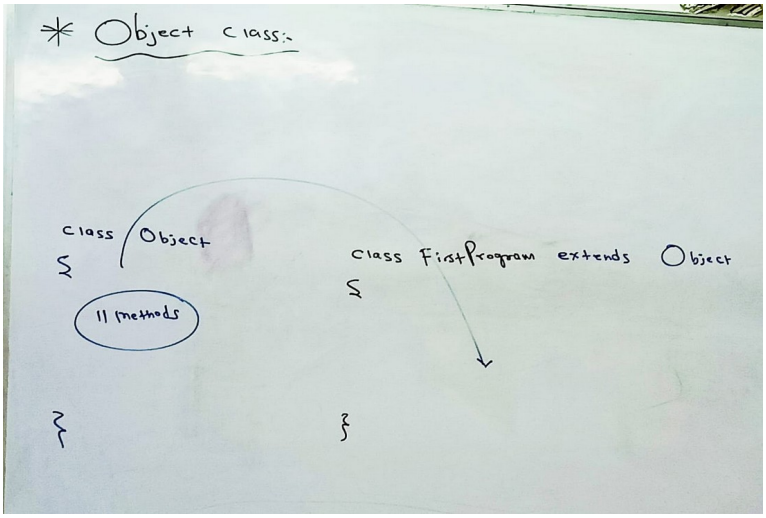
Properties
1. A final method can be overloaded in same class.
2. A final method can be inherited from parent class to child class.
3. A final method cannot be overridden from child class.

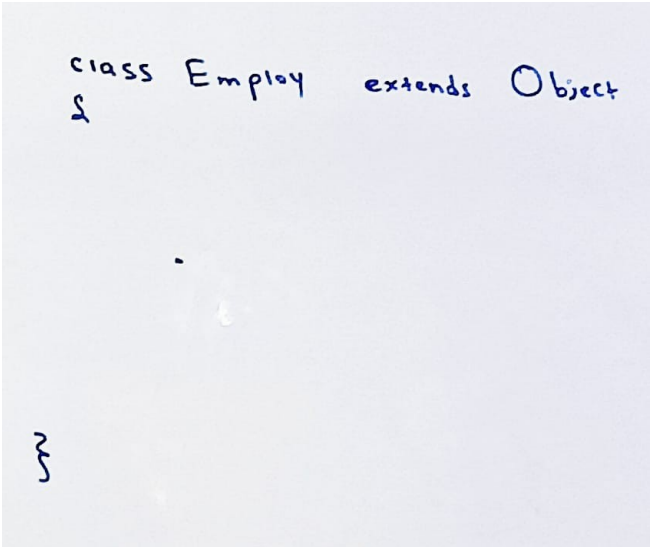
Final Class
If a class in java is declared with the keyword final then such class is called a final class.
If a class is final then it cannot have any child class and a final class can not be inherited from a child class.

```
Final class test
{

}
```

Object Class

Object class is the supermost class of every class in java. Whenever we design a class then by default that class extends to object class of JAVA. Object class has 11 methods which is by default available to every class of JAVA.





Object class is available in java.lang package

<Methods of Object Class>

1. getClass()
getClass() is a method available in object class of java. This keyword is used to get the details about a class. This method is a final type method so it cannot be over-ridden. getClass() method is a native type method which means it is written in JAVA native language (**C, C++**).

//paste code from drive from object class method employ.java and employ driver.java

It can be used while debugging to know the reference of a package which is causing an error.

2. toString()
toString method is available in object class of JAVA. toString is used to get an object address whenever an object is created. Whenever an object is created then object reference calls toString method to print the address of the object. The output of toString() can be changed by over-riding.

Public String toString()
{

}
//paste code from drive from object class method employ.java and employ driver.java

3.hashCode()

Whenever an object is created then it asigned with an unique integer number which is called hashcode.

Design of hashcode

```
public int hashCode()
{

}
```

```
package object_class;
import abstraction.*;
import constructor.*;
import game.*;
public class Employ {
String name;
int eid;
double salary;
Employ()
{
}
Employ(String name, int eid, double salary)
{
this.name=name;
this.eid=eid;
this.salary=salary;
}
public String toString()
{
return "Name is: "+name+" EID. is: "+eid+" Salary is: "+salary;
}
public int hashCode()
{
return eid;
}
}
```

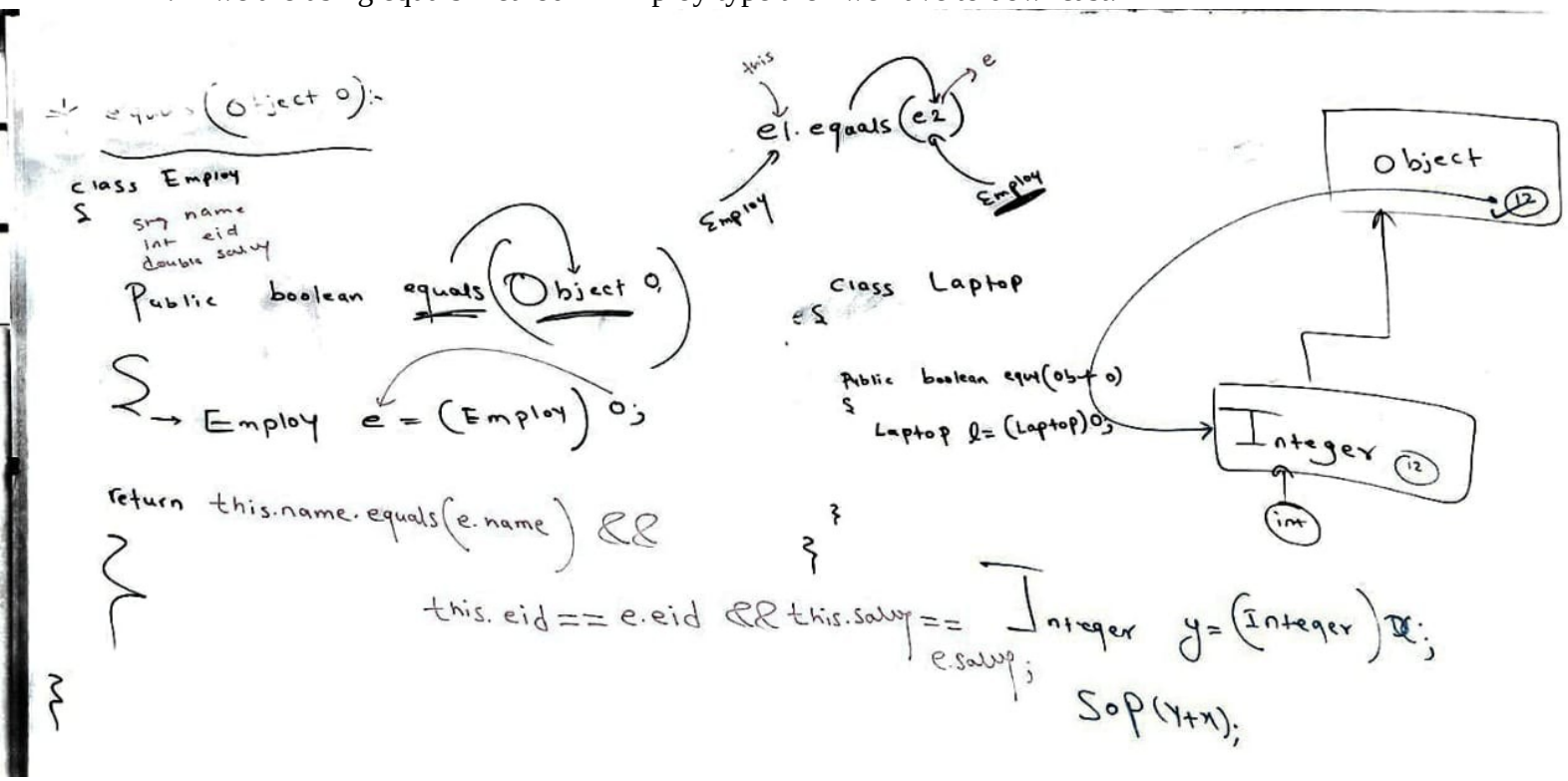
```
package object_class;
import abstraction.ElectricCar;
public class EmployDriver {
public static void main(String[] args) {
Employ e1=new Employ("Mohan", 201, 346677.45);
System.out.println("=====getClass()========");
System.out.println(e1);
System.out.println(e1.getClass());
System.out.println(e1.getClass().getName());
Employ e2=new Employ("Sohan", 205, 996677.45);
ElectricCar e12=new ElectricCar();
System.out.println(e12.x);

System.out.println(e2.getClass());
System.out.println("==========toString()==========");
System.out.println(e1);
System.out.println(e1.toString());
System.out.println(e2);
System.out.println("==========hashCode()==========");
System.out.println(e1.hashCode());
System.out.println(e2.hashCode());
}
}
```

equals(Objects o)

Equals is a method in object class of JAVA this method is used to compare the internal contents of two objects or both objects and if both objects have same content then it gives true otherwise it gives false.

Points:

1. If we are using equals method in Employ type then we have to downcast.



paste code////// from drive

make 5 programs using the same method

***finalize():**

Finalize is a method available in object class of JAVA.

finalize methods is used for the deletion of an object with null value used by garbage collector.

Whenever an object is referred null then garbage collector takes the responsibility to delete the object. Before deleting the object garbage collector calls finalize method to detach all the connectivity with the object. Once finalize method will detach all the conectivity of the object then garbage collector deletes the object.

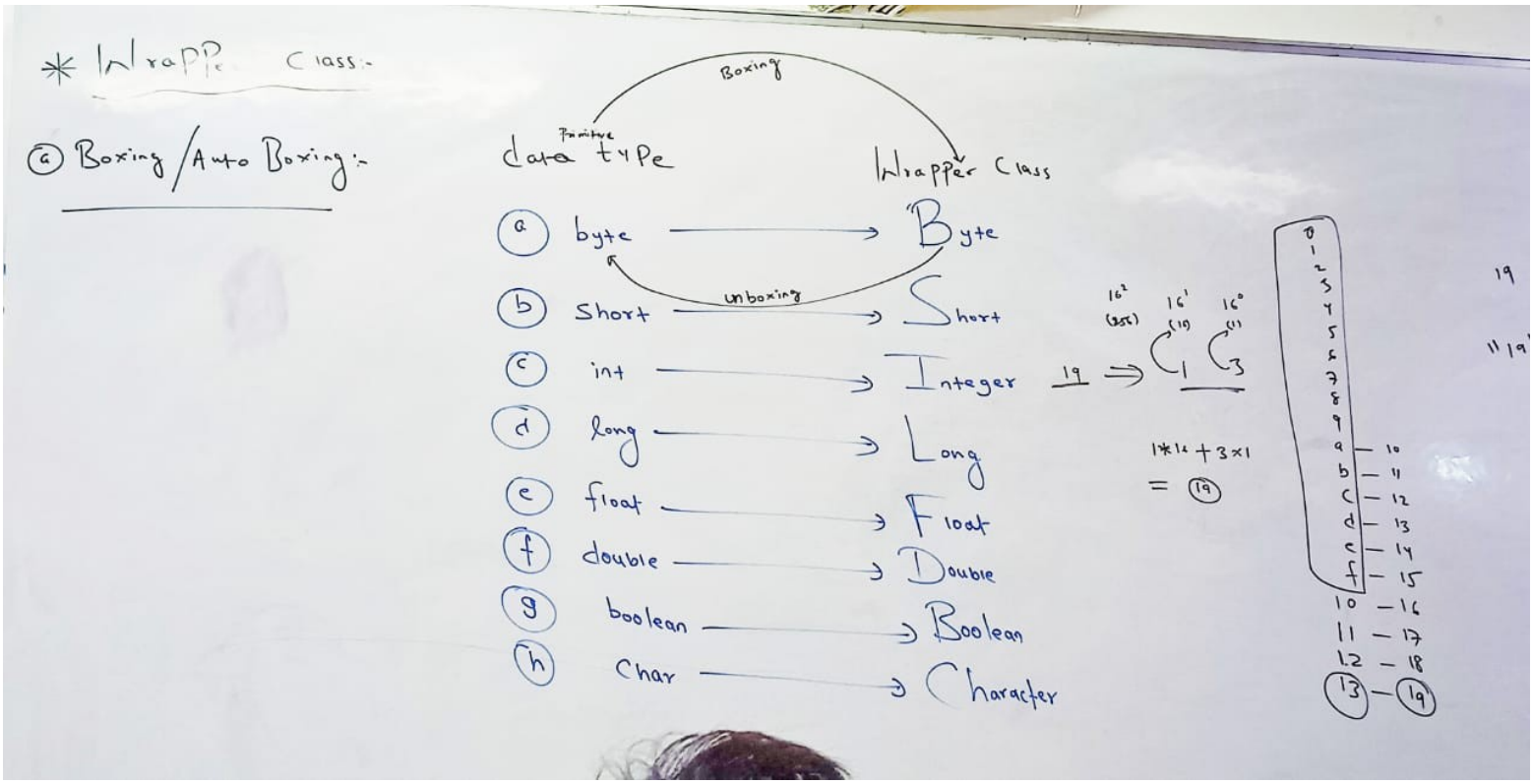
Finalize method can be called programatically as : System.gc();

16-12-2022

Wrapper Class

In JAVA every primitive data has a predefined class which is called wrapper class in JAVA.

Data type	Wrapper class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character



Wrapper class allows us to perform various tasks which data type cannot perform.

```
package wrapper_Class;
public class Program1 {
	public static void main(String[] args) {
		int a = 12;
		System.out.println(a+12*4);
		Integer b = 12;
		System.out.println(b.toHexString(b));
		System.out.println(b+13*4);
		String c = b.toString();// used toString() to convert it into string
		System.out.println(b+100);
		System.out.println(c+100);
	}
}
```

Wrapper class is quite slower compared to primitive data types like int etc as it uses more resources in comparison of data types.

Boxing/Auto Boxing
Boxing is a process to convert primitive type data into its wrapper class. It happens automatically so it is also called Auto Boxing.

```
Int a = 18;
Integer b = a;

Integer b = new Integer(a);//this happens automatically
```

Unboxing / Auto Unboxing
Unboxing is a process in JAVA to convert a wrapper class type data back into primitive type. It happens automatically so it is also called Auto Unboxing.

```
Integer x = 192;
int y = x;

Double p = 19.345;
Double q = p;
```

CONVERT ANY DATA INTO STRING TYPE BY THE HELP OF WRAPPER CLASS
We can convert any type of data into its string form by using toString() method of wrapper class.
int a = 123; **we can't convert it into string**
Integer b = 123; **we can convert it into string**
String x = b.toString();
int a = 123;
Integer b = a;
String x = b.toString();

```
package wrapper_Class;
public class Program4 {
	public static void main(String[] args) {
		Integer a=346;
		String b=a.toString();
		System.out.println(a+100);
		System.out.println(b+100);
		double p=346456.45757;
		Double q=p;
		String r=q.toString();
		System.out.println(r+100);
		System.out.println(r.length());
		int count=0;
		for(int i=0;i<r.length();i++)
		{
			if(r.charAt(i)=='.')
			{
				continue;
			}
			count++;
		}
		System.out.println("Lenght is: "+count);
	}
}
```

CONVERT ANY STRING TYPE DATA INTO ITS ORIGINAL FORM

We can convert a string data back into its original form by using method available in wrapper class.

Example
String p = “1238”
Integer q = Integer.parseInt(p);

String p = “12w875”; **can t convert into integer**

String x = “12.454”;
Double y = double.parseDouble(x);

17-12-2022

String class continued
5. isEmpty()
this method is used to check wether the string is empty or not.
If the length of the string is zero then it return true otherwise it return false.

```
package string;
public class Program12 {
	public static void main(String[] args) {
		String s1 = "";
		String s2 = " ";
		System.out.println(s1.length());
		System.out.println(s2.length());
		System.out.println(s1.isEmpty());
		System.out.println(s2.isEmpty());
	}
}
```

6. isBlank();
isBlank method return true if the length of the string is zero or it has only white space.

```
package string;
public class Program12 {
	public static void main(String[] args) {
		String s1 = "";
		String s2 = " ";
		System.out.println(s1.isBlank());
		System.out.println(s2.isBlank());
	}
}
```

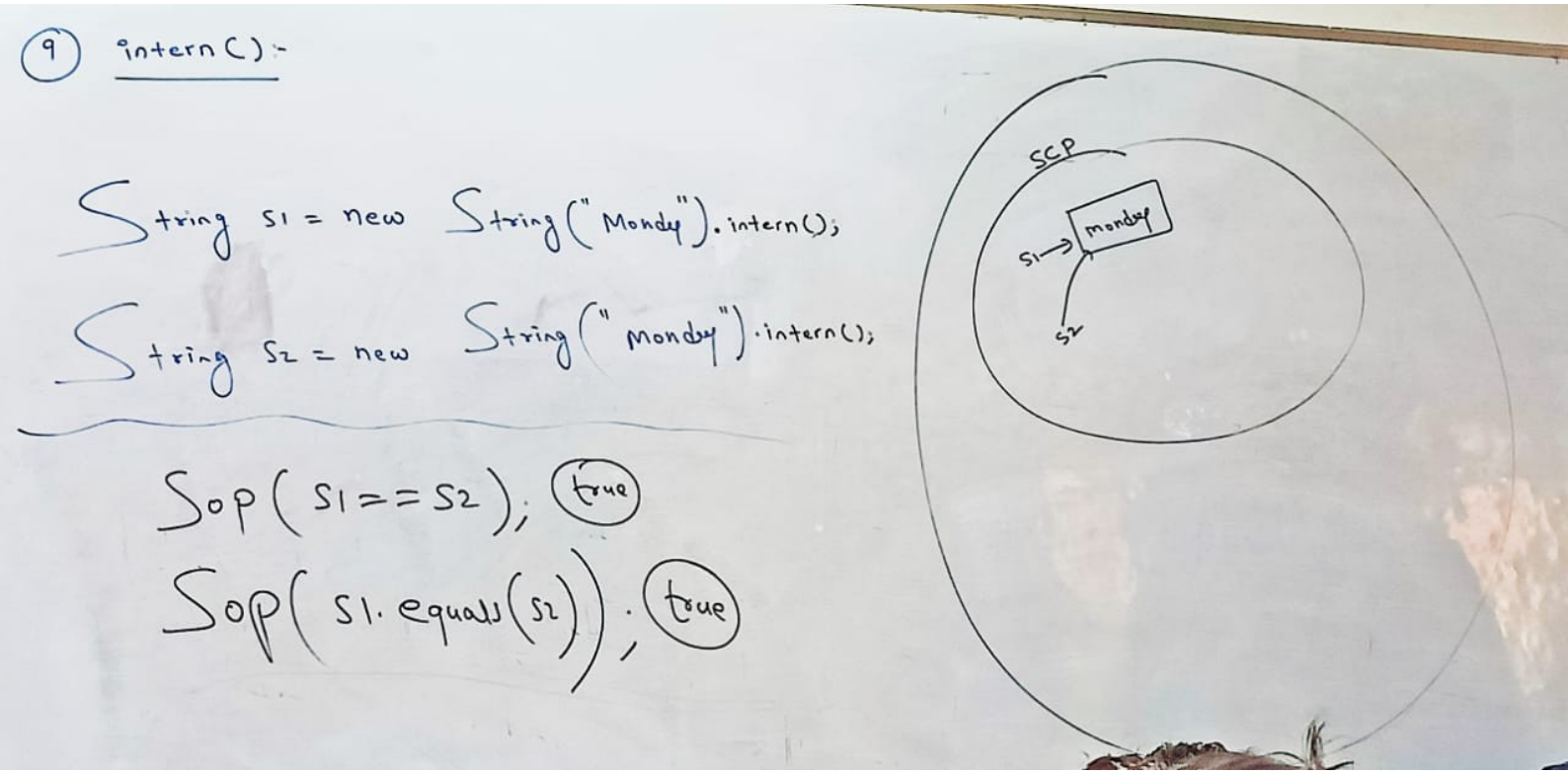
7. **equals(String)**
equals method is used to compare the content of two string and if the content is equal then it returns true otherwise it returns false

```
package string;
public class Program13 {
	public static void main(String[] args) {
		String s1 = new String ("Monday");
		String s2 = new String ("monday");
		String s3 = new String ("Monday");
		System.out.println(s1.equals(s2));
		System.out.println(s1.equals(s3));
	}
}
```

8. equalIgnoreCase(String):
This method is used to compare the content of two string without considering the case of the characters. If the content is same then it return true otherwise it return false.

```
package string;
public class Program13 {
	public static void main(String[] args) {
		String s1 = new String ("Monday");
		String s2 = new String ("monday");
		String s3 = new String ("Monday");
		System.out.println(s1.equalsIgnoreCase(s3));
		System.out.println(s1.equalsIgnoreCase(s2));
	}
}
```

9. intern():
this method is used to create only one object if the content is same. Intern method creates object in **SCP** so that multiple reference can refer only one object with the same content.




```
package string;
public class program13 {
	public static void main(String[] args) {
		String s1 = new String("Monday").intern();
		String s2 = new String("Monday").intern();
		System.out.println(s1.equals(s2));
		System.out.println(s1==s2);
	}
}
```

by using intern method we can send a heap area object into **SCP.**

```
package string;
public class Program15 {
	public static void main(String[] args) {
		String s1 = new String("Monday");
```

```
            s1.intern();
            String s2 = new String("Monday");
            s2.intern();
            System.out.println(s1.equals(s2));//output true
            System.out.println(s1==s2);//output false
      }
}
```

10.*Concat():*

Contcat is a method in a string class which is used to attach one string with another string.

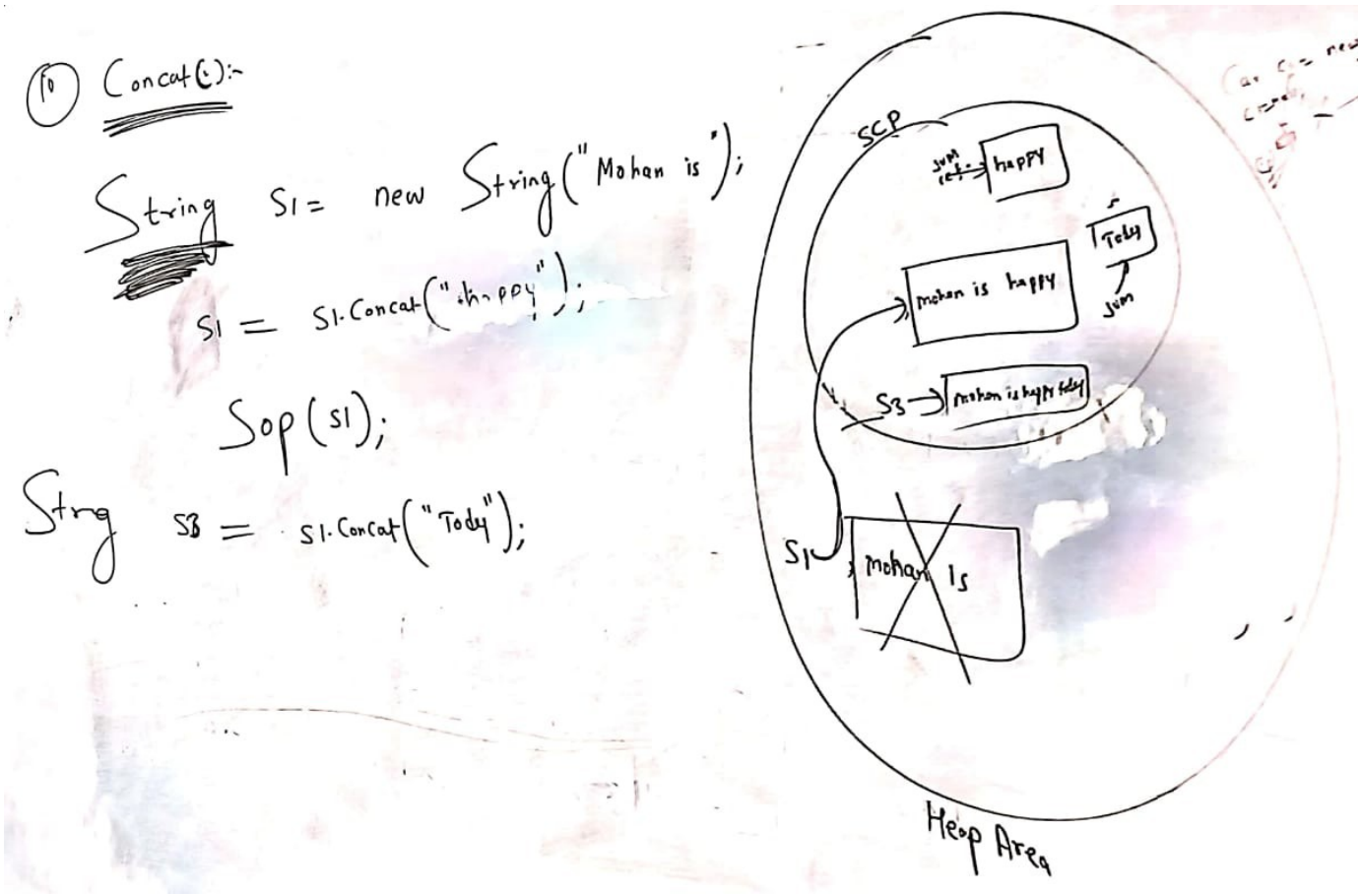
Whenever new keyword is used the object will be created in heap area.

String s1 = new String(“Mohan is”);
s1 = s1.Concat(“happy”);
s1 = s1.Concat(“Today”);
sop(s1);//output Mohan is happy

SCP
Jvm ref |happy|
Jvm ref |Mohan is|
Jvm ref |Today|
s1|Mohan is Happy|

Heap Area
s1 |~~Mohan~~

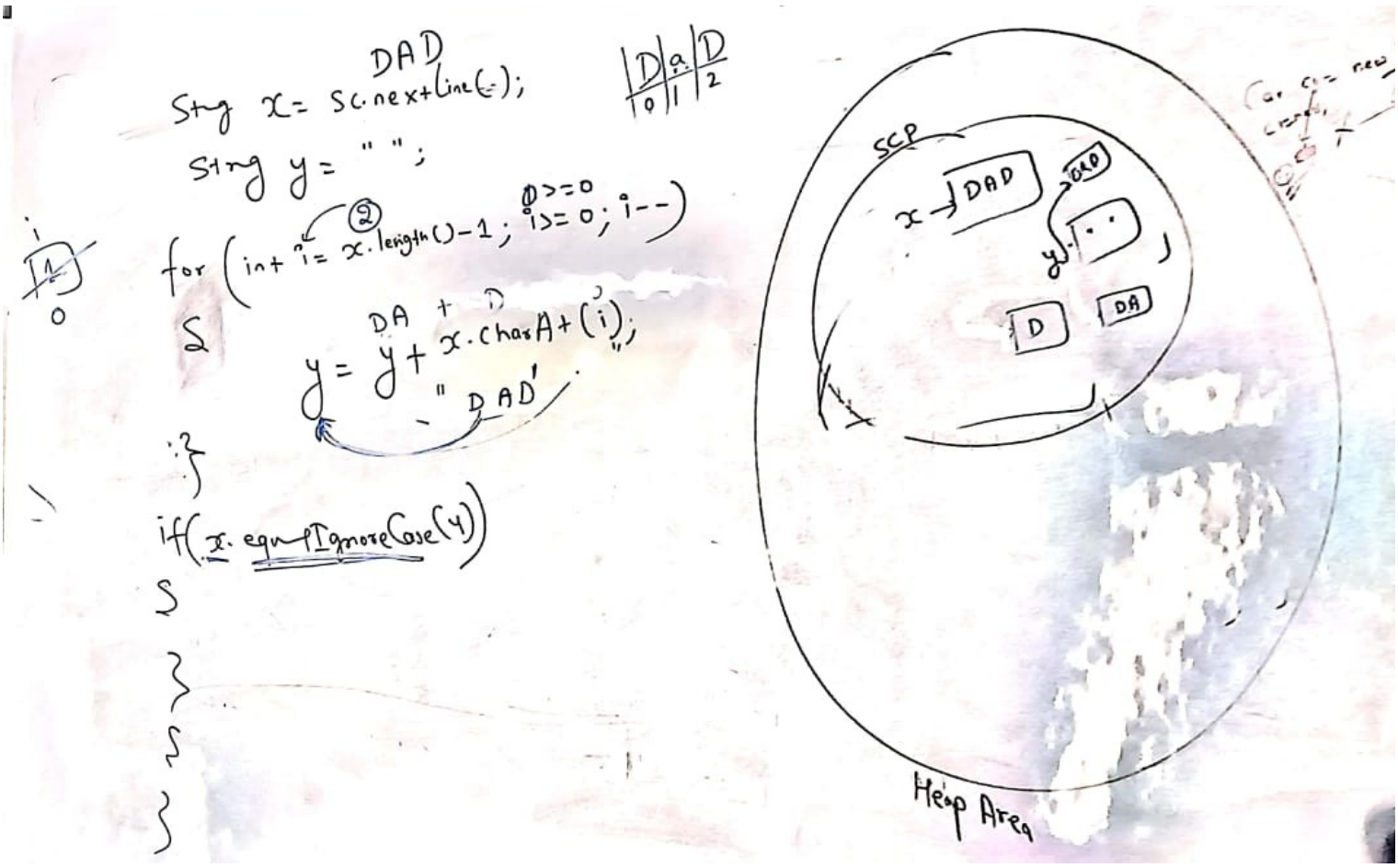
```
package string;
public class Program16 {
      public static void main(String[] args) {
            String s1 = new String ("Mohan is");
            s1=s1.concat("Happy");
            System.out.println(s1);
            String s2 =s1.concat("Today");
            System.out.println(s1);
            System.out.println(s2);
      }
}
o/p
Mohan isHappy
Mohan isHappy
Mohan isHappyToday
```



Question1. WAJP to find out palindrome string?

```
package string;
import java.util.*;
public class PalindromeString {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the String");
		String x = sc.nextLine();
		String y = "";
		for(int i = x.length()-1;i>=0;i--)
		{
			y = y+x.charAt(i);
		}
		if(x.equalsIgnoreCase(y))
		{
			System.out.println(x+" is a palindrome String");
		}
		else
		{
			System.out.println(x+" is not a palindrome String");
		}
	}
}
```

/////paste photo to see the logic of above written questoin



revise immutabilty
revise synchronized and non synchronized

11. toUpperCase()

By using this method we can make all the characters of a string in uppercase.

```
package string;
public class Program17 {
	public static void main(String[] args) {
		String s1 = "Today si a good day";
		s1 = s1.toUpperCase();
		System.out.println(s1);
	}
}
```

12. toLowerCase()

By using this method we can convert all the characters of a string into lowercase.

```
package string;
public class Program17 {
	public static void main(String[] args) {
		String s2 = "Today is bad day";
		s2= s2.toLowerCase();
		System.out.println(s2);
	}
}
```

13. compareTo()

By this method we can compare two string lexicoGraphically. It tells the difference in ascii value of given input. Compareto method compares character of one string with character of another string and when there will be difference between character this method will return the difference of ascii value. If there is no difference then it returns the difference of length in characters present in the string.

Example 1.

```
package string;
public class Program18 {
	public static void main(String[] args) {
		String x= "ball";
		String y = "Dog";
		System.out.println(x.compareTo(y));//output 30 because of the
difference of the ascii value.
	}
}
```

Example 2.

```
package string;
public class Program19 {
	public static void main(String[] args) {
		String s1 = "Monday";
		String s2 = "Mon";
		String s3 = "";
		System.out.println(s1.compareTo(s2));// output is 3 because of the
difference of the length
		System.out.println(s1.compareTo(s3));// output is 6 because of the
difference of the length
	}
}
```

14. compareToIgnoreCase(String)

this method will ignore alphabets case sensitivity. Uppercase alphabets and lowercase alphabets will be considered same after comparison this method returns difference of ascii value. If they are same but different in length then this method will return difference of the length.

```
package string;
public class Program20 {
	public static void main(String[] args) {
		String s1 = "Mohan";
		String s2 = "mohan";
		System.out.println(s1.compareTo(s2));
		System.out.println(s1.compareToIgnoreCase(s2));
	}
}
```

15. indexOf(Char)

This method takes a character and it returns the index value of the character which appear first.

String s1 = " today is a good day"
s1.indexOf('a'); //output is 3

Example

```
package string;
public class Program21 {
	public static void main(String[] args) {

		String s2 = "Today is a good day";
		System.out.println(s2.indexOf('s'));
	}
}
```

16. indexOf(char, index);

By this method we can provided initial index value so that when the character is found after that index then it returns the found index value. Otherwise it returns -1 if the desired charector is not found after given index.

```
package string;
public class Program21 {
	public static void main(String[] args) {

		String s2 = "Today is a good day";
		System.out.println(s2.indexOf('a', 8));
	}
}
```

17. indexOf(String)

By using this method we can find the index of a provided string. If it is not found then this method returns -1.

```
package string;
public class Program21 {
	public static void main(String[] args) {

		String s2 = "Today is a good day";
		System.out.println(s2.indexOf("ay");//output 3
	}
}
```

18. indexOf(String, index)

This method takes initial index value and after this index provided string is found then it prints the index value otherwise it returns -1.

```
package string;
public class Program21 {
	public static void main(String[] args) {

		String s2 = "Today is a good day";
		System.out.println("===========");
		System.out.println(s2.indexOf("ay", 8));
	}
}
```

19. valueOf(data_type):-

Value of method is used to convert any type of data in the form of string. It is a static type method so it can be called on Class name reference.

Exampe:

int a = 12;
String s1 = String.valueOf(a);

double b = 18.5;
String s2 = String.valuueOf(b);

char c= ‘@’;
String s2 = String.valuueOf(c);

```
package string;

public class Program26 {

	public static void main(String[] args) {
		double a = 34.67;
		String b = String.valueOf(a);
		System.out.println(a+100);// output 134.67000000000002
		System.out.println(b+100);// output 34.67100
	}

}
```

String Buffer Class

String buffer is a pre-defined class in java which is used to create mutable string. String Buffer is same like string class to create an string object but it has many differences compared to string class. By using string class a string object is created with fixed-in length, immutable type of object which cannot be modified so that it leads to many unneccessary objects creation in **SCP.**
To overcome this drawback of string JAVA provides **StringBuffer Class** to create String object.
Class Declaration
final class StringBuffer extends Object implements Serializable, CharSequence
{

}

It is available in JAVA.LANG packages.

StringBuffer sp = new StringBuffer();

19-12-2022

EXCEPTION HANDLING

***Exception:** Exception is and unexpected situation which occurs at runtime during execution of a program and because of exception program gets terminated abruptly and normal flow of execution will not be maintained.

***Exception handling:** Exception handling is a mechanism in JAVA by which a program maintains its normal flow even in case exception occures in the program.

```
package exception_handling;
public class Employ {
	String name;
	public static void main(String[] args) {
		Employ e1 = new Employ();
		e1.name = "mohan";
		System.out.println(e1.name);
		e1= null;
		System.out.println(e1.name);
	}
}
```

```
mohan
Exception in thread "main" java.lang.NullPointerException: Cannot read field
"name" because "e1" is null
	at exception_handling.Employ.main(Employ.java:11)
```

Type of Exception

In java exception is of two types:
1. Pre-Defined Exception
2. User Defined Exception

1. Pre-Defined Exception in JAVA: In JAVA the exception which is inbuilt type exception which comes inbuilt together with JDK is called predefined exception in JAVA. Exception is avalable in *JAVA.LANG* package.
In predefined exception the top exception class is **Throwable** which is excetended by a child class called exception.

Throwable
^
extends
^
Exception

Predefined exception in JAVA is three types:

1. CheckedException
2. UnCheckedException
3. Error

1. CheckedException:

If an exception is checked by compiler at compile time and in case exception is not handled at compile time then compiler gives an error and such exception is called Checked Exception in JAVA. The Checked Exception must be handeled at compile time otherwise there will be compile time error.

2. UnChecked Excpetion:

Unchecked Excpetion is a type of exception which is not checked by compiler at compile time and it is check directly by JVM at runtime.

Runtime exception is not manndatory to be handeled at compile time.

3. Error

When a type of exception cannot be handled programatically. The reason of exception can be virtual machine or memory so such exception is called error.

20-12-2022

Hierachy of Exception

Java.Lang
^
Throwable
^
Exception

Checked Exception	Unchecked Exception	Error
1. FileNotFoundException	1. ArithmeticException	1.OutofMemoryError
2. ClassNotFoundException	2. ArrayIndexOutOfBox	2.StackOverFlowError
3. IOException	3. StringIndexoutofBOndsException	3.VirtualMachineError
4. NoSuchMethodException	4. ClassCastException	4.IOError
5. SQLException	5. NullPointerException	Etc
Etc.	etc.	

How an Exception can be handled in JAVA

Exception can be handled in JAVA by using :
1. Try Catch Block
2. Throws keyword

1. Try Catch Block : try catch in Java is used to handle an exception where three keywords are used : **try , catch and finally.**

1.1 Try Block : Try block in java is used to write a statement which may throw an exception. When a statement causes exception from try block then object of such exception is thrown from try block by using throw keyword internally . When an exception object is thrown from try block then it has to be caught by catch block which is written just after try block. One try block can only throw one exception at a time. When an excpetion occurs in try block then any other statement from try block will not be executed. All the statement of try block will only execute when there is no exception from try block. So it is recommended to write only the statements which may cause exception to be written inside try block.

Syntax:
try
{
//statemen1dd
//statemen2
//statemen3
}

We can have multiple try catch block inside a program. Try block must be followed by catch block there should not be anything between try and catch block.

1.2 Catch Block

Catch is a block in java which is used to catch the exception object thrown by try block catch block is written in with parameter where we have to declare the exception. One try block can be followed by multiple catch block.

Syntax:
try

{
	sop(10/0);//**this thorws a new Arithmetic Exception();**
}
Catch(ArithmeticException e)
{

}
Catch(NullPointerException e)
{

}
catch(Exception e)
{

}

At catch block exception sequence should be from child to parent type. If the sequence will be from parent to child then code will become unreachable and compiler will give error.

21-12-2022

***Finally Keyword:**

Finally is a keyword which is used with try catch block. Finally is a block which is written in the end of try catch block and it executes always whether exception occurs or not.
Syntax:
try
{
}
catch(Exception e)
{
}
finally
{
	//finally block
}

Inside finally block the statement should be such that it should execute alaways in any case.
Example : Some system specific task like closing the shutter of ATM after taking the cash.

```
package exception_handling;
public class Program8 {
	public static void main(String[] args) {
		System.out.println("main starts");
		System.out.println(10+45+67);
		try
```

```
		{
			System.out.println(10/2);
		}
		catch (Exception e) {
		}
		finally
		{
			System.out.println("This is finally block");
		}
		System.out.println("Program ends");
	}
}
```

Q Is it possible to use try block without catch block?
Ans Yes try block can be used without catch block and in this case try block should have finally block. If try block is used with finally block then program will execute smoothly if there is no exception from try block, if there will be any exception from try block then there must be catch block to catch the exception object thrown from try block.

__NestedTry__
If a try block is written inside another try block then it is called a nested try block.
If there is an exception in parent statement then the control will not go to nested statement, in this case control will directly go to the catch block. It is not a good practice to neste the try catch block in the catch block, it should only be used to catch the exception not create it.

Syntax:

```
try
{
	try
	{
	}
	catch(Exception e)
	{
	}
}
catch
{
}
```

```
package exception_handling;
public class Program9 {
	public static void main(String[] args) {
		System.out.println("Main Method");
		int[] a = {12, 56, 67, 46, 56, 0};
		try
		{
			System.out.println(a[5]);// if  this shows an error then the
control will not go to nested statement
```

```
                try
                {
                        System.out.println(a[1]/a[5]);
                }
                catch(Exception e)
                {
                        System.out.println("Result is undeifined");
                }
        }
        catch(Exception e)
        {
                System.out.println("indes out of bounds");
                System.out.println(e);// to find the type of exception
                e.printStackTrace();// to find the heirarchy of exception
        }
        finally
        {
        }
        System.out.println("Program ends");
    }
}
```

Exception E = new ArrayIndexOutofBond Exception();
Sop (e); //is used to find which type of exception is there.

e.printStackTrace(); // it gives the hierarchy of exception occured(it tell the name of the files as well in which there is exception)

```
package exception_handling;
public class Program10 {
    public static void main(String[] args) {
        try
        {
                System.out.println(10/0);
        }
        catch(Exception e)
        {
                System.out.println("Result is undefined");
                System.out.println(e);
                e.printStackTrace();
        }
    }
}
```

Throw Keyword
throw is a keyword which is used to throw objects of an exception whenever an exception occurs then internally object is thrown by using keywords throw.

We can use throw keyword to throw an exception which is user defined.

Ex. throw new ArthmeticException();

22-12-2022

Any type of exception(pre-defined exception or custom exception) can be thrown by the keyword throw. By one throw keyword we can throw only one exception at a time.

```java
package exception_handling;
public class Program11 {
	public static void main(String[] args) {
		System.out.println("Main method starts");
		throw new ArithmeticException();//predefined exception
		throw new InsufficientFundException();//custom exception
	}
}
```

throws keyword
Throws is a keyword which is used to handle an exception at method decleration if a method gives exception then instead of using try catch block inside method. We can handle it by using throws keywords. By using one throws keyword we can handle multiple excpetions at a time.

Example
public static void test() throws ArithmeticException, FileNotFoundException, NullPointerExcpetion
{

}

This keyword only solves excpetion at compile time.

Thread.sleep( ) is used to stop the execution for a given period of time. It gives an Inturrupted exception which can be handled by throws keyword or try catch block

Example by throws keyword

```java
package exception_handling;
public class Program13 {
	public static void main(String[] args) throws InterruptedException{
		System.out.println("Main starts");
		for(int i = 1; i<=10;i++)
		{
			System.out.println("I is: "+i);
			Thread.sleep(2400);
		}
		System.out.println("Main ends");

	}

}
```

Difference between throw and throws keyword

1. By using throw keyword exception object is thrown, but by using throws keyword we handle an exception at compile time.
2. By using a throw keyword we can throw only one exception at a time, but by using throws keyword we can handle multiple exception at a time.
3. Throw keywods takes object of an exception but throws keyword takes name of exception.
4. throw keyword is used always inside method but throws keyword is used at method declaration.

Note :

1. Exception is considered the best way to run a program. Because in a program there are a lot file which makes it difficult to handle errors. Which can not be handled with flow.
2. Every user defined exception is checked excpetion which should always be handeld via before compilaton.

```
package exception_handling;
import java.io.*;
public class Program14 {
	public static void main(String[] args) {

		try {
			withdraw(45653);
		}
		catch (Exception e) {
			System.out.println("Excpetion is " +e);
		}
	}

	public static void withdraw(int amount) throws Exception {
		if(amount>0)
		{
			System.out.println("amount can be withdrawn");
		}
		else
		{
			throw new FileNotFoundException();
		}
	}
}
```

23-12-2022

Custom Exception

A user defined exception is called Custom Exception. A Custom Exception is a checked type exception, so it is checked by compiler at compile time. So it must be handled by using throws keyword at method decleration.

^Steps to have custom exception:-

1. We have to create the class for which we want to throw custom exception.

```
package exception_handling;
public class Program15 {
	public static void main(String[] args) {
		throw new UnderAgeException();//here under age exception is throw as
an object for which we have to create a class with the same name

	}

}
```

2. We have to give the properties of exception by extending exception to our class.
for which we have to create class which extends Exception like:

```
package exception_handling;
public class UnderAgeException extends Exception { // here class is now have
properties of exception
```

```
	UnderAgeExcpetion(String x)
	{
		super(x);
	}
}
```

3. We have to handle the custom exception by using throws keyword.
then we have to handle the exception by try catch block or throws keyword like:

```
package exception_handling;
public class Program15 {
	public static void main(String[] args) throws UnderAgeException {

		throw new UnderAgeException();

	}
}
```

4. To print some message through constructor we have have to design a constructor where there should be **super call** statement to print the message through parent class constructor.

```
package exception_handling;
public class UnderAgeException extends Exception {
	UnderAgeException(String x) {
		super(x);
	}

}
```

24-12-2022

STRING CONTINUES

String buffer object creation :

StringBuffer sb = new StringBuffer(String s);

```
package string;
public class Program23 {
	public static void main(String[] args) {
		StringBuffer sb = new StringBUffer("Mohan");
		System.out.println(sb);
		sb.append(" Sohan"); // it concats string buffer string
		System.out.println(sb);
		System.out.println(sb.reverse());// it helps in reversing the string

	}

}
```

***Important methods of String buffer:**

1. Append(): Append method is used to connect or to attach one string with another string.
Ex.
StringBuffer sb = new StringBuffer(“Today is “);
sop (sb); // O/P today is
sb.append(“Saturday”);

sop(sb); //O/P Today is Saturday.

2. Capacity(): Capacity method of string buffer is used to get the capacity of string buffer. The default capacity of string buffer is 16. Whenever a string buffer object is created then it is provided with default capacity 16. Capacity will increase as per the elements will be added in string.

Current Capacity = length +16;

Ex

```
package string;

public class Program24 {

	public static void main(String[] args) {

		StringBuffer sb = new StringBuffer("Mohan");

		System.out.println(sb.capacity());//output in 21
		System.out.println(sb.length());//output in 5

	}

}
```

3. Length(); length method is used to get the actual size of the string buffer.

```
package string;

public class Program24 {

	public static void main(String[] args) {

		StringBuffer sb = new StringBuffer("Mohan");
		System.out.println(sb.length());
	}

}
```

4. CharAt(Index): This method is used to get the character from provided index.

```
package string;

public class Program24 {

	public static void main(String[] args) {

		StringBuffer sb = new StringBuffer("Mohan");
		System.out.println(sb.charAt(2));
		System.out.println(sb.deleteCharAt(3));
	}

}
```

5. deleteCharAt(Index); this method deletes the character from the provided index.

```
package string;
```

```
public class Program24 {

	public static void main(String[] args) {

		StringBuffer sb = new StringBuffer("Mohan");
		sb.deleteCharAt(3);
		System.out.println(sb);// output Mohn
	}

}
```

6. Reverse(); this method is used to reverse the string buffer object.

```
package string;

public class Program24 {

	public static void main(String[] args) {

		StringBuffer sb = new StringBuffer("Mohan");
		System.out.println(sb.reverse());
	}

}
```

7. equals(): equals method in string buffer class only compares the address. So it will give false as the address of the two object is not same.

```
package string
public class Program24 {

	public static void main(String[] args) {

		StringBuffer sb1 = new StringBuffer("Mohan");
		StringBuffer sb2 = new StringBuffer("Mohan");

		System.out.println(sb1.equals(sb2));
	}

}
```

String Builder Class

String builder class in java is used to create mutable string objects. String builder is same as string buffer with one most important difference that string builder is **not synchronized.** So it performs faster as it can user more thread at a time.

Object Creation:
StringBuilder sb = new StringBuilder(String s);

***Difference between String Buffer and String builder**

1. String buffer is available since the initial release of JAVA but String builder is included in JAVA 5.0 version.
2. String buffer class is synchronized but string builder is not synchronized.

3. String buffer can be used by single threads at a time but String builder can be used multiple threads at time.
4. String buffer is slow in performance but string builder is fast in performance.

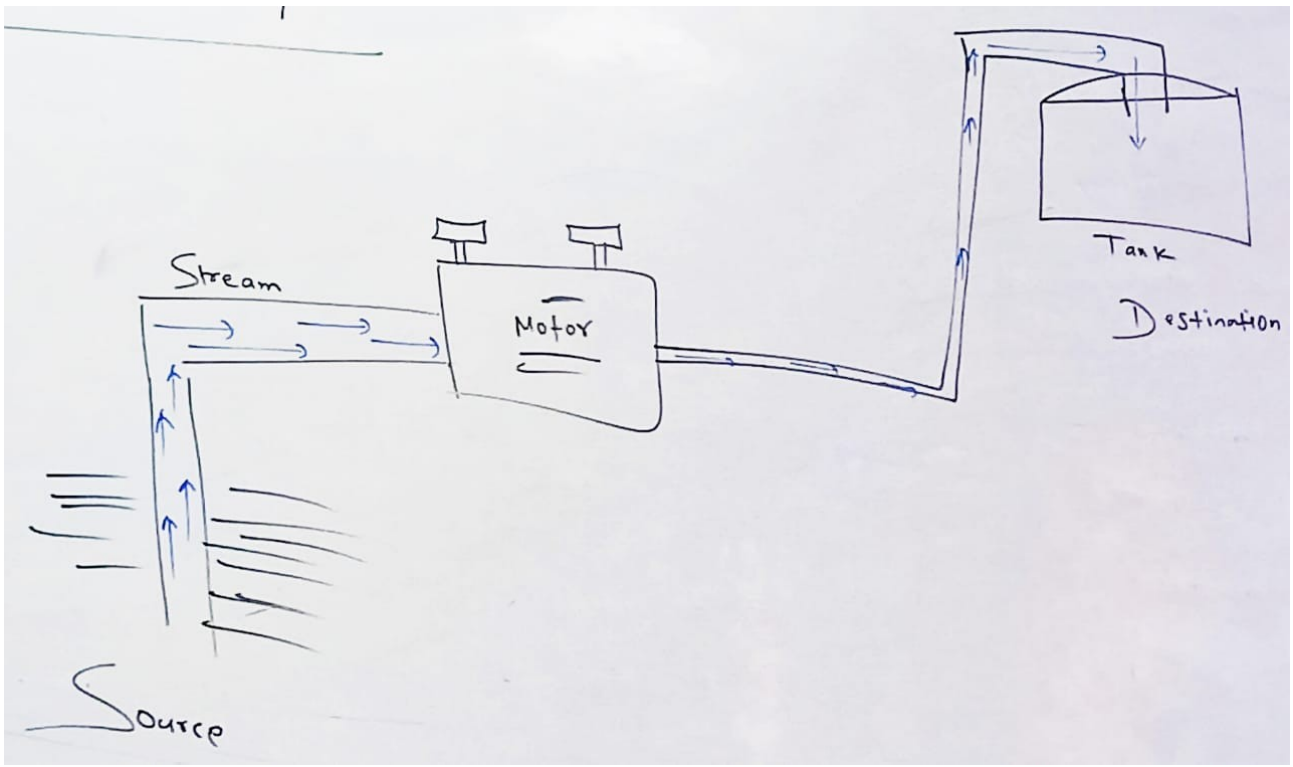
FILE HANDLING / FILE I/O

File handling or file I/O provides a mechanism in JAVA by using which we can perform various types of tasks such as **file creation, file reading or while writing opertions etc.**

To perform file I/O operation there should be source from which input can be accepted and there should be destination where output can be produced.
Example:

Input sources		**Output sources**
Keybooard		Console
Mouse	**JAVA PROGRAM**	Monitor
Storage		Printer
Disk		Speaker
network		Stroage
		Disk
		network

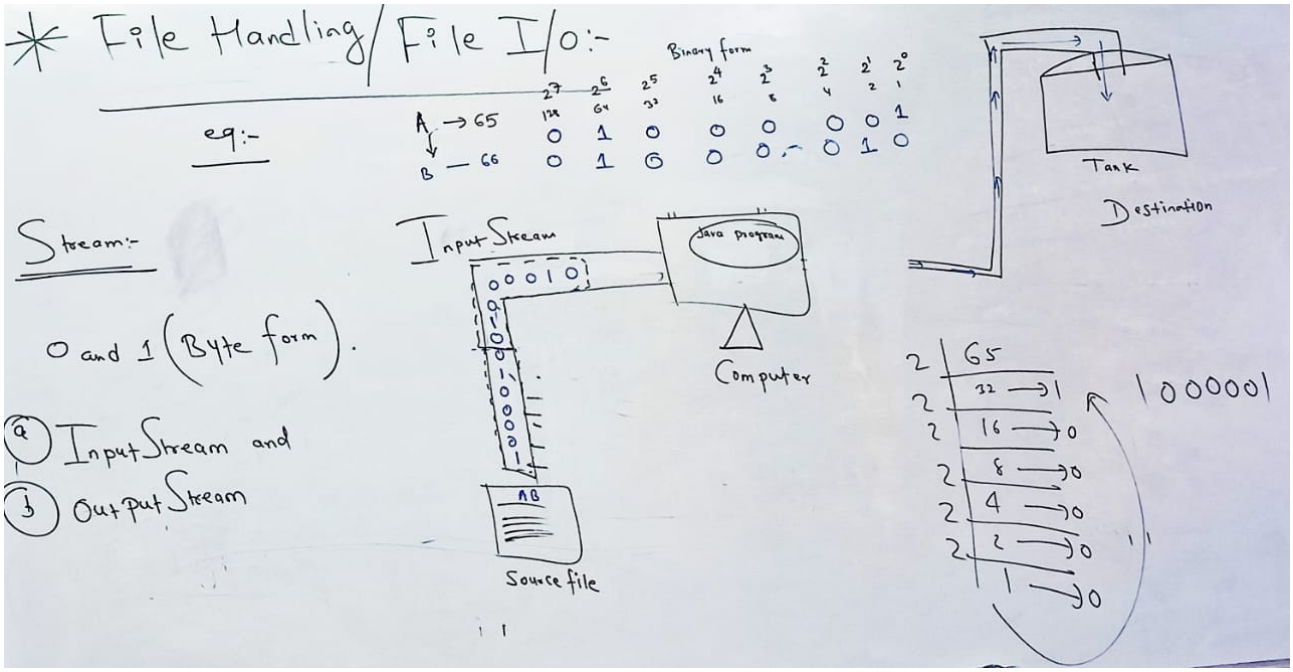
Stream



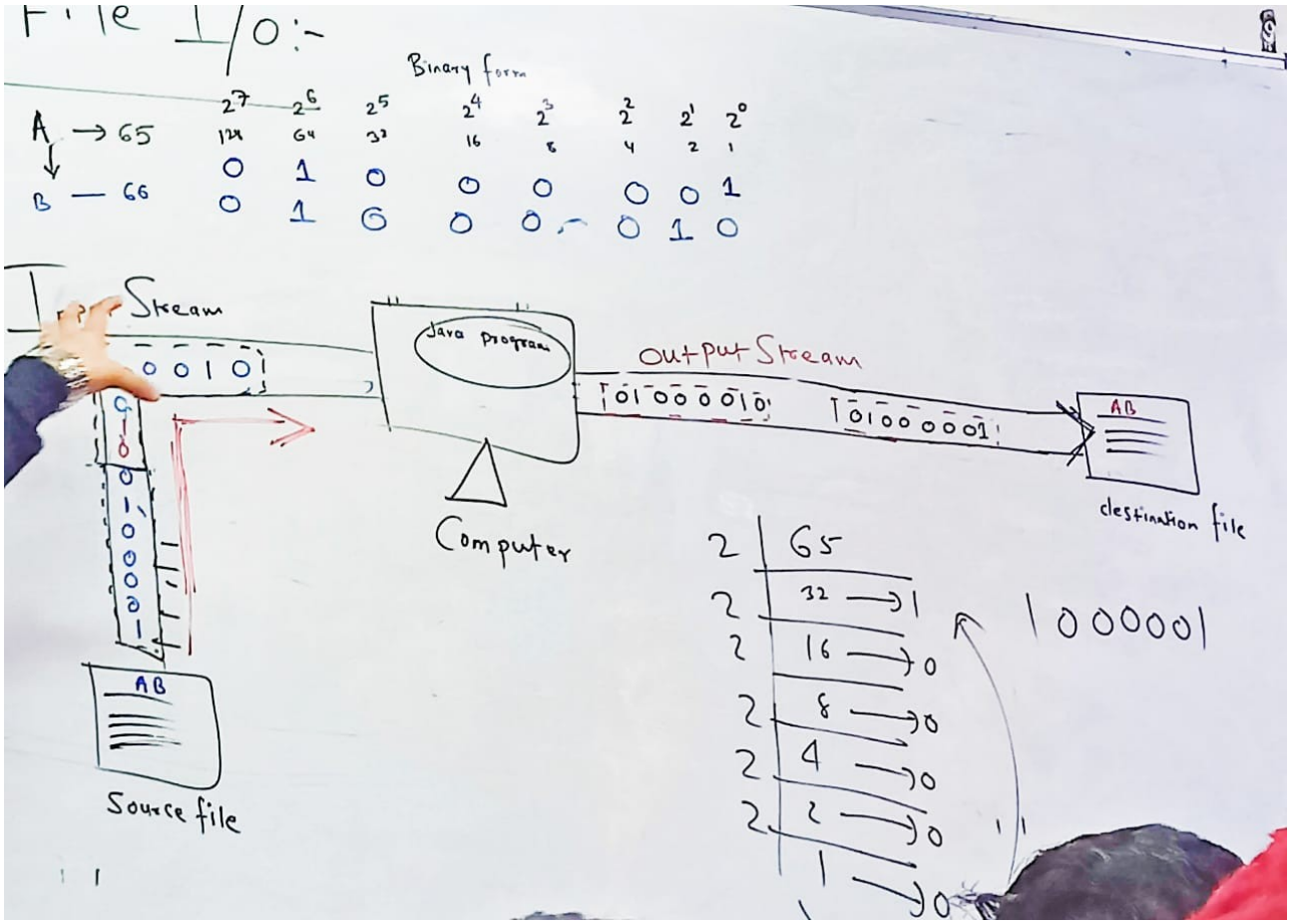
A stream in java is continues squential flow of data in the form of 0 and 1(byte form).
Stream in JAVA is of two types:
1. Input Stream

2. Output Stream

1. Input Stream : if the flow of stream is from source file to program then it is called input stream.



2. Ouput Stream : If the flow of stream is from JAVA program to destination file then it is called output stream.



To support this whole operation JAVA has a dedicated package called Java.io. Java.io package has many inbuilt interfaces and class which are used to **create a file, write into a file and read from a file.**

Create A file through JAVA program

To create a file through java program java.io package has many inbuilt classes and one of those is file class.

```
class File
{

}
```

File class has constructors and different methods which are used to create a file and get all the details about a file.

Constructor:

File f1 = new File(Path Name);

Ex: File f1 = new File(“d:\\ text.txt”);

```
package fIle_handling;
import java.io.File;
import java.io.IOException;
public class Program1 {
	public static void main(String[] args) {
		File f1 = new File("/home/nirmal/Desktop/text.txt");//paste path to the
		try
		{
			System.out.println("Inside try block");
			f1.createNewFile();
			System.out.println("file is created");
		}
		catch(IOException e)
		{
			System.out.println("Inside catch block");
			e.printStackTrace();
		}
		System.out.println(f1.canRead());
		System.out.println(f1.canWrite());
		System.out.println(f1.getName());
		System.out.println(f1.getPath());
		System.out.println(f1.length());
		f1.setWritable(false);// makes the file non writable
		f1.setWritable(true);//makes the file writable
		System.out.println("Program ends");
	}
}
```

By using this constructor we can set the path which is supposed to be created. If there is no file with the given name at provided location then file will be created otherwise file will be not created.
File will be created with the help of method **createFile();** which is available in file class.

Write into a file

We can write into a destination file by using in built class file writer of Java.io package.

Class FileWriter
{

}

Constructor:
FileWriter fw1 = new FileWriter(String Pathname);

By using this constructor a file object is created at the provided path.

```
package fIle_handling;
import java.io.FileWriter;
import java.io.IOException;
public class Program3 {
	public static void main(String[] args) {
		try {
		 FileWriter fw1 = new FileWriter("/home/nirmal/Desktop/text.txt");
		// paste the path of the file
			fw1.write("Today is a christmas day Eve "
				+ " And let me see what it is to be "
				+ " I have no idea what should i write here");
			// paste the text to be edited and
			//to go the the next line we have to use + operator
			fw1.close();
		}
		catch (IOException e) {
			e.printStackTrace();
		}
	}
}
```

Read From a File

We can read a file in Java program by providing file location to file class and by using scanner class of java.

```
package fIle_handling;
import java.util.*;
import java.io.*;
public class Program5 {
	public static void main(String[] args) {
		File f1 = new File("/home/nirmal/Desktop/text.txt");
		/// paste the path of the file
		try {
			Scanner sc = new Scanner(f1);
		// f1 is used as the reference which scanner class have to read
			while(sc.hasNextLine())
		// hasNextLine check does the current line have any character
```

```
                {
                        System.out.println(sc.nextLine());
            // nextLine prints the line and moves to the next line.
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```

31-12-2022
ARRAY

Array is a group of or similar type of collection of elements.

Declearation of an array:
Providing data type to an array is called array decleration.
Syntax :
int[] x;
dattype[] x;
dataType []x;
dataType x[];

int[] x;
doube[] y;
String[] z;
char[] p;

Car c1 = new Car();
Car c2 = new Car();
Car c3 = new Car();
Mobile M1 – new Mobile();
Mobile M2 – new Mobile();

Car[] c;//**we can store only car type objects**

Object[] o;//**We can store any type of objects**

Initialization of an array:

An Array can be initialized:

1. By using new keywords:-

Syntax:
data_type[] VarName = new data_type[size];
Example
int[] x = new int[10];//**10 here is length of array created**

Here, an array object with length 10 is created. Whenever an array object is created then it gets indexing for each elements that starts from 0 and the max index will be length minus one(10-1).

Default value	0	0	0	0	0	0	0	0	0	0
Index	0	1	2	3	4	5	6	7	8	9

We can access or update the elements of an array by using its index.
Ex.
sop(x[0]);//**o/p= 0**
sop(x[4]);//**o/p= 0**

x[4] = 38;
sop(x[4]);//**o/p= 38**

```
package array;
public class Car {
	String name;
	int price;
	String color;
	Car()
	{
	}
	Car(String name, int price, String color)
	{
		this.name= name;
		this.price= price;
		this.color = color;
	}
	public String toString()
	{
		return "Name is "+name+" Price is "+price+" Color is "+color;
	}
}
package array;
public class Program4 {
	public static void main(String[] args) {
		Car[] c = new Car[10];
		Car c0 = new Car("Maruti", 56768, "White");//
		Car c1 = new Car();
		Car c2 = new Car("Audi", 773838, "Black");
		Car c3 = new Car();
		Fruit f0 = new Fruit();
		c[0] = c0;
		c[1] = c1;
		c[2] = c2;
		//c[3] = f0;// we cant add car with fruit type
		System.out.println(c[0]);// here hii is returning with toString
overriding//array.Car@3f8f9dd6
		System.out.println(c[1]);
		System.out.println(c[2]);
		Object[] obj= new Object[10];
		obj[0] = c1;// here we can store car in object because is object is
super most class
		obj[1]= f0;
	}
}
```

output

```
Name is Maruti Price is 56768 Color is White
Name is null Price is 0 Color is null
Name is Audi Price is 773838 Color is Black
```

2. Initialization without using new keywords.

Syntax:
data_Type[] VarName= {e1, e2, e3, e4 .....};
Example
int[] a = {10, 20, 30, 48, 60, 70};

10	12	30	48	60	70
0	1	2	3	4	5

By this manner an array object is created with the length depending on how many elements are passed during array initialization.

```
package array;
public class Program5 {
	public static void main(String[] args) {
		int[] a = {45, 67, 23, 55, 67, 44};
		System.out.println(a[3]);
		System.out.println("Length of array is"+a.length);// a.length tells
the length of an array
		System.out.println("============");
		for(int i= 0; i<a.length;i++)
		{
			System.out.println(a[i]);
		}
	}
}
```

We can get the length of an array by using a variable length.
ex.
int[] a= new int[10];
sop(a.length);// O/P 10

Max index = length -1

We can access the array elements by using:
1. for Loop
2. for each Loop

1. for Loop
We can access elements by using for loop one by one as given below:

Accesssing the elements
Int[] a = new int[10];
for(int i = 0; i<a.length;i++)
{
sop(a[i]);
}

Enter the elements
for(int i = 0; i<a.length;i++)
{
sop(“Enter the element”);
a[i] = sc.nextInt();
}

```
package array;
import java.util.*;
import java.util.Iterator;
public class Program6 {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int count = 0;
		int [] a= new int[10];
		for(int i= 0;i<a.length;i++)// for creating the elements
		{
			count++;
			System.out.println("enter the "+count+"element");
			a[i] = sc.nextInt();
		}
		System.out.println("you entered");
		for(int i= 0; i<a.length;i++) // for accessing the elements
		{
			if(a[1]>=100 && a[i]<=999)
			System.out.println(a[i]);
		}
	}
}
```

Ques. WAJP to find pallindrome armstrong and strong number in the given array and print the same only.

```
package array;
import java.awt.Checkbox;
import java.util.Scanner;
public class Program7 {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int count = 0;
		int [] a= new int[5];
		for(int i= 0;i<a.length;i++)// for creating the elements
		{
			count++;
			System.out.println("enter the "+count+"element");
			a[i] = sc.nextInt();
		}

		for(int i= 0; i<a.length;i++) // for accessing the elements
		{
		       if(checkPalindrome(a[i])) {
				System.out.println(a[i]);
		       }
		}
	}

	public static boolean checkPalindrome(int n)
	{
		int a = n;
		int rev = 0;
		while(n>0)
		{
			int rem = n%10;
			rev = rev*10+rem;
			n=n/10;
		}
```

```
            if(a==rev)
            {
                  return true;
            }
            else
            {
                  return false;
            }
      }
}
```

2. for each loop or enhanced for loop

for each loop is used to access array elements or collection elements one by one. It assignes the value of array to the variable one by one and checks if there is any elements after the present one. It stop itselfs after the array ends.

Syntax
for(data_type VarName: Array/Collection)
{

}

ex

```
package array;
public class Program8 {
      public static void main(String[] args) {
            int[] a = {45, 67, 23, 55, 67, 44};
            for(int p:a)
            {
                  if(p%2==0)
                  System.out.println(p);
            }
      }
}
```

Here “p” will be assigned to every value of “a” array one by one.

Multi Threading

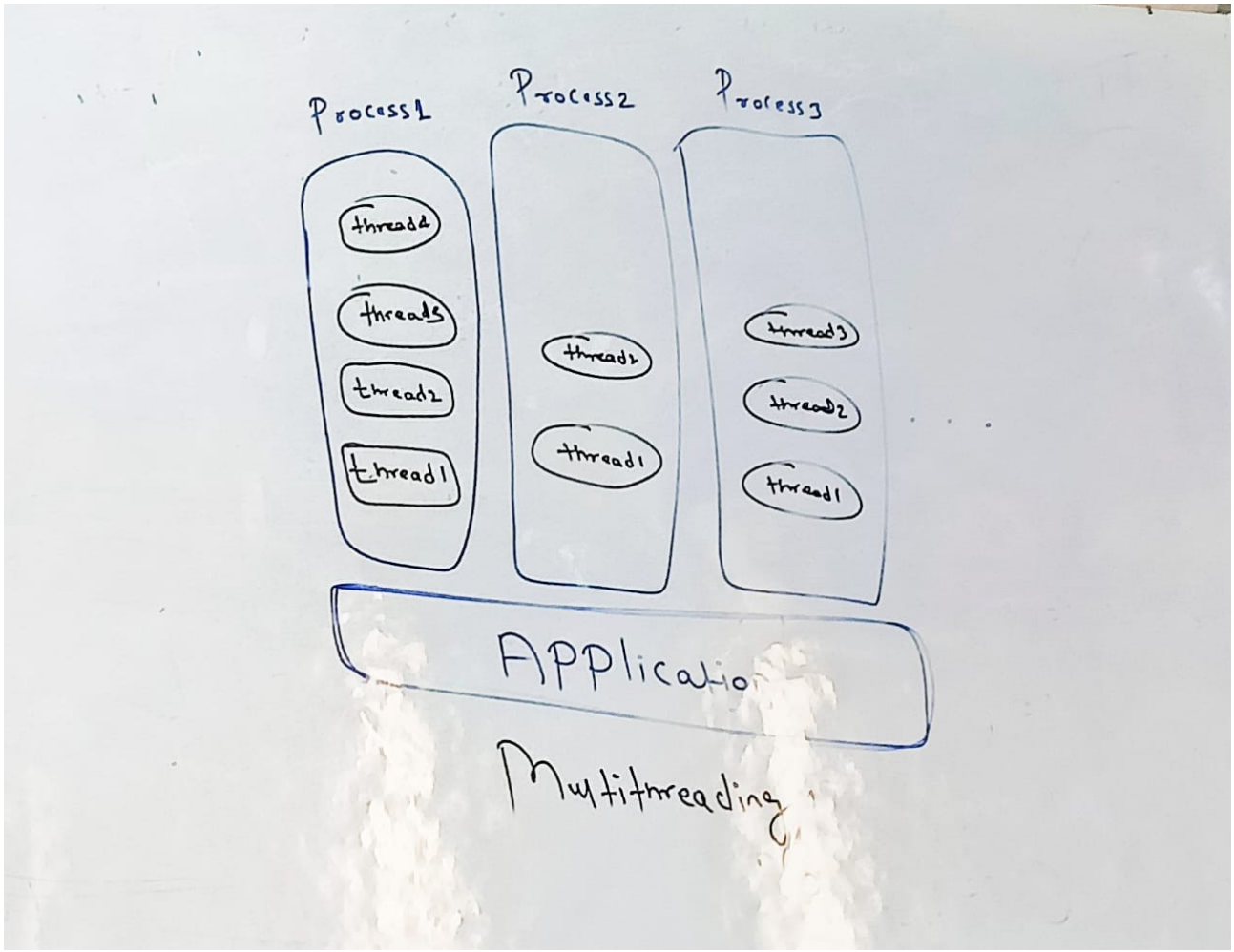
Thread

A thread in java is a smallest sub proccess that is used to complete a process

A task normally uses many threads or sub-process to complete the whole process.

Multi-threading

Multi threading is a concept where many threads (Sub-process) works together to complete a process.



Advantages of Multi Threading:

1. By the help of multi threading we can achieve maximum utilization of CPU.
2. We can perform multiple tasks at the same time parallely.
3. Multi Threading helps to have better user experience.
4. By the help of multi threading we can have fast response from the different tasks.

How to create thread in JAVA

In java threads can be created in two ways:

1. By extending Thread class of JAVA
2. By implementing Runnable interface

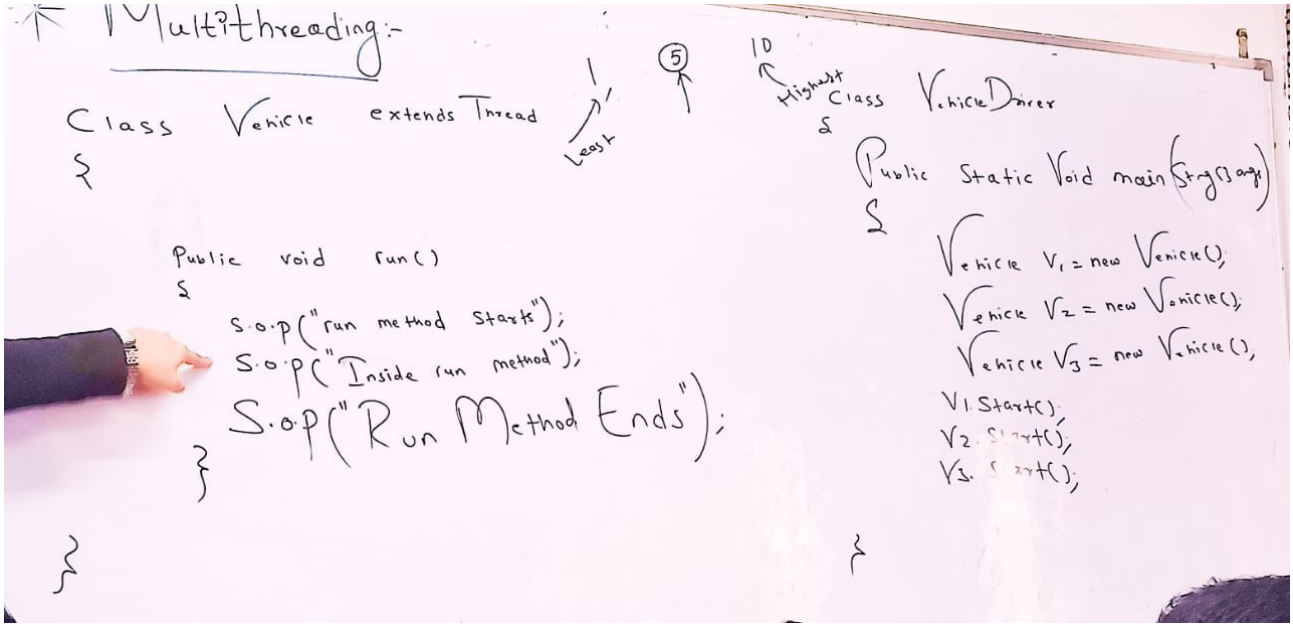
1. By extending Thread class of JAVA

We can create thread by extending thread class in the following steps

Step 1: We have to extend thread class to our class for which we want to create thead.

Step 2: We have to override run method of thread class in our class.

Step 3: We have to create object of our class and we have to call start method on that object reference to start the execution of thread. Start method internally calls run method to execute the thread.



We can set the priority of a method call, to delay or fasten the execution.

```
package multiThreading;
public class Vehicle extends Thread{
	public void run()
	{
		System.out.println(Thread.currentThread().getId());
		System.out.println(Thread.currentThread().getName());
		System.out.println("run method starts");
		System.out.println("Inside run method ");
		System.out.println("Run method ends");
	}
}
```

```
package multiThreading;
public class VehicleDriver {
	public static void main(String[] args) {
		Vehicle v1 = new Vehicle();
		Vehicle v2 = new Vehicle();
		Vehicle v3 = new Vehicle();
		Vehicle v4 = new Vehicle();
		v1.setName("Bullock cart");
		v2.setName("Car");
		v3.setName("Bike");
		v4.setName("Cycle");

		System.out.println("thread name is: "+ Thread.currentThread().getName());
		System.out.println("thread id is : "+Thread.currentThread().getId());
		v2.setPriority(10);
		v1.setPriority(1);
		v1.start();
		v2.start();
		v3.start();
		v4.start();
	}
}
```

2. By implementing runable interface

We can create thread by implementig runnable interface in the following ways:
Step1: We have to implement runnable interface to our class.
Step2: We have to override run method of runnable interface in our class.
Step3: We have to create object of our class for which we want to create thread.
Step4: We have to create object of thread class and at thread constructor we have to pass the reference of object created.
Syntx. Thread t1= new Thread(thread object reference);

Ex.

```
class VehicleDriver {
public static void main (Strng[] args)
{
        Vehicle v1 = new Vehicle();
        Vehicle v2 = new Vehicle();

        Thread t1 = new Thread(v1);
        Thread t2 = new Thread(v2);
}
```

Step5: We have to call start method so that the thread will start execution.

```
package multiThreading;
public class Rocket implements Runnable {

      public void run()
      {
            System.out.println("========================");
            System.out.println("Run method is implemented");
            System.out.println("Thread name is:
"+Thread.currentThread().getName());
            System.out.println("thread priority is
"+Thread.currentThread().getPriority());
            System.out.println("Current thread is terminated");
      }
}

package multiThreading;
public class RocetDriver {
      public static void main(String[] args) {
            Rocket r1 = new Rocket();
            Rocket r2 = new Rocket();
            Rocket r3 = new Rocket();

            Thread t1 = new Thread(r1);
            Thread t2 = new Thread(r2);
            Thread t3 = new Thread(r3);

            t1.setName("Appolo");
            t2.setName("Chandrayan");
            t3.setName("Aryabhatt");

            t1.start();
            t2.start();
            t3.start();
      }
}
```

The process fo execution of thread

A thread in programming undergoes the following steps:-

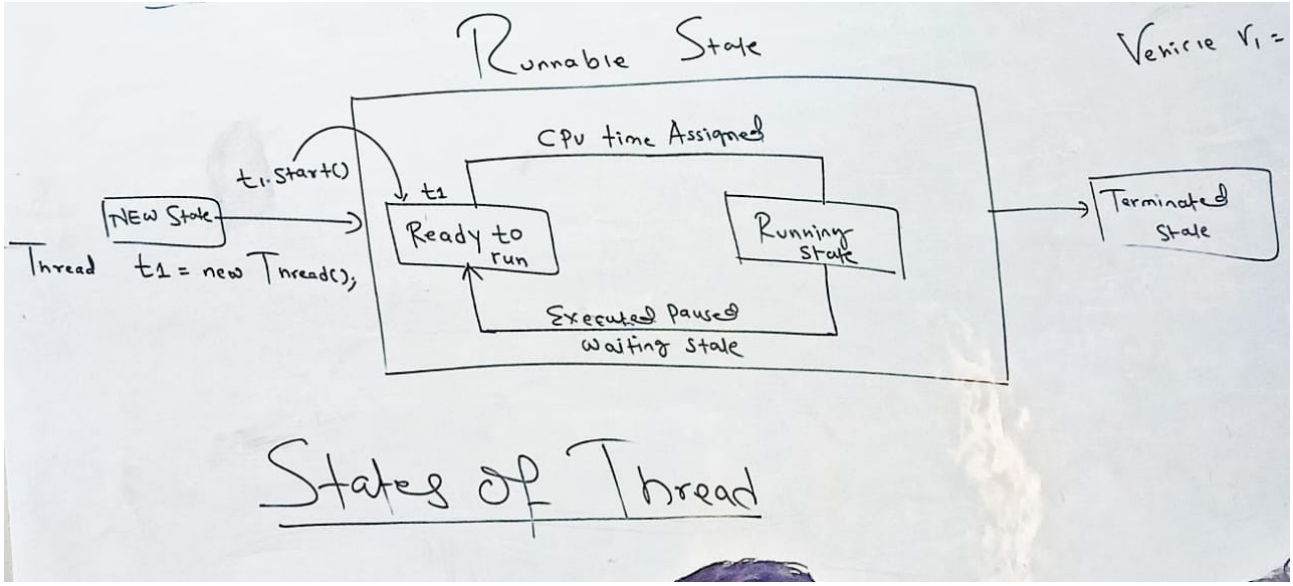
Step1: When a thread is created then its called new state of thread.
Example : Vehicle V1 = new Vehicle();
new state of thread

Step2: When start method is called then thread enters into runnable area.
Example: v1.start();
thread entered into runnable area

Step3: Once a thread enters into runnable area then thread will be ready to run at any moment. It is the responsibility of **Thread Schedular** to decide which thread should be executed first or which thread should get CPU time.

Thread Schedular persues execution of thread on priorty basis. A thread can have priority from 0 to ten. A high priorty thread executes first before low priority thread. If priority is not set for thread then thread shcedular gives priority **5** to each thread.

If multithread will have same priorty then thread schedular executes thread parralelly.



Step4 When a thread completes its execution then it comes out of runnable state and enters into terminated state where it is considered as **Dead thread.**

Important Q1 What is the different state of thread ?

<u>*Different types of thread*</u>

A thread is of two types:
1. User Thread
2. Deamon thread

1. User thread: A user created thread is called user defined thread. A program cannot be terminated until a user defined thread will complete its execution. JVM waits for a user thread to complete its task then only JVM will finish the execution of process.

2. Deamon thread: Deamon thread are background running thread which are used to provide service to an application.

So a background running service provider threads are called deamon threads. Deamon threads are low priority threads. JVM can terminate an application whether deamon thread have completed its execution or not.

07-01-2023
COLLECTION FRAMEWORK

Collection frame work in java is a group of many interfaces and classes which works together to store and perform CRUD operations on object.

Difference between array and collection:

1. Array is fixed in size but collection provides a mechanism by which the size of collection can grow or shrink.
2. We can add only similar types of elements in array but in collection hetrogenious type of element can be added.
3. Array does not have any inbuilt method support but collection has a large number of method support which makes programming simple in collection.
4. In array we can add primitive as well as non primitive types of data but in collection only non-primitive type data is allowed so in collection any element is added in the form of an object.
5. Array provides faster execution but the execution of collection is slower in the comparison of array.

>>To overcome this limitations of array java provides different data structures in the form of collection framework such as:
1. List
2. Set
3. Queue
4. Map

>By using these provided data structure we can efficiently perform CRUD Operations such as:
1. Create a collection and add the elements
2. Remove
3. Update
4. Read
5. Search/Access
6. Sort the Elements of collection
etc.

>To perform the above mentioned operations collection framework provides two different hierarchies:
1. Collection Hierarchy
2. Map Hierarchy

Collection Hierarchy

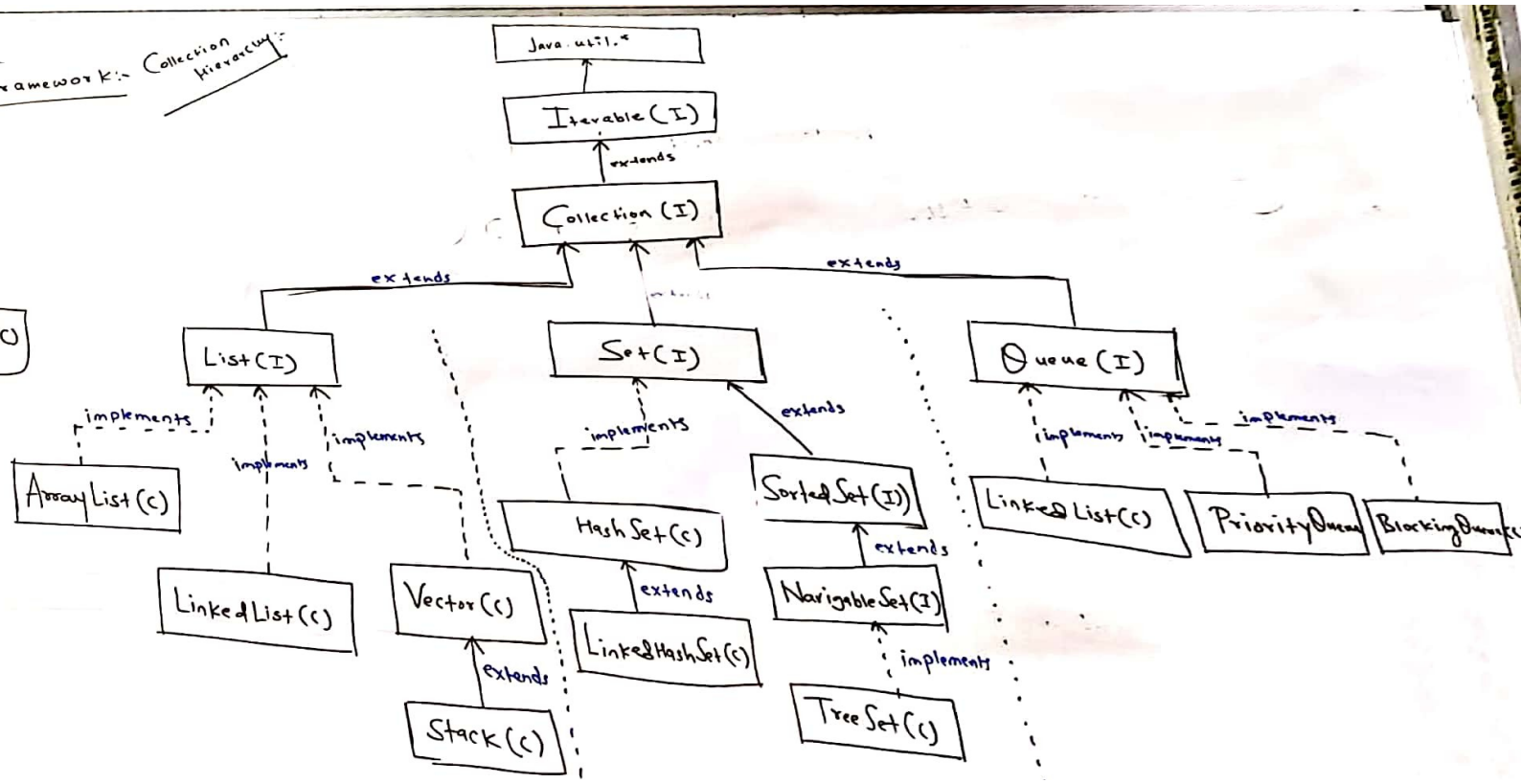
Collection hierarchy in JAVA is available in java.util package.

Collections hierarchy has top most interface called **Collection.** Collection interface is extended by different interface such as:

1. List
2. Set
3. Queue

Collection hierarchy provides a group of many interfaces and classes which are used to add a group of individual objects and perfrom various tasks over them.

Java .util.*;		
Iterable(I)		
Collection(I)		
List(I)**extends collections**	Set(I)**extends collections**	Queue(I) **extends collections**
Arraylist(c) **Implements list**	HashSet(c) **Implements Set**	LinkedList(c) **Implements queue**
LinkedList(c) **Implements list and Queue**	LinkedHashSet(c) **extends HashSet**	PriorityQueue (this is written in c language)**Implements queue**
Vector(c) **Implements list**	SortedSet(I) **extends Set**	BlockingQueue(c) **Implements queue**
Stack(c) **Extends Vector**	NavigableSet(I) **extends Sortedset**	
	TreeSet(c) **Implements NavigableSet**	

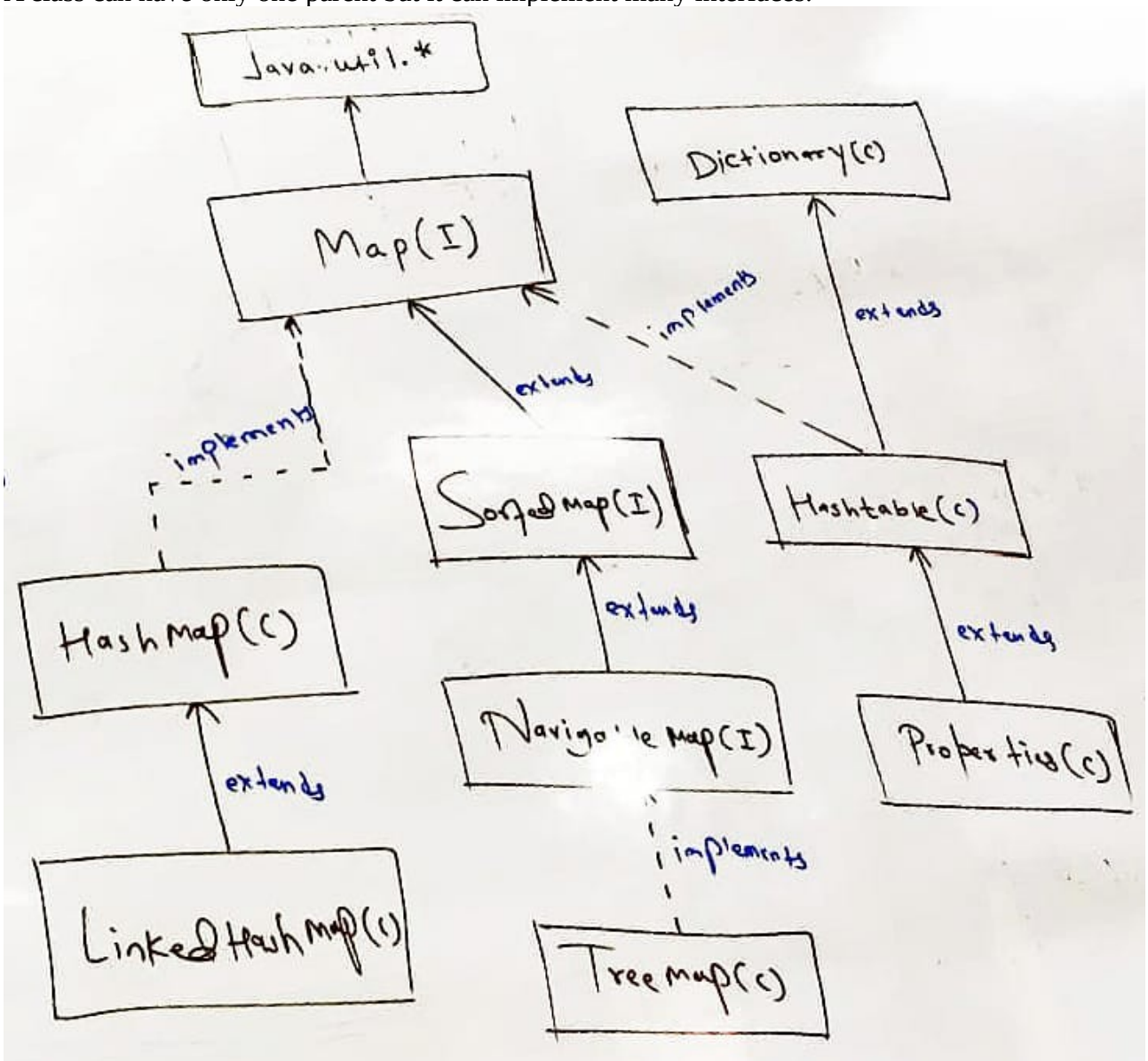


Map Hierarchy

Map heirarchy is availale in java.util package. The top most interface of map hierarchy is **map interface** which is extended by different intefaces and implemented by different classes.

Java .util.*;		
Map(I)		
HashMap(c)**implements Map**	SortedMap(I)**extends Map**	Hashtable(c) **implements Map**
LinkedHashMap(c) **extends HashMap**	NavigableMap(I) **extends SortedMap**	Properties(c) **extends Hashtable**
	TreeMap(c) **implements NavigableMap**	Dictionary(c) **extends Has Hashtable**

class Hashtable extends Dictionary implements Map
{
}
A class can have only one parent but it can implement many interfaces.



Collection Interface

Collection interface is top most inteface of collection hierarchy which is available in java.util package and used to store a group of individual objects. Collection interace has different methods which is used to perform various operations on object.

Collection Methods:-

1. To add object/objects

1.1 add(objects):-

By this method we can add an object into collection.

```
package collection;
import java.util.*;
public class Program2 {
	public static void main(String[] args) {
		Collection c1 = new ArrayList();
		System.out.println(c1);
		c1.add(34);// everything in () is an object
		c1.add(68);
		c1.add(678.67);
		c1.add("Mohan");
		/*Object o=18;
		 * System.out.println(o+100);*/// this will not be possible
because “o” here is an object which has to be downcasted
	}
}
Ouput= [34, 68, 678.67, Mohan]

Everything in c1.add(); will be considered object so to add these object we
have to downcast it to Integer for Example

package collection;
public class Program3 {
	public static void main(String[] args) {
		Object o1 = 34;
		Integer i = (Integer)o1;
		System.out.println(i+100);
	}
}
```

1.2 addAll(Collection C):-

By this method we can add a group of objects or a collection of elements

```
Collection c2 = new ArrayList();
		c2.add(45);
		c2.add(80);
		c2.addAll(c1);// here it is adding all the objects of above written
code of the collection c1 and adding it to c2 on the reference of collecion “c1”
		c2.add(100);

		System.out.println(c2);

Output= [45, 80, 34, 68, 678.67, Mohan, 100]
```

2. Remove object/Objects from method :

2.1 remove(Object):

By using this method we can remove object from collection.

```
package collection;
import java.util.ArrayList;
import java.util.Collection;
public class Program4 {
	public static void main(String[] args) {
		Collection c1 = new ArrayList();
		c1.add(34);// everthing in () is object
		c1.add(68);
		c1.add(678.67);
		c1.add("Mohan");
		c1.add(68);
		System.out.println(c1);
		c1.remove(68);
		System.out.println(c1);
		/* it will remove the first "68" it finds in collection
		 because it doesn't support index so we cannot choose what to remove
		 first or last*/
		c1.remove(70);
		/*remove method returns true or false
		 * so it wont give error if it can't find the given object
		 */
	}
}

Output
[34, 68, 678.67, Mohan, 68]
[34, 678.67, Mohan, 68]
```

2.2. removeAll(Collection c);

By this method we can remove multiple objects from collection at a time. It can also be used to remove the simillar objects from two collections

```
package collection;
import java.util.ArrayList;
import java.util.Collection;
public class Program5 {
	public static void main(String[] args) {
		Collection c1 = new ArrayList();
		c1.add(34);// everthing in () is object
		c1.add(68);
		c1.add(678.67);
		c1.add("Mohan");
		c1.add(68);
		c1.add(120);
		c1.add(200);

		Collection c2 = new ArrayList();
		c2.add(68);
		c2.add(100);
		c2.add("Mohan");

		c1.removeAll(c2);
		/* this is removing all the objects from collection c1
		 * which are present in collection c2 , but it will not change
```

```
             * collection c2
             */

            System.out.println(c1);
            System.out.println(c2);
      }

}

OUTPUT =
[34, 678.67, 120, 200]
[68, 100, Mohan]
```

2.3 retainAll(Collection c):

This method is used to retain a particular collection objects and rest other collection objects will be removed

```
package collection;
import java.util.ArrayList;
import java.util.Collection;
public class Program6 {
      public static void main(String[] args) {
            Collection c1 = new ArrayList();
            c1.add(34);// everthing in () is object
            c1.add(68);
            c1.add(678.67);
            c1.add("Mohan");
            c1.add(68);
            Collection c2 = new ArrayList();
            c1.add(120);
            c1.add(200);
            Collection c3 = new ArrayList();
            c3.add(70);
            c3.add(170);
            Collection c4 = new ArrayList();
            c4.add(400);
            c4.add(440);
            c1.addAll(c2);
            c1.addAll(c3);
            c1.addAll(c4);
            System.out.println(c1);
            c1.retainAll(c3);// this will retain the collection c3 in c1
            System.out.println(c1);
      }
}
OUTPUT :
[34, 68, 678.67, Mohan, 68, 120, 200, 68, 100, Mohan, 70, 170, 400, 440]
[70, 170]
```

2.4 clear();

This method is used to remove every element from collection

```
      System.out.println(c1);
            c1.clear();
            System.out.println(c1);
Output:
[34, 68, 678.67, Mohan, 68, 120, 200, 70, 170, 400, 440]
[]
```

3 Access Object from Collection:

We can access the elements of collection by using different loops and by using method called iterator().

1. for loop
2. for each loop
3. iterator()

When the reference is of collection then we cannot use for loop because for loop will access the elements based on the index **but** collection does not provide indexing.

When the reference is of collection then we can access the elements one by one by using for each loop or by using iterator method.

For Each Loop

For each loop is used for array or collection objects where it has its own algorithm that operates on a group of elements.

Syntax:
for(datatype Varname: Array/collection)
{
}

Example :
int[] a ={12,15,18,20,30};
for(int x:a)
{
	sop(x);
}

Example:
Collection c1 = new ArrayList();
c1.add(12);
c1.add(20);
c1.add("mohan");
for(Object x : c1)
{
sop(x);
}

Question WAJP TO PRINT THE EVEN NUMBERS IN A COLLECTION ARRAY.

```
package collection;
import java.util.ArrayList;
import java.util.Collection;
public class Program8 {
	public static void main(String[] args) {
		Collection c1 = new ArrayList();
		c1.add(34);// everthing in () is object
		c1.add(68);
		c1.add(67);
		c1.add(197);
		c1.add(68);
		System.out.println(c1);
```

```
            System.out.println("====================");
            for(Object x:c1)
            {
                  Integer r= (Integer) x;
                  if(r%2==0)
                  {
                        System.out.println(r);
                  }
            }
      }
}
```

OUTPUT

```
[34, 68, 67, 197, 68]
====================
34
68
68
```

21-01-2023

Miscellaneous Methods:-

1. size(): this method is used to get size of a collection

```
Collection c1 = new ArrayList();
            c1.add(34);// everthing in () is object
            c1.add(68);
            c1.add(67);
            c1.add(197);
            c1.add(68);
            System.out.println("Size is"+c1.size());
```

2. **isEmpty():** is empty method is used to get boolean result true or false after checking whether collection is empty or not.

```
            Collection c1 = new ArrayList();
            c1.add(34);// everthing in () is object
            c1.add(68);
            c1.add(67);
            c1.add(197);
            c1.add(68);
            System.out.println(c1.isEmpty());
```

3. toArray(): this method is used to convert a collection into array.

```
            public static void main(String[] args) {
                  Collection c1 = new ArrayList();
                  c1.add(34);// everthing in () is object
                  System.out.println(c1.hashCode());
                  c1.add(68);
                  c1.add(67);
                  c1.add(197);
                  c1.add(68);
                  int sum = 0;
                  Object[] x = c1.toArray();
                  for(int i = 0;i<x.length;i++)
                  {
                        System.out.println((Integer)x[i]+10);//its adding a 10
value
                  }
            }
```

}
Note: generic is simply parameterizing the collection

4. hasCode(): this method is used to get the HasCode of the collection created.
Hashcode is a unique integer number which is assisgned internally when ever an object is created.

```
Collection c2 = new ArrayList();
System.out.println(c2.hashCode());
```

5. equals(): this method is used to compare two collection and after comparison it gives boolean result. True in case two collection is same and false in case two collection is not same.

```
	public static void main(String[] args) {
		Collection c1 = new ArrayList();
		Collection c2 = new ArrayList();
		c1.add(34);
		c2.add(34);

		System.out.println(c1.equals(c2));
```

LIST INTERFACE

List

List is a child interface of collection. List has properties of collection as well as its own unique properties which are not available to **set or queue.**

Properties of List:

1. List is a odered collecton of elements where insurtion order is maintained. We can access the list element in the same order in which it is added.
2. List has indexing so we can perform different operations by using index.
3. We can add duplicate elements inside list.
4. We can add multiple null values in list.

Methods of Lists:

1. Add the element/ elements:

1.1 add(int index, object value):- By this method we can add an object at the provided index.

Ex. l1.add(2, 38);
=> so 38 will be added to 2nd Index.

1.2 addAll(int index, Collectio c):- By this method we can add a collection from the given index.

2. Remove the element/elements:

2.1 remove(int index):- By this method we can remove the element from given index.

```
	public static void main(String[] args) {
		List l1 = new ArrayList();
		l1.add(12);
		l1.add(42);
```

```
            l1.add(34);
            l1.add(1, 65);
            l1.add(76);
            l1.remove(3);
            l1.add(66);
            l1.remove(1);
            System.out.println(l1);
      }
}
```

o/p

```
[12, 42, 76, 66]
```

3. Search an element:

3.1 indexOf(Object):- By using this method we can get the index of the provided object.

```
System.out.println("Index of 76 is "+l1.indexOf(76));
```

```
O/P = Index of 76 is 2
```

4. Access elements from list:

We can access the elements of list in following ways :

a. for Loop
b. for each loop
c. iterator()

List specific methods are:-

d. listIterator():- By using listIterator we can iterate forward as well as in backword direction.

c. get(int index):- By using this method we can get the elements from the provided index.

ex.

```
List l1 = new ArrayList();
            l1.add(12);
            l1.add(42);
            l1.add(34);
            l1.add(57);
            l1.add(42);

            System.out.println("============");
            System.out.println("by for each loop");
            for(Object x:l1)
            {
                  System.out.println(x);
            }
            System.out.println("============");
            System.out.println("by for loop");
            for(int i = 0;i<l1.size();i++)
            {
                  System.out.println(l1.get(i));
            }
```

Genrics

Genrics in JAVA is parameterized type collection.
In other words, if a collection is parameterized type, then it is called generics.
Genereics in JAVA is intruduced in JAVA 5.0 version

Syntax:
Collection<data_type> c1 = new ArrayList<data_type>();

example:
Collection <Integer> c1 = new ArrayList<Integer>();

Colection <String> c2 = new ArrayList<String>();
etc

Q1. find prime number in List

```
//how to find prime number in a List
package collection;
import java.util.*;
public class Program14 {
	public static void main(String[] args) {
	List<Integer> l1 = new ArrayList<Integer>();
		l1.add(12);
		l1.add(17);
		l1.add(34);
		l1.add(57);
		l1.add(97);
		l1.add(119);
		l1.add(71);
		List l2 = new ArrayList();
		for(int i = 0;i<l1.size();i++)
		{
			if(checkPrime((Integer)l1.get(i)))
			{
				l2.add(l1.get(i));
			}
		}
		System.out.println("the prime number are "+l2);
	}
	public static boolean checkPrime(int n)
	{
		int factor = 0;
		for(int i = 2;i<=n/2;i++)
		{
			if(n%i==0) {
				factor++;
			}
		}
		if(factor==0)
		return true;
		else
		return false;
	}
}
```

Advantage of generics

1. By the use of generics we can achieve more type safety and stability in the program.
2. By the use of generics we don't have to do downcasting.
3. By the use of generics we can handle the issues of program at compile time instead of handling it at runtime. To handle the issues at compile time is much better then handling at runtime.

ArrayList

Array list class is the concrete implementation of list interface. Array list is the child class of list interface in collection framework. By the use of arraylist we can store a group of individual objects which will have different characteristics.

Characteristics of ArrayList

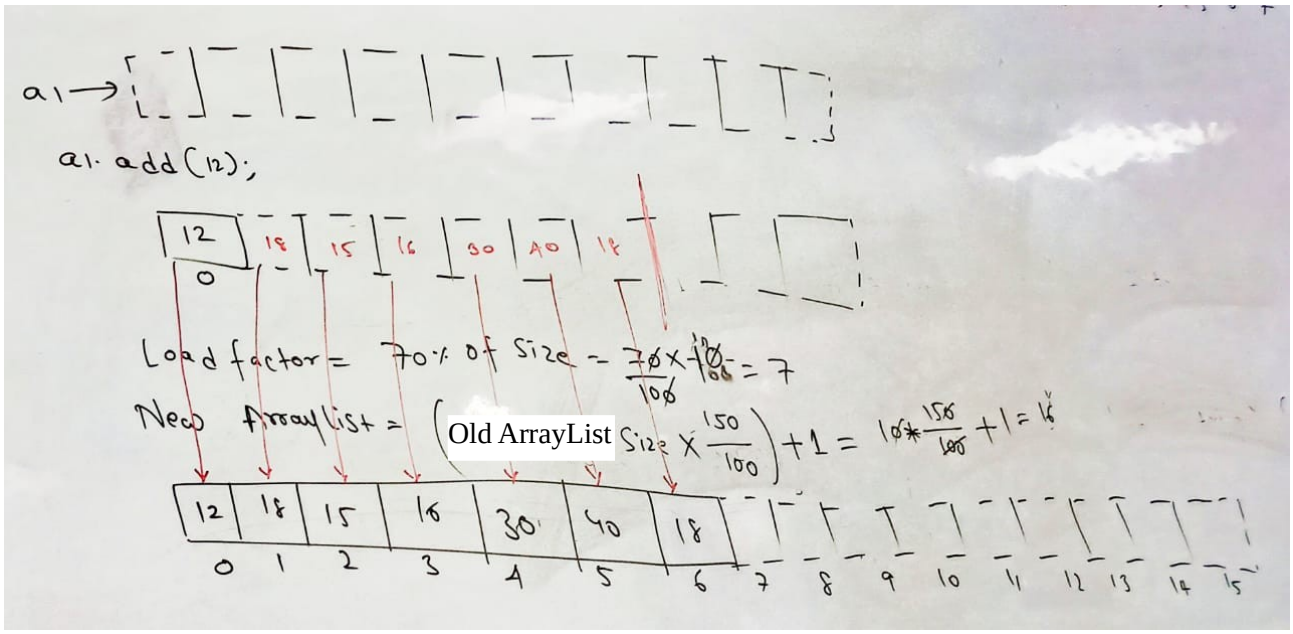
1. In array list insertion order of the elements will be maintained.
2. Array list will have indexing so different operations can be performed based on the index.
3. We can add duplicate elements in arraylist.
4. In array list a dynamic array is used as the data structure to add the elements which can grow or shrink as per the requirements.

4.1Constuctors of ArrayList

a. ArrayList a1 = new ArrayList()**;**
|constructor|

By this constructor internally a dynamic array is allocated with default size 10. When 70% of the size is occupied then it is called the load factor of arraylist. When load factor is reached then internally a new array list is allocated which will increase approximately 50% in size.

When new array list is created then all the elements of old array list will be coppied into new arraylist and the old arraylist will be destroyed by garbage collector.



b. ArrayList a2 = new ArrayList(int Size**);**

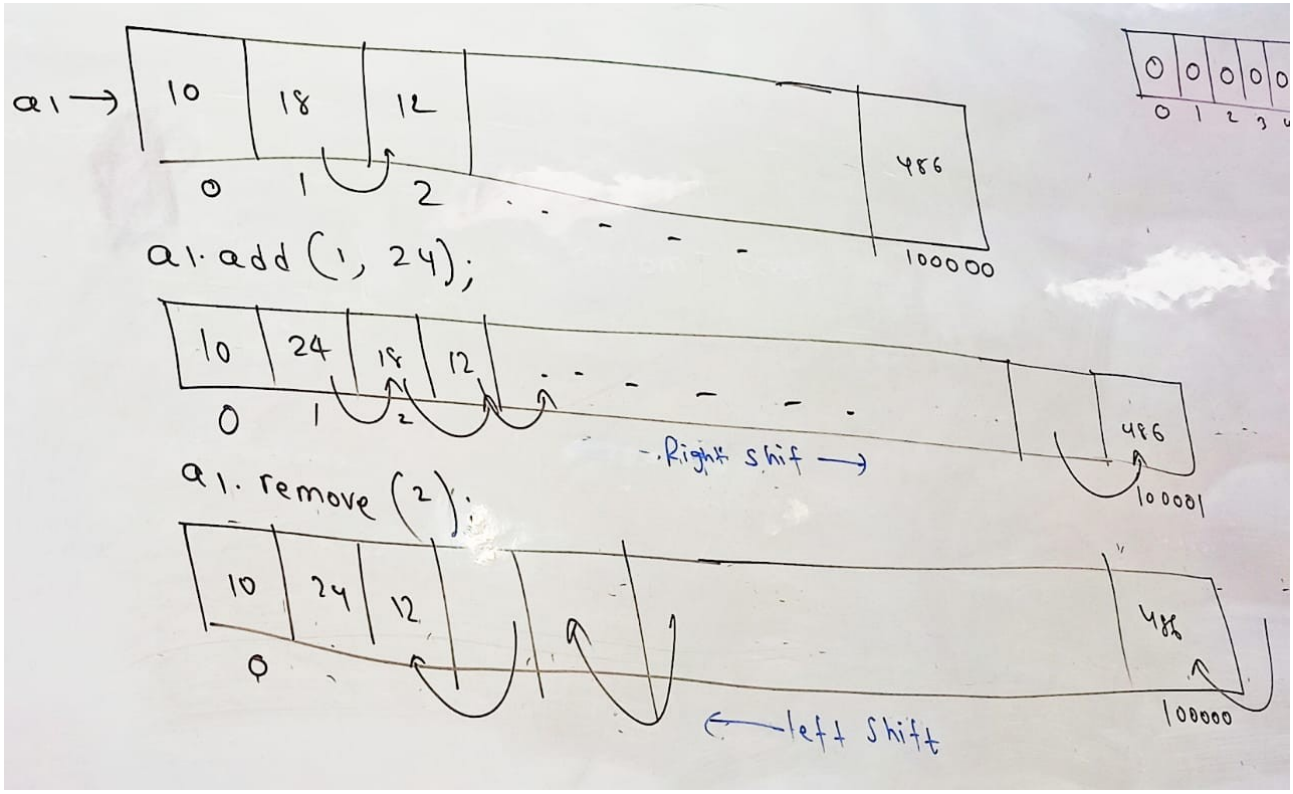
By this Arraylist we can provide the size to an arraylist created.

c. ArrayyList a3 = new ArrayList(Collection c**);**

By this constructor we can add a collection of elements directly inside ArrayList

important
Advantage
5. Array list is recomended if over operation is search operation or access operation. It provides fast search operations because it has indexing and it implements RandomAccess Interface for better accessiblity.

Disadvantage
6. BitShift:- In array list when an element is added or removed then all the remaining elements will be either shifted one place to its right side or to its left side. Which is called **Bit shift** in array list. This bitshift is a slow operation so array list is not recommended if our operation is modification (insertion or deletion operation).



28-01-2023

* Linked list *

Linked list is a class which is available in collection framework of which implements List interface. Linked list internally uses node as the data structure to store the group of individual objects.

Node

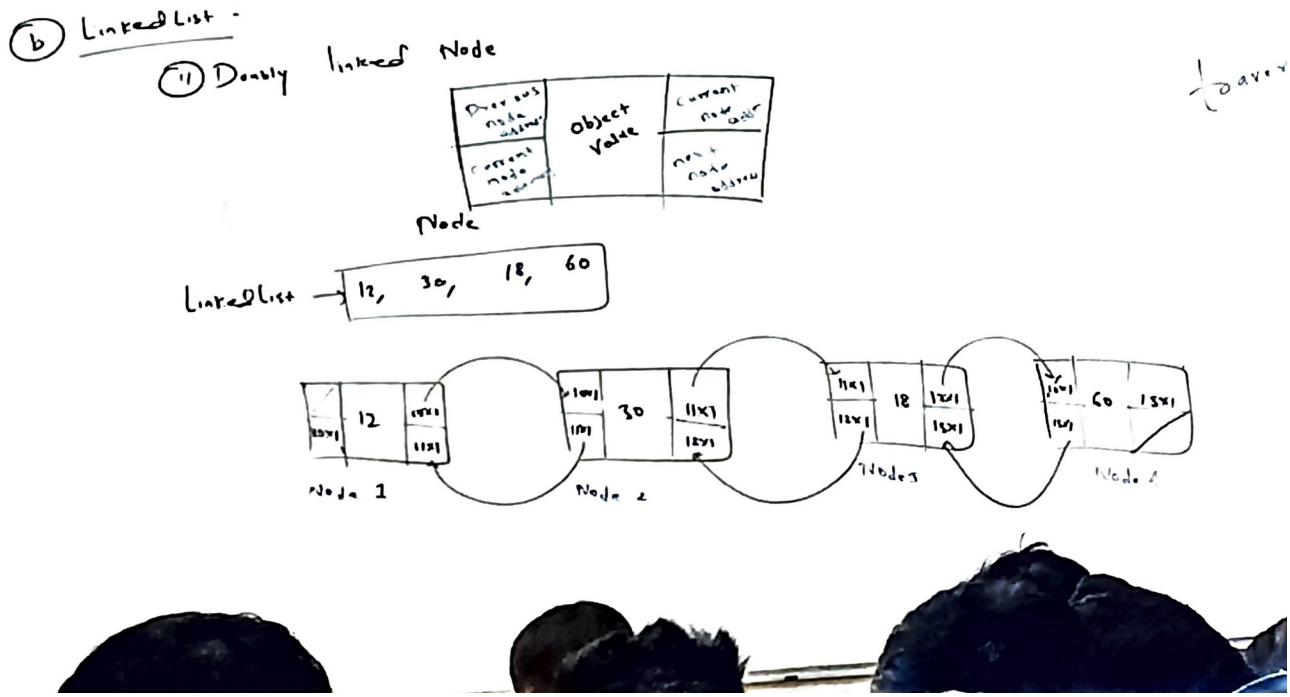
Node represents a block of memory which contains object value and address of the node. Node is of two type:

1. Singly Linked Node
2. Doubly Linked Node

Singly Linked Node

Singly Linked Node is a block of memory which contains object value and address of the previous node as well as address of the current node.
In singly linked nodes we can traverse only in forward direction.

Address of previous node	OBJECT VALUE	Address of current node



Doubly Linked Node

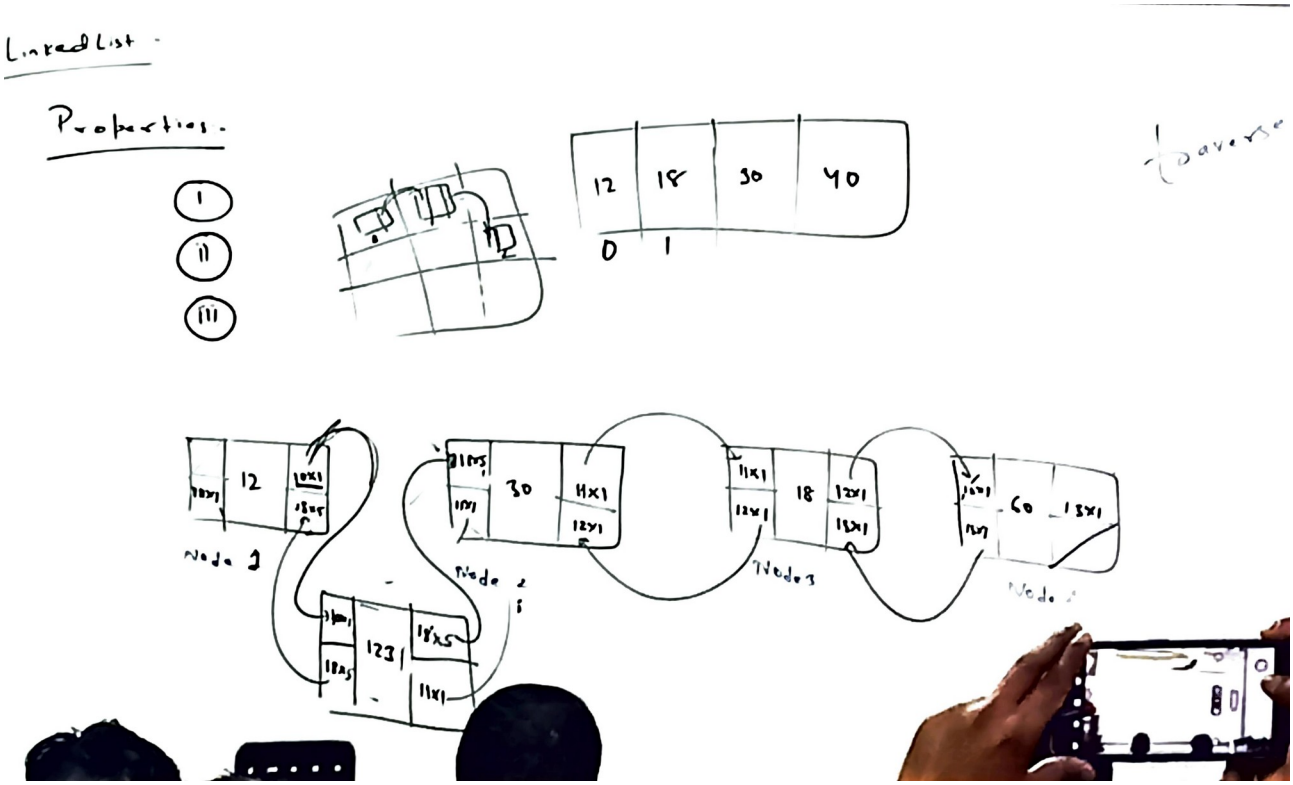
Doubly linked node is used to store the object value along with that it stores previous node address , current node address and next node address.
By using doubly linked node We can traverse forward as well as backward direction.

Previous node address	`OBJECT VALUE	Current node address
Current node address		Next node address

Linked list internally uses doubly linked nodes to store group of individual objects.

Properties of linked List

1. Duplicate members are allowed to be added into linked list.
2. Insertion order of the elements will be maintained.
Imp.
3. In node elements are only accessed either from first node or from last node one by one. So when elements are stored in linked list then its performance is slow if our operation is to access the elements.



Imp
4. In linked list modification operation (**Insertion or Deletion**) is faster because we can easily add or delete a node from anywhere. This addind or removing node does not effect any other remaining nodes so linked list is recomended if our operation is insertion or deletion.

Question: Diffence between Linked List and Array List ?
Answer:
1. Array list implements list interface but linked list implements list as well as queue inteface.
2. Array list internally uses dynamic array to store group of objects in the form of continuous block of memory. But linked list internally uses node as the data structure to store group of objects in the form of Random block of memory.
3. Array list is fast if our operation is to access the elements but linked list is slow if our operation is to access the eleements.
4. Arraylist is slow in insertion or deletion operation but linked list is fast in insertion or deletion operation.
5. If our operation is access then array list is recomended but if our operation is to add or remove then linked list is recommended.

Constructor of LinkedList:

a. LinkedList l1 = new LinkedList( );
we can use genrics here to use index in linked list.
b. LinkedList l2 = new LinkedList(Collection c);

VECTOR

Vector is similar to arraylist with just one difference i.e. Vector is synchronized and arrayList is not. So Vector provides thread safety

Vector is a class which implements list interface of JAVA. Vector uses **dynamic array** as the internal data structure to store the objects.
Vector in JAVA is available since the initial release of java so it is called **legacy class** of collection framework.

Constructor of Vector

a. Vector v1 = new Vector();

By this constructor a dynamic array is created with size 10, when 70% of its size is occupied then new arrayList is created which will be double in length as compared to old size.

b. Vector v2 = new Vector(initial Capacity);

By this constructor we can provide initial capacity to vector.

c. Vector v3 = new Vector(Collection c);

This constructor can accept directly other collection objects.

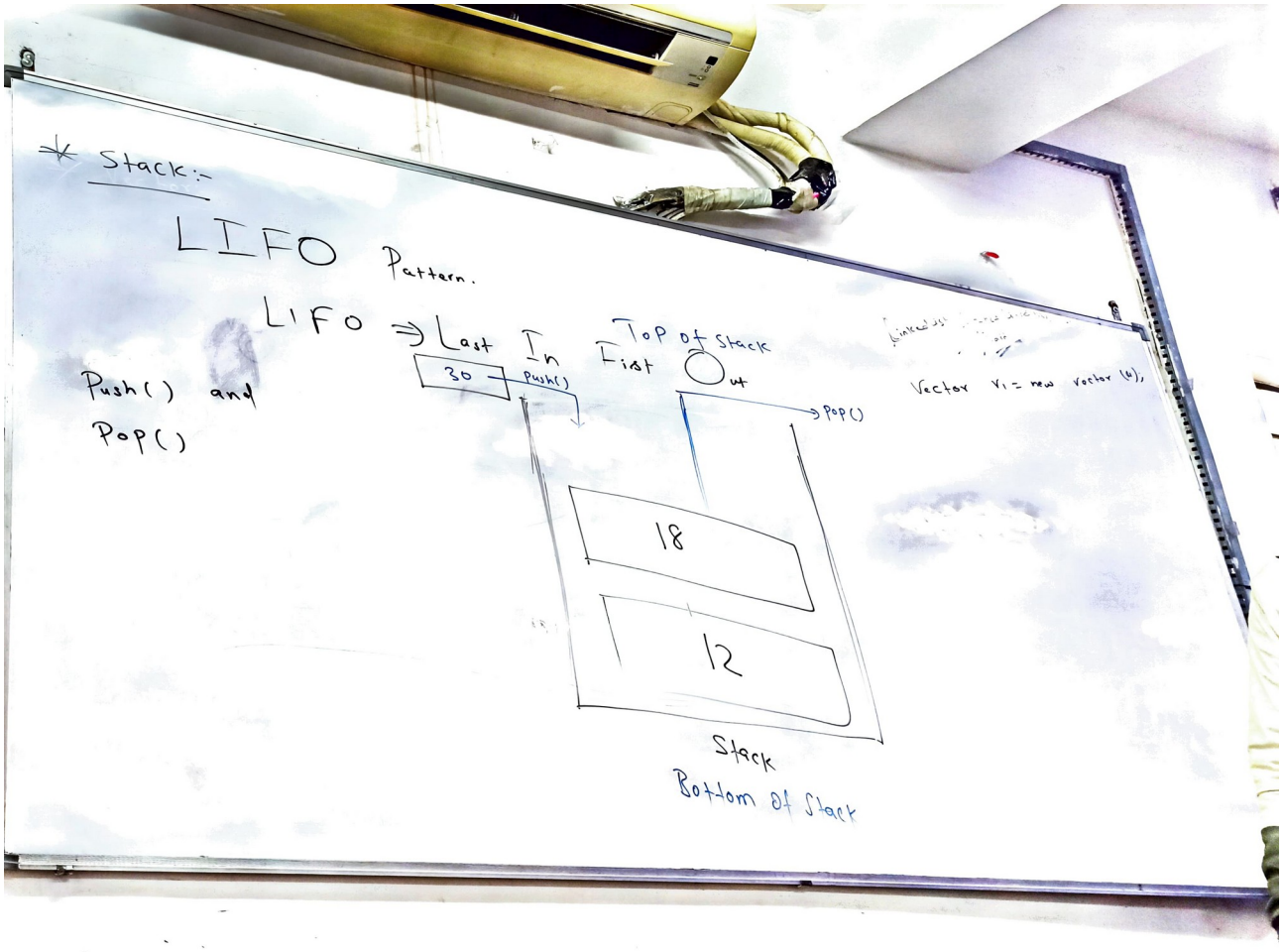
d. Vector v4 = new Vector(int initial capacity, int size increament);

By this constructor we can give initial capacity as well as we can provide the increament size.

Question. Difference between ArrayList and Vector?
Answer:
1. Arraylist is not a legacy class but vector is a legacy class in collection framework.
2. ArrayList is not synchronized but vector is synchronized.
3. ArrayList is not recommended is not recomended for thread safe enviroment but vector is recommended for thread safe enviroment.
4. ArrayList provides better performance but vector is slow in performance.

STACK

Stack is a class of collection framework which extends vector class of java. Stack is also the legacy class of java which is available since initial release of java. Stack works on **L.I.F.O** (**L**ast **I**n **F**irst **O**ut) **Pattern.** Stack works on a mechanism where elements are added one over another and the last added elements will be removed at first. The main feature of stack is **Push() and Pop()** operations.



Constructor of Stack

a. Stack s1 = new Stack();

SET