o History of Java.

Java is OOPL (Object Oriented Programming language)

It is open source language (free to installation and free to updation)

Java is case sensitive language.

Before Java name "OAK" in 1991 from James goasling.

After "Java" from "OAK" in 1995 from James Goasling .

Java language is console app, desktop app etc…

o Features of Java

Java Features :

- **Simple**: Java is easy to learn and use.
- **OOP**: Java uses **objects** to represent real-world things.
- **Interpreter**: Java code is translated into machine code at runtime.
- **Secure**: Java has features to prevent harmful actions and protect data.
- **Robust**: Java is strong and handles errors well.
- **Dynamic**: Java can adapt and change during runtime.
- **High-Performance**: Java runs fast with features like **JIT** (Just-In-Time compilation).
- **Multi-threading**: Java can run multiple tasks at the same time.
- **Platform Independent**: Java code runs on any system with a Java Virtual Machine (JVM).
- **Portable**: Java programs can be easily moved and run on any device.

o Understanding JVM, JRE, and JDK

JVM : java virtual machine. Java virtual machine byte code convert machine code. JVM piece of software that computer run java program.

JRE : java runtime environment. JVM within the JRE work and JRE multiple class libraries include.

JDK : java development kit. JDK collation of tools and libraries that help developer create java application.

o Java Program Structure (Packages, Classes, Methods)

java structure :

```
Class ABC
{
        Public void setData()
        {

        }
        //body part
}
Class Program
{
Public ststic void main(String[] args)
{
        //body part
```

```
}

}
```

o Primitive Data Types in Java (int, float, char, etc.)

primitive datatype : primitive data type provide by language and primitive data type is fix size.

Int : integer size of 4 byte .

Float : float size of 4 byte.

Long : long size of 8 byte.

Double : double size of 8 byte.

Char : character size of 2 byte.

Bool : boolen size of 1 byte.

o Variable Declaration and Initialization.

Variable : variable is container to store the particular value.

Declaration :Int (datatype) age(variable name)=20(value);

Identifier :

1) does not start with digit

2) does not allow reserved keyword as a variable name

3) does not allow space between variable name

4) followed with digit after any letter or "_" or "$"

o Operators: Arithmetic, Relational, Logical, Assignment, Unary, and Bitwise.

Operators : operators is symbol to parform the operation from two value.

1.arithmatic operator : perform the mathematical operation use arithmetic operator.

Ex : +(addition) , -(substarction) , *(multiplication), /(division) , %(parectege).

2. relational operator: perform the two value comparison use relational operator.

Ex : ==(double equal) , !=(not equal) , >(greter then) , <(less then) , <=(lessthen equal) , =>(greterthen equal)

3.logical operator : perform the two or more condition check use logical operator.

Ex : &&(and) , ||(or) , !(not)

4.assignment operator : perform the assign value use assignment operator.

Ex: += , -+, *= , /=

a+=b = a=a+b

5. unary operator : increment operator value+1,decrement operator value-1

Prefix : ++a , --a postfix : a++ , a--

6.bitwise operator : ex : & , | , << , >>

o Type Conversion and Type Casting

Type Conversion : convert from one data type to another data type that type conversion.

1) implicit :automatically type casting convert from smaller data type in size convert into Bigger Data type

2) explicit : user from type casting convert from Bigger data type in size convert into Smaller Data type

o If-Else Statements

if-else statement : if-else statement cheak true or false condition.

Example :

```
public static void main(String[] args) {
        int num=30;
        if(num%2==0){
                System.out.println("number is even:"+ num);}
        else {
                System.out.println("number is odd:"+num);}
        }
```

o Switch Case Statements

switch case : multiple choice but user only one choice use switch case.

Syntex :

```
Switch(expression)
Case 1:
```

```
//body part
Break;
Case 2:
//body part
Break;
Default:
//body part
```

o Loops (For, While, Do-While)

loops : loop is repeatedly execute given condition to be true.

while loop : while loop condition is true loop is execute and condition is false loop is not execute. while loop is entry control loop, entry control loop is check condition before execute code.

Syntex : while(condition) { execute the code }

For loop : for loop condition is true loop is execute and condition is false loop is not execute. for loop is entry control loop, entry control loop is check condition before execute code.

Syntex : for(initialization , condition , increment/decrement)

Do while loop : do while loop condition first time execute and after condition is true loop is execute and condition is false loop is not execute. do while loop is exit control loop, exit control loop is execute code before check condition.

Syntex : do { execute the code } while(condition)

o Break and Continue Keywords

Break : break is keyword and break keyword use the break the code , rest of the code will not executed.

```
//break
public static void main(String[] args) {
    for(int i=0;i<10;i++)
    {
        if(i==6)
        break;
        System.out.println (i);
    }}
```

continue : continue is keyword and continue keyword use the skip the code , rest of the code will be executed.

```
//continue
public static void main(String[] args) {
for(int i=0;i<10;i++)
{
    if(i==6)
    continue;
    System.out.println (i);
}
}
```

o Defining a Class and Object in Java

class : class is the blueprint or template that collection of data member and member function

syntex:

class classname{

data member

member function}

object : object is instant of the class. A basic unit of oop. Object is user assigned memory data member and member function.

syntex:

classname objectname = new classname();

o Constructors and Overloading

constructor : constructor is special member function a same name class name.

constructor is not return type and void .

A create object automatically call constructor.

Overloading : a one class and one to many constructor that constructor overloading.

A constructor overloading constructor name same but parameter are different.

o Object Creation, Accessing Members of the Class

object creating : object create class name object name new keyword and class name.

classname objectname = new clasname();

accessing mebers of class : access member of class object dot method name.

objectname . methodname();

example : class Emp{

```
    int id;
    String name;
    public void setAssignValue() {
        id=45;
        name="shyam";
    }
    public void printValue(){
        System.out.println(id);
        System.out.println(name);
    }
}
public class ClassDemo {
    public static void main(String[] args)
    {
        Emp e=new Emp();
        e.setAssignValue();
        e.printValue();
    }
```

o this Keyword

this keyword is class variable name and argument variable names are same at that time to seprate your class variable with using this keyword .

example :

```
class Data{
        int age;
        String name;
        public void setData(int age,String name){
                this.age=age;
                this.name=name;
        }
        public void displayData(){
                System.out.println(age);
                System.out.println(name);
        }
}
public class ThisKeyword {
        public static void main(String[] args){
                Data ob=new Data();
                ob.setData(20, "ram");
                ob.displayData();
        }
}
```

o Defining Methods

methods : method is block of code to call and execute.

Method are default method , static method , abstract method.

Syntex :

Public void methodname()

{
//block of code}

o Method Parameters and Return Types

method parameters : method parameters are method name and one two many argument that method parameters.

Syntex :

Public void mathodname(int a,int b) //parameters method

{

//block of code

}

Return type : method are return the value from use return type method.

Syntex :

Public int methodname(int a , int b)

{

Return a+b; //return type

}

o Method Overloading

method overloading : method overloading is two or more method in one class and method name same but argument are different that method overloading.

Example :

```
class Overloding{
float pi=3.14f;
public void area(){
        Scanner sc=new Scanner(System.in);
        System.out.println("enter the value:");
        int r=sc.nextInt();
        System.out.println(pi*r*r);//circle formula
        }
public void area(float b,float h){
        System.out.println(b*h/2);//triangle formula
}

public void area(int l,int w){
        System.out.println(l*w);//rect formula
}
public void area(int a){
        System.out.println(a*a*a);//cube formula
}
public void area (double a,double b,double h)
```

```
        {
            double ans =0.5*(a+b)*h;//trapezoid formula
            System.out.println(ans);
        }
    }
    public class FunctionOverlodingPoly {
        public static void main(String[] args)
        {
            Overloding ob=new Overloding();
            ob.area();
            ob.area(9f, 5f);
            ob.area(5,
            ob.area(4);
            ob.area(4,5,6);
        }

    }
```

o Static Methods and Variables

static means constant refrens side.

static method : static method only static data member are execute.

Static variable : static variable is different variable but only one memory execute.

Example :

```
class Sdemo
{
        int no;
        static int  no1;
public static void show()
{
        //no=123;
        //System.out.println(no);
        no1=1234;
        System.out.println(no1);
}
}
public class StaticKeyword {
        public static void main(String[] args)
        {
                Sdemo ob1=new Sdemo();
                Sdemo ob2=new Sdemo();
                Sdemo ob3=new Sdemo();
                ob1.no=10;
                ob2.no=20;
                ob3.no=30;
                System.out.println(ob1.no);
                System.out.println(ob2.no);
```

```
System.out.println(ob3.no);

ob1.no1=10;

ob2.no1=20;

ob3.no1=30;

System.out.println(ob1.no1);

System.out.println(ob2.no1);

System.out.println(ob3.no1);

Sdemo.show();}}
```

o Basics of OOP: Encapsulation, Inheritance, Polymorphism, Abstraction

Encapsulation : wrapping up data into single unit . encapsulation is hide data

Parform the access modifier public , private , protected.

Inheritance : base class propartiy inherit derived class parform inheritance.

Java use mainly three type 1.simple inheritance , 2.multilevel inheritance , 3.herachycal inheritance.

Polymorphism : one form having many form or different form.

Polymorphism two type method overloading(compile time) method overriding(run time)

Abstraction : essential part should be display rest of part will be hide.

Abstraction main purpose hiding data.

o Inheritance: Single, Multilevel, Hierarchical

Inheritance : base class propartiy inherit derived class parform inheritance.

Java use mainly three type 1.simple inheritance , 2.multilevel inheritance , 3.herachycal inheritance.

Simple inheritance : A single base class inherit single derived class that single inheritance.

Syntex : class firstclass{

//body part

}

Class scondname extends firstclass{

//body part

}

Multilevel inheritance : a base class inherit derived class and derived class inherit another derived class that multilevel inheritance.

Syntex: class firstclass{

//body part

}

Class secondname extends firstclass{

//body part

}

Class therdname extends second{

//bodu part

}

Hierarchical : a one base class inherit one to more derived class that hierarchical inheritance.

Syntex : class firstclass{

//body part

}

Class secondclass extends firsclass{

//body part

}

Class thrdclass extends firstclass{

//bodu part

}

o Method Overriding and Dynamic Method Dispatch

Method overriding : method overriding is use inheritance base class method name and parameter and derived class method name and parameter are same and execute the derived class use method overriding.

Syntex : class A{

Public void run()

{
//body part

}

Class B extends A{

Public void run()

{

//body part

}

Dynamic Method Dispatch : Dynamic Method Dispatch is process by overridden method resolved runtime rather then compile time allow java to support.

o Constructor Types (Default, Parameterized)

constructor is 3 type : 1.default constructor, 2.parameterized constructor, 3.copy constructor

1.default constructor : default constructor is no any argument in constructor.

Syntex : class abc{

Public abc(){

//body part

}}

2.parameterized constructor : parametrized constructoris one to many argument in constructor.

Syntex : class abc{

Public abc(int a,int b){

//body part

}}

o Copy Constructor (Emulated in Java)

copy constructor : copy constructor is creating new object as a copy for exiting object.

o Constructor Overloading

constructor Overloading : a one class and one to many constructor that constructor overloading.

A constructor overloading constructor name same but parameter are different.

Syntex : class abc

{

Abc(){

//body part

}

Abc(int a){

//body part

}}

o Object Life Cycle and Garbage Collection

ANS :-

- Garbage collection refers to the automatic process of memory management.
- It automatically removes the unnecessary used memory spaces.

o One-Dimensional and Multidimensional Arrays.

ANS :- Array Is a Collection of Similar Data-Types.

Array is a group of elements which can store multiple value/object in a single variable with same data types.

Array have only two types :-

1) One-Dimessional Array :- it has only one dimensional. Array store single line multiple value. One dimensional array is execute series type . At a time only one loop use.

Example :-

```
public class OneDimensionalArray {
   public static void main(String[] args) {
      int[] arr = {10, 20, 30, 40, 50};
          System.out.println(arr[0]);  // Output: 10
      System.out.println(arr[2]);  // Output: 30
   }
}
```

2) Multi dimensional: it has three or multiple dimensional. Multi dimensional array store row and column multiple value. And multi dimensional array is execute table or matrix.

Example :-

```
public class TwoDimensionalArray {
   public static void main(String[] args){
      int[][] matrix = {
         {1, 2, 3},
         {4, 5, 6},
         {7, 8, 9}
      };
      System.out.println(matrix[0][1]);
      System.out.println(matrix[2][2]);
```

```
 }
}
```

o String Handling in Java: String Class, StringBuffer, StringBuilder

String:

ANS :- IN java, String is basically represents a sequence of character.

String is Immutable in java means After create String not change.

Immutable simply means Unmodifiable or unchangeable.

Stringbuffer:

Stringbuffer is same as the string.

StringBuffer is mutable.

In StringBuffer can modify length and sequence.

Insert , delete and append can be performed.

Stringbuilder:

Stringbuilder is same as the stringbuffer.

StringBuilderr is mutable.

In StringBuilder can modify length and sequence.

Insert , delete and append can be performed.

o Array of Objects

o String Methods (length, charAt, substring, etc.)

```
String s1=new String("My name is meru bharwad..");
System.out.println("Originl String :-"+s1);

1)length() :- returns the length of the String.
System.out.println("Length Of String :- "+s1.length());

2)concat() :- Combine Two Strings.
System.out.println("Concat Of Two String :-
"+s1.concat("i am 21"));

3)compare() :- Compare two strings.
System.out.println(" Compare Of Two String :-
"+s1.compareTo("kasotiya"));
```

4)substring() :- Returns a substring from the specified starting index to the end.

```
System.out.println("Sub-String Of String :-
"+s1.substring(3));
```

5) **substring(int startIndex, int endIndex)** : Returns a substring from the specified `startIndex` to `endIndex` (exclusive).

```
System.out.println("Sub-String Of String Index :-
"+s1.substring(1, 4));

6)repeat() :- Repeat a string in given time.
System.out.println("Repeat of String :- "+s1.repeat(5));

7)charAt() :-  covert string into character.
System.out.println("Character Of String :-
"+s1.charAt(5));
```

8) `toUpperCase()` : Converts all characters of the string to uppercase.

9) `toLowerCase()` : Converts all characters of the string to lowercase.

10) `trim()` : Removes leading and trailing whitespace.

11) `replace(char oldChar, char newChar)` : Replaces occurrences of a character in the string with a new character.

12) `split(String regex)` : Splits the string into an array of substrings based on the given regular expression.

13) `equals(Object obj)` : Compares the string with the specified object for equality.

o Inheritance Types and Benefits

ANS :-

1) Single Inheritance :- single inheritance is one super class inherit one sub class this is single inheritance.

2) Multi-level Inheritance :- In Multi-level Inheritance is one super class inherit sub class and sub class inherit another sub class this is multilevel inheritance.

3) Hierarchical Inheritance :- A Hierarchical Inheritance is super class inherit multiple sub class this is Hierarchical Inheritance.

Benefits :-

Code Reusability :- The main benefits is inheritance in java is code Reusability. Allows reuse of code from parent class in child classes, reducing duplication.

Extensibility: New classes can be easily added without modifying existing code.

Improved Code Organization: Helps organize related classes in a hierarchical structure.

Reduces Redundancy: Common functionality is written once in the parent class and inherited by child classes.

o Method Overriding

method overriding(run time) : method overriding is method should be same in super class as well as in subclass and parameter are same but its behaviors (body part of the method) are different.

```
class Method
{
	public void display()
	{
		System.out.println("This is Method Overriding.");
	}
}
class Method2 extends Method
{
	@Override
	public void display()
	{
		super.display();
		System.out.println("This is also Method Overriding.");
	}
}
public class MethodOverRiddingDemo
{

	public static void main(String[] args)
	{
		Method2 m=new Method2();
		m.display();
	}

}
```

o Dynamic Binding (Run-Time Polymorphism)

Dynamic binding, also known as **run-time polymorphism**, is a concept in Java where the method that is called is determined at **runtime** rather than at compile time.

This allows Java to support **method overriding** and makes the program more flexible and extensible.

Dynamic Method Dispatch is process by overridden method resolved runtime rather then compile time allow java to support.

o Super Keyword and Method Hiding

Super Keyword

when your super class variablename and subclass variable name are same then you used superclass variable used value that time used super keyword with variable

example :

```
class base{
        String name="Rohan";
}
class derived extends base{
        String name="meru";
        public void getvalue(){
                System.out.println("derived class:"+super.name);
        }
}
public class Leb_24_SuperKeyword {
        public static void main(String[] args)
        {
```

```
            derived ob=new derived();

            ob.getvalue();

        }

}
```

Method Hiding

Method Hiding occurs when a static method in the child class hides a static method in the parent class.

method hiding is a feature of **compile-time polymorphism** that occurs when a **static** method in the subclass hides a static method in the superclass.

Abstract Classes and Methods

Abstract classes :- We can't Create object for the Abstract class.

To we an abstract class , you have to inherit it from sub classes.

Example :-

```
abstract class Animal
{
	public abstract void sound();
}
class Dog extends Animal
{
	public void sound()
	{
		System.out.println("Dog is barking..");
	}
}
class Lion extends Animal
{
	public void sound()
	{
		System.out.println("Lion is Roar...");
	}
}
public class Abstract_class_demo
```

```
{

	public static void main(String[] args)
	{
		Dog d=new Dog();
		Lion l=new Lion();
		d.sound();
		l.sound();
	}
}
```

o Abstract Classes and Methods

abstract : only essential part should be display rest of the part will be hide

1) using with class : we can not create object of that class

must inherit into your child class

2) using with method : do no specify body part of the method

your class must be also abstract

must override your abstract method into your child class

o Interfaces: Multiple Inheritance in Java

interface : means same as class

its collection of datamember and member function

data members are final or static by default

member functions are abstract by default

its self a keyword

when you used interface with your class so we used "implements"

interface to interface by using "extends"

to hide data :

main purpose is to resolved the problem of multiple inheritance and hybrid

Multiple Inheritance : multiple interface is two or more interface execute one class.

Using implement keyword.

o Implementing Multiple Interfaces

Multiple Inheritance : multiple interface is two or more interface execute one class.

Using implement keyword.

```
	interface Bycycle
{
	public  void set_Data();
	public void display();
}
interface Bike
{
	String b_name = "Splendor";
	int Price=89000;
	public void bikeInfo();
	public void displayInfo();
}
class Cycle implements Bycycle,Bike
{
	String c_name;
	int price;

	@Override
	public void set_Data()
	{
		c_name="Avon Cycle";
		price=5000;
	}
	@Override
	public void display()
	{
		System.out.println("Cycle name is "+c_name);
```

```
                System.out.println("Cycle price is "+price);
        }
        @Override
        public void bikeInfo()
        {
                System.out.println(b_name);
                System.out.println(Price);
        }
        @Override
        public void displayInfo()
        {
                System.out.println("Bike name is "+b_name);
                System.out.println("Bike Price is "+Price);
        }
}
public class InterfaceDemo3 {

        public static void main(String[] args)
        {
                Cycle c=new Cycle();
                c.set_Data();
                c.display();
                c.bikeInfo();
                c.displayInfo();
        }
}
```

o Java Packages: Built-in and User-Defined Packages

ANS :-

- In java Package is one type of Container , is collection of classes , interfaces and sub-packages.
- in the first line of the program write package keyword with your folder names like package names.
- to used same name class like student in a single package but i can't create that class again in same package so i create package another for that class

In java mainly two types of Package :-

1) built-in package :-

- Built-in package also known as a pre-defined which are already created by java developer people or system provided.
- Java.lang , java.applet , java.awt , java.io , java.util , java.net , java.sql;

2) User – Defined Package :-

- User – defined package created by user it self and not provide by system or develpoer.
- Syntax :- package package_name;
- User – defined package statement must be first line of the program.

o Access Modifiers: Private, Default, Protected, Public

1) Private :- The private access modifier is Accessible only within class. It is not accessible from any other class.

2) Defualt :-If you don't specify any access control specifier , it is default. Accessbile in same Package.

3) Protected :- The protected Access Specifier Only Accessible within Package and outside the package or class to inherit only.

4) Public :- The Public Access Specifier is accessible Everywhere.means All packages , classes and sub classes used to public Access Modifier.

o Importing Packages and Classpath

In java mainly two types of Package :-

1) built-in package :-

- Built-in package also known as a pre-defined which are already created by java developer people or system provided.
- Java.lang , java.applet , java.awt , java.io , java.util , java.net , java.sql;

2) User – Defined Package :-

- User – defined package created by user it self and not provide by system or develpoer.
- Syntax :- package package_name;
- User – defined package statement must be first line of the program.

Classpath :-

- The classpath is a path used by the Java runtime environment (JRE) and Java compiler (javac) to locate classes and resources.

Two ways to use classpath :-

1. java -cp .;myclasses.jar MyProgram
2. set CLASSPATH=C:\myclasses;C:\lib\myjar.jar

o Types of Exceptions: Checked and Unchecked

exception handling is help program to handle error runtime and compile time.

or

- Exception is Unexpected event or abnormal Condition that occurs during Program Executuion.

Two types of Exception :-

1) Checked Exception :-

- Checked Exception is a known as compile time Exception.
- It is an exception that occurs at the compile time.
- Checked Exception type like ClassNotFoundException , IOException , SQLException , FileNotFoundException .

2) UnChecked Exception :-

- Unchecked Exception is a known as Run time Exception.
- It is an exception that occurs at the Run time.
- Unchecked Exception type like ArithmaticException , NullPointerException , ArrayIndexOutOfBoundsException.

o try, catch, finally, throw, throws

try : to find the error from the block.

when you know in my block some line of code having error to use try key word.

whatever error found in try block that error throw to catch block.

Catch : error throw by try block that error handle catch block.

The Try – Catch block is used to handle the exception and prevents the abnormal termination of the program.

catch can be multiple time create in program

finally : finally block always performed that error is execute or not.

Throw : throw keyword create exception custom exception , user define exception,

Throw keyword is use the method in side.

Use new keyword.

At a time one exception called.

Throws : used user or system define exception.

Thows keyword is use method signature.

Multiple exception called.

o Custom Exception Classes

- **We can explicitly make our own exception class by extending the parent Exception Class.**

o Introduction to Threads

ANS:-

- A Threads is a Light Weight Process or Processor. It's totally depend on process.
- A thread is a thread of execution in a program.
- Thread is a pre-defined class which is available in java.lang.package.
- Thread is a basic unit of CPU and it is well known for independent Execution.
- every Java application has at least one thread—the **main thread** that starts when the program begins.

There are main two way to perform thread :-

1) by extending Thread class
2) by implementing runnable interface

o Thread Life Cycle

- As we all know thread is a independent Execution . that life cycle thread to work process.

1) New State (Born)
2) Runnable State (Ready)
3) Running State (Execution)
4) Waiting State (Blocked)
5) Dead state (Exit)

1) New State :-

- When you create Thead Object but not start yet.
- Syntax :- Thread t = new Thread();

2) Runnable State :-

- The Runnable State is Ready to run When you use to Start Method .

- Use start method.
- Syntax :- t.start();

3) Running State :-

- The thread is managed by the thread scheduler, which decides when it can run.

4) Blocked State :-

- Whenever A thread is temporary inactive , it could be blocked or waiting state.
- When you use t.join() , t.sleep() , t.wait() , t.suspend() the thread is switched to Blocked state.

5) Dead State :-

- the thread was terminated abnormally or the thread has either finished its execution.
- Use Stop method.
- Syntax :- t.stop();

o Introduction to File I/O in Java (java.io package)

File input/output permenent to store data into file.

File it self class

You can create any type of file like .jpg , .txt, .docs, .xlsx

its derived from java.io package

having stream

- **Byte Stream**
  - **OutputStream:** to write data into files.
  - **InputStream:** to read data from files.
- **Character Stream**

- **Writer: to write data into files.**
- **Reader: to read data from files.**

o FileReader and FileWriter Classes

ANS :-

1)FileReader :-

- The FileReader class is used to read the content of text files .
- It part of the java.io package and is specifically designed for reading files as streams of characters.
- Syntax :- FileReader fr = new FileReader("output.txt")

Example :-

```
package com.File;

import java.io.FileReader;

import java.io.IOException;

public class FileReaderDemo1 {

	public static void main(String[] args)   {

		try (FileReader fr = new FileReader("output.txt"))  {

			int i;

			while ((i = fr.read()) != -1)  {

				System.out.print((char) i);

			}

		}

		catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

2) File Writer :-

- The FileWriter class is used for writing characters to a text file.
- It is also part of the java.io package and is used when you need to write text-based data (strings or characters) to files.
- Syntax :- FileWriter fw = new FileWriter("output.txt")

Example :-

```
package com.File;

import java.io.FileWriter;

import java.io.IOException;

public class FileWriterDemo {

    public static void main(String[] args)  {

        try (FileWriter fw = new FileWriter("output.txt")){

            fw.write("Hello, World!");

            fw.write("\nThis is a test.");

            fw.flush();

            fw.close();

        }

        catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

o BufferedReader and BufferedWriter

ANS :-

- **BufferedReader** class wraps reader **FileReader** class.
- **BufferedWriter** class wraps around writer **FileWriter** class.

o Serialization and Deserialization

ANS :-

- Serialization is a process of converting an object stat into a byte stream, and save , transmission , reconstructor deserialization
- Deserialization is a reverse process of serialization, converting byte stream to object stat.

o Introduction to Collections Framework

Collection : is a group of object into single object

its derived from java.util package

Collection its self interface

o List, Set, Map, and Queue Interfaces

1) List interface :-

- The List Interface extends the collection interface and adds the method that are specific to lists , which are ordered collections that allow duplicate elements.

- Here are the some methods that are present in the list interface but not in the collection interface.

1) ArrayList
2) LinkedList
3) Stack
4) Vector

2) Set Interface :-

- represent a unordered Collection that are not allow duplicate elements.
- Set interface represents a unique elements. Set interface class is hashset class.

3) Map Interface :-

- Represents a collection of key-value pairs, where each key is unique and maps to a single value.
- It does not allow duplicate keys but allows duplicate values.
- Map interface class is HashMap class.

4) Queue :-

- Represents a collection designed for holding elements prior to processing. It follows the FIFO (First In, First Out) order.
- Queue follows FIFO order, used for holding elements before processing.

o ArrayList, LinkedList, HashSet, TreeSet, HashMap, TreeMap

1) ArrayList :-

- ArrayList Interface Auto – implemented List Interface.
- ArrayList its represent like dynamic array.
- ArrrayList automatically shrink and grow both.
- ArrayList Default size is 0.
- ArrayList duplicate values are allow in Classes.
- add() and remove() method use in ArrayList classes.

- ArrayList default symbol is "[]" (Sqaure bracket).

2) Linked List :-

- LinkedList auto-implements both **List** and **Deque Interfaces**.
- It is implemented as a dynamic array/**doubly linked list** where each element is linked to the previous and next elements.
- linkList Default size is 0.
- Linklist automatically shrink and grow both.
- LinkedList also allows **duplicate values**.
- **Add() , remove() ,** addFirst() and addLast() are used for adding elements to the beginning or end, while removeFirst() and removeLast() remove elements from both ends.
- Often represented by square brackets [] but can be thought of more as a chain of elements.

3)HashSet:

- **Auto-implemented** by Set interface.
- It represents a **dynamic array**.
- **Automatically grows and shrinks** as elements are added or removed.
- The **default size** is 0.
- **Duplicates are not allowed** in a HashSet.
- **All values have a hash key**, which is used to store them in the set.
- The **hash key** is converted into a **hash code** for efficient storage and retrieval.
- All values are displayed **based on their hash code**.
- **Common methods**: add() to add elements and remove() to remove elements.
- The **default symbol** used is [].

TreeSet:

- **Auto-implemented** SortedSet interface.
- It represents a **sorted collection**.
- **Automatically grows and shrinks** as elements are added or removed.
- The **default size** is 0.
- **Duplicates are not allowed** in a TreeSet.

- **All values are displayed** in **ascending order**
- **All values have a unique position** based on their sorted order.
- **Elements are stored in a red-black tree** for efficient searching, insertion, and deletion. add() , remove() , first() , last()
- The **default symbol** used is [] (square brackets), but it is typically displayed in curly braces {} in the actual implementation.

HashMap:

- **Auto-implemented** Map interface.
- It represents data as **key-value pairs** (like a dictionary).
- **Automatically grows and shrinks** as elements are added or removed.
- The **default size** is 0.
- It works by storing **pairs** in the form <key, value>.
- **Duplicate pairs are not allowed**. If the **key is the same**, the **value is overridden**.
- **All pairs have a hash key** associated with them.
- **Hash keys are converted into hash codes** for efficient storage and retrieval.
- **Pairs are displayed based on their hash codes**.
- **Common methods**:
  - put() to add key-value pairs.
  - get() to retrieve values based on the key.
  - remove() to remove key-value pairs.
- The **default symbol** used is {} (curly braces), representing key-value pairs as {key=value}.

TreeMap:

- **Auto-implemented** Navigable Map interface.
- It represents data as **key-value pairs**, stored in **sorted order** based on the keys.
- **Automatically grows and shrinks** as elements are added or removed.
- The **default size** is 0.
- It works by storing **pairs** in the form <key, value>.
- **Duplicate keys are not allowed**. If a key already exists, the **value is overridden** with the new one.

- **All pairs have a unique position** based on their sorted order (ascending or according to the comparator).
- **All keys have a hash key**, and they are converted into **hash codes** for efficient storage and retrieval.
- **Pairs are displayed in ascending order of keys**.
- **Elements are stored in a red-black tree** (a type of balanced binary search tree), which allows for **efficient searching, insertion, and deletion** operations.
- The **default symbol** used is {} (curly braces), representing key-value pairs as {key=value}.

o Iterators and ListIterators

iterators is object that use loop throws collaction arraylist , hashset etc..

Iterator is fetch the value from the collection object like a list , set and map.

Example :

```
Iterator<String> it=fruits.iterator();

while(it.hasNext()){

        System.out.println("Iterator "+it.next());

}
```

ListIterators :-

- ListIterator allows traversal in both forward and backward directions . ListIterators only use in arraylist and linked list.

o Streams in Java (InputStream, OutputStream)

1) Byte stream :-

- for handling input and output of byte.

1) **InputStream :** to read the data from the file.

1) FileInputStream
2) ObjectInputStream
3) Buffered Input stream

2) OutputStream : to write the data into the file

1) FileOutputStream
2) ObjectOutputStream
3) buffered output stream

1) file Input stream :- The input stream is read the data byte from a file.

Example :

```
public class FileInputStreamDemo {


public static void main(String[] args) {
        try {
                FileInputStream fs = new FileInputStream("fisrtFile.txt");
           int byteRead;
           while ((byteRead = fs.read()) != -1) {
              System.out.print((char) byteRead);
           }
             fs.close();
        }
        catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

FileOutputstream :- to write the data from byte stream in a file.

Example :-

```
public class FileOutputStreamDemo
{
    public static void main(String[] args) {
        File f1=new File("fisrtFile.txt");
        try {
            FileOutputStream fio=new FileOutputStream(f1);
            String s="Hello Devloper";
            byte[] b=s.getBytes();
            fio.write(b);
            fio.flush();
            fio.close();
            System.out.println("Data Is write into file...");
        }
        catch (Exception e) {
```

```
                    e.printStackTrace();
                }
            }
        }
```

o Reading and Writing Data Using Streams

ANS :-

To read the file data

- FileInputStream
- BufferedInputStream
- FileReader
- BufferedReader

To write the file data

- FileOutputStream
- BufferedOutputStream
- FileWriter
- BufferedWriter

o Handling File I/O Operations

```
system.out.println("return Boolean value of file dorectory:"+f.isdirectory());

system.out.println("return the file yes or no:"+f.isfile());

system.out.println("return the file name is:"+f.getname());

system.out.println("can file execute or not the file:"+f.canExecute())

system.out.println("return the location of file:"+f.getAbsolutepath());

system.out.println("file path is:"+f.getpath());
```

```
system.out.println("file is readble or not return:"+f.canRead());

system.out.println("file is writable or not return:"+f.canWrite());

system.out.println("return the file size is:"+f.length());
```