Java programming

```
class firstprogram {
                  → class name
    public static void main(String[] args) {
    System.out.println("Hello world"); } }
```

→ Basic Question ;

1. Write a program ~~print~~ to get the following Output:
Hey there,
I am data!

```
Class print {
    public static void main (String [] args) {
    System. out. println ("Hey there, \n I am data");
    } }
```

2. Write a program to print the sum of no. 2, 4 and 5.

```
class Sum {
    public static void main (String [] args) {
System. out. println (2+4+5); } }
```

```
class datatype {
public static void main (String [] args) {
int x = 10 ;
double y = 10.23 ;
char z = 'z';
System.out.println("int : "+x+"\nfloat :"
+y+"\nchar : "+z);
```

Q Print the ~~ASCII~~ ASCII value of the character 'h'.

```
class ASCIIvalue {
public static void main (String[] args) {
System.out.print((int) 'h'); }}
```

Write a program to add 3 to the ASCII value of the character 'd' and print the equivalent character.

```
class New {
    public static void main(String[] args){
        int x = 3+ 'd';
        system.out.println((char)x); }}
```

Find power :-

```
import java.lang.Math;

class Power {

public static void main (String[] args) {

    ~~System.pr~~in
System.out.println (Math.pow(7,3)); }}
```

```
println ("  " + a);
```

Page No.
Date

// Input from user

```
import java.util.Scanner;

class Input {

  public static void main(String[] args){

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a no. : ");

    int a = scanner.nextInt();

System.out.println("You entered" + a);

  scanner.close(); }}
```

// Create array

```
int[] arr ;
```

// declare an array allocated memory space

```
int[] arr = new int[5];
```

// Another way to declare

```
int[] arr = {1, 2, 3, 4, 5};
```

use string :-

String a = "Helloworld";

Input methods of datatypes

1. int a = scanner.nextInt(); // integer
2. char a = scanner.next().charAt(0); // character
3. float a = scanner.nextFloat(); // Float
4. double a = scanner.nextDouble(); // Double
5. short a = scanner.nextShort(); // Short
6. int arr[0] = scanner.nextInt(); // array

Primitive Data types

default value '0'

- Byte → 1 byte — range -128 to 127
- short → 2 bytes — range = $2^{((2\times8)-1)}$ to $2^{((2\times8)-1)-1}$
- int → 4 bytes — range $-2^{((2\times8)-1)}$ to $2^{((2\times8)-1)-1}$
- long → 8 bytes
- float → 4 bytes — storing 6 to 7 decimal digit
- double → 8 bytes — storing 15 decimal digit
- boolean → 1 bit — // use smallest size
- char → 2 bytes — // store a single character

default value false

ASCII - American standard code for Information Interchange

In C language char size is 1 byte because its store only character but in java we access the data by unique code therefore char size is 2 bytes.

Primitive data type

- Integral
  - byte
  - short
  - int
  - long
- floating
  - float
  - double
- character (char)
- boolean

In float and long we and f in end of the float value and l in the end of the long.

Ex : float = 46.3f;
long = 5634568928L;

Literals :

A constant value which can be assigned to the variable is called as a literal.

→ Types of operators

i) Arithmetic operators → +, -, *, /, %, ++, --
ii) Assignment operators → =, +=
iii) Comparision operators → ==, >=, <=
iv) Logical operators → &&, ||, !
v) Bitwise operator → &, | (operates bitwise).

Arithmetic operators cannot with booleans.

Ex – b = 2;
b += 3;
$\boxed{b = 5}$

Precedence and Associativity

Operator	Associativity	Precedence
++, --	Right to left	13
*, /, %	left to Right	12
+ -	left to Right	11

Highest precedence is 14. Ex: Function call ()
Array Subscript [], Dot ., Arrow →

* - STRING - *

A string is sequence of a character.

```
String name;
name = new String("Hello");
```

String is a class but can be used like a data type :-

```
String name = "Harry";
```

Reference → object

String are cannot be changed.

* Different ways to print in Java :-

System.out.println() → New line at the end
System.out.print() → no new line at the end
System.out.printf()
System.out.format()

→ Its behaves like C function

Ex System.out.printf("%c", ch);

```
String name = "Dhruv";
```

0 1 2 3 4

1. name.length() → Returns length of string

Ex. int value = name.length();

Escape sequence character :-

\n , \t , → tab
↳ newline

2. name.toUpperCase() = Convert the string into Uppercase.

Ex. int a = name.toUpperCase();

3. name.toLowerCase() = Convert the string into lower case

4. name.trim() = Used to remove spaces from string.

5. name.replace('char', 'char')
Old char New char

Used for replace the character to another character.

Ex :- String name = "hello";
System.out.print(name.replace(l,k));

Output : hekko

6. name.charAt(index) → return character at given index.
int

Ex → String name = "hello";
System.out.print(name.charAt(1));

Output : e

D.K

Page No.
Date

INPUT IN STRING

Two way of input

1. String a = scanner.next();

In this way in string store only first word not be store after space word.

Example :

String a = scanner.next();

Input :- Hello How are you.

Output :- Hello

↳ only ~~single~~ store single word.

2. String a = Scanner.nextLine();

This store all the string

Ex.

String a = Scanner.nextLine();

Input : Hello How are you

Output : Hello How are you

Write a program in java for replace spaces with underscore :-

```
class Replace {

  public static void main (String[] args) {

    String a = "Hello, How are you" ;
System.out.println( a ) ;

    a = a.replace (
```

For - each loop (For array traversal)

~~int[] a = new int [6]~~

```
int [] arr = { 5, 8, 9, 6, 4 }

  for ( int element : arr) {
                ham isko change kar sakte
  system.out.print(element); }   isko name a,b,c kuch bhi rakh sakte
```

Output :- 5 8 9 6 4

Note : for each loop me ham ak element Rakhte jo array me se value uthata jata h.
se java ki taraf se ak feature jisse ham array traversal ko fast kar sakte.

D.K

Page No.
Date

Method or function in java

```
class Main {

static void or int name ( ) {
    // program
}

~~static void~~ public static void main (String[] args)
    name ( ) ;
}
```

Parameter

call to function

argument

Example :- Write a sum program with the help of method ;

```
class Used-Method {

Method return Integer → static int Sum (int a, int b) {
        int sum = a + b ;
        return sum ; }

main class → public static void main (String [] args) {
    call to Method → Sum (4, 5) ;
    System.out.print (sum) ; } }
```

Parameter

arguments

The values 4, 5 are copied and then sum = a + b is return in sum which is an integer.

Note: In case of Array, the reference is passed. Same is the case for object passing to methods.

Example:-

```
class Used_fun {

static int sum (int a){
        int a = 85 ; }
public static void main (String[] args){
        int a = 5 ;
        sum (a) ;
System.out.print (a) ; } }
```

Output : 5

↳ Because in integer function not be the change the value of in integer but in array we pass the address therefore array value will be changed.

Page No.
Date

→ **METHOD OVERLOADING**

Two or more methods can have same name but different parameter. such methods are called Overloaded methods.

Ex :-

```
void name ( )
void name (int a)
int name (int a, int b)
```

→ **Variable arguments** → Varargs

```
class Main {

static int sum (int ... arr) {
    int result = 0;
```

→ use 3 dot, used for handle many no. of inpu

for each loop ← `for( int a : arr) {`

```
    result + = a; }
    return result; }

public static void main (String[] args) {
System.out.print (sum (2, 1, 5, 8, 9)); } }
```

Output = 25

Variable arguments used handles the many no. of inputs. its store all the arguments.

Parameter :- It is like a blank slot in a function or program that can be filled with different values.

Argument :- It's the actual value you into that blank slot (parameter) when using the function.

Recursion

A function in java can call itself. Such calling of function by itself is called recursion.

Example :- Factorial of a no.

factorial (n) = n* factorial (n-1) ;

Switch case

Syntax :-

```
switch (choose) {
case 1:
//code
break;

default:
break; }
```

choose is variable you can change this.

if choose value = 1 case 1 execute

value = 2 then execute case 2

then other value execute default fn

Page No.
Date

OOP

Solving a problem by creating objects is one the most popular approaches in programming. This is called Object Oriented Programming (OOP)

DRY → Don't Repeat yourself

- // focus on code reusability

CLASS

A class is a logical template to create objects that share common properties and methods.

Class ⇒ Object Instantiation ⇒ Object

The processes of creating a new object for a class using a new keyword

OBJECT

Object is an instantiation of a class.

~~Where a class~~ A java object is a member of a class.

→ Each object has an identity, a behaviour and a state.

→ The state of an object is stored in variables
→ While methods (function) display the objects behaviour.

ATTRIBUTES :-

Attributes are data members inside a class or an object.

Example :-

Additional
class // Class Employee {

Attributes // Int name, age, salary ;

Method // void getsalary() {
print (name, age salary) ;
}

public class Main {

Main // public static void main (String [] args) {

Method call // get salary () ; } }

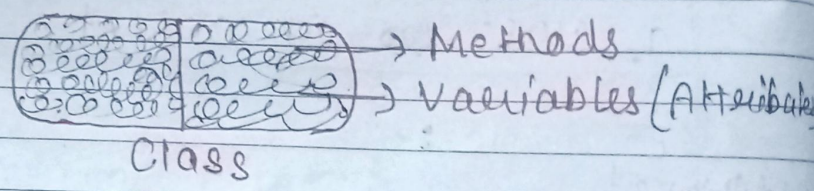
Page No.
Date

OOPs Terminology :-

Abstraction :- Hiding internal details

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Encapsulation :- The act of putting various components together (in a capsule).



In Encapsulation, a class variable are hidden from other classes and can only be accessed by the methods of the class.

Inheritance : Creating new classes based on existing ones.

Ex - Rickshaw ⇒ E-Rickshaw
Phone ⇒ Smart phone

Ex → Smartphone → Calculator, torch, Radio, Phone, Music player

Polymorphism :- Allow us to perform a single action in different ways.

polymorphism allow you to define one interface and have multiple implementations.

Writing a Custum Class

```
public class Student {
    int Roll_no. ;
    int Section ;
    string Name ;
}
```

Attributes

Class with methods :-

```
public class student {
    int Roll no.
    string class
    public void print () {
        System.out.println ( Rollno.+ class);
    }
}
public class Main {
    public static void main (String[] args) {
        print ();
    }
}
```

Page No.
Date

Access Modifiers

There are four types of access modifiers in Java :-

- Private
- Public
- Default
- Protected

Getters and Setters

Accessor → Getter → Return the value
Mutator → Setter → set/updates the value

Accessor and Mutator

→ Accessor methods, also known as getter methods are methods that allow you retrieve the value of an object's private instance variable. (These methods provide read only (Print) to the object state)

→ Mutator methods, also knowns as setter methods that allow you to modify the value of an object's private instance variables.
(This methods provide that write-only access to the Object's state.)

→ Example : getter and setter

```
public class Student {

    private String name;

    public String getname() {
        return name; }

    public void setname (int name) {

        this.name = name; } }
```

Constructors in Java

A constructor is a special type of method, whose name is same to class name.

It run automatically without to called.

Constructor Overloading

Constructors can be overloaded just like other methods in java

because can create many no. of constructor with different parameters.

Page No.
Date

Example of constructor

```
new class — class Student {

        private String name ; ⇒ Attribute

Getter ⇒ public String getname () {
            return name ; }

                                Parameter
Setter ⇒ public void setname ( String name) {
            this.name = name ; }

        Student() {
Constructor
  ⇒ System.out.print (" I am a Constructor ") ; }

public class Main {
    public static void main (String[] args) {

        Student Ram = new Student () ;

        String name = "Ram" ;
                                Argument
Call to function ← Ram.setname ( name ) ;

System.out.print ( Ram.getname () ) ; }}
```

Output : I am constructor
Ram

Inheritance :-

Inheritance is used to borrow properties and methods from an existing class.

Phone → Smartphone
↓ ↓
Superclass → Sub class

In inheritance we can extends the the and sub class to super class.

① If we extends the super class in sub class, then access the method, variable, properties of the superclass.

EXAMPLE :-

(Super class) → class vehicle {
public void key () {
System.out.print ("Start the vehicle");}

(Extend class) ← class Car extends Vehicle { }
(Sub class) → public void tyres () {
System.out.print ("Start the car");}}

class Main {
public static void main (String[] args) {
Car x = new Car ();
x.key(); → we use the super class methods in subclass.
x.tyres ();
}}

Page No.
Date

Constructor in Inheritence

When a create a contructor in superclass and subclass, then we access all the constructor in hierarchy

for, we create a C1 superclass and C2 is subclass of C1 and C3 is a subclass of C2 superclass.

C3 → C2 → C1
Grandchild Child Parent class

In we can access in the form of constructor up-down to upper.

Example :-

```
Class Principle {
        Principle () {
system.out.print ("This is Principle class") ; }}

class Teacher {
        Teacher () { super() ;
system.out.print ("This is Teacher class") ;
                }}
```

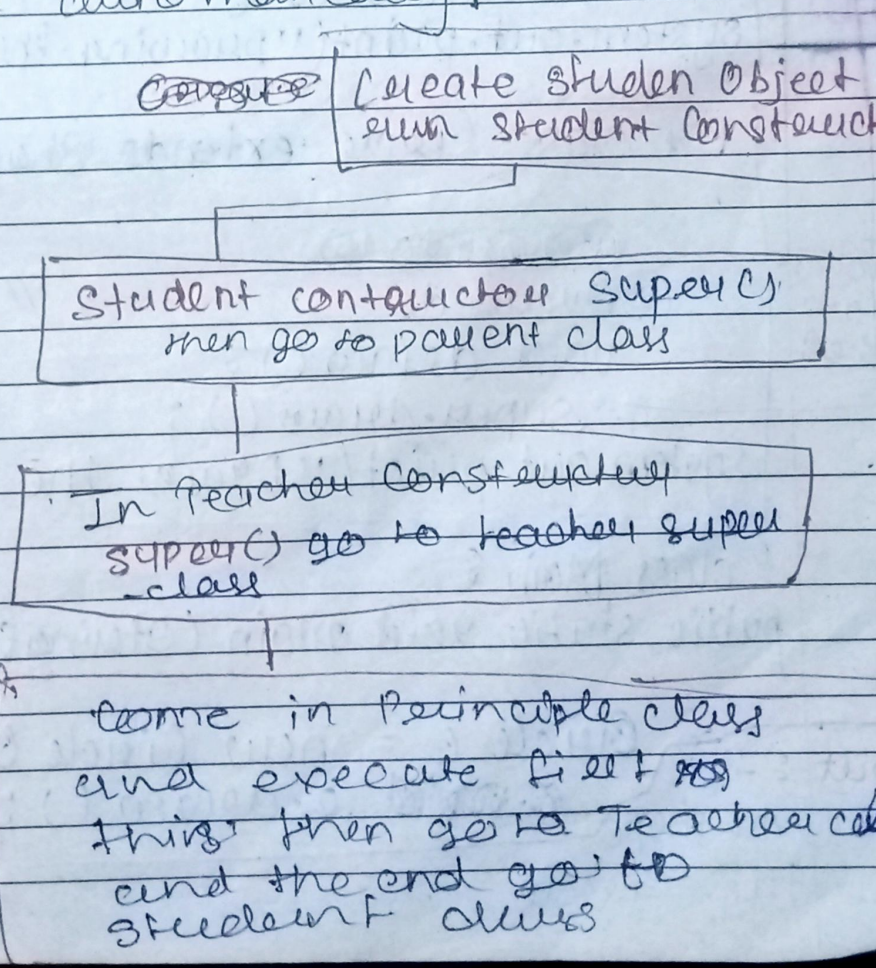
```
class Student {
Student () {
    super ();
system.out.print ("This is student Class") }}

class Main {

public static void main (String[] args) {

    Student s = new Student () ; }}
```

We can create a object of student class, then student constructor run automatically.



Page No.
Date

⇒ Super keyword

The Super keyword refers to superclass (parent) objects.

It is used to call super class methods, and to access the superclass constructor.

Example :-

```
class Shape {                    // super class
    void draw() {
        System.out.print("Drawing the shape");}}

class Circle extends Shape {     // sub class
    @Override
    void draw() {
        super.draw();
        System.out.print("Draw the circle");}}

class Main {
    public static void main (String[] args) {
        Circle c = new Circle();
        c.draw();  }}
```

call to superclass draw method

Output :-
Drawing the shape
Draw the circle

⇒ this keyword

The this keyword refers to the current object in a method or a constructor.

The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name.

Example :-

// The of para-meter and attributes are same therefore create a confusion then we use this.

```
class Print {
    private String name;
    attribute

    void setname (String name) {
        this.name = name;
        parameter
    // this define its variable comes from Print class
    } }

class Main {
public static void main (String [] args) {

    Print p = new Print ();

    String n = "Hello";

    p.name (n); } }
```

Page No.
Date

Dynamic method Dispatch

```
// Parent class
class Animal {
    void sound () {
System.out.print ("Animal makes sound"); }}
// child class
class Dog extends Animal {
    @Override
void sound () {
System.out.print ("Dog barks") ; }}
```

override the function create the same fn from superclass

Main class ←

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

    Animal animal 1 = new Dog () ;
```

Reference of animal class super class

object from dog class sub-class

Output :-

Dog barks

Abstract Classes

Abstract classes serve as a blueprint for classes and abstract methods.

define a common interface that must be implemented by concrete subclasses.

Desi - English

Simply, jaise manlo koi superclass h maine uske (method) aur (class) ko abstract bna diya, phir ab jo is class ki jitni bhi subclass hongi unko bhi same name se method over-ride karne padenge.

Abstract class ak blue-print ha baki sabhi class ka jo methods isme honge vo methods subclass ko bhi override karne padenge.

We can't be create the object of abstract classes, but we create create object of sub classes than neither be sub class is abstract

Page No.
Date

Abstract Example

```
abstract class animal {

    void eat () {
        System.out.print ("eat") ; }

    void sleep () {
    System.out.print ("sleep") ; }

    abstract void sound () ;  }

    class dog extends animal {
        void sound () {
        System.out.print ("Bark") ; } }

class main {

public static void main (String[] args) {

    dog pet = new dog () ;

    pet.sound () ;
    pet.eat () ;
    pet.sleep () ; } }
```

This is abstract method therefore its compulsorily to implement this method