Java Basic Interview Questions

1. What is Java?

Java is the **high-level**, **object-oriented**, **robust**, **secure** & **platform-independent**, **high performance, Multithreaded, and portable programming language**. It was developed by **James Gosling** in 1995.

2. What is the difference between C++ and Java?

C++	Java
C++ is platform-dependent.	Java is platform-independent.
C++ supports multiple inheritance.	Java doesn't support multiple inheritance To achieve use interface
C++ supports operator-overloading.	Java doesn't support operator overloading.
C++ supports pointer You can write pointer programs in C++.	Java supports pointers internally. However, you can't write the pointer program in java. It means java has restricted pointer support in Java.

3. What are the features of Java?

- **Simple:** Java is easy to learn. The syntax of Java is based on C++ which makes it easier to write the program in it.
- **Object-Oriented:** Java follows the object-oriented paradigm which allows us to maintain our code as the combination of different types of objects that incorporates both data and behavior.
- **Portable:** Java supports read-once-write-anywhere approach. We can execute the Java program on every machine. Java program (.java) is converted to bytecode (.class) which can be easily run on every machine.
- **Platform Independent:** Java is a platform independent programming language. It is different from other programming languages like C and C++ which need a

platform to be executed. Java comes with its platform on which its code is executed. Java doesn't depend upon the operating system to be executed.

- **Secured:** Java is secured because it doesn't use explicit pointers. Java also provides the concept of ByteCode and Exception handling which makes it more secure.
- **Robust:** Java is a strong programming language as it uses strong memory management. The concepts like Automatic garbage collection, Exception handling, etc. make it more robust.
- **Interpreted:** Java uses the Just-in-time (JIT) interpreter along with the compiler for the program execution.
- **High Performance:** Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++).
- **Multithreaded:** We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multimedia, Web applications, etc.
- **Dynamic:** Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

4. What are the differences between JDK, JRE and JVM?

JDK→ JDK stands for Java Development Kit. It is a software development environment which is used to develop Java applications.It physically exists. It contains JRE + development tools.

JRE→ JRE stands for Java Runtime Environment. It is the implementation of JVM. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

JVM→ JVM stands for Java Virtual Machine; it is an abstract machine which provides the runtime environment in which Java bytecode can be executed. It is a specification which specifies the working of Java Virtual Machine

5. How many types of memory are allocated by JVM?

- **Class(Method) Area:** Class Area stores per-class structures such as the runtime constant pool, field, method data, and the code for methods.
- **Heap:** It is the runtime data area in which the memory is allocated to the objects
- **Stack:** It holds local variables and partial results.
- **Program Counter Register:** PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.
- **Native Method Stack:** It contains all the native methods used in the application.

6. What is the JIT compiler?
The Just-In-Time (JIT) compiler is **a component of the Java Runtime Environment that improves the performance of Java applications at run time.**

7.What is Data Type?
Data types means which type of data you are assigning to the particular variable called data type.basically there are two types of data.

1. **Primitive type**

(Numeric data type)

- byte(1 byte)
- short(2 byte)
- int(4 byte)
- long(8 byte)
- float(4 byte)
- double(8 byte)
- char(2 byte)
- boolean(size depends on jvm)

2. **Non-Primitive type**

- String
- Array
- List
- Set
- Stack
- Vector
- Dictionary

8. What is an Identifier?
The name in the java program is called identifier. identifiers can be class name, method name and variable name.

9 .What is variable?
A variable is the container which is used to hold/store the value called variable and it is three types.

1. **Local variable**→ local variable which is created inside the body of the method called local variable and we can access local variables directly.all the local variable will be stored inside the stack section.
   Ex:-
```
class Main{

    public static void main (String[] args) {
    int x=10;//local variable
    System.out.println(x);
      }
    }
```

2. **Static variable**→ static variable is a variable which is created **inside the class with the help of static keyword and outside the main method** is called static variable and we can access static variable using class name,by making object or directly but alway try to access with class name because static belongs to class not objects.**all static variable will be stored inside heap area.**
   Ex:-
```
class Main{
    static int y=12;// static variable
    public static void main (String[] args) {
    Main m=new Main();//creating object
   System.out.println(m.y);//accessing through object
   System.out.println(y);//accessing directly
   System.out.println(Main.y);  //accessing through class name }}
```

3. **Instance variable**→ Instance variable is which is **created inside the body of the class and outside the main method called instance variable**.if we want to access the instance variable then we need to **create the object** for that.we can not access directly or with the help of class.

   EX:-class Main

```
     {
      int z=30;//instance variable
      public static void main (String[] args) {
      System.out.println(z);//we will get error
      System.out.println(Main.z);// we will get error

      Main m=new Main();
      System.out.println(m.z);//this is valid to access instance variable
       }
      }
```

When to use static and when to use an instance variable out program.
→ if our data changes frequently then we need to use an instance variable.
Ex-student name sometime ramesh,hira and sanjeet
→ if our data is common for all the objects then use a static variable.
Ex-college name,company name

10. How many keywords in java?
Total 53 keywords in java in which 3 is literal(true, false, null) so rest 50 keywords in which 48 are used keywords and 2 (goto and const) are unused keywords.

11. What is an array?
- Array is a collection of similar data items
- Array stores only homogeneous data items
- Array is fixed in size
- We can access array through index

Advantage→ **code optimization and random access .**
Disadvantage→ **Fixed in size**

Type of array→ **1 Dimensional array → int[] arr=new int[size];**
Multi-dimensional array → int[][] arr=new int[size][size];

12.What is a string?
- String is an object in java which is immutable

- String is also a group of characters called string.

13.How many ways to create string objects in java?
There are two ways in java to create string object

1. **Using string literals**→ wherever we will create a string object in java by using literals then it will create only one object **inside string constant pool** string constant poll is a place where all string literal objects will be stored.
   Ex:-
   ```
   class Main
      {
      public static void main (String[] args) {
      String s="amarjeet";// using string literal
      System.out.println(s);
       }
      }
   ```

2. **Using new keyword**→ wherever we create a string object in java by using new keyword then it will create two objects one object for literal and another one for new keyword it will be stored inside **the heap area**.
   Ex:-
   ```
   class Main
      {
      public static void main (String[] args) {
      String s=new String("amarjeet");// using new keyword
      System.out.println(s);
      }
     }
   ```

14.Why string object is immutable?
In Java string every object is immutable which means that we can not change the value once an object is created.
Ex:-
```
class Main{
   public static void main (String[] args) {
     String s="amarjeet";
     s.concat("singh");//value cannot be contacted
     System.out.println(s);//amarjeet
    }
   }
```

15. What is the difference between == and .equals() ?

- **== is an operator** used for reference comparison or address comparison whether two objects pointing to the same reference/address or not.
  Ex:-class Main{
```
      public static void main (String[] args) {
       String s1="amarjeet";
     String s2="amarjeet";
     System.out.println(s1==s2);//true because both pointing same address
       }
     }
```

- **.equals() is a method** used for content comparison whether two objects' content is the same or not.

  Ex:-class Main{
```
      public static void main (String[] args) {
     String s1="amarjeet";
     String s2="amarjeet";
     System.out.println(s1.equals(s2));//true because content is same
      }
     }
```

16.What is the difference between String, StringBuffer and StringBuilder?

String	StringBuffer	StringBuilder
The String objects are immutable.	The StringBuffer objects are mutable.	The StringBuilder object is mutable.
String objects stored in Heap area as well inside string constant pool.	StringBuffer objects are only stored inside the Heap area.	StringBuilder objects are only stored inside the Heap area.
String objects are synchronized, means thread safe.	StringBuffer objects are synchronized, means thread safe.	StringBuilder objects are non-synchronized, which means they are not thread safe.
Performance is slow.	Performance is slow.	Performance is fast.

17. What is type casting?
Type casting means whenever you are assigning/converting one type of data to another type called type casting, There are two types of casting.

1. Widening or automatic casting→ **smaller into larger.**

2. Narrowing or manual casting→ **larger into smaller.**

Ex-:
```
public class TypeCasting {
   public static void main(String[] args) {
         double d=12.20;
         int i=(int)d;//double into int

         int i1=20;
         byte b=(byte)i1;//int into byte

         System.out.println(i);
         System.out.println(b);
         }
         }

         Or
```

Ex:-
```
public class TypeCasting {
      public static void main(String[] args) {


//byte -> short -> char -> int -> long -> float -> double
/*boolean bool = true;
byte b1 = (byte) (bool ? 1 : 0);
System.out.println("boolean to byte: " + b1);

byte b=65;
short s=b;
System.out.println("byte to short: "+s);

char c=(char)s;
System.out.println("short to char: "+c);

int i=(int)c;
System.out.println("char to int: "+i);

long l=i;
```

```
		System.out.println("int to long: "+l);

		float f=l;
		System.out.println("long to float: "+f);

		double d=f;
		System.out.println("float to double: "+d);
		*/
		//double->float->long->int->char->short->byte->boolean

		double d=65;
		float f=(float)d;
		System.out.println("double to float: "+f);

		long l=(long)f;
		System.out.println("float to long: "+l);

		int i=(int)l;
		System.out.println("Long to int: "+i);

		char c=(char)i;
		System.out.println("int to char: "+c);

		short s=(short)c;
		System.out.println("char to short: "+s);

		byte b=(byte)s;
		System.out.println("short to byteL: "+b);

		byte b1 = 1;
		boolean bool = (b1 != 0);
		System.out.println("byte to boolean: " + bool);

	}
}
```

18.What is the wrapper class?

Wrapper class provides support to convert primitive into object and object into primitive called wrapper class called.There are two ways to wrap the data.

1. Autoboxing→ **wrap primitive into object.**

```
Ex:-class Main
    {
  public static void main (String[] args) {
  int x=10;
  Integer i=Integer.valueOf(x);//doing manually
  Integer i1=x; //compiler will do automatically

  float f=12.90f;
  Float f1=f;
  System.out.println(f1);
  System.out.println(i);//manually converting
  System.out.println(i1);//Compiler will automatically
    }
      }
```

2. Unboxing→ **object into primitive.**

```
Ex:-class Main
  {
  public static void main (String[] args) {
  Integer i=12;
  int i1=i;
  System.out.println(i1);// object into int
  }
  }
```

19.What is object class and its methods?
Object class is the parent class of all the classes in java by default and object class is present in **java.lang package** and it is also the top most class in java and all the **classes are derived from this object class**.

Methods of object class:-

- **hashcode()**
- **toString()**
- **equals()**
- **getClass()**
- **notify()**
- **notifyAll()**
- **finalize()**

- **clone()**
- **wait()**

20.Explain public static void main(String[] args)?
public→ It is an **Access modifier**, which specifies where and who can access the method. Making the *main()* method public makes it globally available. It is made public so that JVM can invoke it from anywhere.

static→ It is a keyword and the main() method is static so that JVM can invoke it without creating an object of the class.This also saves the unnecessary wastage of memory.

void→ It is a keyword and is used to specify that a method doesn't return anything.

main→ It is the name of the Java main method. It is the identifier that the JVM looks for as the starting point of the java program. It's not a keyword.

String[] args→ It stores Java command-line arguments and is an array of type **java.lang.String** class. Here, the name of the String array is *args* but it is not fixed and the user can use any name in place of it.
21.Explain System.out.println();
System→ It is a final class present in the **java.lang package.**
out→ it is a static variable present in the **PrintStream** class of type **PrintStream.**
println() → it is a method present in the **PrintStream** class.

Java oops Interview Questions:

22.What are the main features of oops and explain it?
There are six main features of oops in java
1. Class
2. Object
3. Inheritance
4. Polymorphism
5. Encapsulation
6. Abstraction

Class→ class in a **group of objects** and class is **not a real world entity** it is just **blueprint or template** and **class does not occupy memory.**
Ex:-Animal,vehicle

Object→ object is an **instance of a class** and object is a **real world entity** and **object occupies memory** and every object consists of **identity**(name of object), **attribute**(color, age, breed) and **behavior**(run, eat, walk, sleep)**.**
Ex:-**inside the animal class dog,cat,rat is an object name which identity,attribute and behavior.**

Inheritance→ inheritance is a process in which a **child class acquires the properties of the parent class** called inheritance and it has three types.
Ex:-son acquired the properties from father.
1. Single level
2. Multi level
3. Hierarchical
4. Hybride(not supported by java)
5. Multiple(not supported by java)

Polymorphism→ The word **polymorphism means having many forms** called polymorphism in another way polymorphism in which one object has many forms called polymorphism and there are two types of polymorphism.
1. Compile time/static polymorphism**(achieved by method overloading)**
2. Runtime/Dynamic polymorphism**(achieved by method overriding)**

Encapsulation→ wrapping a **data member and member function** together in a **single unit** called encapsulation.We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use **setter and getter methods to set and get the data** in it. **java bean class is an example of a fully encapsulated class**.

Abstraction→ **is** a process of **hiding the implementation details** and showing only **functionally** to the user called abstraction and there are two ways to achieve abstraction in java.
1. **Abstract class**(0-100 percent abstraction we can achieve).
2. **Interface**(100 percent abstraction we can achieve).

Ex:-atm machine, whatsapp how they are working we don't know we are just using it like several functions and services are available.

23.What is constructor and its types?

- constructor is a block (similar to method) having the same name as that of class name called constructor.
- constructor does not have any return type even void.
- constructor execute automatically when we create an object.
- Only four modifiers are applicable for constructors: public,private,protected and default.

There are 3 type in java

1. Default constructor
2. User define constructor
3. Parameterized constructor

24.explain about single,multilevel and hierarchical inheritance with example?

Single level→ child class inherits the properties of parents called single level inheritance.

```
Ex:-public class A{
      public void m()
     {
   System.out.println("this is inside m()...");
    }
   }
  class B extends A
 {
    public void m1()
    {
        System.out.println("this is inside m1()..");
    }
       public static void main (String[] args) {
        B b=new B();
        b.m();
        b.m1();
    }
  }
```

Multilevel inheritance→ when we inherit the properties of multiple classes one by one in child class, this process is called multilevel inheritance.

Ex:-
```
public class A{
      public void m()
      {
        System.out.println("this is inside m()...");
      }
      }
      class B extends A
     {
     public void m1()
     {
        System.out.println("this is inside m1()..");
     }
     }
     class C extends B
     {
         public void m3()
         {
         System.out.println("this is inside m3()..");
         }
         public static void main (String[] args) {
        C c=new C();
        c.m();
        c.m1();
         c.m3();
      }}
```

Hierarchical inheritance→ when two or more child classes inherit the properties of a single parent class called hierarchical inheritance.

Ex:-
```
public class A{
      public void m()
     {
        System.out.println("this is inside m()...");
     }
    }
    class B extends A
```

```
{
public void m1()
{
   System.out.println("this is inside m1()..");
}
}
class C extends A
{
    public void m3()
    {

            System.out.println("this is inside m3()..");
    }
    public static void main (String[] args) {
   C c=new C();
   c.m();
            c.m3();

            B b=new B();
            b.m1();
            b.m();
}
}
```

25.why multiple inheritance is not supported by java?

Whenever **one single class is inheriting more than two class properties** so ambiguity problems come this is the reason java does not support multiple inheritance.**we can achieve multiple inheritance in java** using **interface**.

Ex:-class Parent1

```
        {
     void fun()
        {
        System.out.println("Parent1");
        }
        }
    class Parent2
        {
    void fun()
```

```
{
System.out.println("Parent2");
}
}
class Test extends Parent1, Parent2
{
public static void main(String args[])
{
Test t = new Test();
t.fun();
}
}
```

25.Explain methods overloading and method overriding with examples?

Method overloading	Method overriding
Method with same name with different parameters called method overloading.	Method with same name and same parameters called method overriding.
We can use method overloading in the same class.	We can use method overriding in different classes or can extend.

Method overloading program:-

```
class Main{
   public void m1()
   {
      System.out.println("i am inside m1() with empty parameter..");
   }

   public void m1(int a,int b)
   {
      System.out.println("i am inside m1() with integer
parameter..");
   }

   public void m1(String name,String city)
   {
      System.out.println("i am inside m1() with String parameter..");
```

```
    }
    public static void main (String[] args) {
        Main m=new Main();
        m.m1();
        m.m1(1,2);
        m.m1("amarjeet","rajnish");

     }
    }
```

Method overriding program:-

```
class Main
{
    public void m1(int a,int b)
    {
        System.out.println("i am inside m1()rajnish...");
    }
}
class B extends Main
{
    public void m1(int a,int b)
    {
        System.out.println("i am inside m1()tuntun...");
    }
}
class C extends B{
    public void m1(int a,int b)
    {
        System.out.println("i am inside m1() hira...");
    }
}
class Test{
    public static void main (String[] args)
    {

         Main m=new Main();
         m.m1(1,2);
        B b=new B();
        b.m1(1,2);

        C c =new C();
        c.m1(1,2);
         }

    }
```

OR

```
interface inter {
        static int x = 10;
        final int y = 20;

        abstract void eat();
        public void run();
        default void default_bark()
        {
                System.out.println("I am default method...");
        }

        static void static_bark()
        {
                System.out.println("I am static method...");
        }

}
class Amarjeet implements inter{

        @Override
        public void eat() {
                System.out.println("I am eating...");

        }

        @Override
        public void run() {
                System.out.println("I am running...");
        }
}
class Test{
public static void main(String[] args) {
                Amarjeet am=new Amarjeet();
                am.eat();
                am.run();
                am.default_bark();
                inter.static_bark();
                System.out.println(am.y);
                System.out.println(inter.x);

        }
}
```

26.Explain abstract class and interface with example?

Abstract class	Interface
abstract class must be declare with abstract keyword	interface must be declared with interface keyword.
abstract classes can have an abstract and non-abstract method.	interface can have only abstract methods.
we can not create objects of abstract class.	we can not create objects of interface.
abstract class does not support multiple inheritance.	interface supports multiple inheritance.
abstract class used to achieve (0-100) percent abstraction in java.	interface used to achieve (100) percent abstraction in java.
abstract class can have (1)abstract and (2)non-abstract method as well as (3)simple block, (4)static block, (5)constructor, (6)static method,	interface can have **static method,Default method since java 8**,by **default every variable in interface is public,static and fina**l and **every method**

(7)final method and (8)static and (9)final variable.	**is public and abstract** we can not use block and constructor inside the interface.

Note:-we can not use constructor and block inside interface.

Abstract class program:-

```
abstract class Bike {
        {
                System.out.println("i am block...");//simple block
        }
        Bike()  //constructor
        {
                System.out.println("this is constructor...");
        }

        abstract void run(); //abstract  method


        void changeGear() //non-abstract method
        {
                System.out.println("this is non-abstract method...");
        }

        static int x=2000;//static variable

        static void static_method()//static method
        {
                System.out.println("this is static method method...");
        }

        static{
                System.out.println("this is static block...");//static block
        }
        final void final_method()//final method
        {
                System.out.println("this is final_method method...");
        }
    }


   class Honda extends Bike
   {
        void run()
        {
                System.out.println("calling abstract method...");
        }
  }

  class TestAbstraction2
  {
        public static void main(String args[])
        {
                System.out.println(Bike.x);


                Honda obj = new Honda();
                obj.run();
                obj.changeGear();
                obj.static_method();
                obj.final_method();
                }
                }

                OR


public abstract class AbastractClassDemo {
        static int x = 10; //static variable
```

```
	final int y = 20;  //final variable

	abstract void run1();  //abstract method

	void run2() {   //Non-abstract method
		System.out.println("non abstract method..");
	}

	AbastractClassDemo() {  //constructor
		System.out.println("this is constructor...");
	}

	{   //block
		System.out.println("this is block");
	}
	static {  //static block
		System.out.println("this is static block");
	}

	static void static_method() {  //static method
		System.out.println("this is static method");
	}

	final void final_method() {  //final method
		System.out.println("this is final method");
	}

}
class Amarjeet1 extends AbastractClassDemo{

	@Override
	void run1() {
		System.out.println("this is abstract method");
	}
	public static void main(String[] args) {
		Amarjeet1 am=new Amarjeet1();
		System.out.println(am.y);
		System.out.println(AbastractClassDemo.x);
		am.final_method();
		am.run2();
		am.run1();
		AbastractClassDemo.static_method();
	}
}
```

Interface program:-

```
	interface InterfaceDemo {
	abstract void m1();
	public void m2();

	default  void default_method()
	{
	System.out.println("Default method");
	}

	static void static_method()
	{
		System.out.println("This is static method...");
	}

	 }
	 class B implements InterfaceDemo
	 {
	public void m1()
	{
		System.out.println("calling m1() of interface...");
	}
	public void m2() {
		System.out.println("calling m2() of interface...");
	}
	public static void main(String[] args) {
		B b=new B();
		b.m1();
		b.m2();
		InterfaceDemo.static_method();
	}
	}
```

27.Explain encapsulation why we are using it with an example?
Encapsulation is a process in which we are wrapping data member and member function together in a single unit called encapsulation.We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use **setter and getter methods to set and get the data** in it. **java bean class is an example of a fully encapsulated class**.

```
Ex-:public class EncapsulationDemo {
	private String name;
	private String city;
	private int age;

	public void setName(String name)
	{
		this.name=name;
	}

	public String getName()
	{
		return name;
	}
	public void setCity(String city)
	{
		this.city=city;
	}
	public String getCity()
	{
```

```
            return city;
        }

        public void setAge(int age)
        {
            this.age=age;
        }

        public int getAge()
        {
            return age;
        }

        public static void main(String[] args) {
            EncapsulaitonDemo e=new EncapsulationDemo();
            e.setName("amarjeet");
            e.setCity("Hyd");
            e.setAge(22);

System.out.println(e.getName()+"\n"+e.getCity()+"\n"+e.getAge()
);
        }
        }
```

Note:-we are using encapsulation for security purposes or data hiding.

28.What is access modifier and how many types of modifier explain?
Access modifiers are **used to set the accessibility of classes, constructors, methods, and other members of Java** called access modifier.there are 4 types of modifier.

1. **Public** → it is accessible from anywhere like within the class,outside the class,within the package and outside the package called public.
2. **Private**→ it is only accessible from within the class called private.
3. **Protected**→ it is only accessible from within the package and outside the package through child class.
4. **Default**→ it is only accessible from within the package.

29.What is the super keyword and why are we using it?

- super can be used to refer to the immediate parent class instance variable.
- super can be used to invoke the immediate parent class method.
- super() can be used to invoke immediate parent class constructor.

Ex-:class SuperDemo{

```
    String color="white"; //parent class variable
    public void m1()
     {
        System.out.println("method of m1()...");//parent class method
     }
     SuperDemo()
     {
        System.out.println("constructor..");//parent class constructor
      }
     }
    class Dog extends SuperDemo{
    public void m1()
    {
        System.out.println("method of child class of m1()..");// child class method m1()
    }
    String color="black";
    void printColor(){
    System.out.println(color);
    System.out.println(super.color);//calling parent class variable
    super.m1(); //calling parent class method
    }
    }
    class TestSuper1{
    public static void main(String args[]){
    Dog d=new Dog();
    d.printColor();
    d.m1();// method of child class
    }}
```

Note: we are using super to call parent call variable,method and constructor.The most common use of the super keyword is to eliminate the confusion between superclasses and subclasses that have methods with the same name.

30.What is this keyword and why are we using it?

- this can be used to refer to the immediate current class instance variable.
- this can be used to invoke the immediate current class method.
- this() can be used to invoke immediate class constructor.

31 What is the difference between this and super?
This is used to refer to the current class while super is used to refer to the parent class; this is the main difference between this and super.

32.what is the static keyword and why are we using it?

- **Static is a non-access modifier** applicable with **variable,method and block**.
- Main use of static keywords is **memory management.**
- Static keyword We can not use static keywords with local variables.

33.Explain static variable,method and block?
Static variable→ any variable which is declared with a static keyword called static variable and it is class level variable.

- The static variable can be used to refer to the common property of all objects.
  Ex-:company name,college name etc.
- The static variable gets memory only once in the class area at the time of class loading.

Static variable Program:-

```
public class StaticDemo {
        String name;
        static String village=" Sakari "; //command for every object
        int age;

        StaticDemo(String name,int age)
        {
                this.name=name;
                this.age=age;
        }
```

```
	public void show()
	{
		System.out.println(name+"\n"+village+"\n"+age);
	}

}
class Test
{
	public static void main(String[] args) {
		StaticDemo s=new StaticDemo("rajnish", 27);
		s.show();
		StaticDemo s1=new StaticDemo("amarjeet",22);
		s1.show();
		StaticDemo s2=new StaticDemo("hira",18);s2.show();}}
```

Static method→ static method **belongs to the class** not an object and we can call the **static method without creating an object** we can call with the help of class name and we can also **access and change the static data value.**

Static method program:-

```
public class StaticDemo {
	String name;
	static String village=”Sakari”;
	int age;

	static void changeData()
	{
		village="Turki";
	}

	void display()
	{
		System.out.println(name+"\n"+village+"\n"+age);
	}
	StaticDemo(String name,int age)
	{
		this.name=name;
		this.age=age;
```

```
        }
    }
   class Test
   {
        public static void main(String[] args) {
                StaticDemo.changeData();
                StaticDemo s=new StaticDemo("rajnish", 22);
                s.display();
                StaticDemo s1=new StaticDemo("amarjeet",22);
                s1.display();
                StaticDemo s2=new StaticDemo("hira",18);
                s2.display();
        }
     }
```

Static block→ static block will **alway execute first means before main method** at the time of class loading.if multiple static block will be in the program then it will execute from **top to bottom** means those block will come first will execute first.

Ex:-**class Main{**

```
   static{
   System.out.println("static block is invoked1");//called first
   }
   static{
   System.out.println("static block is invoked2");//called second
   }
   public static void main(String args[]){
   System.out.println("Hello main"); //then main method will called
   }
   }
```

34.what is the final keyword and why are we using it?

Final is a modifier which is applicable with **variable,method and class** and it is used to restrict the user means final **variable→ restrict the user to change value of variable,final method→ you can't override the method and final class→ you can't inherit or extend that class.**

35.what is the package and its advantages and disadvantages?

Package→ it is an encapsulation mechanism to group the related class and interface into a single unit called package.and always write the package name as the **reverse of the website domain(google.com)->(com.google)**.
Main purpose of use→ resolve naming conflict,provide security to the class and interface so outside persian can not be accessed and it improves readability of the code.

Ex:-
```
com.amarjeet.singh.package.tutorial//package name
Import java.util.*;//importing package
class public PackageDemo{
public static void main(String[] args)
{
   System.out.println("amarjeet");
}
}
```

Note:- Java(parent package)→ java.util,java.lang,java.io(sub package).

36.what is Exception Handling?

Exception→ an exception is an event which occurs during the execution of a program at run time that disturbs the normal flow of the program called exception.
Handling→ the way we are handling the exception called handling .

37.what is the difference between exception and error?

Exception	**Error**
Exceptions occur because of our program.	Error occurs because of lack of system resources.
Exceptions are recoverable means the program can handle them using try,catch blocks.	Error are not recoverable means we programmers can not handle the error.
Exception are two types:- ● Compile time/Checked Exception ● Run time/Unchecked Exception	Errors are one types: ● Run time/unchecked Exception.

Note:- Throwable is the parent class of Exception and Error.

38.what is compile time/checked exception and runtime unchecked exception?

Compile time/checked exception→ checked exception are those exceptions that are checked at compile time called compile time/checked exception.

Ex:- ClassNotFoundException,FileNotFoundException,IOException,SQLException etc.

Runtime/unchecked exceptions → unchecked exceptions are those exceptions that are checked at runtime called runtime/unchecked exceptions.
Ex:- ArithmeticException,NullPointerException,ArrayIndexOutOfBoundsException etc.

39.what is the difference between final,finally and finalize()?

final	finally	finalize()
final modifier or keyword applicable with: ● variable ● method ● class	finally is a block always associated with: ● try() ● try() & catch() block ● Perform clean up activities	finalize() is a method alway called by garbage collector just before destroying an object to perform clean up activities.

40.what is the difference between throw & throws?
throw:

- throw keyword is used to throw or create an exception manually otherwise default exception mechanism is responsible to create object.
- Using throw keyword We can throw only one exception in a program
- throw keyword used within the method body.
- throw followed by a new instance.
- throw keywords mainly for runtime/unchecked exceptions.

Syntax→ throw new Exception("you can not divide by zero");

throws:

- throws used to declare an exception in a program
- Using throws keywords we can declare multiple exceptions in a program.
- throws followed by method signature.
- throws keyword mainly used for compile time exception/checked exception.

Syntax→ throws IOException,ArithmeticException

41.what is multitasking,multiprocessing and multithreading?
Multitasking→ To perform multiple tasks at a single time called multitasking.

Multiprocessing→ When one system is connected with multiple processors in order to complete the task called multiprocessing.

Multithreading→ Executing multiple threads simultaneously at a single time called multithreading.

42.what is the difference between process and threads?
Process→
- Program under execution called process.
- Process is heavy weight.
- Each process has a different address space.
- Process takes more resources.
- Processes are not dependent on each other.

Threads→
- Thread is a sub part of the process.
- Thread is lightweight.
- Threads share the same address space.
- Thread takes less resources and threads are dependent on each other.

43.Life cycle of thread in java?
New → Runnable→ Running→ Non-Runnable→ Terminate.
Note:- Thread class present in java.lang package.

44.How many ways to create threads in java?
There are two ways in java to create thread.
1. **By extending Thread class(step is given below that how we are extending thread class)**.
    - Extend Thread class.
    - Override run() method.
    - Create an object of thread class.
    - Start() the thread.

Extending Thread Class Program:
```
public class ThreadDemo extends Thread{// extending thread class
public void run()//overriding run method of thread class
 {
     System.out.println("Thread is Running by extending Thread class...");
 }
 public static void main(String[] args) {
     ThreadDemo t=new ThreadDemo();//creating object of thread class
     t.start();//starting thread
}
}
```

2. **By Implementing Runnable interface(steps are given below that how we are implementing runnable interface).**
    - Implement Runnable interface.
    - Override the run() method.
    - Create the object for the main class .
    - Create the object for Thread class and pass the parameter of main class object to Thread class.
    - Start() the thread.

Runnable interface program:

```
        public class RunnableDemo implements Runnable//implementing Runnable
interface
        {
        public void run()//overriding the run method
        {
            System.out.println("Thread is Running by implementing Runnable...");
        }
        public static void main(String[] args) {
            RunnableDemo r=new RunnableDemo();//creating object for main class
            Thread t=new Thread(r);//creating Thread class object and passing the
main class parameter to Thread class
            t.start();//staring the thread
         }
        }
```

45.Which is better implementing a runnable or extending thread?
If you want to implement or extend any other class then Runnable interface is most preferable, otherwise, **if you do not want any other class to extend or implement the Thread class is preferable**. **When you extend the Thread class, after that you can't extend any other class which you required.**

46.what is daemon thread?
Demon threads are those threads which are running in the background called demon thread.and it is providing service for thread,life of the demon thread is dependent on the main thread whenever main thread will die then automatically the daemon thread will die.

47.what is the difference between sleep(),yield() and join()?
sleep()→ if any thread doesn't want to perform any task for a specific time then use sleep.

Ex:-Timer

Yield()→ it stops the current execution of the thread and gives a chance to another thread to complete the task.
Ex:-Billing counter

Join()→ it waits for another thread to complete and join.
Ex:- License department counter

Collection Interview Question:-

48.what is collection?

- Collection is a group of objects in a single unit called collection.
- Collection is a group of classes and interface collection collection.
- Collection is a root interface of the collection framework present in java.util.package.
- Collection framework provides ready-made architecture means to perform several operations on a group of objects.

49.why collection we are using in java?
→ **array is fixed in size** so collection **is growable in nature. To overcome this problem we are using collection** and if we want to perform any operation like sort,reverse,searching then the programmer needs to write the logic manually but in the collection interface all the ready made methods are available to perform all these operations.

50.what is the difference between collection and array.

array	**collection**
array is fixed in size.	Collection is growable in nature.
array can hold only homogeneous data	Collection can hold homogeneous and

elements.	heterogeneous data elements
array don't have any predefined method for sorting,searching operation .	Collection have ready made methods to perform searching and sorting operations.
Memory point of view array not recommended to use.	Memory point of view collection recommended to use.
Performance point of view array is recommended to use.	Performance point of view collection not recommended to use.

51.what is the difference between Collection and Collections?
Collection→ collection is a **root interface** of the collection framework it is present in **java.util.package** and collection is a group of **classes and interface** in a single unit,collection provides ready made data structure to perform several operations called collection.

Collections→ Collections is a **class** present in **java.util.package** and Collection provides several utility methods for collection interface like sorting and searching operations sort(),min(),max().

53.explain collection hierarchy?
Collection→ Here **i**→ means interface and **c**→ means class

1. **List(i)**
    - ArrayList**(c)**
    - LinkedList**(c)**
    - Vector**(c)**
    - Stack**(c)**
2. **Set(interface)**
    - HashSet**(c)**
    - LinkedHashSet**(c)**
    - SortedSet**(i)**
    - NavigableSet**(i)**
    - TreeSet**(c)**
3. **Queue(interface)**
    - PriorityQueue**(c)**
    - BlockingQueue**(c)**

Note:- Vector and **Stack** is legacy class means old class.

54.what is the difference between ArrayList & Vector?

ArrayList	Vector
● **Arraylist method is not synchronized.**	● **Vector method is synchronized.**
● **ArrayList increases its size by 50 percent.**	● **Vector increases its size by 100 Percent.**
● **ArrayList performance is high.**	● **Vector performance is low**
● **ArrayList was introduced in the 1.2 version so it is not a legacy class.**	● **Vector was introduced in the 1.0 version so it is a legacy class.**

Notes:- ArrayList and Vector Class implements RandomAccess interface.
How to get synchronize arraylist→ ArrayList al=new ArrayList();
List l1=Collections.SynchronizedList(al);

Disadvantages of ArrayList→ if our frequent operation is to insertion and deletion of data in the middle of arraylist then ArrayList is not recommended to use.

When to use a Vector→ if our operation is retrieval to get the data searching.
When not to use Vector→ when we need to insert and delete the data in middle

When to use ArrayList→ if our operation is retrieval to get the data or searching.
When not to use ArrayList→ when we need to insert and delete the data in middle

When to use LinkedList→ if we need to insert and delete in the middle.
When not to use LinkedList→ if our operation to retrieve or search the element

55.what is the difference between ArrayList & LinkedList?

ArrayList	LinkedList
ArrayList internally uses a dynamic array to store the elements.	**LinkedList internally uses a doubly linked list to store the elements.**
Manipulation with ArrayList is slow because it internally uses an array. If any element is removed from the array, all the bits are shifted in memory.	**Manipulation with LinkedList is faster than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.**

ArrayList class implements a List interface.so it acts as a list.	**LinkedList class implements both the List interface and the Deque interface so it acts as a list and a deque.**
ArrayList is better for storing and accessing data.	**LinkedList is better for manipulating data.**

56.what is the difference between List & Set?

List	Set
• **List is the child interface of Collection.**	• **Set is the child interface of Collection.**
• **List allow duplicate elements**	• **Set does not allow duplicate elements.**
• **List insertion order is preserved.**	• **In Set insertion order is not preserved.**

57.what is the difference between Enumeration, Iterator and ListItirator?

Enumeration	Iterator	ListIterator
• Applicable for legacy classes.	• Applicable for any collection objects.	• Only applicable for List objects.
• Single direction cursor means move forward only.	• Single direction cursor means move forward only.	• Bi-directional cursor means move forward and backward.
• Only can read the elements.	• Only can read and remove the elements.	• Read,remove,add and replace.
• Methods name→ hasMoreElements(). nextElement().	• Methods name→ hasNext(). next(). remove().	Methods name→ 9 methods are there in ListIterator cursor.

58.what is the difference between HashSet & LinkedHashSet?

HashSet	LinkedHashSet
● It uses a Hashtable data structure to store the elements	● It uses a HashTable and doubly linked list Data structure to store and maintain the insertion order of the elements.
● Insertion order is not preserved.	● Insertion order is preserved.
● Default initial capacity is 16.	● Default initial capacity is 16.
● It extends the AbstractSet class.	● It extends the HashSet class.

59.what is the difference between HashSet,LinkedHashSet and TreeSet?

HashSet	LinkedHashSet	TreeSet
● Using data structure HashTable.	● Using Data structure Hash Table+doubly Linked List	● Using a data structure balance tree.
● Insertion order not preserved.	● Insertion order is preserved.	● Insertion order not preserved.
● Duplicate not allowed	● Duplicate not allowed	● Duplicate not allowed
● Only one null value is possible to insert.	● Only one null value is possible to insert.	● Null insertion is not allowed.

Note:- if you try to insert null value in TreeSet then it will generate a NullPointerException.

Note:- Stack worked on the principle of LIFO(last in first out).
Methods→push()--> push elements
pop()--> pop or delete elements
peek()-->return top of the stack

empty()--> return true and false if stack is empty
search()--> search object in stacks

60.explain Map hierarchy ?
Map→ is used to store the data in the key and value pair called map

- **HashMap**
- **LinkedHashMap**
- **IdentityHashMap**
- **Hashtable(legacy)**
- **WeakHashMap**
- **SortedMap(i)**
- **NavigableMap(i)**
- **TreeMap**

61.what is the difference between HashMap and Hashtable?

HashMap	Hashtable
● HashMap is not synchronized.	● Hashtable is synchronized
● HashMap performance is high.	● Hashtable performance is slow.
● HashMap allows one null key and multiple null values.	● Hashtable doesn't allow any null key and null value.
● It was introduced in the 1.2 version so it is not a legacy class.	● Hashtable was introduced in the 1.0 version so it is a legacy class.

• **HashMap inherits the AbstractMap class.**	• **Hashtable inherits Dictionary class.**

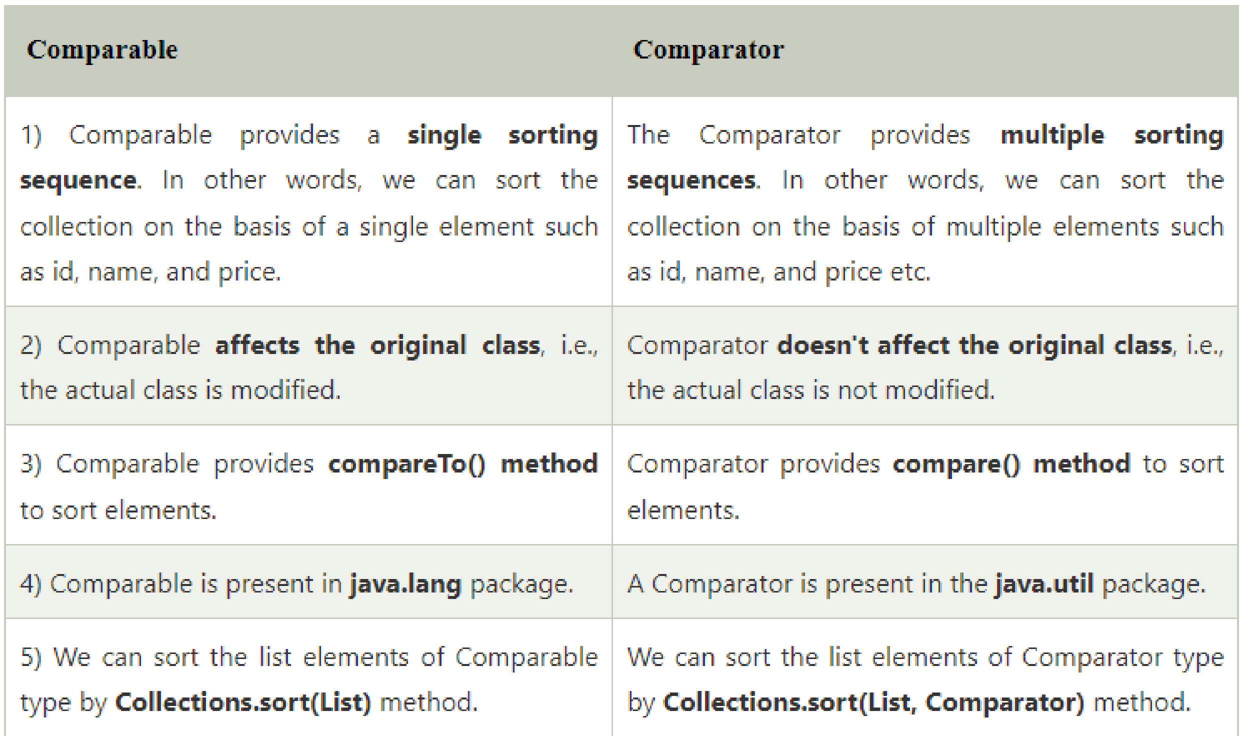
65.what is the difference between HashMap & LinkedHashMap?

HashMap	LinkedHashMap
• **Hashmap uses array of buckets along with a technique called hashing**	• **Using LinkedList and Hashtable data structure.**
• **Insertion order is not preserved,it is based on hash code only.**	• **Insertion order is preserved.**
• **Introduced in 1.2. Version.**	• **Introduced in 1.4 version.**
• **HashMap inherit AbstractMap class**	• **LinkedHashMap inherits HashMap class.**

62.How to get a synchronized version of map?
HashMap h=new HashMap()
Map m=Collections.synchronizedMap(h);

63.what is an entry in map?
Entry is a key & value pair called entry.

64.what is TreeMap?
→ TreeMap using Red-Black-Tree and insertion order not preserved ,TreeMap is non synchronized.



Comparable	Comparator
1) Comparable provides a **single sorting sequence**. In other words, we can sort the collection on the basis of a single element such as id, name, and price.	The Comparator provides **multiple sorting sequences**. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc.
2) Comparable **affects the original class**, i.e., the actual class is modified.	Comparator **doesn't affect the original class**, i.e., the actual class is not modified.
3) Comparable provides **compareTo() method** to sort elements.	Comparator provides **compare() method** to sort elements.
4) Comparable is present in **java.lang** package.	A Comparator is present in the **java.util** package.
5) We can sort the list elements of Comparable type by **Collections.sort(List)** method.	We can sort the list elements of Comparator type by **Collections.sort(List, Comparator)** method.

<u>Java 8 Features:</u>

- **Lamda Expression**
- **Functional Interface**
- **Method Reference**
- **Static and Default methods**
- **Stream API**
- **Optional Class**
- **Date & Time API**

66.explain lambda expression with an example?

Lambda expression is an anonymous function which does not have.

- Name
- Modifier
- Any return type

(→) symbol used to lambda expression

Example1:-
```
public void add()
{
  System.out.println("without lambda expression ");
 }
```

→ here public is modifier,void is return type,add is method name

Example2:- (a,b)->System.out.println(“with lambda expression ”);

1. Hello world using lambda expression:

```
interface inter{
   public void m1();//functional interface

}
public class Main{
   public static void main (String[] args) {
      inter m=()->System.out.println("Hi Amarjeet..");//lambda expression
      m.m1();
   }}
```

2.Sum of two numbers using lambda expression.

```
interface inter{
   public void m1(int a,int b);

}
public class Main{
   public static void main (String[] args) {
      inter m=(a,b)->System.out.println("sum of two number is:"+(a+b));
      m.m1(10,20);
   }
}
```

3.square of two number using lambda expression

```
interface sqrt{
   public void sqrt1(int n);
}
class Main{
   public static void main (String[] args) {
      sqrt m=n->System.out.println(n*n);
      m.sqrt1(10);
   }
}
```

Lambda expression→ functional interface→ steam api

Note:without a functional interface we can not call the lambda expression.

67.what is the advantage of lambda expression?
One of the most benefits of a lambda expression is to reduce the amount of code. We know that lambda expressions can be used only with a functional interface. For instance, Runnable is a functional interface, so we can easily apply lambda expressions.

68.if lambda expression does not have a name then how to call lambda expression and who will call?
Functional interface will call lambda expression
69.What is a functional interface in java 8?

- Functional interface used to call lambda expression.
- Functional interfaces contain a single abstract method(SAM).
- An interface which has only one abstract method called functional interface.
- We can use any number of static and default methods in a functional interface.

70.what is @FunctionalInterface annotation in java 8?
It is used to identify or indicate if the interface is functional interface or not.

Note:-

- **Till 1.7 version** Every method present inside the interface is always **public and abstract.**
  Ex:- void m1();
  ```
  public void m1();
  abstract void m1();
  public abstract void m1();
  ```

- **Till 1.8 version** every method present inside the interface is **default+static method.**
- **Till 1.9 version** private methods are allowed in the interface.
- abstract class **does not support private methods** but **default,static and final is supported.**
- Every variable present inside interface is **always public,static and final**

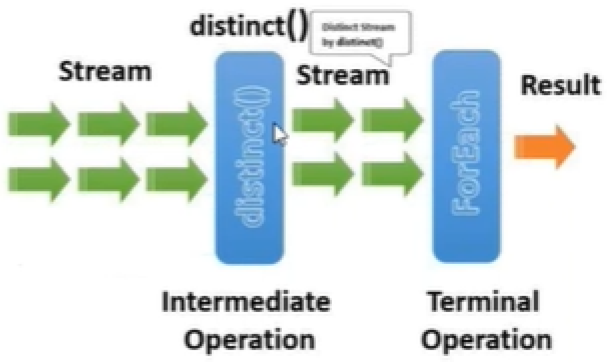
71.what is the default method in java 8?

Without affecting the implementation class if we want to add a new method inside the interface then the default method is best.

```
Ex:- interface InterfaceDemo {
        public void m1();
        public void m2();
        public void m3();
        public void m4();
        public void m5();
        public default void m6()
        {
                System.out.println("I am default method..");
        }
        static void m7()
        {
                System.out.println("I am static method..");
        }
}
class Test implements InterfaceDemo{
        public void m1() {
                System.out.println("i am inside m1()....");

        }
        public void m2() {
                System.out.println("i am inside m2()...");

        }
        public void m3() {
                System.out.println("i am inside m3()...");

        }
        public void m4() {
                System.out.println("i am inside m4()...");

        }
        public void m5() {
                System.out.println("i am inside m5()...");

        }
}
```

```
class Test1{
        public static void main(String[] args) {
                Test t=new Test();
                t.m1();
                t.m2();
                t.m3();
                t.m4();
                t.m5();
                t.m6();
                InterfaceDemo.m7();
        }

}
```

Intermediate operations:
map(),filter(),distinct(),sorted(),limit(),skip(),toUpperCase,toLowerCase().
Terminal operations:
forEach(),toArray(),reduce(),collect(),min(),max(),count(),anyMatch(),allMatch(),
noneMatch(),findFirst(),findAny().
Short circuit operations: findFirst(),findAny(),anyMatch(),allMatch(),noneMatch(),limit().



Q.Can we use multiple intermediate operations?
Ans-> We can use any number of intermidiate operations but only one terminal operation.

Java tricky interview questions

1.When to use static and when to use instance variable out program.

→ if our data changes frequently then we need to use an instance variable.
Ex-student name sometime ramesh,hira and sanjeet
→ if our data is common for all the objects then use a static variable.
Ex-college name,company name

2.Can we overload the main method?
Yes we can overload the main method but we can not override main method
Ex:-**public class MainMethodOverload1**

```
{
public static void main(Integer args)
{
System.out.println("Overloaded main() method invoked that parses an integer argument");
}
public static void main(char args)
{
System.out.println("Overloaded main() method invoked that parses a char argument");
}
public static void main(String[] args)
{
System.out.println("Original main() method invoked");
MainMethodOverload1 m=new MainMethodOverload1();
m.main(10);
}
}
```

3.can we override the main method?
No, we can not override the main method. Because main method is static and static method can not be overridden became static belongs to class and share common memory for all if you do any changes in overridden method it will affect all.

4.can we overload constructor?
Yes! Java supports constructor overloading.

5.can we override the constructor?
No,we can not override constructor in java because if you override the constructor by extending class then you have to create the constructor for child class on for base class this is the reason you could not override the main method.
Ex-

```
public class A {
        A() {
                System.out.println("I am base class constructor");
        }
}
```

```
class B extends A {
        A(){  //error while overriding constructor
                System.out.println("I am child class constructor");
        }
        public static void main(String[] args) {
                B b = new B();
        }
}
```

6.can we overload static methods?
Yes, we can overload static method.We can have two or more static methods with the same name, but differences in input parameters.

7.can we override the static method?
No, we cannot override static methods because static is belong to the class and static common for all if you change anything by overring then it will affect all the properties this is the reason method overring is not possible.

8.can we write concrete methods inside abstract methods?
Yes,we can use concrete method inside abstract class

9.why java does not support operator overloading?
If you allow programmers to do operator overloading they will come up with multiple meanings for the same operator which will make the learning curve of any developer hard and things more confusing and messing. Or we can say **because it's just a choice made by its creators who wanted to keep the language more simple.**

10.Which is better implementing a Runnable interface or extending thread?
If you want to implement or extend any other class then **Runnable interface is most preferable**, otherwise, **if you do not want any other class to extend or implement the Thread class is preferable**. When you extend the Thread class, after that you can't extend any other class which you require.

11.What if I write `static public void` instead of the `public static void`?
Yes,we can change the modifier order in the program.
Ex:-

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

```
        }

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

12. Can we inherit constructor?
Constructors are not member functions, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.

13.can we make the constructor final?
No, a constructor can't be made final.

14.why the main method is static?
The main() method is static because JVM can call main without creating the object of the class.

15.can we execute a program without a main method?
Yes You can compile and execute without the main method By using static block.

16.can we make the constructor static?
No, we cannot define a static constructor in Java.

17.Can we make abstract methods static?
No

18.can we override private methods?
No, we cannot override private because the scope of the private is within the class if you override then you have to extend it in another class and from another class it is not possible to access private method this is the reason private method is not overridden.

```
Ex:- public class A{
        private void run() {
                System.out.println("I am running");
        }

}
class B extends A{
        private voud run() error while overriding
        {
                System.out.println("I am eating");
```

```
		}
		public static void main(String[] args) {
			B b=new B();
			b.run();
		}
}
```

19.can we make the main() method final.
Yes

20.can we make the interface final?
If you make an interface final, you cannot implement its methods which defies the very purpose of the interfaces. Therefore, you cannot make an interface final in Java.

JDBC

1.what is JDBC and why do we use it?

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database.

2.what is JDBC driver and its type?

JDBC Driver is a software component that enables java applications to interact with the database. There are 4 types of JDBC drivers.

- JDBC-ODBC Bridge Driver(we use it and version is 4.3),
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

3.What are the steps to connect to the database in java?

- **Registering the driver class:**

The forName() method of the Class class is used to register the driver class. This method is used to load the driver class dynamically. Consider the following example to register OracleDriver class.

Ex:- Class.forName("oracle.jdbc.driver.OracleDriver");

- **Creating connection:**

  The getConnection() method of the DriverManager class is used to establish the connection with the database. The syntax of the getConnection() method is given below.

  Ex:- Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/databaseName", "Username", "Password");

- **Creating the statement:**

  The createStatement() method of Connection interface is used to create the Statement. The object of the Statement is responsible for executing queries with the database.

  Ex:- Statement st=connection.createStatement();

- **Executing the queries:**

  The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

  Ex:- ResultSet rs=stmt.executeQuery("select * from emp");

```
        while(rs.next()){
        System.out.println(rs.getInt(1)+" "+rs.getString(2));
        }
    st.executeUpdate("create table student(id int primary key auto_increment,
                            name varchar(25), city varchar(25), company varchar(25))");
```

4.what are the JDBC API components?
Interfaces:-

- **Connection:** The Connection object is created by using the getConnection() method of DriverManager class. DriverManager is the factory for connection.
- **Statement:** The Statement object is created by using createStatement() method of Connection class. The Connection interface is the factory for Statement. It is used to execute static queries. It is slow because it needs to compile every time..
- **PreparedStatement:** The PrepareStatement object is created by using prepareStatement() method of Connection class. It is used to execute the parameterized query and it is faster because it needs to compile only once..
- **CallableStatement:** CallableStatement interface is used to call the stored procedures and functions. We can have business logic on the database through the use of stored procedures and functions that will make the performance better because these are precompiled. The prepareCall() method of Connection interface returns the instance of CallableStatement.
- **ResultSet:** The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points before the first row. The executeQuery() method of Statement interface returns the ResultSet object.

Classe:

- **DriverManager:** The DriverManager class acts as an interface between the user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. It contains several methods to keep the interaction between the user and drivers.

5. Difference between executeQuery(), executeUpdate() and Execute()

executeQuery()	**executeUpdate()**	**Execute()**
The execute method can be used for any SQL statements(Select and Update both).	The executeQuery method can be used only with the select statement.	The executeUpdate method can be used to update/delete/insert operations in the database.
Return type would be int	Return type would be ResultSet obj	Return type would be boolean

Note: Core Java & Jdbc Completed.

Servlet Interview questions

1.What is servlet?

- **Servlet is a technology which is used to create web app applications.**
- **Servlet is an API which provides several classes and interfaces including documentation.**
- **Servlet is a class that handles requests, processes them and reply back with a response**

2.What is the lifecycle method of servlet?

init()→it will be called only once at the first request and it is used to initialize a servlet object.

service(ServletRequest request,ServletResponse)→it will call as many times as the user will hit for the request and it takes two parameter requests and responses.

destroy()→it will call only once.

3.Who is responsible for creating servlet objects?

Web container or servlet container.

4.when servlet object will be created?
At the time of first request

5.what is the difference between get and post?

Get	Post
● Get is sending limited data because data is sent in the header.	● Large amount of data can be sent because data is sent in the body.
● Get data is not secure because data is exposed in the url bar.	● Post data is secure data is not exposed in url bad.
● Get can be bookmarked	● Post cannot be bookmarked.

6.what is the difference between PrintWritter and ServletOutputStream?
PrintWriter is a character-stream class whereas ServletOutputStream is a byte-stream class. The PrintWriter class can be used to write only character-based information whereas the ServletOutputStream class can be used to write primitive values as well as character-based information**.**
7.what is the difference between GenericServlet and HttpServlet?
HttpServlet is protocol independent whereas HttpServlet is protocol specific.

8.what is servlet collaboration?
When one servlet is communicating with another servlet called servlet collaboration.there are many ways of servlet collaboration.

1. **RequestDispatcher interface.**
    - Include method
    - Forward method
2. **SendRedirect method.**

9.difference between Forward and SendRedirect?

Forward	SendRedirect
● Forward method works at the server side.	● SendRedirect works at the client side.
● Forward always sends the	● SendRedirect always sends

same request and response object to another servlet.	new requests.
• Forward works within the servlet only.	• SendRedirect works within and outside the servlet

10.what is the difference between ServletConfig and ServletContext?
The container creates an object of ServletConfig for each servlet whereas an object of ServletContext is created for each web application.

11.what is the difference between cookie and HttpSession?
Cookie works at client side whereas HttpSession works at server side.

12.what is session tracking?
Session Tracking is a way to maintain state of an user.Http protocol is a stateless protocol.Each time user requests to the server, server treats the request as the new request.So we need to maintain the state of an user to recognize a particular user.

13.what is cookie?
A cookie is a small piece of information that is persisted between multiple client requests. A cookie has a name, a single value, and optional attributes.

JSP Interview Questions

1.what is Jsp?
Java Server Pages technology (JSP) is a server-side programming language used to create a dynamic web page in the form of HyperText Markup Language (HTML). It is an extension to the servlet technology.a jsp page is internally converted into the servlet.

2.what are the lifecycle methods of jsp?

- **jspinit()**
- **_jspService()**
- **jspDestroy()**

3.what is the difference between hide comment and output comment?
The JSP comment is called hide comment whereas HTML comment is called output comment. If a user views the source of the page, the JSP comment will not be shown whereas HTML comment will be displayed.

4.what are the jsp implicit objects?

1. `Out→ jspWriter`
2. `Request → HttpServletRequest`
3. `Response → HttpServletResponse`
4. `Config → ServletConfig`
5. `Session → HttpSession`
6. `Application → ServletContext`
7. `pageContext → PageContext`
8. `Page → Object`
9. `Exception → Throwable`

5.what are jsp directives and their types?
The jsp directives are messages that tells the web container how to translate a JSP page into the corresponding servlet

1. Page directives
   `Syntax: -<%@ page attribute="value" %>`
2. Include directives
   `Syntax: <%@ include file="resourceName" %>`
3. Taglib directives
   Syntax: <%@ taglib uri="uriofthetaglibrary"prefix="prefixoftaglibrary" %>

6.what are jsp action elements?

1.	**Jsp:forward**	**jsp:useBean**	**jsp:getProperty**	**jsp:param**
2.	**Jsp;include**	**jsp:setProperty**	**jsp:plugin**	**jsp:fallback**

Note: J2SE & J2EE Completed with below topics Importanat topics.

1. String
2. Exception Handling
3. Multi-Threading
4. OOP'S
5. This, Super, Final, Static, Transient, Volatile
6. Collection
7. Map
8. Jdk8
9. Servlet
10. JSP