Java

→ popular & widely used

→ Object oriented (however not purely object oriented as it supports primitive data types)

→ When every data is represented in form of ~~data~~ object is called Object oriented (100%)

100% OOP language ex is HTML (rest C++, Python, Java are semi object oriented)

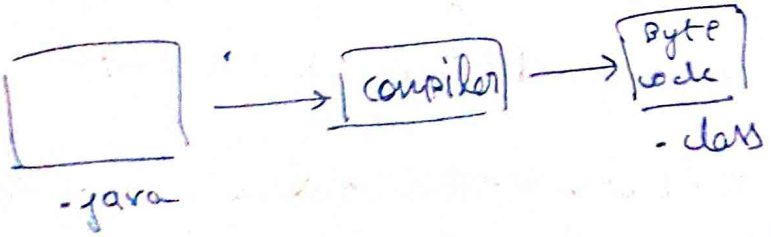
→ Compilation & Execution

Process of Java program

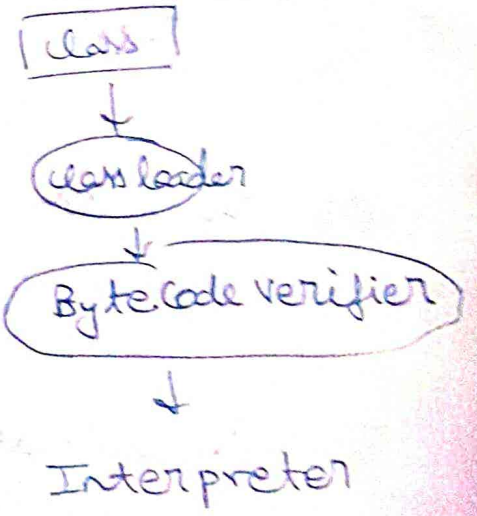
Compile time

java code → Byte Code

java compiler



Runtime



* Java codes compiled to byte code (machine independent code)

byte code runs on JVM

Java syntax similar C / C++ but does not provide low level features like pointers

Java when compared to C++, are more maintainable because Java does not allow many things which may lead to inefficient programming

---

Introduction

→ Java is class-based, object oriented programming language

→ Write once run everywhere compiled java code can run on all platforms that support Java.

James Gosling → Father of Java

Terminologies

Java Virtual Machine → Execution process

written → compile → run

compilation is done by JAVAC compiler

↑ included in JDK

JVM executes bytecode generated by JAVAC compiler.

Java as platform independent language

Each OS has different JVM but output they produce is same across all OS.

JDK (Java Development Kit)
include compiler (JAVAC), JRE (Java runtime env), java debuggers, docs

JRE → it allows Java program to run
includes, browser plugins, applet.

Garbage collector → Java programs can't delete objects therefore to delete JVM has Garbage C
∴ helps in memory management.

Features of Java

1. Platform independent →
2. OOP language → collection of objects (instance of class)
   - 4 features
     - ↳ Abstraction
     - ↳ Encapsulation
     - ↳ Inheritance
     - ↳ Polymorphism
3. Robust → reliable (errors check as early as possible)
4. Secure → Array out of Bounds Exception (flaws stack corruption)

Distributed → easily distributed over connected networks.

Multi-threading → concurrent execution of 2 or more parts for max utilization of CPU.

Portable →

// Single line comment

/* multi line comment */

import java.io.* → import all classes of iopackage

for input output streams

class → contains data & methods

Static → keyword tells us that this method is accessible without instantiating the class.

void → method will not return anything

System.in → standard input stream

Setting up env. in Java

1. JDK → includes development tools
2. JRE → contains part of Java libraries
3. JVM → runtime for Java Byte code to be executed

Java Basic Syntax

Object → is an instance of a class have behaviour and state.

Class → blueprint of objects.

Method → behaviour of an object

Instance variables → unique set (state of an object)

Note → program file name should exactly match the class name

Class Names → 1st letter Capital
↳ Camel Casing

public static void main (String [] args)

↳ Java program processing starts with method main()

Methodnames → start with Lowercase ch. 5)
↳ Camelcasing.

Identifiers → any letter A to Z a, to z or _ (und. score)
↳ keyword cannot be used.

White spaces → blank line (can have comments)
compiler ignores it

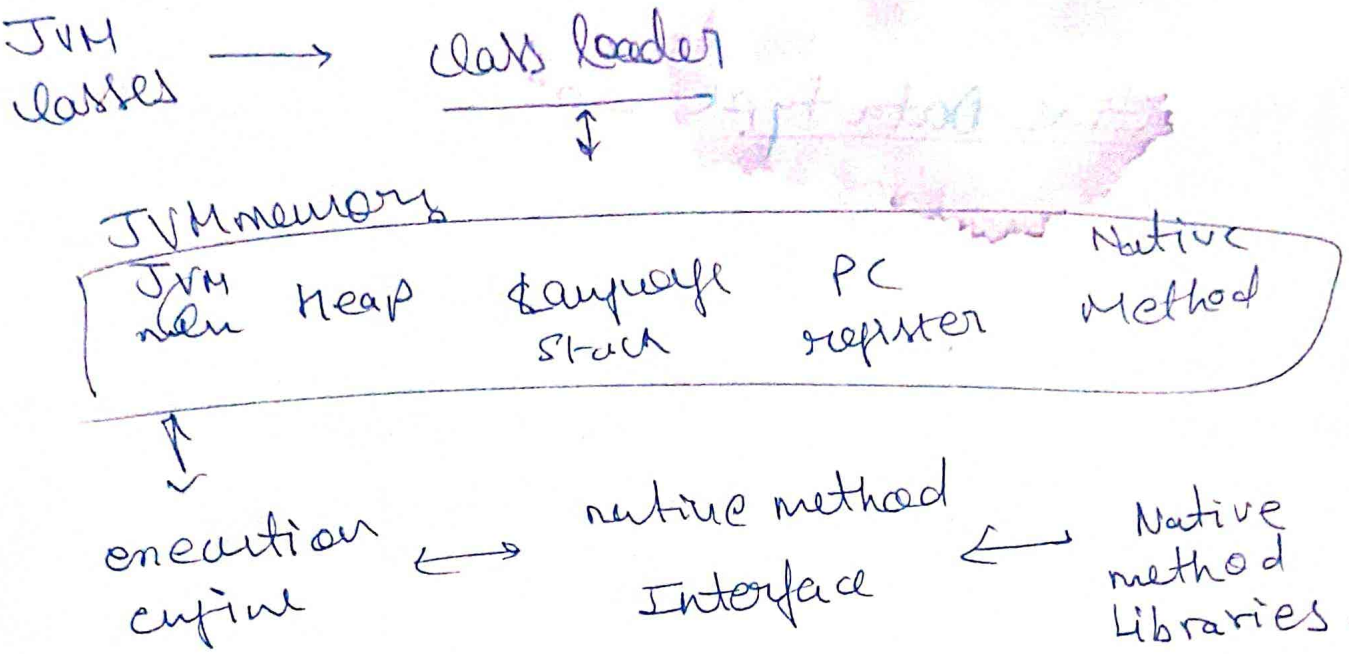
AccessModifiers → control scope of class & methods.
default public protected private.

keywords → predefined words (reserved words)
special meaning to Java compiler.

JVM Architecture

JVM is part of JRE

WORA (Write once run Everywhere)



Data types in Java

There are majorly 2 types of languages

↳ Statically typed language (once declared of one data type cannot hold values of other data type) ex C C++ Java

↳ Dynamically typed languages - different data types over time. ex Ruby, Python.

* Java is statistically typed & strongly typed language.

Java has 2 categories of data

↳ Primitive data type - boolean, char, int, sho byte, long, double and float

↳ Non-primitive - String, array

Primitive Data type - single values with no special capabilities

type	Syntax	Size	values	Default value
boolean	booleanname	virtual machine dependent	True false	false
byte	byte name	1byte (8 bits)	-128 to 127	0
short	short name	2byte (16bits)	-32768 to 32767	0
int	int var	4byte (32bit)	-2^{32} to 2^{31}	0
long	long var	8byte (64bit)		0
float		4byte (32bits)	upto 7 decimal digits	0.0
double		8byte (64bits)		0.0
char		2byte (16bits		

Why char is 2byte in Java?

In other languages C/C++ (uses only ASCII characters) & to represent all ASCII 8 bits (1byte) is enough.
But Java (uses Unicode system)
↳ represents most of the world's written languages.

Non-primitive data type or Reference data type

contain memory address of variable value because reference types wont store the variable directly in memory ex- String, objects, arrays

String → array of characters

```
String s = "hi";
```

using new keyword

```
String s = new String ("hi");
```

Class → set of properties & methods that are common to all objects

Object → real life entities

Interface → like a class, interface has properties & methods but methods are by default abstract- specify what a class must do but not how.

Array

work differently in Java as they do in C/C++

→ Arrays are objects in Java (∴ find their length using length) whereas C/C++ we use size

→ superclass is Object

→ dynamically allocated

Variable in Java

basic unit of storage

type name;

→ local variables → within a block
(destroyed when call returns)
(initilisation of is mandatory)

→ instance variables → non-static, declared in class outside any method.
(created as object is created & viceversa)
To access them object has to be created.

→ Static variables → like instance but with static keyword
classname.variable_name (without making object)

Comments
→ Single line //
→ Multi line /* */
→ Documentation comments
/** comment start *

comment end/

Operators in Java

Arithmetic → *, /, %, +, −

Unary → ++
−−
!

Assignment → =

[Compound Statement → a=a+5; ↓ a+=5]

Relational → ==
!=
<
>
>=
<=

Logical → &&
||

Ternary → condition ? true : false

Bitwise → &, |, ^, ~

Shift → << left shift
→ >> right shift } sign fills 0 in void places leftmost on sign
→ >>> unsigned

→ precedence then associativity

Ways to read input

① Buffered reader class

```
import java.io.*
BufferedReader name = new BufferedReader(new
InputStreamReader(System.in));
name.readLine();
```

② Scanner class (preferred)

```
Scanner SC= new Scanner (System.in);
SC.nextLine(); // String
SC.nextInt(); // integer
SC.nextDouble(); // dauble
SC.nextFloat(); // float
```

③ for competitive coding →

parseInt converts string into integer

Decision Making (if, if-else, switch, break, continue, jump)

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

```
nested if
if (—)
{ if(—)
  {
  }
}
```

```
ladder
if(——)
{ }
else if (——)
{
}
else
{
}
```

witch case →

```
switch (expression)
{ case value1:
        ---------
        break;
  case value2:
        ---------
        break;
  default:
        ---------;   ] optional
}
```

JUMP statements

Break → exit a loop
can use as goto statement break second;

continue → control to top of the loop.

return →

LOOPS

① while loop → repeating if statement

while (condition)

{

}

(also called entry control loop)

② for loop → easy debugging than while

for (initialization, condition, updation)

③ ~~expand~~ enhanced for loop / for each loop

for (Type name : Collection)

{

Statement

}

④ do-while

do {

} while (condition

1st do once then check

pitfall of loops → Infinite loops

→ out of memory

Java

↳ widely used programming language & platforms

Enums - Enumerations → purpose of representing a group of named constants

→ define our own data → constructors
→ variables
→ methods

→

Syntax

```
enum Color {
    RED, GREEN, BLUE
}
```

] outside class

```
Color C1 = Color.RED;
```

class Box

Box B1 → [ ]

B1 = new Box(); → [ ] → width, height, depth

reference: Box B2 = B1; B2 → [ ]

B2.height = 20;

B1 → [ ] — height = 20

B2 →

Addition & concatenation

2+0+1+6 + "uttu" → 9uttu

"uttu" + 2+0+1+6 → uttu2016

2+0+1+5 + "uttu" + 2+0+1+6 → 8uttu2016

2+0+1+5 + "uttu" + (2+0+1+6) → 8uttu9

Conditional operator → numeric type promotion

```
Object O1 = true ? new Integer(4) :
                   new Float(4.0);
```

4:0

Scanner Class

① Scanner SC = new Scanner(System.in);

② To read number(numerical data types XYZ)

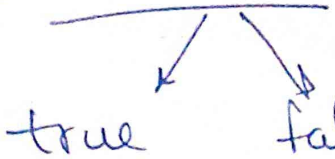
```
SC.nextXyz(), ex
sc.nextShort()
SC.nextInt()
SC.nextDouble()
```

③ To read strings SC.nextLine();

④ single character sc.next().charAt(0).~~next()~~;

check if value is of certain type

hasNextXYZ() is used



or hasNextInt()
hasNextLine()
hasNext().charAt(0)

Command Line Arguments

cmd> Hello.java

java Hello Utkarsh Shukla

- Hello: name of class running the program
- Utkarsh Shukla: command line arguments
- JVM wraps & supply to args[]

```
public static void main (String args[])
{
    if (args.length > 0)
    {  for (String val : args)
       { System.out.println(val);
       }
    }
```

Output Utkarsh Shukla

I/O Stream

Stream - sequence of data

Input Stream - reads data from source

Output Stream - writes data to destination

Character Stream

Characters stored as Unicode

Note → In Scanner if we call

① nextLine() after any nextXYZ()

② then it does not read value & skip that step.

Solution

use next instead of nextLine();

Output Formatting

printf → takes multiple arguments unlike print, println.

```
System.out.printf("%-d, %-f, %-c, %-C, %-s, %-S, %n", {5, 5.0, a, A, ba, BA.  change line});
```

Type Conversion in Java

Automatic conversion (widening)

① Smaller to bigger

② Byte → Short → Int → Long → Float → Double

③ ex- `int i = 100; long l = i;`

Explicit conversion (Narrowing)

① Double → Float → long → Int → short → Byte

② Bigger to smaller

③ ~~`char ch = 'c';`~~ ex- `double d = 100.04;`
`long l = (long) d;`

Note → char and no. are not compatible

Type promotion

① While evaluating an expression it automatically promotes ans to higher datatype.

Widening primitive conversion

"Y" + "O" = "YO"

↳ treated as String

+ ↳ concatenated

Note

~~'Y' + 'O'~~

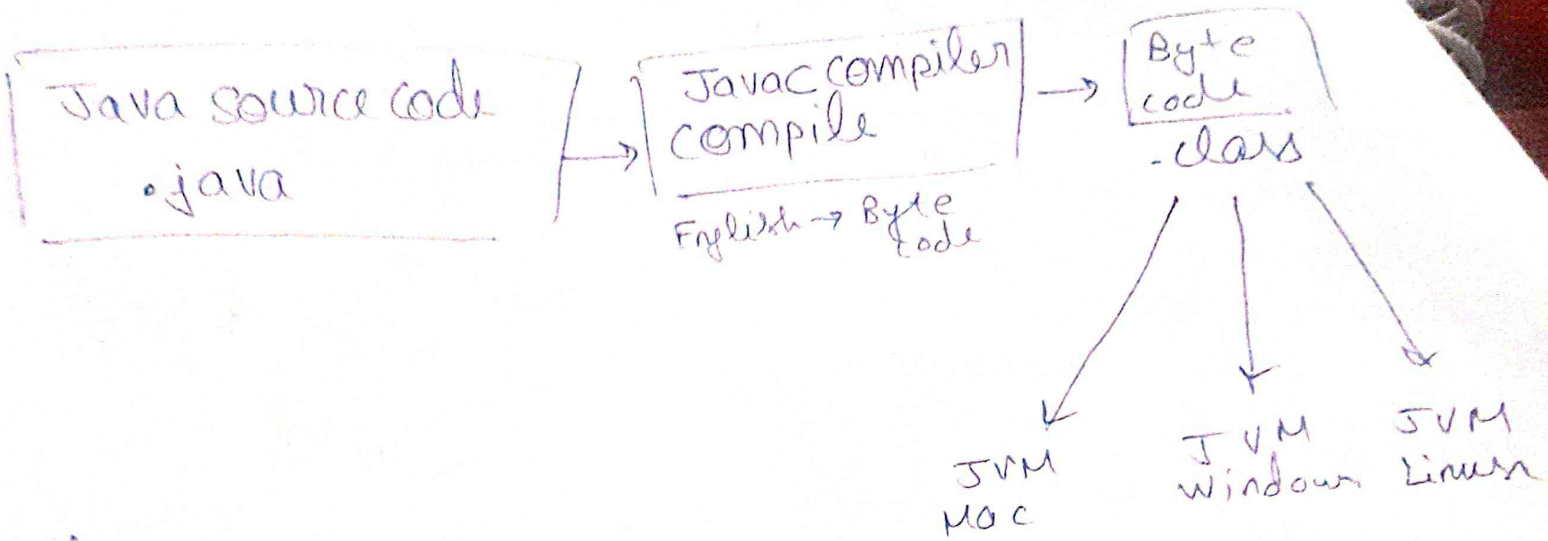
'L' + 'O' = 155

↳ characters converted to int

L is 76
O is 79

↳ IDE (Integrated Dev ENV.) → debug, mat... workflows.

How Java Works?



Note:

main is an identifier not a keyword

String [] args

↳ command line argument

Java Program 1 2 3 — args

JDK → compiler (convert to Byte code)
→ JVM (run Byte Code in OS)

1. Write the Program
2. Save with .java extension
3. javac program_name.java
4. .class file formed
5. java program_name

handled by IDE

Project → package → class

Data Types & Variables

Data types (Primitive)

Integer type
- byte
- short
- int
- long

~~Float~~
Decimal type
- float
- double

Character type
- char

- boolean

Binary System

1 Byte = 8 bits outcomes 2^8 → 0 to 256 range
→ -128 to 127

for n bits range 2^n
outcome -2^{n-1} to $2^{n-1}-1$

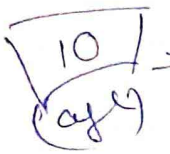
1 Byte = 8 bits
1 Short = 2 Byte
1 int = 4 Byte
1 long = 8 Byte

use as per size to save memory

variables

→ Data type variable_name = value ;

int age = 10 ;



(memory name)

Type casting (conversion of primitive data type to another)

Widening (Implicit) Automatic

byte → short → int → long → float → double

char → int

Narrowing (Explicit) Manual

boolean

double my = 3.14 ;
int mx = (int) my ;

User Input

→ use Scanner class import java.util.Scanner ;

Scanner SC = new Scanner (System.in);
↳ read from keyboard.

int a = SC.nextInt();
.nextDouble();
.nextFloat();
.next() → read 1st word (till space)
.nextLine() → read entire line

Taking String after int not possible

int L,
int ⑤ —— SC.nextInt()
String —— SC.next()
String → your input → SC.nextLine()

Arrays (Data Structure) - Storage of data in memory

Multidimensional array →
int b[][] = { { }, { }, { } };
2D-array
Index out of bounds error
different lengths

Matrix → ImageProcessing

Sorting → Bubble
→ Selection →

Bubble sort

[3, 2, 1, 5, 4]

2, 3, 1 5 4
2 1 3 5 4
2 1 3 5 4
2 1 3 4 5

2 1 3 4 5 — last element biggest

Note → get sorted from back

(n-1)

```
for(int i=0, i<n-1; i++)
{ for(int j=0; j<n-1-i; j++)
  { if a[j+1] < a[j]
    { if // swap
    }
  }
}
```

Strings (Non-primitive data types)

String animal 1 = new String ("Dog");

String animal 2 = "Cat";

String animal 3 = "Cat";

String animal 4 = "Dog"

String → array

String.toCharArray()

Not in heap

Immutable

Not changed in memory.

Not changes but creates new

Garbage collection → when no reference remains.

==

Object reference (heap, pool)

.equals

↳ checks for value

→ String.isEmpty() → true / false

→ String.replace('A', 'C')

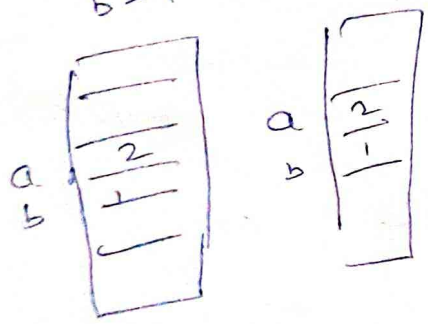
old new

→ String allCars[] = cars.split(";");

Anagrams (same no. characters with same repition)

aab, aba

a-2 b-1 | a-2 b-1



1) form array & check.

```
int a[] = new int[256];
int b[] = new int[256];
for (char c : a.toCharArray()){
    int index = c;
    a[index]++;
}
```

similarly for b

convert char to int

a ^ a = 0

take XOR should give 0.

Bubble sort

1-1 5 9 8 2 1 → 2 1 5 8 2 9 → 3 1 5 2 1 8 9 → 4 1 2 1 5 8 9

1-2 5 9 8 2 1 → 2 1 5 8 2 9 → 3 2 5 1 8 9 → 2 1 5 8 9

1-3 5 8 9 2 1 → 2 1 5 2 8 9 → 5 2 1 8 9

1-4 5 8 2 9 1 → 2 1 5 2 8 9

```
for(int i=0; i<n-1; i++)
{
 for(int j=0; j<n-i-1; j++)
 {  if(arr[j+1] < arr[j])
    {  swap(j, j+1);
    }
 }
}
```

(like bubbling)

worst case $O(N^2)$

Selection Sort

1-1 5 9 8 1 2 → 2-1 1 9 8 5 2 → 3-1 1 2 8 5 9

1-2 5 9 8 → 2-2 1 8 9 5 2

1-3 5 9 8 1 2 → 2-1 1 9 8 5 2

1-4 5 9 8 1 2 → 2-2 1 9 8 5 2

2-3 1 2 8 5 9

swap(i, m)

(select min & place)

$O(N^2)$

```
for(int i=0; i<n-1; i++)
{  int min = i;
 for(int j=i+1; j<n; j++)
 {  if(arr[j] < arr[min])
    {  min = j;
    }
 }
 swap(i, min);
}
```

Insertion sort

Date:
Page:

① 2 9 5 1 3

② 2 9 | 5 1 3

③ 2 9 5 9 | 1 3

④ 1 2 5 9 | 3

1 2 3 5 9

Har no. pe jaate hain aur uske sorted wale part me sahi jagah daate hain (reverse bubble)

already sorted check once only
∴ more optimized than bubble & selection

```
for (int i = 1 ; i<n ; i++)
{
 for (int j = i-1 ; j >=0 ; j--)
 {
    if (arr[j] > arr[j+1]
    {swap (j, j+1)}
    else{
      break;
      }
 }
}
```

worst
$O(N^2)$

Best
$O(N)$ for sorted

Merge Sort

7 4 1 3 6 8 2 5

7 4 1 3 → 1 3 4 7
6 8 2 5 → 2 5 6 8

merge 2 sorted
1 2 3 4 5 6 7 8

7 4 1 3 | 6 8 2 5

```
int[] mergesort(arr, lo, hi)
{
    int mid = (lo + hi)/2;
int a[] = mergesort(arr, lo, mid);
int b[] = mergesort(arr, mid+1, hi);

int ans[] = merge2sorted(a, b);
return ans;
}
```

```
if(lo == hi)
{ int[] base = new int[1];
  base[0] = arr[lo];
  return base;
}
```

Quicksort

```
    i  j
    ↓  ↓
    7 9 4 8 3 6 2 1
```

0 to j-1 < pivot
j to i-1 > pivot
i to end unknown

a[i] > pivot i++
increase in greater

a[i] < pivot
swap(i,j) i++, j++
< = increase
> shift

```
partition(arr, pivot)
{
int i=0;
int j=0;
while(i < arr.length)
{  if(arr[i] > pivot)
   { i++;
   }else{
      swap(i, j);
      i++;
      j++;
   }
}
```

0 — j-1 | Smaller or equal

j — i-1 | larger area

8 5 1 3 7 2 9 (6) ← last element as pivot

Date: Page:

↓

~~8 5~~
5 1 3 2 (6) 8 7 9
(1235) (789)
5 1 3 2 | 8 7 9

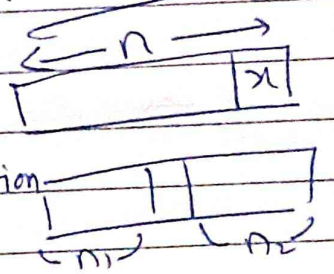
Note
pivot comesat right position

```
quickSort (arr, lo, hi)
{ if (lo >= hi)
    return;
  int pivot = arr[hi];
  int pi = partition (arr, pivot, lo, hi);
  quickSort (arr, lo, pi-1);
  quickSort (arr, pi+1, hi);
}
```

pi = j-1 in partition

Time Complexity

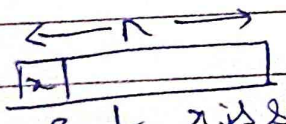
$T(n) = T(n_1) + T(n_2) + (n$ — partition



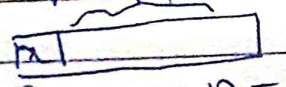
$\Rightarrow n_1 + n_2 + 1 = n$

$\Rightarrow n_1 + n_2 \simeq n$

Worst case

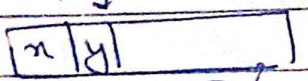


p ↓ x is smallest



$n_1 = 0$ | $n_2 = n-1$ ∴ sorted array

p ↓



$n \rightarrow$

$1+2+3+\ldots+n$

$\frac{n(n+1)}{2}$

~~Ans~~ $O(N^2)$

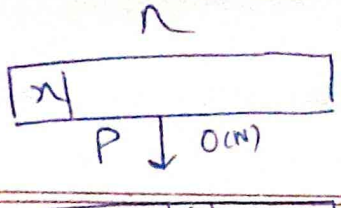
$T(n) = T(0) + T(n-1) + (n)$

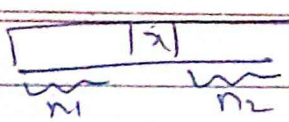
$T(n-1) = T(n-2) + (n-1)$

$T(n) = C\left(\frac{n(n+1)}{2}\right)$

$O(N^2)$

Best Case





$n_1 \simeq n_2 \simeq \frac{n}{2}$

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + C(n)$$

$$T(n) = 2T\left(\frac{n}{2}\right) + Cn$$

$$\boxed{T(n) = O(n \log n)}$$

Merge sort → $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + T(\text{merge}) \rightarrow O(N)$

$\frac{n}{2}$ $\frac{n}{2}$

② $T(n) = 2T\left(\frac{n}{2}\right) + T(\text{merge}(\frac{n}{2}, \frac{n}{2})) \rightarrow O(\frac{N}{2})$

2^{nd} $T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + 2T(\text{merge}(\frac{n}{4}, \frac{n}{4})) \rightarrow O(\frac{N}{2})$

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

$$T(n) = 2^n + O(N) + 2 * \frac{N}{2} + 4 \times \frac{N}{4} + \dots)$$

$$T(n) = 2^i T\left(\frac{n}{2^i}\right) + iN$$

~~$T(4) = 2 \times T(2) + 4$~~

ultimately let $\frac{n}{2^i} = 1$ $\quad n = 2^i$

$\log n = i \log 2$

$$\boxed{i = \log_2 n}$$

$i \rightarrow \log_2 n$

$$T(n) = n\,T(1) + n \log n$$

$$T(n) = n + n \log_2 n$$

$$\simeq O(N \log_2 N)$$

Count sort

→ Many elements but within a small range
ex- Rank list

Date:
Page:

① find min & max

② Freq array (size max-min+1)

9 6 3 5 3 4 3 9 6 4 6 5 8 9 9

0[3] 1[4] 2[5] 3[6] 4[7] 5[8] 6[9]
3 2 2 3 0 1 4

③ Prefix array of freq.

3 5 7 10 10 ~~11~~ ~~15~~
10 ~~14~~
12

first 3 position ~~have~~ fills 3 position & 4 | 5 ka ilaka kahan tak hai

Note position not index

33344

④ Loop main array from back
we get 9
check 9 in farr 15 is present as position
∴ index = 14
& decrease its position
8 9 9
11 13 14

next is 9 in farr ∴
next 8 farr 8 → 11

→ All this to make count sort stable
[(Jo pehle appear kar rha wo pehle Rare)]

```
int range = max-min+1;
int farr[] = new int[range];

for(int i=0; i<n; i++)
{
  int idx = arr[i]-min;
  farr[idx]++;
}
for(int i=1; i<range; i++)          // prefix sum
{  farr[i] += farr[i-1]+farr[i];
}
int ans[] = new int[n];
for(int i=n-1; i>=0; i--)
{  int idx = arr[i]-min;
   int fidx = farr[idx]-1;
   ans[fidx] = arr[i];
   farr[idx]--;
}
```

$T(n) = O(n + range)$

Radix Sort

213 97 718 123 37 443 982 64
375 685

sort on basis of units place then tens then hundreds then thousands ... using count sort approach

```
min = 0
max = 9
```

```
for(int e: arr)
{  max = Math.max(max, e);
}
int exp=1;
while(exp <= max)
{  countSort(arr, exp);
   exp = exp*10;
}
```

arr[i] / exp % 10

OOPS
Object Oriented Programming

Date:
Page:

Fundamental Units → classes
→ Objects

4 pillars → Encapsulation
→ Abstraction
→ Inheritance
→ Polymorphism

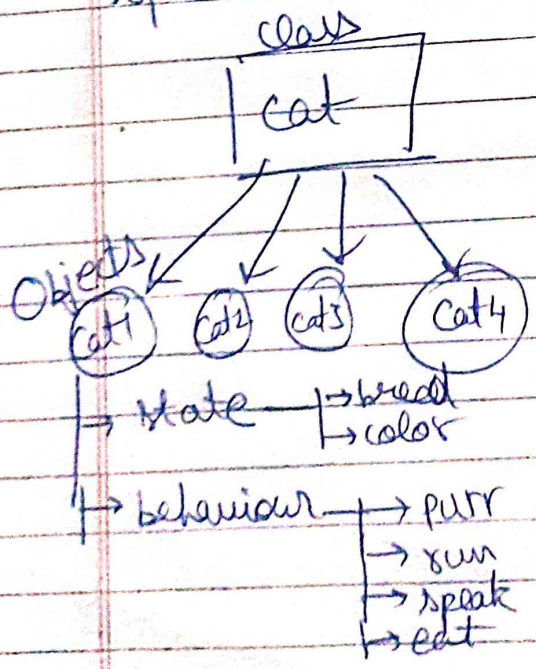
→ Object map onto real world entities
i.e have → states (blackboard → black, size - 40m)
→ behaviour (can write on it / rub on it)

→ so Java being OOP language tackles real world problems easily.

→ Object defined as instance of class

→ makes code more humanly readable.

→ Object contains address ↑ takes up some space



en → Class is like
(Note chapne ki machine)
object is notes

→ Class is a blueprint for the object

→ Logical Entity → doesn't get stored in memory. (no space consumed)

→ create many objects from class.

→ only one public class in a java file because name of javafile is same as that of public class

Making class

```
class Cat {
    boolean hasFur;
    int legs;
    String color, breed;
    public void walk() {
    }
}
```

```
class Dog {
}
```

Making object

```
Cat c1 = new Cat();
```

↳ allocates space in memory to object (object go in heap memory and address reference goes to object name).

Methods

→ functions in OOPS are methods

→ DRY principle
↳ do not repeat yourself
easy debugging - change code only at a place instead of everywhere.

behaviour part in class is methods

Syntax

access modifier return type method name (parameter list) {

}

→ Thread execution starts from main method of public class public static void main (String args[]).

→ static functions can be called in static functions

Method Overloading

→ When the name of the function remains same as existing one but with different parameters (signature).

→ is Java Pass By value or Pass by reference?

It looks Java is
→ pass by value for primitive data types.
→ pass by reference for Non primitive

but is actually pass by value.

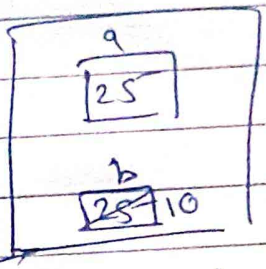
→ Pass By value → method parameters values are copied to another variable & then copied object is passed.

Page:

→ Pass By Reference → Reference to actual parameter is passed to the method.

pass by value eg

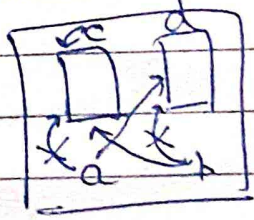
```
int a=25
funct(a);
void funct(int b)
{
  //scope of b b=10
}
```



changing b doesn't change a

In non-primitive

Its also pass by value but the value passed is reference.



but i can change values of original

Constructors

Initialize ~~set~~ default values of state of a class

→ name same as that of class
→ NO return type
→ invoked automatically when object is instantiated.

Types

→ No argument constructor

Default constructor gives default values to states. ex- int 0 boolean false double 0.0

→ Parameterized constructor
(customize values)

this keyword
↳ use to point at current object of the class.

this.wheels = wheels

Constructor overloading
↳ similar like function overloading.
(same name, diff. arguments)

Note → Default constructor can't be called after you have made a constructor of your own

Packages & Access Modifiers

part of Encapsulation (hiding data i.e restricted access of data to rest of code)

Encapsulation

↳ process of wrapping

Date:
Page:

[code + data] together into single unit

just like a capsule. (data of class is hidden from other classes) also data-hiding.)

Achieved through → packages
→ Access Modifiers

Packages → group of similar types of → sub packages
↓ ↓
classes Interfaces

Inbuilt
eg- util, io, lang

↳ removes naming collision

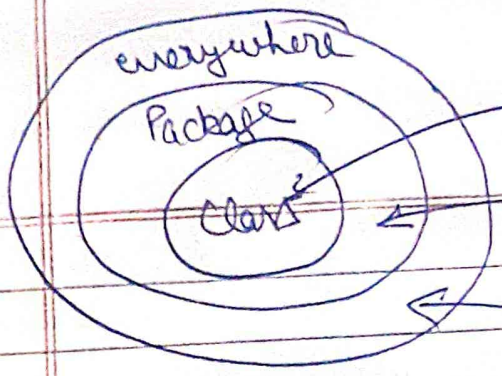
using a class file to diff. package class using import

Access Modifiers

↳ specifies scope of field, method, constructor or class

4 types

1. Private → only within the class. ~~cannot~~ ↳ can't be accessed outside class.
2. Default → only within package
3. Protected → within package & to the subclasses
4. Public → anywhere



Date:
Page:

More about Encapsulation

☆ → makes it easy to maintain (cleaner code)

if you want restrictions on data
like age can't be less than 18
names length should be greater than 5

① Declare class variables as private
② provide public getter & setter methods used to write & read their values.

```
public class Student {
    private int age;
    private String name;
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        if (age > 18)
        { this.age = age;
        } else {
            print("grow old man!");
        }
    }
```

```
main (String args[])
{
    Student s1 = new Student();
    s1.setAge(20);
print (s1.getAge());  ⟶ (20)
```

→ helps in decoupling of system. (loosely coupled interdependency)

→ dev, tested, debugged independently

→ Encapsulations main work is to group your data code together as a result of which Data hiding is achieved;

Static Keyword

→ related to class not object
(particular members (class, method, variables) belong to class & not object)

→ helps in memory management
only 1 copy at class level rather than multiple copies.
→ used on member not changing in objects.
like species.

→ Static members are not declared in non-static inner class
but can be in public class

→ Make private constructor & then no one can make object of the class. eg - Math.max
Math class

→ cannot declare top level class as static (but nested classes can be)

Date:
Page:

if→ Why nested class to be static?

```
public class A {
    class B {

    }
    static class C {

    }
}
```

diff.

to access B make object of class A then class B while for C simply make object of class C.

```
A obj A = new A();
A.B obj B = new B();
```

↳ jabtak classes nahi banaoge school nahi exist kar sakta

```
C obj C = new A.C();
```

Static blocks

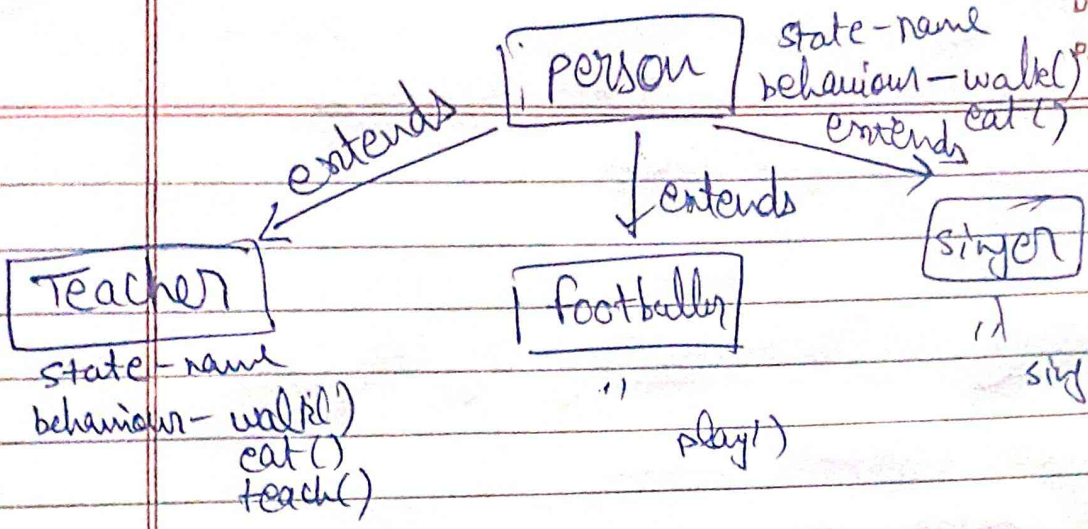
```
static {

}
```

gets executed first

just like componentDidMount

Inheritance

Date:
Page:



as name, walk, eat is getting repeated. (DRY principle)

—— is a ——

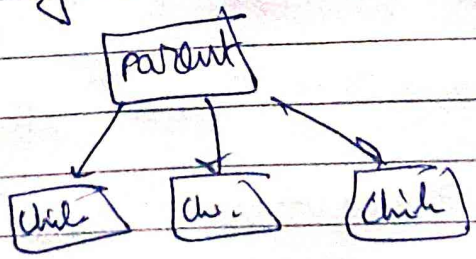
eg- Teacher is a person

Protected

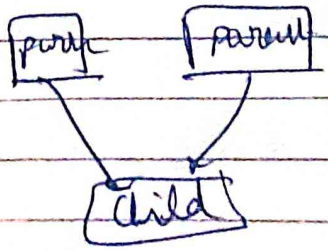
- within class
- within package
- within subclass

* object class is parent of all
eg- hashcode
getClass

Java only allow hierarchical inheritance



✗ but not multiple inheritance



Method overriding

Date:
Page:

↳ same name function
↳ where child function overrides parent function.

Upcasting & Downcasting of Objects

Animal parent class
Cat child class

```
Cat c = new Cat();
Animal a = c; // upcasting
```

implicit

```
Cat c = new Cat();
Animal a = c;

Cat c1 = (Cat) a; // downcasting
```

(explicit)

↳ checked runtime (executed)

```
Person p = new Person();
Teacher t = (Teacher) p;
```

↳ runtime error cause p is not teacher

Super Keyword

refer to immediate parent class of child class

Date:
Page:

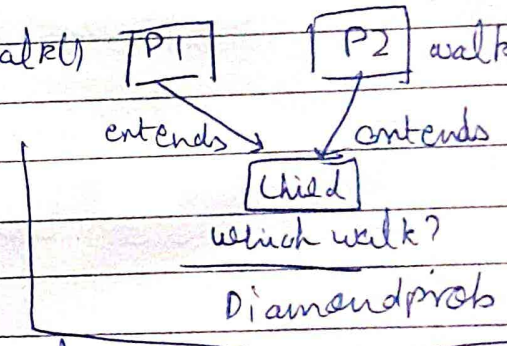
```
public void eat()
{
    super.eat(); → parent eat
                 → child eat
}
```

super() → parent constructor

Singleton Pattern

Problem with Multiple Inheritance is that 2 classes may define different ways of doing same thing, and subclass can't pick one

→ Diamond problem



Polymorphism

Ability to take multiple forms (1 method behaving in multiple ways)

① Compile time polymorphism
② Run Time Polymorphism

```
Dog d = new Dog();
Pet p = d;
Animal a = d;
Dog d1 = (Dog) p;
```

runtime

```
d.walk()
p.walk() → this also calls dogs function
```

runtime poly

Abstraction

→ reducing system complexity

Date:
Page:

Repair Shop

repair car

WagonR
accelerate brake

Audi
accelerate break

→ abstract methods in class are to implemented by its children class
abstract methods are present in abstract class

```
public abstract class Car {
    public abstract void accelerate();
```

→ Cannot create objects of Abstract classes
→ when parent is only concept
→ abstract class can have concrete methods
(Non abstract methods)

final Keyword

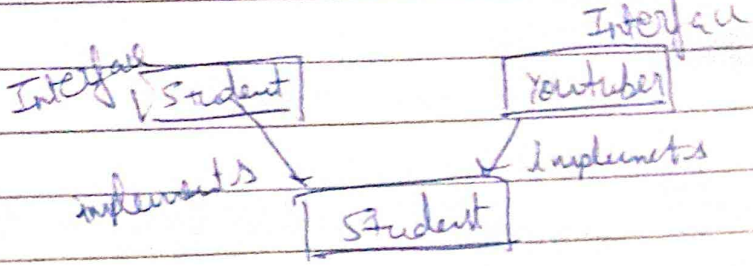
→ value of variable can't be changed
→ class cannot have children
→ final method can't be overriden

Interface

Date:
Page:

→ To overcome diamond problem

→ & achieve abstraction



Multiple interfaces can be implemented

→ Specifications that other classes have to implement

→ functions in interfaces are by default Abstract

```
interface Name {
    public void getArea();
}
```

Low Level Design

Parking lot system

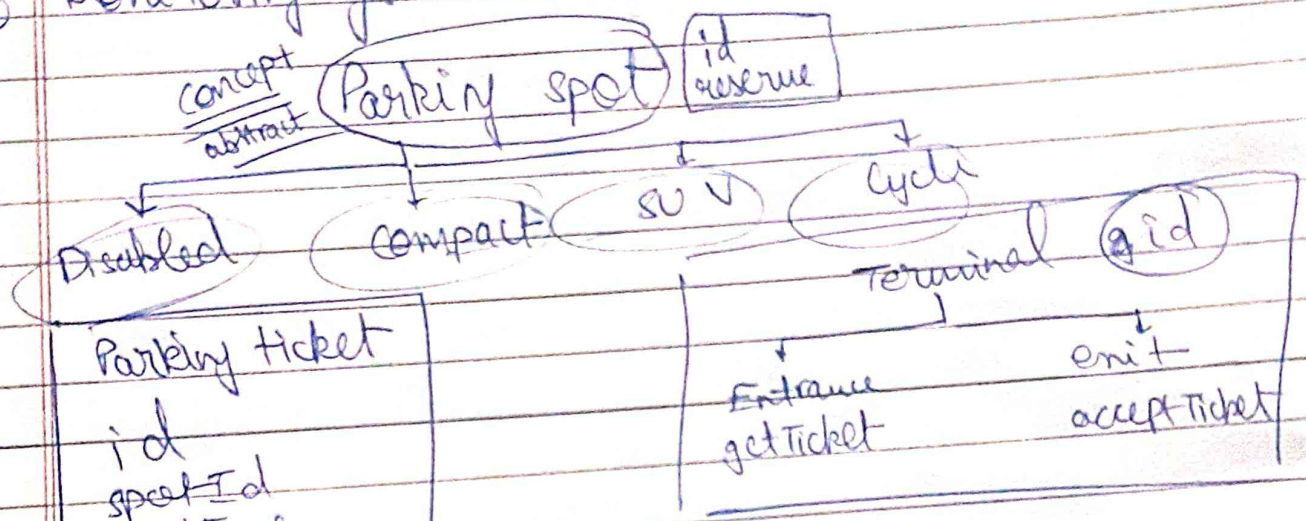
Req.

1. Big parking lots 30K-40K (size)
2. Entrance-4 exit-4
3. Ticket at entrance
4. spot nearest to entrance
5. Limit
6. Different spots → Disabled, Compact, SUV, motor cycle
7. fee on time → cash, credit
8. Monitoring

structural design → Behavioral design

1. Parking lot System
2. Entry / Exit Terminals
   - → printers
   - → payment processor
3. Parking spot
4. ticket
5. Database
6. monitoring system

Date: _____
Page: _____



Singleton pattern
↳ having single object of the class
1. make constructor private (so no object)
2. getInstance ()

compiled languages → Java, C, C++
Interpreted lang — Python, JS