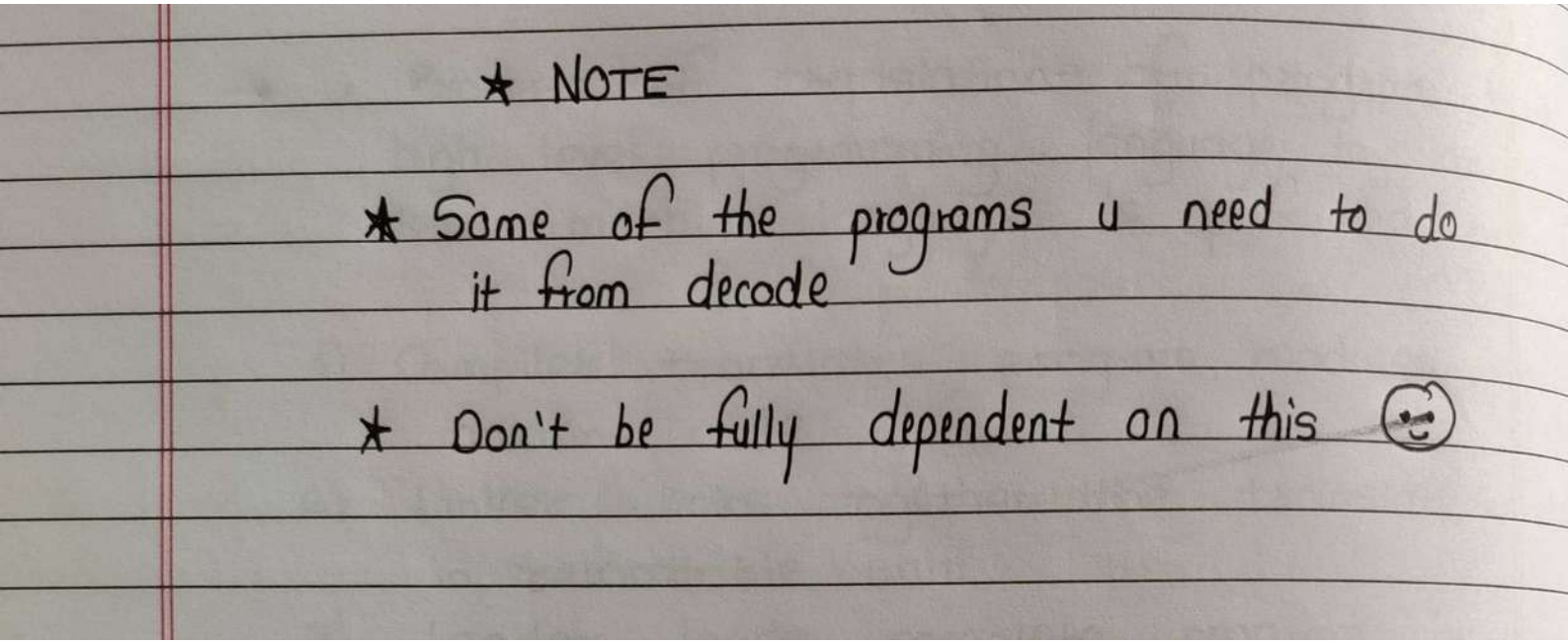

★ NOTE
★ Some of the programs u need to do it from decode
★ Don't be fully dependent on this

UNIT-III

JAVA AS OOPL - Overview

Q.1 Justify the meaning of line,
"Java is simple, architecture neutral portable, interpreted and robust and secured programming language"

→ • Java is simple -
Java is simple programming language. Even though you have no programming background, you can learn this language comfortably.

• Architecture Neutral Portable -
Programs in Java can be executed on variety of platforms. Primitive data types used in Java are machine independent.

• Interpreted -
Java is a language which can be compiled AS WELL AS interpreted.

• Robust and Secure -
Following reasons make Java more secured:
i) It does not support concept of pointers directly
ii) Output of Java is a bytecode
iii) Memory management is done automatically using garbage collection

★ Arrays -

→ • Array is collection of similar type of elements.

• Syntax of declaring array is:
data_type array_name [size];

• To allocate memory:
array_name = new data_type [size];

• Arrays are allocated Dynamically in Java.

• Two-dimensional arrays are those in which elements are stored in rows as well as columns

• Syntax is,
data_type array_name [][] = new data_type [size][size]

Program which reads matrix of size 3x3 and performs addition of elements in each row & column. and prints the result.

→
```
public class Test
{
    public static void main (String [] args)
    {
        int rows, cols, row_sum, col_sum;
```

```
int a[][] = {
            {1,2,3},
            {4,5,6},
            {7,8,9}
          };

rows = a.length;
cols = a[0].length;
// Calculation of row addition
for (int i=0; i<rows; i++)
{
    row_sum = 0;
    for( int j=0; j<cols; j++)
    {
        row_sum = row_sum + a[i][j];
    }
    System.out.println ("Sum of" + (i+1) + "row:" + row_sum);
}
    // Calculation of column addition
for (int i=0; i<cols; i++)
{
    col_sum = 0;
    for (int j=0; j< rows; j++)
    {
        col_sum = col_sum + a[j][i];
    }
    System.out.println("Sum of" + (i+1) + "column:" + col_sum);
}
}
};
```

★ String Handling -

→ • String is a collection of characters
• In Java, string defines object.

• Operation of class strings:

1) Find the length:
We can find length of given string using method length().

```
Eg,
class Demo
{
    public static void main (String [] args)
    {
        char Str [] = {'H', 'I'};
        String S = new String (Str);
        System.out.println (" Length of string " + S.length());
    }
}
```

2) Comparing 2 strings:
We use method equals() of Boolean type for this purpose.

```
Eg,
class Demo
{
public static void main (String [] args)
  {
    String s1 = new String ("HI");
```

```
        String s2 = new String ("hi");
        if (s1.equals(s2) == true)
            System.out.println ("Two strings are equal");
        else
            System.out.println ("Two strings are not equal);
    }
}
```

3) Extraction of character from string -

We use the method charAt(index) to extract the character.

Eg,

```
String fruit = new String ("mango");
char ch;
ch = fruit.charAt(2);
System.out.println (ch);
```

- Definition -

  Object -

  Object is nothing but instance of class. Hence, block of memory gets allocated for this instance

  For creating object in Java, operator 'new' is used

```
Test obj;          // Declaration
obj = new Test();  // obj gets instantiated
```

★ Constructors -

→ • Constructor is specialized method for initializing objects

• Name of constructor is same as that of its class name.

• It is called constructor because it creates values for data fields in class

• Types of Constructors -

1) Default Constructor -

• No arguments / parameters are passed in default constructor

• Eg,

```
public class Demo
{
    int a;
    Demo()          // Default constructor
    {
        a = 0;
    }
}
class MainClass {
    public static void main(String args[])
    {
        Demo obj = new Demo();
    }
}
```

2) Parameterized Constructor -

- Parameters are passed to constructor function
- Eg,

```
public class Demo
{
    int a;
    Demo (int x)        // parameterized constructor
    {
        a = x;
    }
}
class MainClass {
    public static void main (String args[])
    {
        Test obj = new Test(10);
    }
}
```

3) Copy constructor -

- It is kind of constructor which can be initialized using object of same class.
- Eg,

```
class Complex
{
    private double real, img;
    public Complex (double r, double i) // parameterized
    {
        this.real = r;
        this.img = i;
    }
```

```
Complex (Complex c)      // copy constructor
{
   real = c.real;
   img = c.img;
}
```

★ 'This' Keyword -

→ • The 'this' keyword is used by a method to refer the object which invoked that method.

• It is used inside method definition to refer the current object.

• Eg,

```
public class Demo
{
   int a,b;
   public void sum (int a, int b)
   {
      this.a = a;
      this.b = b;
   }

   public void display()
   {
   int c = a+b;
```

```
    System.out.println("Sum = "+c);
}
public static void main(String args[])
{
    Demo obj1 = new Demo();
    obj1.sum(10,20);
    obj1.display();
```

* Garbage Collection -

→ • It is a method of automatic memory management

• It works as follows -

1) When application needs free space to allocate nodes and there is no free space available, then garbage collector is called.

2) It then searches for nodes that are no longer accessible from external pointer. These nodes are then made available for reuse.

• Garbage collection is usually done in 2 phases

1) Marking phase -
Garbage collector scans entire system and marks all nodes that are accessible

2) Collection phase -
Memory is scanned sequentially and unmarked nodes are made free.

* finalize() method -

→ • Inside the finalize() method, you can specify the actions that must be performed before an object is destroyed.

• Syntax is,
finalize()

• Code is,
```
void finalize()
{
    finalization code
}
```

• finalize() method is called just before the garbage collection

• It is not called when object goes out-of-scope.

■ Definition -
Overloading -
It is a mechanism in which we can use many methods having same function name but different no. / types of parameters

★ Object as Parameter -

→ • Object can be passed to method as argument

• Using dot operator, object's value can be accessed

• Syntax is,

```
data_type name_of_method (object_name)
{
    // body
}
```

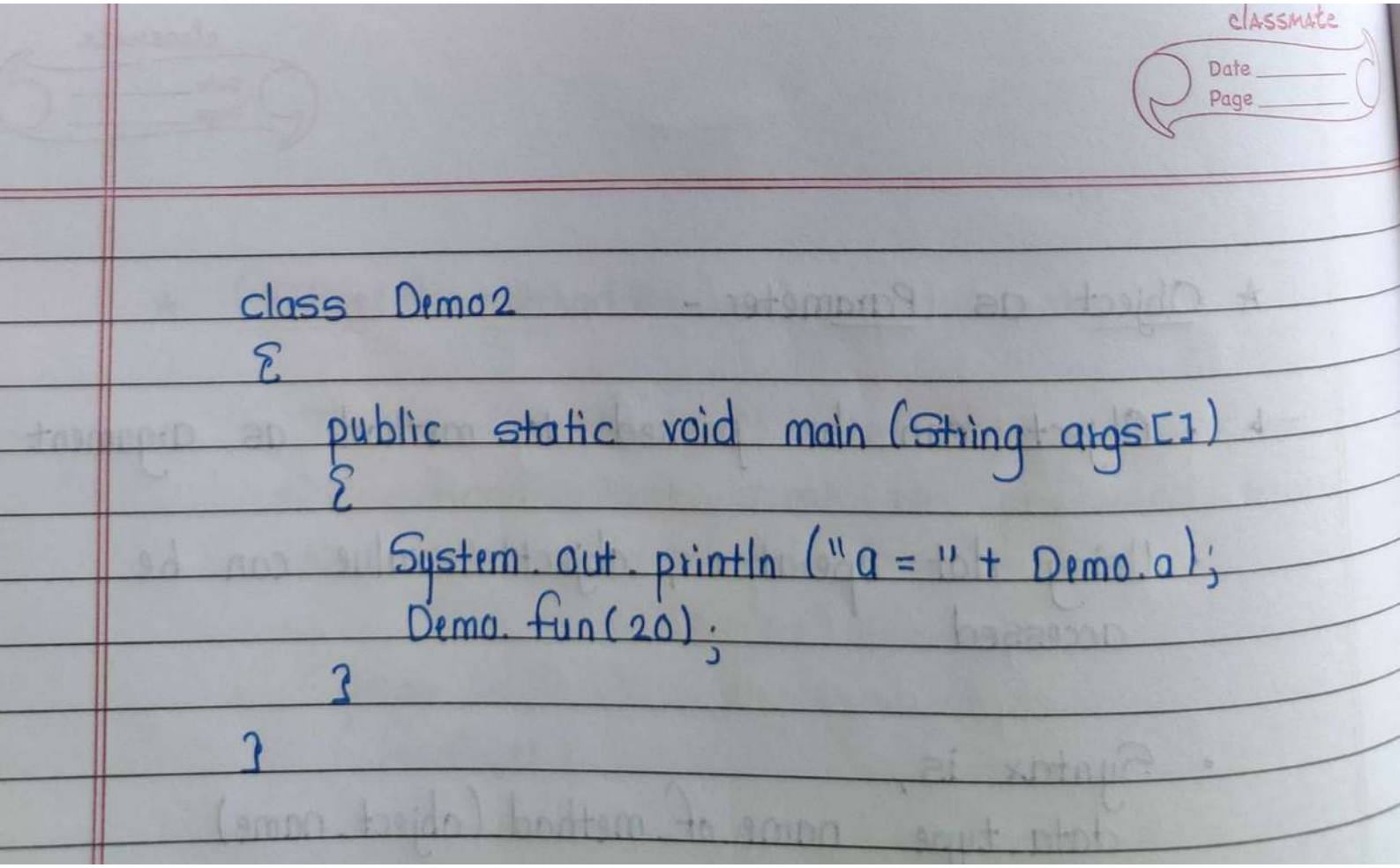
★ Static Members -

→ • Static members can be static data or method

• Static members are those members which can be accessed without using object

• Eg,

```
class Demo
{
  static int a = 10;
  static void fun (int b)
  {
    System.out.println ("b = " + b);
    System.out.println ("a = " + a);
  }
}
```



```
class Demo2
{
    public static void main (String args[])
    {
        System.out.println ("a = " + Demo.a);
        Demo.fun(20);
    }
}
```

UNIT – IV

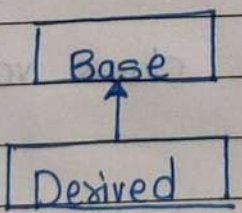
Inheritance, Packages & Exception Handling Using Java

* Inheritance -

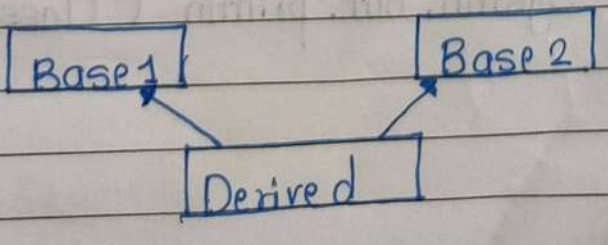
→ • It is a mechanism in Java by which derived class can borrow properties of base class and at same time, derived class can have additional properties

- Advantages :
  1) Reusability
  2) Extensibility
  3) Data Hiding
  4) Overriding

- Types of inheritance :
  1) Single inheritance -
     There is one parent per derived class

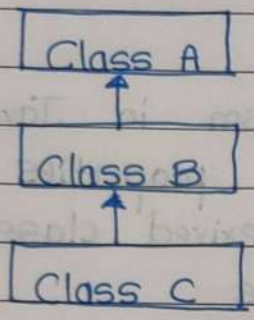


  2) Multiple Inheritance -
     Derived class is derived from more than one base class



3) Multilevel Inheritance –

Derived class is derived from base class which itself is a derived class



★ Super Keyword –

→ • 'Super' is a keyword used to access immediate parent class from subclass.

• There are 3 ways by which 'super' keyword is used.

1) Used to access class variable of immediate parent class.

Eg,

```
class A
{
    void fun()
    {
        System.out.println ("Class A");
    }
}
```

```
class B extends A
{
    void fun()
    {
        System.out.println("Class B");
    }
    void disp()
    {
        super.fun();
    }
    public static void main(String args[])
    {
        B obj = new B();
        obj.display();
    }
}
```

2) Used to invoke immediate parent class constructor

Eg,

```
class A
{
    A()
    {
        System.out.println("Constructor");
    }
}
class B extends A
{
    B()
    {
        super();
```

```
        System.out.println("Constructor of B");
    }
    public static void main(String args[])
    {
        B obj = new B();
    }
}
```

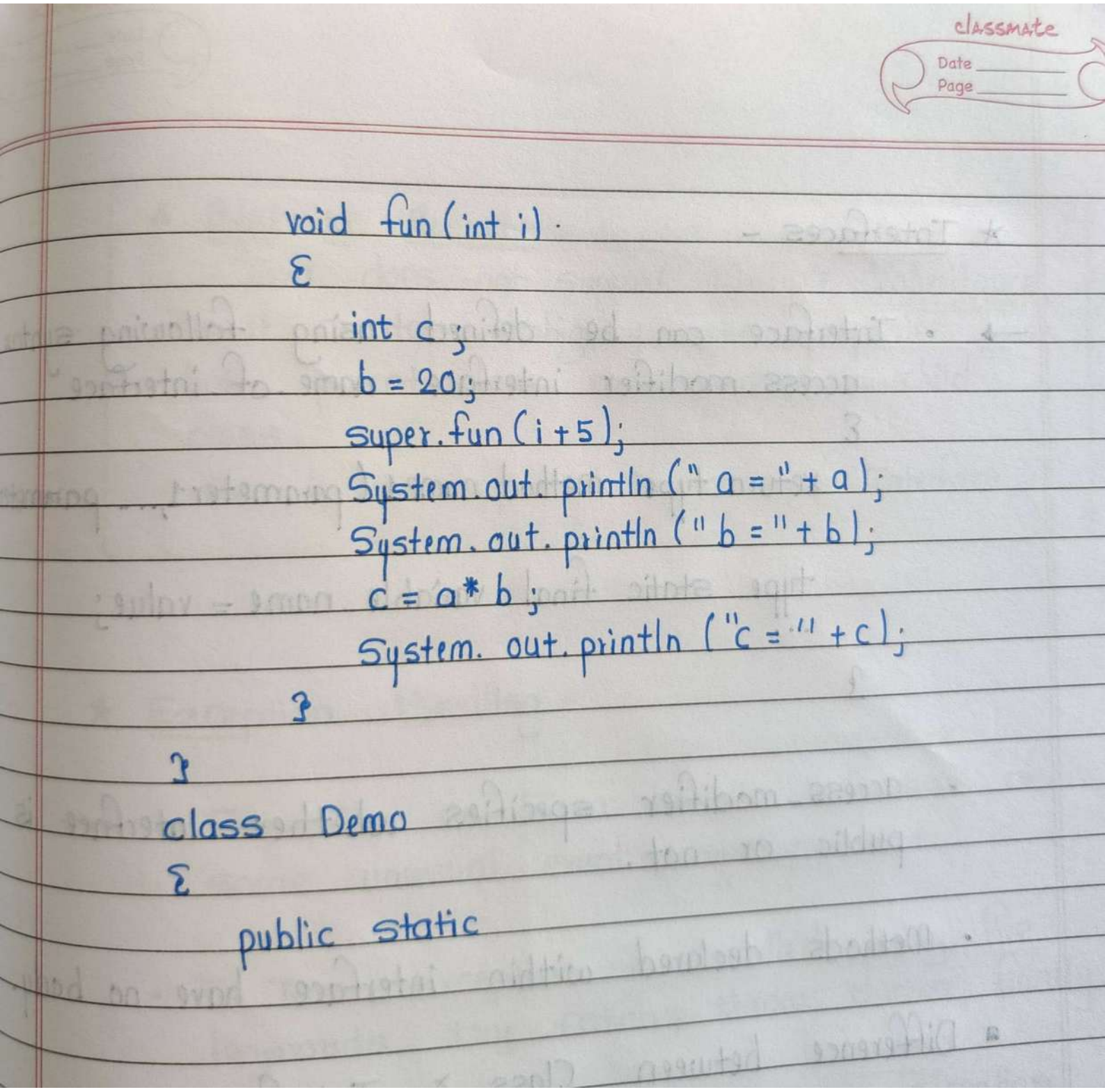
★ Method Overriding -

→ • It is a mechanism in which subclass inherits methods of superclass and sometimes subclass modifies implementation of method defined in superclass.

• Method of superclass which gets modified in subclass has same name and type signature.

• Eg,

```
class A
{
    int a=0;
    void fun(int i)
    {
        this.a = i;
    }
}
class B extends A
{
    int b;
```



```
void fun(int i)
{
    int c;
    b = 20;
    super.fun(i+5);
    System.out.println("a = "+a);
    System.out.println("b = "+b);
    c = a*b;
    System.out.println("c = "+c);
}
}
class Demo
{
    public static
```

?

★ Polymorphism –

→ • It is a mechanism which allows to have many forms of method having same name.

• Types of polymorphism –
1) Compile - time polymorphism
2) Run - time polymorphism

• Compile time polymorphism can be achieved by function overloading and runtime polymorphism can be achieved by method overriding

★ Abstract Class –

→ • In inheritance hiearchy, superclass is supposed to be most general / less specific.

• Sometimes, superclass is so general that it does nothing but lists out only common features of

other classes.

- Then such superclass is referred as abstract class.

★ Package -

→ • Package is defined using keyword 'package'.

- Syntax for declaring package is,
  package name_of_package
- package statement defines name space in which classes are stored.
- Basically Java creates a directory and name of this directory becomes package name.

★ CLASSPATH -

→ • For locating specified package, java run time system makes use of current working directory as starting point

- Thus if required package is in current working directory then it will be found
- Otherwise you can specify directory path setting the CLASSPATH statement.

★ Interfaces –

→ • Interface can be defined using following syntax,

```
access_modifier interface name_of_interface
{
    return_type method_name1 (parameter1,... parametern);
    type static final variable_name = value;
    ...
}
```

- access_modifier specifies whether interface is public or not.
- Methods declared within interface have no body.

▪ Difference between Class & Interface

Class	Interface
i) Denoted by 'class' keyword	i) Denoted by 'interface' keyword.
ii) By creating instance of class, the class members can be accessed	ii) You cannot created an instance of interface
iii) Class can use various access specifier like public, private, protected	iii) Interface makes use of only public access specifier
iv) Members of class can be constant or final	iv) Members of interfaces are always constant

▲ Point to Remember :
Java does not support Multiple Inheritance, as it creates ambiguity when properties from both parent class are inherited in child class.
But, it is supported in case of Interface

★ Exception Handling –

→ • Exception in Java is an indication of some unusual event, usually error

• In Java, exception is handled using five keywords try, catch, throw, throws, finally.

• Java code that may produce an exception is placed within try block.

• Benefits :
i) Using exception, main application logic can be seperated out.
ii) When error occurs, exception can be thrown, avoiding crashing of entire application
iii) Various types of errors in source code can be grouped together

- Type of Exception :

1) Checked Exception -
These type of exception need to be handled explicitly by code itself.

2) UnChecked Exception -
These type of exception need not be handled explicitly. The JVM handles these type of exceptions.

- Keywords used :

1) try - Block of source code that is to be monitored for exception is placed here

2) catch - It handles specific type of exception along with try block

3) finally - Specifies the code that must be executed even though exception may occur / not

4) throw - Used to throw specific exception from program code

5) throws - Specifies exceptions that can be thrown by particular method.

★ Program to calculate value of ((x+y)/(x-y))
It should prevent condition x-y = 0
(Can be used as example for try(), catch() keywords)

→
```
import java.util.Scanner;
public class Expression
{
    public static void main(String args[])
    {
        int x, y, N, D;
        float result;
        Scanner input = new Scanner(System.in);
        System.out.println("Enter value of x:");
        x = input.nextInt();
        System.out.println("Enter value of y:");
        y = input.nextInt();
        try
        {
            N = x + y;
            D = x - y;
            result = N/D;
            System.out.println("Result = " + result);
        }
        catch(ArithmeticException e)
        {
            System.out.println(e);
        }
    }
}
```

★ Uncaught Exceptions –

→ • If there exists some code in source program which can cause exception & if programmer does not handle it, then Java raises the exception

• This is known as uncaught exception

• Eg,

```
class Demo
{
    static void fun(int a[])
    {
        int c;
        c = a[0] / a[2];    // Index out of range
    }
    public static void main(String args[])
    {
        int a[] = {10, 5};
        fun(a);
    }
}
```

• When we use index which is beyond range of index then ArrayIndexOutOfBoundsException occurs

★ Built - In Exceptions -

→ Exception	Description
NullPointerException	Caused when attempt to access an object with NULL reference is made
IndexOutOfBoundsException	Index when gets out of bound, this exception is caused.
ArrayIndexOutOfBoundsException	Array index when gets out of bound, this exception is caused.

- **Definition -**

  Stream -

  - It is basically a channel on which data flow from sender to receiver
  - Input object that reads stream of data from file is called Input stream
  - Output object that writes stream of data from file is called Output Stream

★ Difference - Byte Stream v/s Character Stream

→

Byte Stream	Character Stream
i) It is used for inputting & outputting bytes	i) It is used for inputting & outputting characters
ii) There are 2 super classes used in byte stream - InputStream & OutputStream	ii) There are 2 super classes used in character stream - Reader & Writer
iii) It never supports Unicode characters	iii) It supports Unicode characters.

★ Predefined Stream -

→ • For displaying messages on console, we make use of System.out statement.

• There is class named System which is defined in java.lang package

• System contains 3 predefined variables - in, out & err

• System.out defines standard OutputStream
System.in refers to standard InputStream
System.err are objects for standard input stream and error.

- Definition -

  PrintStream Class -

  This class provides methods to write data to another stream.

  PrintWriter Class -

  This class is useful in writing contents to console just like System.out

Q] What is use of CharactexArrayReader() & CharacterArrayWriter() methods in Java? Write program which reads string of 10 characters from usex. and prints substring from given string using above method

→

```
import java.io.*;
import java.util.*;
class Demo
{
    public static void main (String arg[]) throws IOException
    {
        System.out.println ("Enter string: ");
        Scanner input = new Scanner (System.in);
        String str = input.nextLine();
        if( str.length() > 10)
            str = str.substring (0,9);
        char c[] = str.toCharArray();
        CharArrayReader car = new CharArrayReader (c);
```

```
int i;
System.out.println ("Displaying string");
while ((i = car.read()) != -1)
System.out.print((char)i);
    System.out.println (" Displaying substring");
    CharArrayWriter caw = new CharArrayWriter();
    caw.write(str,2,5);
    System.out.println (caw.toString());
}
}
```

UNIT-II

Multithreading in Java

- **Definition -**

Thread -

- It is a tiny program running continuously It is sometimes called as light-weight process
- In Java, we can implement thread programs using two approaches,
  - Using Thread class
  - Using runnable interface

★ **Difference - Multiprocessing v/s Multithreading**

Multiprocessing	Multithreading
i) In this, more CPUs are added to increase the computing power	i) In multi-threading, multiple threads are created to increase computing power
ii) Multiple processes execute concurrently	ii) Multiple threads of single process execute concurrently
iii) It can be symmetric and asymmetric	iii) It is not classified
iv) Creation of process is slow and resource specific	iv) Creation of multiple thread is economical and resource & time

★ Thread Synchronization -

→ • When two or more threads need to access shared memory, then there is a way to ensure that access to resource will be by only 1 thread at time.

• This process of ensuring one access at time by one thread is called synchronization

• There are 2 ways to achieve synchronization-

1) Using synchronized methods

Syntax:

```
synchronized return_type fun_name(parameters)
```

2) Using synchronized blocks (statements)

Syntax:

```
synchronized (object reference)
{
statement;      // block of code to synchronize
:
}
```

★ isAlive() & join() methods –

→ • isAlive() method is used to check if the thread is alive or not.
• This method returns true if thread is still running else it returns false

• Syntax :
public final boolean isAlive()

• Using join() method, we tell our thread to wait until specified thread completes its execution.

• Eg,

```
public class Demo extends Thread
{
    public void run()
    {
        System.out.println ("Begin");
        try {
            Thread.sleep(500);
        }
        catch ( InterruptedException ie)
        {
        }
        System.out.println ("End");
    }

    public static void main (String[] args)
    {
```

```
Demo t1 = new Demo();
Demo t2 = new Demo();
t1.start();
System.out.println("Is t1 alive?"+t1.isAlive());
if (t1.isAlive() == true)
    try
    {
      t1.join();
    }
    catch (InterruptedException ie)
    {
    }
    System.out.println("Starting another thread");
    t2.start();
    System.out.println("Is t2 alive?"+t2.isAlive());
}
}
```

★ Features of Javascript -

→ 1) Browser Support :
Almost all popular browsers support JavaScripting

2) Structure Programming Syntax :
Javascript supports commonly the structured language type syntax

3) It automatically inserts semicolon at end of statement

4) Dynamic Typing -
Data type is bound to value & not to variable

5) Run-time Evaluation

6) Support for Object

7) Supports use of regular expression

★ Angulax JS -

→ • Angulax JS is powerful Javascript framework
• It is an open source front-end enabled web application framework.

- Advantages :-
  1) Built by Google Engineers
  2) mvc support -
     It is based on model view controller architecture
  3) Intuitive - because it makes use of HTML
  4) Comprehensive -
     It does not need any other plugins/frameworks
  5) Rich Features - like Data Binding, dependency injection, etc
  6) Reusable and Less code
- Disadvantages :-
  1) It is not secure
  2) There are multiple ways to do same thing which can be hard for novices

- Features :-

1) Data Binding -
Allows automatic synchronization between model & view components

2) Scope -
Some objects from model can be correlated with view & controller

3) Controller -
functions that are bound to particular scope

4) Directives

5) Filters - allows to select subset from array of items

6) Routing - allows to switch between multiple views

★ ReactJs -

→ • ReactJs is an open-source, component-based front end JS library maintained by Facebook

• It is responsible only for view layer of application

▪ Features -

1) Virtual Dom -
Dom means Document Object Model. It provides way to update the content, structure and style of document.
ReactJs's Dom manipulation is faster than other Frameworks

2) Components -
Allows web developer to create custom elements that can be reused in HTML

3) JSX -
It is extension of JS syntax and can be seen as combination of JS and XML

4) One way Data Binding -
Data is allowed to flow in one direction at a time, helping to achieve greater control.

★ VueJS -

→ • Vue.js is open-source, progressive framework for JS used to build web interfaces & one-page applications

• It is also used for both Desktop & Mobile app as front end derelopment tool

Features -

1) Event Handling -
Any event can be handled with help of v-on attribute

2) Animation -
It helps to show animation of elements on web page

3) Computed properties -
Feature that can immediately sense changes made in UI & take necessary actions

4) Lightweight & Efficient.

★ Comparison -

Sr. No	Angular JS	React JS	Vue JS
1)	It is completely based on mvc	It is library that just offers view	It is library that allows you to create interactive web interfaces
2)	It is steep learning curve	Easier to learn than Angular JS	There is small learning curve
3)	Uses 2-way data binding	Uses 1-way data binding	Uses 1-way data binding
4)	Written in TypeScript	Written in Native JavaScript	Written in HTML and JS
5)	Mainly used by Google, Instagram, whatsapp	Mainly used in Facebook, Netflix	Mainly used by Alibaba, Baidu

UNIT-VI

Logical & Functional Programming

★ Functional Programming -

→ • Functional languages are those languages in which basic building block is functions

- Features -

1) Functional programming languages are modelled on concept of mathematical functions and make use of only conditional expression & recursion

2) Programs are constructed by building functional applications

3) Functional programming language use neither variables nor assignment statements

4) Can be categorized in two types
   - (a) Pure functional language - supports only functional paradigm
   - (b) Impure functional language - can be used to write imperative style programs.

• Commonly used Functional Programming Languages are LISP, Haskell, ML, APL, etc.

★ LISP Functions –

→ ▪ Mathematical Functions :

- These are various mathematical functions like,

```
>(sqrt 25)
 5
>(min 10 20)
 10
>(max 10 20)
 20
```

▪ Eval Function :

- Helps in evaluating an expression & return the value of evaluation
- Eg, `>(eval (+ 10 20))`
  `30`

▪ car function :

- returns first element of non-empty list
- eg, `>(car '(1 2 3 4))`
  `1`

▪ cdr function :

- returns rest of list after first element is removed
- eg, `>(cdr '(1 2 3 4))`
  `(2 3 4)`

Cons function :

- We can combine two lists using cons
- eg, > (cons 10 '(20 30 40))
  (10 20 30 40)

Lists :

- List is represented as,
  > (10 20 30)

- Various functions used for manipulation of lists:

1) Length -
   > (length '(10 20 30 40 50))
   5

2) Append -
   > (append '(10 20 30) '(40 50))
   (10 20 30 40 50)

3)

★ Definitions -

→ • The operator 'defun' is used to define a procedure

• Syntax is,
defun (proc_name (argument1 argument2))
body of procedure

• Eg, > defun (mysum (a, b)
(+ a b))

> (mysum 2 3)
5

★ Predicates -

• Equality Predicates :

1) EQUAL :
• It is true, if its arguments are structurally similar objects.
• Eg, > (equal "dk" "dk")
=> T
> (equal "dk" "DK")
=> NIL

2) EQ :

- It is true, iff x and y are same identical object i.e share same memory location
- Eg, >(eq 'a 'a)
  =>T

  >(eq 'a 'b)
  => NIL

3) EQL :

- It is true iff they are numbers of same type with same value, or character object that represent same character
- Eg, >eql (10, 10)
  => T
  > eql (10, 10.0)
  => NIL

4) = :

- It compares only numbers regardless of type
- Eg, (= 10 10.0)
  => T

■ Number predicates:

1) NUMBERP :

- It takes one argument and returns True if argument is number
- Eg, > (numberp 10)
  T

2) EVENP :

- Takes one argument & returns True if argument is Even
- Eg, >(evenp 10)
  T

3) ODDP :

- Takes one argument & returns True if argument is Odd
- Eg, >(oddp 11)
  T

* LISP program to find factorial of n numbers using recursion

→
```
(defun fact(n)
(if (= n 1)
1
(* n (fact(- n 1))))
```

- Definition :

Bound -

A variable x is bound in expression E, if meaning of E remains unchanged by replacement of variable x by y for every x in E

Free -

Variable x is free in expression E, if meaning of E changes by replacement of x by y for every x in E

★ Recursive Function :-

→ • Recursive function is function which calls itself for more than once

• Eg, (defun power (x y)
(if (= y 0) 1
(* x (power x (- y 1)))))

> power (3 4)
81

★ Association Lists -

→ • An association list or a-list, records a mapping from keys to values

• Using 'assoc' we can obtain most recent binding of key to value

• Eg, > (assoc 'one '((one 1)(two 2)(one 3)))
(one 1)

★ Arrays –

→ • Array is sequence of contiguous memory locations in which we can store objects of any kind

• In LISP, we can write using ARRAY function

```
(ARRAY MyTable 20 20)
MyTable
```

This states MyTable is array that has 2 indices from 0 to 19 each.

★ setf and get –

→ • Properties are created using 'get' within a 'setf' form

• Eg,
```
> (setf (get 'student 'name) '(DK))
(DK)

> (setf (get 'student 'rollno) '(22))
(22)
```

* Lambda Function -

→ • The lambda function is used to define a function which returns some value

• Syntax is,
(lambda (parameters) Body)

• Eg, > ((lambda (x y) (* x y)) 10 20)
200

• Similarly we can assign value to function, by return value of some function

• Eg, > (define (add n) (lambda (x) (+ x y)))
add
> ((add 10) 20)
30

* Printing & Reading Operations -

→ • The print, write are output functions used to display values / text

• Read function is for reading values from input streams

• Eg, defun Myfun ()
(print "Enter number:")
(setq a (read))

```
(print "You have entered...")
(write a))
```

- Output is :

```
Enter number : 22
You have entered ... 22
```

★ Logical Programming –

→ • It is programming paradigm in which set of sentences are written in logical form.

- Logic programs consists of facts and rules about some problem domain
- Concept of logic programming is linked up with language called prolog.

■ Features of prolog -

1) Used to solve complex problems in artifical intelligence
2) Used on pattern matching algorithm
3) Used to develop system based on OOP concept
4) Used to build decision support systems

- Commonly used logical programming languages are PROLOG, ASP, Datalog.

* Meanings -

1) Terms :

- Prolog term is constant, variable or structure
- Constant is either atom or integer
- Variable is any string of letters, digit, etc

2) Goals :

- Goal is a statement starting with a predicate and followed by its arguments
- Purpose of submitting goal is to find out whether statement represented by goal is true according to knowledge database.

3) Fact :

- Fact must start with predicate and end with fullstop
- Syntax of fact is,
  pred(arg1, arg2, ... argN).
- Eg,
  man(dk).
  parent(anand, parth)
- Result is either true or false depending upon facts.
- Eg,
  ?- parent(anand, parth)
  True

4) Rule :

- Rule is extension of fact with added conditions that also have to be satisfied for it to be true.
- Rule is no more than a stored query
- Syntax is,
  head :- body

- Eg,
  father(F, C) :- man(F), parent(F, C)

★ List, Operator & Arithmetic -

→ ■ List -

- List is an ordered sequence of objects
- In prolog, list is written as its elements seperated by commas and enclosed in brackets

- Eg, [1, 2, 3, 4]

■ Operators -

- Predicate calculus is fundamental notation for representing and reasoning with logical statements.

1) Propositional Symbols - P, Q, R, S, T ... denote propositions about world that may be True or False

2) Truth symbols - true, false
3) Connectives - ∧(and), ∨(or)
4) Propositional sentences - $p \wedge q$
5) Conjucts & Disjuncts.
6) Symbols - () and []

★ List Structures / Manipulation -

→ • List is ordered sequence of objects
• Eg, []
[a,b,c,d]

• The list contains Head and Tail part.
Eg, if we type following query
?- [H][T] = [10,20,30,40]

We get,
H = 10
T = 20, 30, 40

* Difference - Functional v/s Logical

Functional Programming	Logical Programming
i) It uses functions	i) It uses predicates
ii) Functions can return a value	ii) Predicate cannot return value, only true & false
iii) Variable denotes a concrete value	iii) Variable do not need to refer to concrete value.
iv) Normally used for modelling reasoning	iv) Normally used for modelling knowledge
v) Eg, ML, Haskell	v) Eg, PROLOG, ASP

My Notes Are Free Of Cost ☺

But As a Act of Kindness

You can give me some money
if you liked my Notes and
has helped you

if (money >= 50)

then only TRUE , xD ☺

SPPU-SE-COMP-CONTENT – KSKA Git