MODULE 6 :- CORE JAVA

1. Introduction to Java

Q1.History of Java

ANS :-

- In 1991 ,Java Create by Sun Microsystem and Founder Name Was James Gosling to Called by 'OAK'.
- The OAK Name was Already in trademarked so James Gosling Changed name was JAVA in 1995.
- Java Is OOP(Object Oriented Programming) Language and Java Is Very Popular Language Because it's Features.
- Java is Open-Source Language because Any system Easy to free Installation And free to Update in our windows ,linux etc.
- Java Used to console app , Android , mobile app etc.

Q2.Features of Java (Platform Independent, Object-Oriented, etc.)

Ans :-

- **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.

Q3.Understanding JVM, JRE, and JDK

ANS :-

JVM :-JVM Stands For Java Virtual Machine is compiled Byte code and then It is Interpreted into Machine Code .

JRE :- JRE stands for Java Run-Time Environment is Help in Executing Programs Developed in java and include many libraries in java.

JDK :- JDK stands For Java Development Kit is collection of Tools used for Developing and running Java Programs and Desktop Application.

Q4.Setting up the Java environment and IDE (e.g., Eclipse, IntelliJ)

ANS :-

First Install JDK (Java Development kit) install in your device.

Step 1:- After JDK Install Go to Download Page and Search Eclipse – IDE In your Device.

Step 2:- Download Eclipse- IDE In Your Device , windows , linux etc.

Step 3:- Choose Java IDE for java Developers And Run the Installer.

Step 4:- After Choose the WorkSpace (Which folder You store the project) and launch.

Step 5:- After Successfully Installed Eclipse IDE Open and Create new java project.

Q5.Java Program Structure (Packages, Classes, Methods).

ANS :-

```
packagecom.CoreJava;  // Packages

publicclass HelloWorld {        //Classname

        publicstaticvoid main(String[] args)            //methhod
        {
        // block of code
                System.out.println("Hello World!");
        }
}
```

2. Data Types, Variables, and Operators

Q1.Primitive Data Types in Java (int, float, char, etc.)

ANS :-

primitive datatype is provide by Language And Fixed size of Datatypes.

- **Integer :** this data type is used to store a integer value and size of bytes is 4 bytes.
- **Float :** this data type is used to store decimal value and size of bytes is 4 byte.
- **Char :**This data type is used to store a single character and size of bytes is 2 bytes.use to store the character in code and use to ‘’(single quotes).
- **Bytes :-** This Data Type store The Integer value in 1 bytes.

- **Long integer :** declare store the positive value high range.size of 8 bytes.
- **double :** declare store the decimal value high range.size of 8 bytes.

Q2.Variable Declaration and Initialization

ANS :-

variable : means is nothing but to store some value or variable is Container Store the Particular value.

Declaration :-Datatype_nameVariable_name = Value_name;

int x=10 , String name=”Zeel”,char ch=’z’;

Many Rules In java For Variable :-

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 "_"

Q3.Operators: Arithmetic, Relational, Logical, Assignment, Unary, and Bitwise

ANS :

In Java programming operator means to perform some operations on the data or values.
Many types of operator in Java language :

1. arithmetic
2. relational
3. logical
4. assignment
5. increment/decrement
6. bitwise

1. Arithmetic operator : The arithmetic operator are used perform the some operations on the value. Arithmetic operator mainly used to mathematical or logical operation.

Operators	meaning
+	Addition
-	substraction
*	multiplication

/	division
%	moduls

2.Relational operator : Relational operator is also known as a comparison operator. Relational operator used to comparison two values and make the relation between.

Operators	meaning
>	Greater than
<	Less than
>=	Greater than & Equal to
<=	Less than & Equal to
==	Equal to
!=	Not equal to

3.logical operator : Logical operators are used to test more than one condition and make decisions.

operator	meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

&& - And (All the expressions must be true)
|| - Or (One of the any condition must be true)
! - Not (true expression will turn into false)

4.Assignment operator : The assignement operator is used to assign the value to the variable.

+=	a=a+b	a+=b
-=	a=a-b	a-=b
*=	a=a*b	a+=b
/=	a=a/b	a+=b
%=	a=a%b	a+=b

5.Increment / Decrement operator :

operator	meaning
++a	Prefix increment
a++	Postfix increment
--a	Prefix increment

a--	Postfix increment

Prefix : ++i, --i
Postfix :i++, i—

unary op. - a++, b-- (1 operand, 1 operator)
binary op. - a+b (2 operands, 1 operator)

Q4.Type Conversion and Type Casting

ANS :-

Type conversion :

- convert from one data type to another data type known as a Type Conversion.
- There are mainly 2 types :-

1) implicit :automatically : convert from smaller data type in size convert into Bigger Data type is Called Implicit Type Casting.

2) explicit :type casting : convert from Bigger data type in size convert into Smaller Data type is called Explicit Type Casting.

3. Control Flow Statements

Q1.If-Else Statements

ANS :-

Conditional statement are allow developers to control the flow of program based on specific conditions.

1. If stetment : The condition is true code to be executed otherwise Nothing.

Syntax :

```
if(condition)
{
        //code to b execute
}
```

2.if_else stetment : IF statement condition is false so else statement condition is true.

Syntax :

```
if(condition)
        {
        //code to be execute
        }
        Else
        {
        // code to be execute
        }
```

Switch Case Statements

ANS :-

switchstetment : Switch Statement use to choose only one Choice by user from multiple choice.

Not use relation operator , switch use integer , character data type .

switch use keyword switch , break , case , default.

Use with switch statement to select one of many code blocks to be executed.

Syntax :

```
switch(choice)
{
        Case 1:  // stetment
        Break;
        Case 2 : //stetment
        Break;
        Default : //stetment
```

```
        Break;

}
```

Q2.Loops (For, While, Do-While)

ANS :-

in Java language loops is repeat the same code a number of times.

in c language two types of loops :
1)entry loop
2)exit loop

1)entry loop :1)while loop
2)for loop
2)exit loop :1)do..while loop

1)while loop : A while loop is the an entry controlled loop. in while loop given condition is true then the loop is executed.and given condition false the loop is not executed.
Syntax of while loop :

```
intialization
While(condition)
{
        //block of code
                Increment / decrement
}
```

Example :

```
Inti = 1;
While(i<=10)
{
        System.out.println(i+1);
        i++;}
```

2) for loop :for loop is easier to compare than while loop. A for loop is the an entry controlled loop. in for loop given condition is true then the loop is executed and given condition false the loop is not executed.

Syntax :

```
For(initialization ; condition ; increment/decrement)
{
        //block of code;
}
```

Example:

```
For(inti=1;i<=10;i++)
{
        System.out.println(i+1);
}
```

3)do-while loop : do-while loop is a exit control loop . in do – while loop given condition is true d0-while loop first time execute and after loop is repeat given condition is false loop is not execute.

Syntax :

```
initialization
do
{
        //block of code
        Increment/decrement

}while(condition)
```

Q3.Break and Continue Keywords

ANS :-

The break statement :

- The break statement is mainly used in the switch statements .it is also useful for immediately stopping a loop.
- IN simple words break statement is used to break the code and rest of the code will not be executed.
- **Example of Break statement :**

```
Class Break
{
Public Static Void main(String[] args)
{
        for(inti=5;i>0;i--)
{
        if(i==3)
        break;
        System.out.println(i);
}
}
}
```

2) The continue statement :

- When you skip the current iteration but remain in the loop you should use the continue statement .

- IN simple words you skip the current iteration and rest of the code will not be executed then.

Example of Continue statement :

```
Class Continue
{
Public Static Void main(String[] args)
{
        for(inti=5;i>0;i--)
{
        if(i==3)
        Continue;
        System.out.println(i);
}
}
}
```

4. Classes and Objects

Q1.Defining a Class and Object in Java

ANS :-

class : -

- A class is a template or blueprint that specifies data member and data function.
- A class is a prototype or blueprint from which objects are created.
- Data member know as a class member declared in variable inside the class.
- Data function known as a method declared in function inside the class.

object:

- object is a instance of the class.
- Object is a Basic unit of Object Oriented programming.
- It is basically used to assigned the memory to the class (data member, member function).
- Class and object are dependent on each other.
- its uses new keyword and class constructor to create object.
- We cannot access private properties of class.

Syntax :-

classnameobjectname = new constructor();

Q2.Constructors and Overloading

ANS :-

- Constructor is A special member function because it's same name as class name.
- Constructor does not return any value but only void function use to return value.
- Constructor have three types :-

1. Default Constructor
2. Parameter Constructor
3. copy Constructor

- Constructor Does not Support Copy Constructor.
- When you create class object Constructor automatically called.
- Constructor can be overloaded.

Constructor Overloading :-

- Constructor also supported Constructor Overloading in java.
- in java create one class constructor or more than same method name to different arguments known as constructor overloading.

Q3.Object Creation, Accessing Members of the Class

ANS :-

Object Creation :-

- In java Object is a instance of class .When you create object use to new keyword.

Syntax :-

Class_nameObjectname = new Class_name();

How to Access member of The class :-

- After create object use to (.) to access class data members and class methods.

Syntax :-

Object .classdata_member();
Object .classmember_function();

Q4.this Keyword

ANS :-

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

Example :-

```
class Car
{
        String C_name;
        intprice , year;
        public Car(String C_name ,intprice ,intyear)
        {
                this.C_name=C_name;
                this.price=price;
                this.year=year;
        }
        publicvoid display()
```

```
        {
                System.out.println("Car Name Is :- "+C_name);
                System.out.println("Car Price is :- "+price);
                System.out.println("Car Year Is :- "+year);
        }
}
publicclassThisDemo
{

        publicstaticvoid main(String[] args)
        {
                Car c=newCar("Scorpio" , 180000, 2023);
                c.display();
        }
```

5. Methods in Java

Q1.Defining Methods

ANS :-

Methods are defined inside classes, is a block of code that performs a specific task.
There are mainly two types method in java :-
1) Pre-defined method
2) user –defined method

Syntax :-
Return_typeMethod_name();

Q2.Method Parameters and Return Types

ANS :-

- **Parameters** are used to pass information into a method.
- **Return Type** indicates what type of data the method will return.

Q3.Method Overloading

ANS :-

- the two or more method name should be same in a single class but its behaviors(data types, arguments) are different .
- **Method overloading** is a concept in object-oriented programming where you can have multiple methods with the same name but different parameters (either in number or type).

Static Methods and Variables

ANS :-

- A **static variable** is common to all instances of a class.
- A **static method** belongs to the class rather than to any specific object, and it can be invoked directly using the class name.

6. Object-Oriented Programming (OOPs) Concepts

Q1.Basics of OOP: Encapsulation, Inheritance, Polymorphism, Abstraction

ANS :-

Encapsulation :-

- Wrapping up data into single unit is known as Encapsulation.
- In Java the data is not accessible to the outside world.
- Only those functions can access it which is wrapped together within single unit.

Inheritance :-

- Inheritance is extends or inherit properties and methods of one class to another class.
- The new class is called derived class and old class is called base class.
- The main purpose of Inheritance use to reusability ,extendsiblity.
- The 'extends' key word used to through create inheritance.
- The derived class may have all the features of the base class.

polymorphism :

- ability to take one name having many forms or different forms.
- Polymorphism allows a single name to be used for more than one related purpose.
- The concept of polymorphism is characterized by the idea of 'one interface, multiple methods'.

- **there are mainly 2 types :-**

1) method overloading(compile time) : the two or more method name should be same in a single class but its behaviors(data types, arguments) are different.

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

Abstraction :-

- only essential part should be display rest of the part will be hide known as Abstraction.
- It refers to the showing only relevant information to the outside world.
- In simple words, hiding any background information from the outside world.
- It is used to implement in class to provide data security.

Q2.Inheritance: Single, Multilevel, Hierarchical

1)Single Inheritance :- Single Inheritance is nothing but inherit only one single super class to sub class.

2)Multi-level Inheritance :- In Multi-level Inheritance we have only one super class and multiple sub class called multilevel inheritance.

3)Hierarchical Inheritance :- A Hierarchical Inheritance Which contain only one super class and multiple sub class and all sub class directly extends super class called Hierarchical Inheritance.

Q3.Method Overriding and Dynamic Method Dispatch

1) method Overriding :-

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

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

	publicstaticvoid main(String[] args)
	{
		Method2 m=newMethod2();
		m.display();
	}

}
```

Dynamic Method Dispatch :-

ANS :-Dynamic method dispatch is the mechanism by which a call to an overridden method is resolved at run time, rather than compile time.

```
class Phone
{
	publicvoidshowTime()
```

```
        {
                System.out.println("Time is 10 pm..");
        }
        publicvoid On()
        {
                System.out.println("Phone is On mode...");
        }
}
classSmartPhoneextends Phone
{
        publicvoid music()
        {
                System.out.println("Playing music");
        }
        publicvoid On()
        {
                System.out.println("Smart Phone is On mode...");
        }
}
publicclassDynamic_Dispatch_Method
{
        publicstaticvoid main(String[] args)
        {
                Phone obj=newSmartPhone();
                obj.showTime();
                obj.On();

                Phone obj1=newPhone();
                obj.On();
                obj1.On();
        }
}
```

7. Constructors and Destructors

Q1.Constructor Types (Default, Parameterized)

ANS :-

1) Default Constructor :-

- A Constructor that has no any arguments known as Default Constructor.

2)parameterized Constructor :-

- A Constructor has One or more Arguments or parameters Known as Parameterized Constructor.

Example :-

```
class Sum
{
        inta,b;
        public Sum()
        {
                inta=10,b=20;
                System.out.println(a+b);
        }
        public Sum(intx , inty)
        {
                a=x;
                b=y;
                System.out.println(a+b);
        }
}
publicclassConstructorDemo
{
        publicstaticvoid main(String[] args)
        {
                Sum s=newSum();
                new Sum();
                new Sum(11,12);
        }
}
```

Q2.Copy Constructor (Emulated in Java)

ANS :-Java does not have an automatic **copy constructor** like C++.

Q3.Constructor Overloading

ANS :-

- Constructor also supported Constructor Overloading in java.
- in java create one class constructor or more than same method name to different arguments known as constructor overloading.
- We can have multiple **constructors** with different parameters & its datatype.

Q4.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.

8. Arrays and Strings

Q1.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-DimessionalArray :-it has only one dimensional. Array store single line multiple element. One dimensional array is execute series type. At a time only one loop use.

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

Q2.String Handling in Java: String Class, StringBuffer, StringBuilder

ANS :-

- IN java, String is basically an object that represents a sequence of character values.
- An Array of characters works same as java string.
- String is Immutable in java means After create String not change.
- Immutable simply means Unmodifiable or unchangeable.
- Once a String Object is created its data or state can't be changed.

StringBuffer :-

- StringBuffer is used to create a mutable string object.
- In StringBuffer can modify length and sequence.
- Insert , delete and append can be performed.
- StringBuffer are synchronized , so Thread-safe.can use in multi-threaded application.

StringBuilder :-

- StringBuilderr is used to create a mutable string object.
- In StringBuilder can modify length and sequence.
- Insert , delete and append can be performed.
- StringBuilder are not synchronized , so not Thread-safe.cannot use in multi-threaded application.

Q3.Array of Objects

ANS :-

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

ANS :-

```
String s1=newString("My name is merubharwad..");
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(intstartIndex, intendIndex): 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.

9. Inheritance and Polymorphism

Q1.Inheritance Types and Benefits

ANS :-

1)Single Inheritance :- Single Inheritance is nothing but inherit only one single super class to sub class.

2)Multi-level Inheritance :- In Multi-level Inheritance we have only one super class and multiple sub class called multilevel inheritance.

3)Hierarchical Inheritance :- A Hierarchical Inheritance Which contain only one super class and multiple sub class and all sub class directly extends super class called 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.
- Method Overriding: Child classes can provide their own implementation of methods from the parent class.
- Extensibility: New classes can be easily added without modifying existing code.
- Improved Code Organization: Helps organize related classes in a hierarchical structure.

Q2.Method Overriding

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

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

```
publicclassMethodOverRiddingDemo
{

	publicstaticvoid main(String[] args)
	{
		Method2 m=newMethod2();
		m.display();
	}

}
```

Q3.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 :-

- Dynamic method dispatch is the mechanism by which a call to an overridden method is resolved at run time, rather than compile time.

Q4.Super Keyword and Method Hiding

ANS :-when your super class variable name and subclass variable name are same then you used superclass variable want to used value into subclass at that time used super keyword with variable.

Example :-

```
class Shape
{
	String s_name="Square";
	publicvoiddraw()
	{
		System.out.println("can't say shape type");
	}
}
class Square extends Shape
{
	@Override
	publicvoid draw()
	{
		super.draw();
		System.out.println(super.s_name+" Shape ");
	}
}
publicclassSuperDemo
```

```
{

        publicstaticvoid main(String[] args)
        {
                Shape s=newSquare();
                s.draw();
        }

}
```

- **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.

10. Interfaces and Abstract Classes

Abstract Classes and Methods

ANS :-

- **Abstract Class** can have both abstract methods and non-abstract methods.
- We can't Create object for the Abstract class.
- To we an abstract class , you have to inherit it from sub classes.

Example :-

```
abstractclass Animal
{
        publicabstractvoid sound();
}
class Dog extends Animal
{
        publicvoid sound()
        {
                System.out.println("Dog is barking..");
        }
}
class Lion extends Animal
{
        publicvoid sound()
        {
                System.out.println("Lion is Roar...");
        }
}
publicclassAbstract_class_demo
{
```

```
        publicstaticvoid main(String[] args)
        {
                Dog d=newDog();
                Lion l=newLion();
                d.sound();
                l.sound();
        }
}
```

Abstract Methods :-

- A method declared without implementation in an abstract class is **an abstract method**.
- It can only be used in Abstract class. It does not contain any body "{}" and always End.
- Abstract Method must be Ovverridden in sub-classes otherwise it will also become a abstract class.

Example :-

```
abstractclass Programming
{
        publicabstractvoiddevloper();
        publicabstractvoid rank();
}
abstractclass HTML extends Programming
{
        @Override
        publicvoiddevloper()
        {
                System.out.println("HTML Devlop by Tim Berners-Lee");
        }
}
class Java extends HTML
{
        @Override
        publicvoid rank()
        {
                System.out.println("Java rank is 2nd.");
        }
}
publicclassAbstact_method_demo {

        publicstaticvoid main(String[] args)
        {
                Programming obj=newJava();
                obj.devloper();
                obj.rank();
        }

}
```

Q2.Interfaces: Multiple Inheritance in Java

Interface :-

- Interface is just like a class , which collection of data members and member Functions.
- Interface contain only interface Methods.
- To achieve interface java provides a key Word called 'implements'.

Notes :-

1) Interface methods are by default public and abstract .

2) interface variable are by default public , static and final.

3) interface method must be overridden inside the implementing class.

4) interface nothing but deals between client and developer.

Multiple interface :-

In Java, a class can implement multiple interfaces to achieve multiple inheritance. This allows the class to inherit the abstract methods from all the interfaces and, if necessary, override them.

Q3.Implementing Multiple Interfaces

ANS :-

- An **interface** in Java is a reference type, like a class. By default, all methods in an interface are abstract.
- In Java, a **class can implement multiple interfaces** to achieve multiple inheritance. This allows the class to inherit the abstract methods from all the interfaces and, if necessary, override them.

Example :-

```
interfaceBycycle
{
        publicvoidset_Data();
        publicvoid display();
}
interface Bike
{
        String b_name = "Splendor";
        intPrice=89000;
        publicvoidbikeInfo();
        publicvoiddisplayInfo();
}
```

```
class Cycle implementsBycycle,Bike
{
	String c_name;
	intprice;

	@Override
	publicvoidset_Data()
	{
		c_name="Avon Cycle";
		price=5000;
	}
	@Override
	publicvoid display()
	{
		System.out.println("Cycle name is "+c_name);
		System.out.println("Cycle price is "+price);
	}
	@Override
	publicvoidbikeInfo()
	{
		System.out.println(b_name);
		System.out.println(Price);
	}
	@Override
	publicvoiddisplayInfo()
	{
		System.out.println("Bike name is "+b_name);
		System.out.println("Bike Price is "+Price);
	}
}
publicclass InterfaceDemo3 {

	publicstaticvoid main(String[] args)
	{
		Cycle c=newCycle();
		c.set_Data();
		c.display();
		c.bikeInfo();
		c.displayInfo();
	}}
```

11. Packages and Access Modifiers

Q1.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 icant 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.

Q2.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.

Q4.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

12. Exception Handling

Q1.Types of Exceptions: Checked and Unchecked

- 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 :-

- Checked 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.

Q2.try, catch, finally, throw, throws

1) try :-

- The try key word use to find the error from the block.
- when you know in my block some line of code having error to use try key word.
- in which line to find the error that line to reamining line in the block are skipped and code execute.
- whatever error found in try block that error throw to catch block.

2) **catch :-**
- whatever error thrown by try block that error will handled with appropriate class.
- 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.

3) **Finally :-**
- The finally Keyword used to if error will be come or not but my code is always run.
- It is always executed whether an exception is handled or not.

4) **throw :-**
- We use the throw key word to throw custom Exception.
- The throw keyword write inside the method. at a time only 1 exception to called.

5) throws :-
- it is used to declare an exception. It is required only for checked exception.
- Throws key word used to system defined or user defined Exception.

Q3.Custom Exception Classes

ANS :-

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

13. Multithreading

Q1.Introduction to Threads

ANS:-

- A Threads is a Light Weight Proess 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 :-

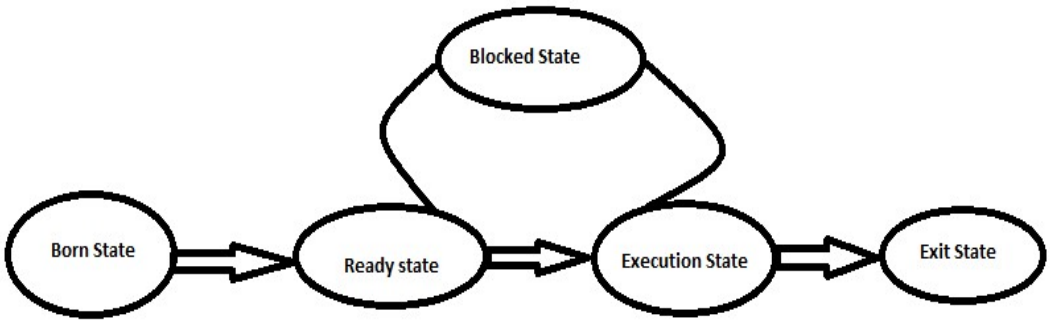
1) by extending Thread class
2) by implementing runnable interface

Q2.Thread Life Cycle

ANS :-

- As we all know thread is a independent Execution.During the life cycle thread can move from different states.

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 .
- A thread enters the Runnable state when the start() method is invoked the thread switch to the thread state.
- 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 to Stop() method.
- Syntax :- t.stop();

14. File Handling

Q1.Introduction to File I/O in Java (java.io package)

ANS :-

- File Input/Output (I/O) is an essential concept when dealing with files in Java.
- File Handling defines how we can read and write data on a file. File work in stream.
- Stream is a sequence or Continuous flow of data on the basic of 'java.io'.
- There are mainly two types stream 1) byte stream 2) character 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.

Q2.FileReader and FileWriter Classes

ANS :-

1)FileReader :-

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

Example :-

```
packagecom.File;
importjava.io.FileReader;
importjava.io.IOException;
publicclass FileReaderDemo1
{
        publicstaticvoid main(String[] args)
         {
        try (FileReaderfr = newFileReader("output.txt"))
              {
        inti;
        while ((i = fr.read()) != -1)
                 {
        System.out.print((char) i);
                  }
              } catch (IOExceptione)
              {
        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 :- FileWriterfw = **new**FileWriter("output.txt")

Example :-

```
packagecom.File;
importjava.io.FileWriter;
importjava.io.IOException;
publicclassFileWriterDemo
{
        publicstaticvoid main(String[] args)
          {
        try (FileWriterfw = newFileWriter("output.txt"))
            {
        fw.write("Hello, World!");
        fw.write("\nThis is a test.");
        fw.flush();
        fw.close();
            }
        catch (IOExceptione)
            {
        e.printStackTrace();
            }
         }
}
```

Q3.BufferedReader and BufferedWriter

ANS :-

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

Q4.Serialization and Deserialization

ANS :-

- **Serialization** is a process of converting an object into a byte stream. This allows us to save objects into files.
- **Deserialization** is a reverse process of **serialization**, converting byte stream to object.

15. Collections Framework

Q1.Introduction to Collections Framework

ANS :-

- Collection are group of Objects Into single unit.
- The java Collections Framework provides a set of interfaces and classes to implements various data structures and algorithms.
- Collection are group of objects that can be manipulated as a single unit.
- Collection are essential for sorting , storing , searching and manipulating data.
- ItsDerived from Java.util.Package. Collection it self Interface.

Q2.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.

Q3.ArrayList, LinkedList, HashSet, TreeSet, HashMap, TreeMap

1) **ArrayList :-**
   - ArrayList Interface Auto – implemented List Interface.
   - ArrayListits represent like dynamic array.

- ArrrayListautomatically shrink and grow both.
- ArrayListDefault size is 0.
- ArrayListduplicate values are allow in Classes.
- add() and remove() method use in ArrayList classes.
- ArrayListdefault symbol is "[]" (Sqaure bracket).

2) Linked List :-

- LinkedList auto-implements both **List** and **Deque Interfaces**.
- It is implemented as a **doubly linked list** where each element is linked to the previous and next elements.
- This allows fast insertion or removal from both ends of the list.
- Like ArrayList, the default size starts as 0 and grows dynamically.
- 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 [] (square brackets), but it is often displayed in curly braces {} in the actual implementation.

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 16, and it doubles when the threshold is crossed.
- 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}.

Q4.Iterators and ListIterators

ANS :-

- Iterator is fetch the value from the collection object like a list , set and map.
- It is part of the java.util package and provides methods to iterate over the elements of a collection.

```
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.

16. Java Input/Output (I/O)

Streams in Java (InputStream, OutputStream)

ANS :-

- Stream is a sequence or Continuous flow of data on the basic of 'java.io'.
- There are mainly two types stream 1) byte stream 2) character stream

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) 1)FileOutputStream
2) ObjectOutputStream
3) buffered output stream

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

```
publicclassFileInputStreamDemo
{

        publicstaticvoid main(String[] args)
        {
```

```
                try
                {
                        FileInputStreamfs = newFileInputStream("fisrtFile.txt");
                intbyteRead;
                while ((byteRead = fs.read()) != -1)
                  {
                System.out.print((char) byteRead);
                  }
                fs.close();
                }
                catch (IOExceptione)
                {
                e.printStackTrace();
                }
        }

}
```

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

Example :-

```
publicclassFileOutputStreamDemo
{
        publicstaticvoid main(String[] args)
        {
                File f1=newFile("fisrtFile.txt");
                try
                {
                        FileOutputStreamfio=newFileOutputStream(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();
                }
        }
}
```

Q2.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

Q3.Handling File I/O Operations

ANS :-

```
System.out.println("return Boolean value of File Directory  :- "+f.isDirectory());

System.out.println("return the  File yes  Or Not :- "+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 Locaton 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 writeble or not return :- "+f.canWrite());

System.out.println("return the File size is :- "+f.length());
```