24/5/2025

Java.

* Java is object oriented prg Language
* Four pillers
  - Encapsulation
  - polymorphism
  - Inheritance
  - Abstraction
* Java both compiled as well as interpreter
* JVM → convert byte code into machine code.
* Java platform independent.
* Java is architecturally neutral which mean it can run anytime and anywhere.
* Java is robust. handle exception
  . handle memory efficiently
* Java performs multithreading
  → multiple task at a same time
* JDK → Java Development Toolkit is used to execute java program.
* JDK is a tool which consists of Java compiler, interpreter, applet viewer, and other tools.

History of Java

→ developed by James Grossling

sunmicro system

→ Oak. initial name for Java

Initial browser.

Internet explorer, Safari old name

→ in all e-device java is used initial like nokia,

⇒ Valid identifier. * start with -, alphabets, $ * No keyword.
* case-sensitive
* but it contain numbers
* No space blw identifier

⇒ class name always start with uppercase letter

else it show warning

⇒ ~~I/O~~ etc.

⇒ you need to save the file as same as classname

(i.e) main class name.

Structure of Java prg

```
class <Classname>
{
 public static void main (String args[])
 {
 //statements;
 }
}
```

public → main method accessable by any other class

static → main method associated with the class where it is declared

void → main method does n't return anything

main → name of the method which can execute at first

String args[] → accept array of strings from the command line

Java keywords - reserved keywords (48)

abstract	Do	implements	short
boolean	Double	import	static
break	else	Int	super
byte	extends	Interface	switch
case	final	Long	this
Catch	finally	new	throw
char	float	private	try
class	for	Protected	true
Continue	false	public	void

Variable declaration

int → int a = 5;

float → float b = 5.6f;

(or)

float b = (float) 3.14;

char → must enclose in single quotes and it does nt contain more than one character

Note
* float precision is limited (~7 decimal digits)
* double precision is higher (~ 15 decimal digits)
* double more accute than float

Input / output

For getting input at runtime you need

`import java.util.Scanner;`

```
import java.util.Scanner;
Getting number
Class Main {
    P.S.V.M (String[] args) {
        Scanner sc = new Scanner (System.in);
        System.out.println ("Enter Employee no: ");
        int eno = sc.nextInt (); //int I/P
        S.OP("Emp.no:" +eno);
    }
}
```

Enter Employee no:
5

Getting String

```
String ename = sc.next (); // Single word
(or)
String ename = sc.nextLine () // entire line
```

Getting Char

```
char gen = sc.next ().charAt (0); //char
```

Getting float

```
float height = sc.nextFloat (); // float
```

Getting Double

```
double weight = sc.nextDouble (); Double
```

Getting Boolean → nextBoolean ()

Typecasting → converting one datatype into another datatype

(i) Implict typecasting → done automatically

```
Int intialVal=100;
double doubleval = intval; // implicit (int to double)
```

(ii) Explict typecasting → require manual cast

```
double doubleval=99.99;
int intval = (int) doubleval; // explict (double to int)
```

Type conversion Precedence (highest to lowest)

double → float → Long → int → char → short → byte

Arithmetic operators (+, -, *, /, %, ++, --)

→ to perform numerical calculation

operator precedence

Swaping two number without temporary variable

```
num1 = num1 + num2;
num2 = num1 - num2;
num1 = num1 - num2;
```

```
a = a ^ b;
b = a ^ b;
a = a ^ b;
```

(compiler convert int to binary and then it perform operation and then convert to desired form

```
a = a * b;
b = a / b;
a = a / b;
```

Relational operation / Comparison operator [==, !=, <, >, <=, >=]

Logical operator (&&, ||, !)

bitwise Logical operator (&, |, ~, ^, <<, >>, >>>)

XOR, Not/complement, left shift, right shift, unsigned right shift

~a = -(a+1) ~a = -(30+1) ⇒ -31

a << 3 ⇒ a = 30

30 x 2 = 60 x 2 = 120 x 2 = 240 ans

a >> 2 ⇒ a = 30

$\frac{30}{2} = \frac{15}{2} = 7$ ans

Practice question

Swap two number with and without 3rd variable

Area of circle

type convertion int to float,
float to int,
int to char,
char to int

Conversion Type	Direction	Follow precedence?	Cast needed?	Risk
Implict (Widening)	Lower → higher	☑ Yes	☒ No	☒ No
Explict (Narrowing)	Higher → Lower	☒ Not based on order	☑ Yes	⚠ Yes (data loss, overflow)

Lower ~~double → float~~ → higher

byte → short → char → int → long → float → double

Farhenit to celcius problem.

$$celcius = \frac{5}{9} \times (Fahrenheit - 32) \quad | \quad Fah = \left(\frac{9}{5} \times C\right) + 32$$

in java I need to give $C = \frac{5.0}{9} \times (F - 32)$ & $F = \left(\frac{9.0}{5} \times C\right) + 32$

5/5/25

Assignment operator (=, +=, -=, *=, /=, %=, &=, |=, ^=)

Conditional operator / Ternary operator (condition)? value if true : value if false;

~~Bitwise operator~~.

Control Statements

(i) Conditional statement

simple if and else

```
if (condition) {
  // code
}
else {
  // code
}
```

else if / ladder if

```
if (condition1) {
  // code
}
else if (condition2) {
  // code
}
else {
  //code
}
```

switch

```
switch (expression) {
  case x:
    //code
    break;
  case y:
    //code
    break;
  default:
    // code
```

(ii) Unconditional statement

while

```
while (condition) {
  // code
}
```

do-while

```
do {
  //code
}
while (condition);
```

for

```
for (initialization ; condition; increment/dec) {
//code
}
```

for each

```
for (type variableName : arrayName) {
  // code
}
```

~~Switch~~

eg)
```
int num = 5;
if (num > 0)
  System.o.p("positive");
S.o.p ("Greater than zero"); // This is always executed
```

Best practice to use { } curly brackets

eg)
```
String[] colors = {"r", "g", "b"};
for (String color : colors) {
  S.o.p (color);
}
```

```
int[] nums = {1,2,3};
for (int n : nums) {
  S.o.p(n);
}
```

o/p	o/p
r	1
g	2
b	3

Note → no. of iteration known in "for" Loop

→ no. of iteration not known in "while" Loop

26/05/2025 Practice

```
if (10 < 20);
    S.O.P("False");
```
⇒ It print false even if the condition fails.

```
if (1 == 1.0)
    S.O.P("False");
else
    S.O.P("True")
```

ASCII

American Standard code for information interchange

Printing A to Z

```
for(char ch='A'; ch<='Z'; ch++){
    system.out.println(ch);
```

A B C D ... E

A - 65
Z - 90
a - 97
z - 122

0 - 48
9 - 57
Space - 32

Armstrong number

153 → $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$ (end of string)

It is armstrong_number

123 → $1^3 + 2^3 + 3^3 = 1 + 8 + 27 = 36 \neq 123$

strong number

145 ↑ 1! 4! 5! ⇒ 1 + 24 + 120 ⇒ 145

123 ↑ 1! 2! 3! ⇒ 1 + 2 + 6 = 9 ≠ 123

perfect number

6 → divisible of 6 is 1, 2, 3 → 1+2+3 ⇒ 6 (perfect number)

12 → 1, 2, 6, 3, 4 ⇒ 16 ≠ 12 (Not perfect number)

→ Printing prime number

Pattern

3
0 0 0
1 1 1
2 2 2

3
0 1 2
0 1 2
0 1 2

3
1 2 3
4 5 1
7 8 9

3
A B B
B A B
B B A

3
A
A
A

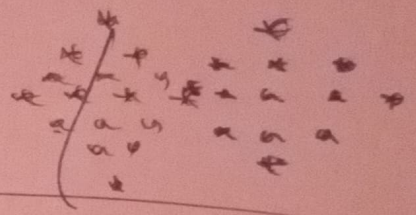
Pascal triangle

3
*
* *
* * *

3
*
* *
* * *

3
* * *
* *
*

3 , diamond.



Loop Control Statement
break, continue

Practice

printing A-Z without vowels

27/5/25

Array

* Array is collection of similiar data which is stored in sequencial order
* Array are static

Rule of array.

1. array name must be valid identifier.
2. Array size are the subscript always be a integer value. It should never be negative.

(eg) datatype array_name[] = {value};

(or)

datatype[] array_name = {value};

(or)

datatype array_name[] = new int[size];

→ Sum of the array
→ Highest and Lowest number in aarray
→ Counting a vowels in a array in character array and string array.
→ Ascending and descending order in array.
→ 2D - matrix addition, sum of number in a matrix,
→ Matrix transpose.

Practice

28/5/25 2D-array

Printing sum of diagonal

```
for(int i=0; i<size; i++){
    sumD1 += matrix[i][i];
    sumD2 += matrix[i][n-i-1];
}
```

1	2	3	sumD1 = 15
4	5	6	sumD2 = 25
7	8	9	

Reverse each string in array

i/p {"welcome", "to", "java"}

o/p {"emocleW", "ot", "avaj"}

29/5/25 power of a number using Recursion.

i/p → pow(2,4) o/p = 16

```
class Main{
    int fact
```

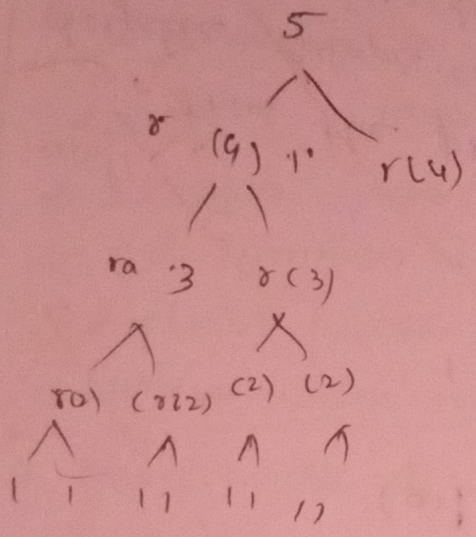
* if you call from static to non-static you need to create a object.

```
class Main{
    int fact(int n){
        // code
    }
    P.S.V.M(String[] args){
        Main m = new Main();
        S.O.P(m.fact(5));
    }
}
```

* object not needed for

Calling method	called method
Static	- Static
non-static	- non-static
non-static	- Static

fibbonaci using recursion -> next page



30/5/2025

Function

Set of instruction that perform a task can be executed at any time. The block of code only execute when it is called.

Syntax

```
access-specifier static returntype MethodName (argument list)
{
    Local variable declaration;
    Executable part;
    return (expression);
}
```

Types of method declaration

i) with argument with return type
(ii) with argument without return type
(iii) without argument with return type
(iv) without argument without return type.

OOPS

Class and objects

Why mostly a class is created without access specifier in java.

→ because java supports package-level encapsulation.

→ It's common well structured codebases to restrict acess using the default modifier unless wider access is needed

---

→ basic code with public and private specifier

Practice

1. Find sum of array using recursion.

31/5/25

* Getting emp no and emp name in another class
* Getting empno and emp name using array in another class

Method overloading

Same method name but different argument

Note:- no. of argument, sequence of the argument and type of argument should be distinct

```
class calcus{
Arith(int a, int b) {
    return a+b;
}
Arith(int a, int b, int c){
    return a+b+c;
}
}
class main{
p.s.v.m (string[] arg){
calcus c= new calcus;
S.OP (Arith(4,5));
```

Constructor

Special method : it called automatically when an object is created

1. constructor name and class name should be same
2. Constructor must be public.
3. Constructor must have no return type
4. arguments passed are optional
5. It can also be overloading

(i) Default constructor (No argument)

(ii) parametarized constructor

Practice session

→ reverse the string using recursion

→ reverse the digit using recursion

→ Binary value for the given number using recursion

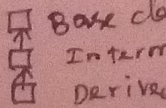
Inheritance

16/2025

[one class inherits the properties and methods from another class]

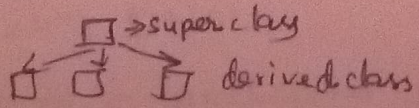
→ single Inheritance

→ Multilevel

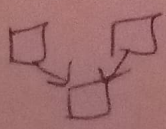


sum → base class
diff → immediate class
div
mul → derived class

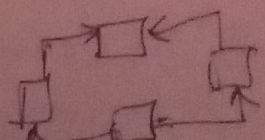
→ Hierarchical



→ multiple



→ Hybrid.



Super keyword

→ used to access the members of super class inside the subclass

* Parameter constructor in inheritance
* non-parameterized constructor in inheritance.

Method Overwriting overriding

A subclass (or) child class implements a method that is already defined in the superclass or base class.

final method

cannot be overriden

cannot be overriden.

Abstract class

→ Abstract class cannot be initiallized directly.
→ It declared with Abstract keyword.
→ It serve as a blueprint for another class. providing a common base with shared functionality and it require subclass to implement specific method.
→ Abstract class contain abstract method.

Interface

An Interface is used to define constant and abstract methods, which are implemented by a class.

2/6/2025

try... except block.

finally → it is always executed wheather the exception throwons or not.

throw → is used to through an exception.

throws

User-defined exceptions

String methods

Time and work

pipes and cistern

work equivalence

Number (Divisibility, LCM, GCD, HCF)

Percentage increase/dec

S.I

C.I

Verbal

Reading Comprehension
Sentence correction
Sentence completion
parts of speech
Vocabulary.
Voices & speech.

Verbal reasoning
Coding Decoding/Number Series/
Data arrangement
Blood relation
Logical connectives.
Data Sufficiency.
Clocks/calander.
Logical connectives tick & cross method

data interpretation & Sufficiency

Set theory
Algebraic expressions

Complex

Easy

AVG
Time, speed Distance
Prob train, Age
Profit Loss
Permutation combination
Ratio proportion
Area.

probability
Discount
Clock & Calandar
Height and distance
Volume & Surface