Class 1

Java Programming

Date
Page

What is Java programming?

Ans:- Java is a popular programming language.
Java is high level programming language.
Java is object oriented and secure programming language.
Java is used to develop mobile app web apps, destop and games.

History of Java :-

⇒ Java was developed by James gosling who is known as father of Java in 1995.

⇒ The History of Java is very interesting Java was originally designed for interactive television, but it was too advanced technology for the digital cable television industry at the time.

Features of Java:-

⇒ (i) It is easy to learn language.
⇒ (ii) It is simple plogramming lang.
⇒ (iii) It is based on object oriented programming language.
⇒ (iv) It is portable.
⇒ (v) It is Robust.
⇒ (vi) It is Interpreted programming

=> (viii) It is distributed language.
=> (ix) It is secure language.
=> (x) It is platform independent language.
=> (xi) It is high performance language.
=> (xii) It is multithreaded language.
=> (xiii) Java is a case sensitive language.

Difference b/w Java and C++

=> Java

(i) Java is platform independent language.

(ii) Java is mainly used for application programming.

(iii) Java does not support goto statement.

(iv) Java does not support operator overloading.

(v) Java support only call by value.

C++

(i) C++ is platform dependent language.

(ii) C++ is mainly used for system programming.

(iii) C++ supports the goto statement.

(iv) C++ supports the operator overloading.

(v) C++ support both call by value and call by reference.

(v) Java does not Supports structure and union.	(vi) C++ Support structure and union.

⇒ Extension of Java file = .java

⇒ Extension of byte code = .class

Syntax :-

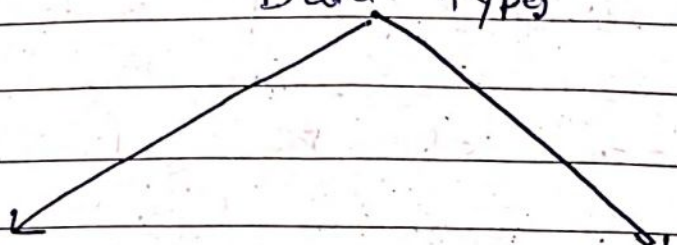
```
Public class main
{
    Public static void main (String [] args)
    {
        System.out.println ("Hello, world");
    }
}
```

Variables in Java :-

⇒ Variable are used to store the value in a single variable.

```
Public class main
{
    Public static void main (String [] args)
    {
        String name = "Ankit";
        int age = 30;
    }
}
```

Data types in Java:-



(i) Primitive data types:-

byte - 1 [-128 to 127 numbers ko store kar skate hai.

Short - 2 byte

int - 4 byte

long - 8 byte

Float - 4

double - 8

Char - 2

boolean - true / false - 1 byte

```
Public class main
{
    Public Static void main(String[] args)
    {
        byte age=30;
        int phone = 12345678,90;
    }
}
```

(11) Non primitive types:-

```
String name = "Aman";
System.out.println(name.length());
```

String:-

// Concatenate

```
String name1 = "Ankit";
String name2 = "Raj";
String name3 = name1 + name2;
System.out.println(name3);

name1 + " and " + name2;
```

// CharAt

```
String name = "Ankit";
System.out.println(name.charAt(0));
```

// replace

```
String name.replace('a', 'b');
```

=) Java is a immutable types

// SubString

```
String name = "Ankit and Anshu";
System.out.println(name.substring(93));
```

→ total 68 Keywords are in Java.

⇒ Byte code Provide Security because the code is not visible.

⇒ Java is a platform independent with the help of byte code. [magic of Java]

⇒ It follows oops properties.

⇒ In Java when the program is written inside the class then it is known as method and when the program is written outside the class, then it is known as function.

⇒ Is Java can be pure object oriented language?

⇒ No because it involves primitive datatypes such as int, char, float.

⇒ To overcome the problem of primitive datatypes.
We can create wrappers classes [First letter should be capital of

Such as Float, Integer, Double, Boolean, Short, long, char.

Program 1:

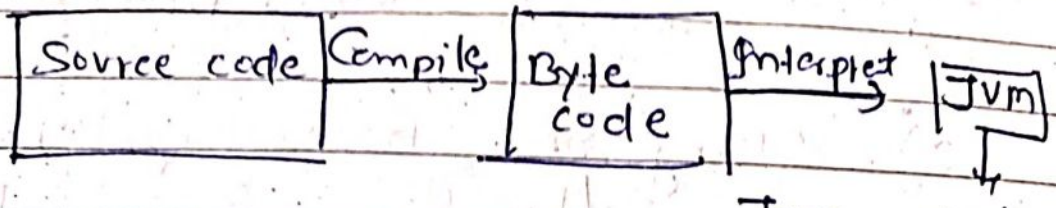
```
Public class ABC
{                                    Predefined class.
    Public static void main(String args[])
    {
        System.out.println("Hello");
    }
}
```

Note:-

⇒ When class will be public so, the class name should be save as Class name.

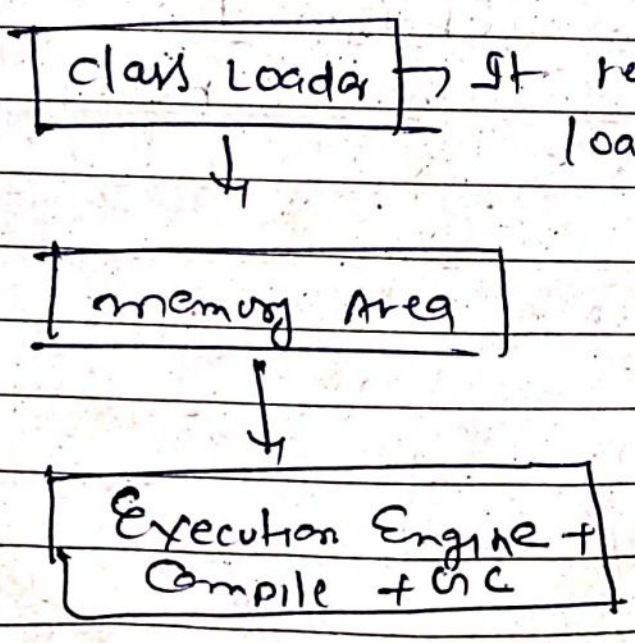
⇒ But, when there is no public, So the class name not mandatory For saving a file.

Java Arechitecture :-



- (i) Jvm (Java virtual machine)
- (ii) JRE (Java runtime Environment
- (iii) JDK (Java development kit)

(i) Jvm



* JRE → Java runtime Environment

JRE = JVM + Libraries
provide Libraries.

* JDK → Java development kit
JRE + Javac, debugging, Java doc
monitoring tools

Difference between JDK, JRE, JVM

JVM	JRE	JDK
(I) It stand For Java virtual machine.	(I) It stand for Java runtime Environment	(I) It stand for Java development kit.
(II) It is used for execute byte code.	(II) JRE transfer Library to JVM.	(II) JDK is composed of tools that are required.
(III) Class Loader + memory Area + execution Area	(III) JRE is JVM + runtime Libraries	(III) JDK is JRE + development tools.

Ankit Rej

Lec-2

Java Variables and Data Types:-

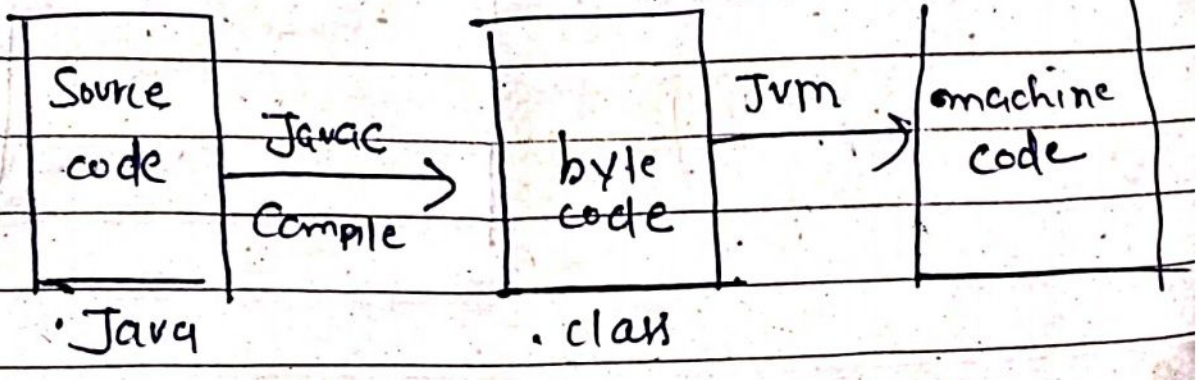
- Java output
- Java Variable
- Java identifiers
- Java Datatypes

① Java output:-

- System.out.println();
- System.out.print();

First program:-

```
class A
{
  Public Static void main(String args[])
  {
    System.out.println("Hello world");
  }
}
```



Public static void main (String [] args)
{
}

- Public → access modifier
- static → allow to call this method without creating an object of code.

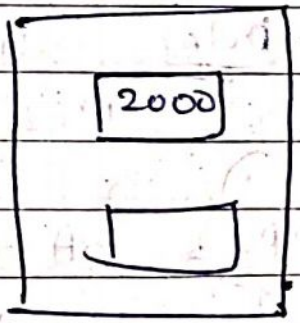
void :→ return type
main → function name
String [] args :→ Command line arguments.

System . out . println()
↓ ↓ ↘
class object of method in
Print Stream print stream class

Java variable :-

Java variable are title of reserved memory location.

age = 18;



Syntax for Declaring a Java variable.

Type variable - name [= value];

int mony = 1000;

String

Java Naming Convention

- (i) Lowercase
- (ii) Uppercase.
- (iii) Camelcase.

Java में जो भी function और variable होते है उन्हे Camel case में लिखते है।

(iv) Pascal case
College Wallah.

Rules for Naming variables in Java

Rule 1: Variable names should not begin with a number.

Rule 2: Whitespace is not permitted in variable names.

Rule 3: A Java keyword cannot be used as a variable name.

Rule 4: When creating variable, it is preferred to give them meaningful names.

Rules All lowercase letters should be used when creating one word variable name.

Java Identifiers :-

→ Java identifiers core name any class package, function module that is known as identifiers.

→ Just name for any component in Java is called Identifiers.

Points to remember about identifiers:-

Rule1 :- All identifiers should begin with a letter (A to z or a to z), currency, character or an underscore.

Rule2. After the first character, identifiers can have any combination of character.

Rule3. A keyword cannot be used as an identifiers.

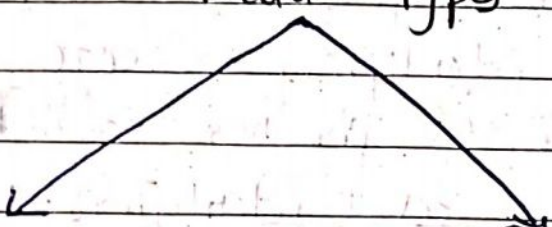
Rule4. The identifiers are case sensitive.

int money = 200

Rule5. - Whitespace is not permitted.

Java Data types

⇒ There are two types of data types in Java



① Primitive data types
↓
→ built in Java
→ Cannot be divide
→ hold single value

e.g → boolean
char
byte → 1 byte or 8 bits
↳ -128 to 127
↳ Range
Short
int
long
float
double

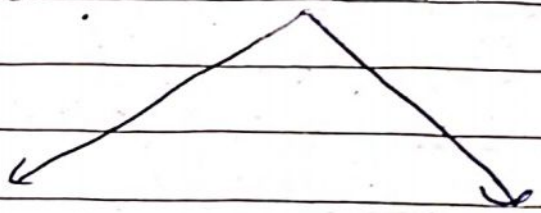
Short ← Size 2byte/16bits

② Non primitive data types.
↓
→ It is also known as User defined data types.
→ memory address of a variable.

→ Strings
→ Arrays
→ classes
→ Interface

Lec-11

Introduction to OOPs in Java:-



① Class

↓

→ class ij the user defined datatype.

→ class ij blueprint and templa

→ objects are the instances of the class.

→ objects ij the real world quantity.

Characterstics of object:-

3 characteristics ɣ object:-

① Identity ij the name ɣ object

② state → attribute → red → model

③ Behaviour → method → accelera

Create a class

- Access modifier
- class keyword
- class name
- Body name.

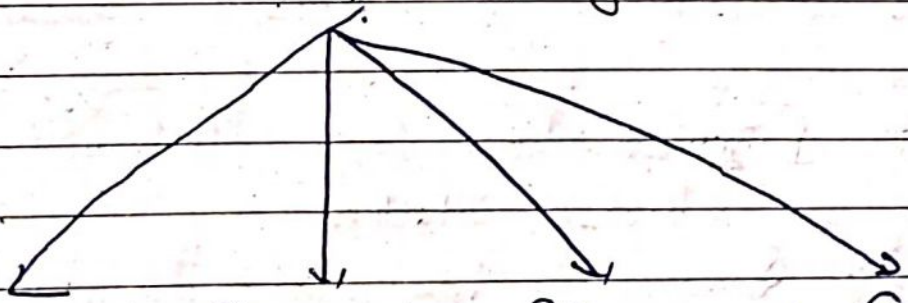
Syntax of class create in Java

```
Public class Car
{
    // class body
};
```

Access modifiers:-

→ Access modifiers of the property and method that can be used to control the visibility of field, methods and constructors in a class.

The four types of Access modifiers in Java



(i) Public → Road

(ii) private → phone

(iii) protected → assets like property

(iv) default → Tvs within home.

Lec-12

Java methods & function :-

→ Java method is the block of code performance some action which runs only when it is called.

Why are methods important in Java?

→ write once, and reuse many time
→ time Save
→ duplicate code reduced

What is types of method in Java?

→ There are two types of method in Java
(i) user defined methods
(ii) Standard Library methods.

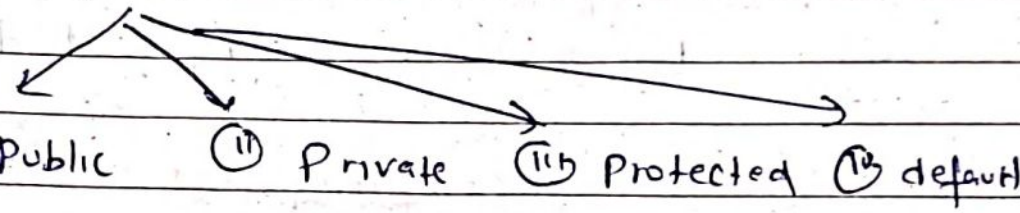
How to declare methods in Java.

```
Public class main
{
    Public int Sum (int a, int b)
    {
        // code
    }
}
```

(1) method signature :- · Sum (int a, int b)

method name + Parameter list this called as method signature.

(2) Access specifies.



(3) Return type :- int

(4) method name :- Sum

(5) Parameter : (int a, int b)

(6) method Body

Call a method

```
Public class main
{
    Static void welcome()
    {
        SOP("welcome to BCIIT");
    }
    Public static void main(String[] args)
    {
        welcome();
    }
}
```

21/02/2024

String in Java:-

```
class String
{
    Public Static void main(String args[])
    {
        String name = " Ankit";
        String fullnm = " AnkitRaj";
        String sentene = " mynam is ting";
    }
}
```

Input From user in Stng:-

```
import java.util.*;
class String
{
    Public static void main(string args[])
    {
        Scanner sc = new Scanner (System.in);
        String name = sc.next();
        System.out.println(" Your name is:"+
            name);
    }
}
```

→ take all line → Ankit Raj → nextLine();

Concatenation:
It is used to Combine the two string into single strings

```
class string
{
  Public static void main(String[] args)
  {
    String firstname = "Ankit";
    String lastname = "Raj";
    String fullname = "firstname + " "
                      + lastname;
    System.out.println(fullname);
  }
}
```

length():⇒ length function are used to find length of string

```
class Strings
{
  Public static void main(String[] args)
  {
    String Firstname = "Ankit";
    String lastname = "Raj";
    String fullname = "~~Ankit~~ firstname+
                      lastname;
    System.out.println(fullname.length());
```

CharAt :-

```
class Strings
{
    Public static void main(String arg[])
    {
        String firstname = "Ankit";
        String lastname = "Raj";
        String fullname = "Ankit Raj";
                          firstname+lastname;
        System.out.println(fullname.length());

        For (int i=0; i< fullname.length();i++)
        {
            System.out.println(fullname.charAt(i));
        }
    }
}
```

=

Compare of two string

//1. $S_1 > S_2$: +ve value

//2. $S1 == S_2$: 0

//3. $S1 < S_2$: -ve value

Array in Java:-

→ Array is the collection of homogenous data type.

→ Arrays are used to store multiple values in a single array.

Defining an array.

type [] arrayName = new type [size];

Syntax

↳ (i) int [] marks = new int[3];

(ii) int marks[] = new int[3];

Program:-

```
Public class array
{
  Public static void main (String args[])
  {
    int [] marks = new int[3];
     marks[0] = 97;
     marks[1] = 98;
     marks[2] = 95;
    System.out.println (marks[0])
    System.out.println (marks[1]);
  }
}
```

Apna college

OOPS in Java

Date
Page

```
Public class oops.
{
    class pen
    {
        String color;
        String type;
        Public void write()
        {
            System.out.println("Writing something");
        }
    }
    Public static void main(String args[])
    {
        Pen pen1 = new Pen();
        pen1.color = "blue";
        pen1.type = "gel";
        pen1.write();
    }
}
```

this Keywords :-

→ The this Keyword can be used to refer current class instance varable. If there is ambiguity b/w the instance varable and paramdes, this keyword resolves the problem of ambiguity.

→ this can be used refers current class instance variable.

→ this can be used to invoke current class method.

→ this can be used to invoke current class constructor.

→ this can be passed as an argument in the method call.

Example :-

```
class pen
{
    String color;
    String type;
    Public void display()
    {
        System.out.println(this.color);
        System.out.println(this.type);
    }
}
class oops
{
    Public Static void main(String
```

S

```
Pen ob1 = new pen()
ob1. color = "yellow";
ob1. type = "gel";

Pen ob2 = new pen()
ob2. color = "black";
ob2. type = "ballpant";
}
}.
```

Constructors in Java:-

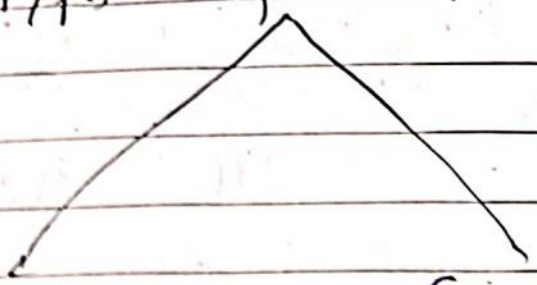
→ Constructors in Java A constructor is a block of code similar to the method. It is called when an instance of the class is created.

→ Constructor name must be the same as class name.

→ Constructor are used to initialize the data members.

→ Constructor are this automatically called.

Type of Constructor in Java



(i) Default Constructor (ii) Parameterized

(i) Default Constructor :-
Default Constructor when it does not have any parameter.

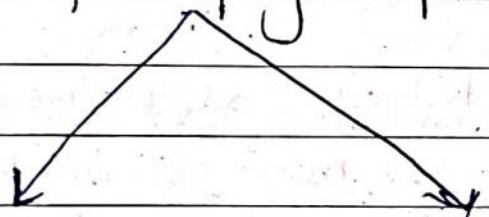
```
class Student
{
    int rollno;
    String name;
    public void display()
    {
        System.out.println(this.color);
        System.out.println(this.type);
    }
}
public class oop
{
    Pub
```

Polymorphism :-

↳ poly + morphism
many forms

→ polymorphism mean many forms.
⇒ polymorphism in java can be defined as the ability of an object to take many forms.
⇒ This helps us perform the same action in different ways.

Types of polymorphism :-



(i) Function overloading → Compile time polymorphism

(ii) Function overriding → Run time polymorphism

(i) Function overloading :- Function overloading in java when multiple function have a same name but different parameters are passed that is called function overloading.

Inheritance:-

→ Inheritance in java is a machanism in which one object acquire all the properties and bhavours of a our parent object.

→ Inheritance allows us to define a class that inherits all the method and properties from another class.

~~Types of Inheritance~~:-

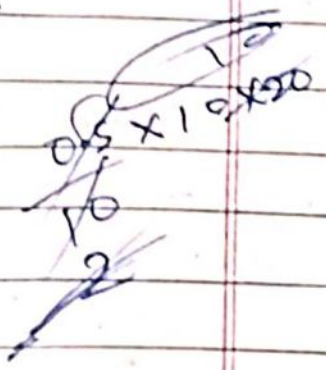
Example:-

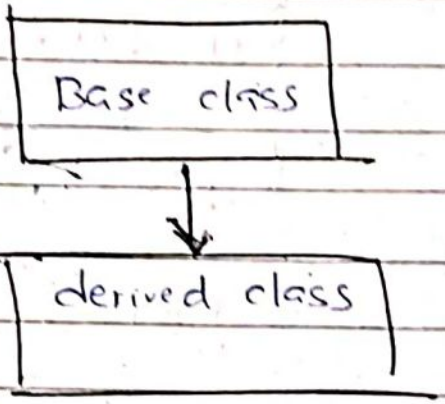
```
class Shape
{
    String color;
}
class Triangle extends Shape
{
}
Public class oops{
    public static void main(String args
    {
        Triangle t1 = new Triangle();
        t1.color = "red";
    }
}
```

Types of Inheritance

① Single Level Inheritance :-

→ In single level inheritance acquiry 1 base class and 1 derived class



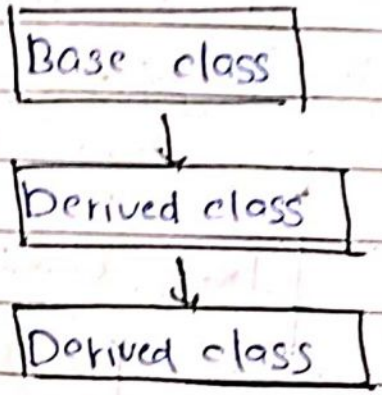



```
class shape{
    public void area()
{
    System.out.println("displays area");
    }
}

class Triangle extends shape
{
    Public void area(int l, int h)
    {
        System.out.println(1/2*l*h);
    }
}

Public class oops{
    Public static void main(String args[])
    {
    }
}
```

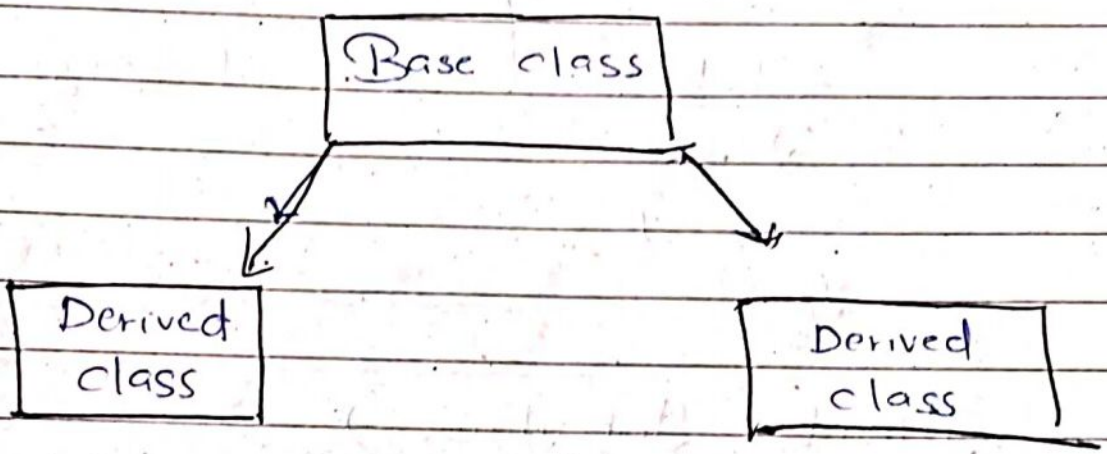
(II) Multi Level Inheritance :-




```
class Shape{
    Public void area()
    {
        System.out.println("displays area");
    }
}
class triangle extends shape
{
    Public void area(int l, int h)
    {
        System.out.println(1/2*l*h);
    }
}
class Equilateral Triangle extends triangle
{
    Public void area(int l, int h)
    {
        System.out.println(1/2*l*h);
    }
}
```

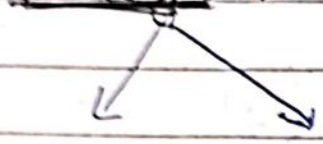
```
Public class oops
{
    Public static void main(String args[])
    {
    }
}
```

(III) Hierarchical Inheritance :-




```
class shape{
    Public void area()
    {
      System.out.println(".displays area");
    }
}
class Triangle extends shape{
    Public void area(int l, int h)
    {
        System.out.println(1/2*l*h);
    }
}
```

Packages :→



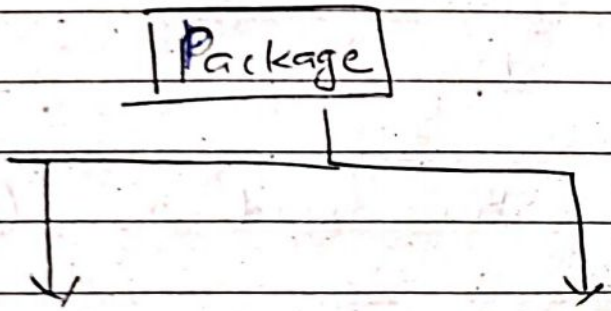
Inbuilds Package

User defined Package.

→ A package in Java is used to group related classes. Think of it as a folder in a file directory.

→ A Java package is set of classes, interfaces and sub packages that are similar.

→ A set of classes and interfaces grouped together are known as package in Java.



Interface:-

⇒ How to declare Interface in Java.

interface

* method ⇒ abstract void add(int a, int b);

Ex-

```
void add(int a, int b);
```

* method → Public abstract hoga.
* Variable By default public static, final.
* interface of

```
void add(int a, int b);
void sub(int a, int b);
```

* In interface private method is allowed.
* interface a

```
{
    private void add(int a, int b)
    {
        SOP(a+b);
    }
}
```

Java Programming

Date
Page

Q दर्ज Abstract class के अंदर At least one Abstract class होना Compulsory है

```
abstract class bf
{
}
class a
{
    public static void main (String aaa[])
    {
        System.out.println ("hello");
    }
}
```

Q. दर्ज है Abstract class का कोई object create कर सकते है /

Q Difference b/w abstract class and Interface.

Q Can we create an interface inside another interface.

⇒ Yes

Q Can an interface inherited another interface.

⇒ Yes

Q Can an interface inherited in class

⇒ No

Static keywords :-

⇒ The static keywords in java is used for memory management mainly.

⇒ We can apply static keyword with variables, methods, blocks and nested classes.

Static
- (i) static variable
- (ii) Static method
- (iii) Static block.

(1) Static variable :-

⇒ If you declare any variable as static it is known as a static variable.

⇒ The static variable can be used to refer to the common property of all objects for example the company name of employees, college name of student.

For example:-

```
class Student
{
    int rollno;
    String name
    static String college = "BCIIT";

    Student (int rn, string n)
    {
        rollno = rn;
        name = n;
    }
    void display()
    {
    System.out.println(rollno+" "+name+" "+
                                    college);
    }
}
Public class staticvariable
{
    public static void main(String args[])
    {
    Student s1 = new Student(1, "Ankit");
    Student s2 = new Student(2, "Aman");
    s1.display();
    s2.display();
    }
}
```

(II) Static method:-

→ If you apply static keyword with any method it is known as static method.

→ A static method belongs to the class rather than the object of a class.

→ A static method can access static data member and can change the value of it.

```
static void change()
{
    college = "TIPS";
}
```

(III) Static block:-

→ Static block is a set of instructions that is run only once when a class is loaded into memory

```
Public class 9
{ static
  {
      // static block
      System.out.println("Hi I am a static");
  }
}
```

Final Keywords:-

→ The final keyword in Java is used to restrict the uses

→ The Java final keyword can be used in many context.

→ Final can be variable method class

→ Final keyword

- (I) final instance variable → class define
- (II) final static variable → final fi
- (III) final local variable → functio
- (IV) final class
- (V) final method.

(I) final instance variable

```
Public class Example
{
    Private final int x;
    Example()
    {
        x=10;
    }
    Public static void main(String[] args)
    {
        Example e1 = new Example();
    }
}
```

→ final variable can be assigned only once.

(II) final static variable:

```
Public class Example
{
    Private final static int y;
    Static
    {
        y=20;
    }
    Public static void main(String[] args)
    {
        Example e1 = new Example();
    }
}
```

Super keywords :-

→ The super keyword in Java is a reference variable which is used to refer immediate parent class object.

→ Super can be used to refer immediate parent class instance variable.

→ Super can be used to invoke immediate parent class method.

→ Super can be used to invoke immediate parent class constructor.

```
class Animal
{
    String color = "white";
}
class dog extends Animal
{
    String color = "black";
    Void display()
    {
        Sop (color);
        Sop (super.color);
    }
}
```

Interfaces and packages:-

→ An interface in Java is a blueprint of a class.

→ It has static constants and abstract method.

→ Interfaces cannot be used to create as object.

Syntax:-

```
Interface < Interface_name>
{
    //
    //
    //
}
```

→ All the field in interface are public, static and final by default.

→ All methods are public & abstract by default.

→ Interfaces supports the functionality of multiple Inheritance.

→ Interface do not have constructor.

Difference between abstract class and interface.

Abstract class	Interface
(i) Abstract class does not support multiple inheritance.	(i) Interface supports multiple inheritance.
(ii) Abstract class can have final, non final, static and non static variable.	(ii) Interface has only static and final variable.
(iii) The abstract keyword is used to declare abstract class.	(iii) The interface keyword is used to declare interface.
(iv) A java abstract class can have class members like private, protected.	(iv) members of a java interface are public by default.
(v) Abstract class can have abstract and non abstract methods.	(vi) Interface can have only abstract methods.

Wrapper classes

→ Wrapper classes provide a way to use primitive datypes as objects

→ The wrapper class in java used to convert perimitive data types into objects.

Perimitive type	Wrapper class
boolean	Boolean
char	Character
byte	Byte
Short	Short
long	Long
float	Float
double	Double

Example:-

```
public class WrapperExample
{
    Public static void main(String[] args)
    {
        Int primitiveInt = 10;
        Integer wrappedInt = Integer.valueOf(
                                primitiveInt);
```

Method overriding in Java:-

```
class vechicle
{
    void run()
    {
        System.out.println("Vechicle is running");
    }
}
class Bike extends vechicle
{
    void run()
    {
        System.out.println("Bike is running safely");
    }
}
class oop
{
    Public Static void main (String args[])
    {
        Bike obj = new Bike();
        obj.run();
    }
}
```

→ Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.

→ method overriding is used for runtime polymorphism.

Exception Handling:-

⇒ Java exception Handling is a mechanism to handle runtime error such as classNotFoundException, IOException, SQL Exception, RemiteException etc.

```
Public class main
{
    public static void main (string[] args)
{
    int [] marks = {10, 40, 70};
    try {
      System.out.println(marks[5]);
    }
    catch ( Exception , e)
    {
        //
    }

    System.out.println(" The name is Ankit");
    }
}
```

Multi-threaded programming

- → multithreading fundamentals.
- → thread class and runnable interface
- → the life cycle of thread
- → create of single and multiple threads.
- → implementation of thread methods
- → Synchronization (using Synchronized methods, Synchronized statement).

⇒ Introduction to thread

→ A thread is an independent path of execution within a program.

→ Multithreading refers to two or more tasks executing concurrently within a single program.

→ Every thread in Java is created and controlled by the java.lang.Thread class.

InputStream()

Public abstract int read()
Public int read(byte[])
Public int read(byte[], int, int)
byte[] readBytes()
Public byte[] readNBytes(int)
Public long transferTo(java.io.OutputStream)
Public void close()

byte[] readAllBytes()

OutputStream()

Public void write(int)
Public void write(byte[])
Public void write(byte[], int, int)
public void flush()
public void close()

Public, java.io.FileInputStream(java.lang.String)

java.io.FileInputStream

Paper

Page No.

Java Programming :-

Swings

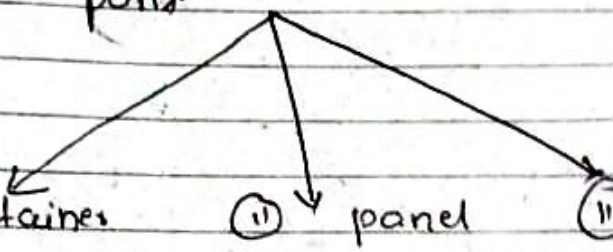
- Introduction to GUI

→ It stand for Graphical user Interface.

→ using awt

→ using swing

Graphical user Interfaces are divided into three parts.



Java Swing is a part of Java foundation classes that is used to create window based applications. It is built on the top of AWT API and entirely written in Java.

→ Java Swing provides plate form independent and lightweight components.

→ The javax.swing package provides classes for Java Swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckBox, JMenu, JColorChooser etc.

⇒ Differentiate b/w AWT and Swing

AWT	Swings
(i) AWT components are plateform dependent.	(i) Java Swing components are plateform independent
(ii) AWT components are heavyweight.	(ii) Swing components are lightweight.
(iii) AWT provided less components than swing	(iii) Swing provided more powerful components.
(iv) AWT does not follow MVC	(iv) Swing follows MVC.
(v) It is an API used to develop window based application.	(v) Swing is a graphical user interface to develop and design the application.

Date : / /
Page No.

#
```
import javax.swing.*;
class Login
{
Public Static void main(string[] args)
{
    JFrame J1 = new JFrame();
    J1. setVisible(true);
    J1. SetDefaultCloseOperation(JFrame.EXIT_ON_
                CLOSE);
}
}
```

→ J1. SetSize(500, 500);

#
```
import javax.swing.*;
class Login extends JFrame
{
    Login(string s1)
    {
        Super(s1);
    }
    Login()
    {
    }
Public static void main(string[] args)
{
    Login S1 = new Login("welcome to bc
}
}
```

Date : / /
Page No

Unit - 04

Swings fundamentals :-

```
#. import javax.swing.*;
   class Login extends JFrame
   {
     Login (String s1)
     {
       super (s1);
     }
     Login()
     {
     }
     void setComponents()
     {
       JLabel l1 = new JLabel ("welcome");
       JTextField t1 = new JTextField ();
       setLayout (null);
       l1.setBounds (200, 100, 100, 30);
       t1.setBounds (200, 200, 100, 30);
       add (t1);
       add (l1);
     }
     Public static void main (String[] args[])
     {
       Login s1 = new Login ("welcome to bcit");
       s1.setVisible (true);
       s1.setSize (700, 700);
       s1.setComponents ();
       s1.setDefaultCloseOperation (JFrame.EXIT_on_close
     }
   }.
```

Login Page:

```
import java.awt.event.*;
import javax.swing.*;
class Login extends JFrame
{
    JLabel l1, l2, l3 l4;
    JTextField t1, t2;
    JPasswordField b;
    JButton b1, b2;
    Login (String s1)
    {
        super(s1);
    }
    Login()
    {
    }
void setcomponents()
{
    l1 = new JLabel ("Welcome to coding");
    l2 = new JLabel ("Username:");
    l3 = new JLabd ("password:");
    l4 = new JLabd ();
    t1 = new TextField();
    t2 = new Textfield();
    b1 = new JButton ("Login");
    b2 = new JButton ("clear");
    Set Layout (null);
    add (l1);
    add (l2);
    add (l3);
    add (l4);
    add (t1);
    add (t2);
    add (b1);
    add (b2);
```

Date : / /
Page No.

```
l1. SetBounds (100, 50, 300, 25);
l2. —  (100, 200, 100, 30);
l3. —  (100, 350, 100, 30))
l4. —  (100, 350, 100, 30)
t1. —  (350, 200, 100, 30)
t2. —  (350, 350, 100, 30)
b1. —  (200, 460, 100, 30)
b2 —  (400, 450, 100, 30)
b1. addActionListener (new log());
b2. addActionListener (new.dem());
Public static void main (String [] args)
{
    Login s1 = new Login ("welcome to bcil");
    s1. setvisible (true);
    s1. setSize (700, 700);
    s1. setcomponents ();
    s1. SetDefaultcloseOperation (JFrame. EXIT_ON_
                                  CLOSE);
}
class log implements ActionListener
{
    Public void actionPerformed (ActionEvent e1)
    {
        String s1 = . t1. getText ();
        String s2 = t2. getText ();
        if (s1. equals ("") && s2. equals (""))
        {
            l4. SetText ("Login Successful");
        }
        else
        { l4. SetText ("Invalid");
        }
    }
}
```

```
class clear implement ActionListener
{
Public void actionPerformed (ActionEvent e1)
{
    t1. setText (" ");
    t2. setText (" ");
}
}

JButton b3;
Object
b3 = new JButton (" Add ");

add (b3);
b3. setBounds ( )

class Add implements ActionListener
{
Public void actionPerformed (ActionEvent e1)
{
try{
int a = Integer. parseInt (t1. getText());
int b = Integer. ParseInt (t2. getText());
}
catch (Exception e2)
{
}
int c = a+b;
l4. setText (" Addition is " +c);
}
```