@jobspandit

Java Simple (Freshers Level) Interview questions and Answers

1. What is Java?

Java is a popular object-oriented programming language. It is defined as a complete collection of objects. By using Java, we can develop lots of applications such as gaming, mobile apps, and websites.

2. Explain in brief the history of Java?

In the year 1991, a small group of engineers called 'Green Team' led by James Gosling, worked a lot and introduced a new programming language called "Java". This language is created in such a way that it is going to revolutionize the world.

In today's World, Java is not only invading the internet, but also it is an invisible force behind many of the operations, devices, and applications.

3. What is the difference between C and Java?

The differences between C and Java are as follows:

C- Language	Java
C language was developed by Dennis M. Ritchie in the year 1972.	Java was developed by James Gosling in the year 1995.
C is Procedural-Oriented	Java is Object-Oriented
Functions play a major role in the C language.	Objects play a major role in the Java language.
It is a middle-level language	It is a high-level language.
C language does not support OOPS Concepts.	Java supports OOPS concepts, in which Inheritance the main property used for code reusability.
In C, memory is allocated using Malloc	In Java, memory is allocated using a new keyword.
The threading concept is not supported by C	Java supports Threading
It is not portable	It is portable
Default members of C are public	Default members of Java are private
Storage classes are supported by C	Storage classes are not supported in Java

4. What is the reason why Java came into existence?

Java is the first programming language that is used to write code on a virtual machine, so that is the reason why it is called JVM (Java Virtual Machine). This JVM is

a new concept that is introduced in Java. It also provides a new feature called code reusability which is not possible in C.

5. Compare both C++ and Java?

Java	C++
Java is platform-independent	C++ is platform dependent
Goto statement is not supported by Java	C++ supports the goto statement
Multiple inheritances are supported in java by using interfaces	Multiple inheritances are supported in C++
Java doesn't support structures and unions	C++ does support unions and structures
Java has built-in support for threading	C++ does not provide built-in support for threading
Java is used for building applications	C++ is used for system programming
Java uses both interpreter and compiler	C++ uses compiler only

6. What are the prominent features of Java?

The following are the notable features in Java:

Dynamic: Java is more dynamic when compared to C++ and C. Java is designed in such a way that it is adaptable to any the evolving environments.

Simple: Java is very easy to learn and code.

Object-oriented: In Java, everything is based on objects.

Secure: Java provides a secure platform to develop safe and virus-free applications.

Platform Independent: Unlike C and C++, when we compile Java code it is compiled into platform-independent bytecode rather than the platform-specific machine code.

Portable: Java provides no implementation aspects for the specifications which make Java portable.

Multithreaded: By this feature, we can perform multiple tasks simultaneously in Java.

High-Performance: With the built-in Just-in-Time compiler, Java provides high performance.

Robust: Java makes more efforts to eliminate errors by concentrating more on runtime checking and compile-time check.

7. What is a Class in Java?

Class is defined as a template or a blueprint that is used to create objects and also to define objects and methods.

8. Define Object in Java?

The instance of a class is called an object. Every object in Java has both state and behavior. The state of the object is stored in fields and the behavior of the objects is defined by methods.

9. How to write a basic Hello World program in Java?

```
class Career4freshers

{

public static void main(String args[ ])

 {

System.out.println("Hello World");

}

}
```

10. Define JVM?

JVM is the abbreviation for Java Virtual Machine. It is a virtual machine that provides a runtime environment to write code. JVM is a part of JRE (Java Runtime Environment) and is used to convert the bytecode into machine-level language. This machine is responsible for allocating memory.

11. What do you understand by Java virtual machine?

Java Virtual Machine is a virtual machine that enables the computer to run the Java program. JVM acts like a run-time engine which calls the main method present in the Java code. JVM is the specification which must be implemented in the computer system. The Java code is compiled by JVM to be a Bytecode which is machine independent and close to the native code.

12. What is the difference between JDK, JRE, and JVM?

JVM

JVM is an acronym 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. Its implementation has been provided by Oracle and other companies. Its implementation is known as JRE.

JVMs are available for many hardware and software platforms (so JVM is platform dependent). It is a runtime instance which is created when we run the Java class. There are three notions of the JVM: specification, implementation, and instance.

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.

JDK

JDK is an acronym for Java Development Kit. It is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools. JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

Standard Edition Java Platform

Enterprise Edition Java Platform

Micro Edition Java Platform

13. How many types of memory areas are allocated by JVM?

Many types:

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: Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return. Each thread has a private JVM stack, created at the same time as the thread. A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

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.

14. Define the JIT compiler in Java?

Just In Time Compiler is the component of JRE, which is used to compile the bytecodes of the particular method into the native machine code. This compiled code of the method is directly called by JVM without interpreting it

15. What are the differences between Heap and Stack Memory in Java?

Stack is generally used to store the order of method execution and local variables. In contrast, Heap memory is used to store the objects. After storing, they use dynamic memory allocation and deallocation.

16. Define variables in Java and explain with example?

Variables in Java can be defined as a basic storage unit of a program. It is a storage unit that holds the value during the program execution. Always the variable is assigned with a datatype.

For Example:

int a = 10;

17. What are the different types of variables in Java?

There are mainly three different types of variables available in Java, and they are:

Static Variables

Local Variables

Instance Variables

Static Variables: A variable that is declared with the static keyword is called a static variable. A static variable cannot be a local variable, and the memory is allocated only once for these variables.

Local Variables: A variable that is declared inside the body of the method within the class is called a local variable. A local variable cannot be declared using the static keyword.

Instance Variables: The variable declared inside the class but outside the body of the method is called the instance variable. This variable cannot be declared as static and its value is instance-specific and cannot be shared among others.

Example:

```
class A{
int num=30;//instance variable
static char name=pranaya;//static variable
void method(){
int n=90;//local variable
}
}//end of class
```


18. What is Typecasting?

Typecasting in Java is done explicitly by the programmer; this is done to convert one data type into another data type.

Widening (automatically) - conversion of a smaller data type to a larger data type size.

byte -> short -> char -> int -> long -> float -> double

Narrowing (manually) - converting a larger type to a smaller size type

double -> float -> long -> int -> char -> short -> byte

19. What is Type Conversion?

Type conversion can be defined as converting one data type to another data type automatically by the compiler.

There are two types of type conversions, and they are:

Implicit type conversion

Explicit type conversion.

20. Will the program run if we write static public void main?

Yes, the program will successfully execute if written so. Because, in Java, there is no specific rule for the order of specifiers

21. Why Java is platform independent?

Java is called platform independent because of its byte codes which can run on any system irrespective of its underlying operating system.

22. Why Java is not 100% Object-oriented?

Java is not 100% Object-oriented because it makes use of eight primitive data types such as boolean, byte, char, int, float, double, long, short which are not objects.

23. What are the data types in Java?

Datatypes in Java specify the values and sizes that can be stored in the variables. There are mainly two types of data types; they are:

Primitive Data Types

Non-primitive Data Types

24. What are primitive data types?

Primitive data types in Java are the major constituents of data manipulation. These are the most basic data types that are available in Java. The primitive data types include int, char, byte, float, double, long, short, and boolean.

25. What are the various access specifiers in Java?

In Java, access specifiers are the keywords which are used to define the access scope of the method, class, or a variable. In Java, there are four access specifiers given below.

Public The classes, methods, or variables which are defined as public, can be accessed by any class or method.

Protected Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.

Default Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope.

Private The private class, methods, or variables defined as private can be accessed within the class only.

26. What is the purpose of static methods and variables?

The methods or variables defined as static are shared among all the objects of the class. The static is the part of the class and not of the object. The static variables are stored in the class area, and we do not need to create the object to access such variables. Therefore, static is used in the case, where we need to define variables or methods which are common to all the objects of the class.

For example, In the class simulating the collection of the students in a college, the name of the college is the common attribute to all the students. Therefore, the college name will be defined as static.

27. What if we write public static void as static public void?

Both the compilation and execution of the programs are done correctly because in Java specifiers order doesn't matter.

28. How many types of operators are available in Java?

Eight types of operators are available in java, and they are:

Arithmetic operators

Assignment operators

Logical operators

Relational operators

Bitwise operators

Unary operators

Ternary operators

Shift operators

29. What is the difference between ++a and a++ increment operators?

++a is a prefix increment and a++ is the postfix increment. The prefix increment is used to return the value after incrementing the present value. Whereas in postfix increment, the value is returned before incrementing it.

30. What are the advantages of Packages in Java?

There are various advantages of defining packages in Java.

Packages avoid the name clashes.

The Package provides easier access control.

We can also have the hidden classes that are not visible outside and used by the package.

It is easier to locate the related classes.

31. What are Loops in Java? What are three types of loops?

Ans: Looping is used in programming to execute a statement or a block of statement repeatedly. There are three types of loops in Java:

1) For Loops

For loops are used in java to execute statements repeatedly for a given number of times. For loops are used when number of times to execute the statements is known to programmer.

2) While Loops

While loop is used when certain statements need to be executed repeatedly until a condition is fulfilled. In while loops, condition is checked first before execution of statements.

3) Do While Loops

Do While Loop is same as While loop with only difference that condition is checked after execution of block of statements. Hence in case of do while loop, statements are executed at least once.

32. How to reverse a string in Java?

```
String str = ""Hello"";

String reverse(String str){
  StringBuilder sb = new StringBuilder();
  sb.append(str);
  sb.reverse();
  return sb.toString();
}
```


33. What is object-oriented paradigm?

- It is a programming paradigm based on objects having data and methods defined in the class to which it belongs.
- Object-oriented paradigm aims to incorporate the advantages of modularity and reusability.
- Objects are the instances of classes which interacts with one another to design applications and programs. There are the following features of the object-oriented paradigm.

- Follows the bottom-up approach in program design.
- Focus on data with methods to operate upon the object's data
- Includes the concept like Encapsulation and abstraction which hides the complexities from the user and show only functionality.
- Implements the real-time approach like inheritance, abstraction, etc.
- The examples of the object-oriented paradigm are C++, Simula, Smalltalk, Python, C#, etc.

34. What is an object?

The Object is the real-time entity having some state and behavior. In Java, Object is an instance of the class having the instance variables as the state of the object and the methods as the behavior of the object. The object of a class can be created by using the new keyword.

35. What is the difference between an object-oriented programming language and object-based programming language?

There are the following basic differences between the object-oriented language and object-based language.

Object-oriented languages follow all the concepts of OOPs whereas, the **object-based language** doesn't follow all the concepts of OOPs like inheritance and polymorphism.

Object-oriented languages do not have the inbuilt objects whereas Object-based languages have the inbuilt objects, for example, JavaScript has window object.

Examples of object-oriented programming are Java, C#, Smalltalk, etc. whereas the examples of object-based languages are JavaScript, VBScript, etc.

36. What is the difference between Array list and vector in Java?

ArrayList	Vector
Array List is not synchronized.	Vector is synchronized.
Array List is fast as it's non-synchronized.	Vector is slow as it is thread safe.
If an element is inserted into the Array List, it increases its Array size by 50%.	Vector defaults to doubling size of its array.
Array List does not define the increment size.	Vector defines the increment size.
Array List can only use Iterator for traversing an Array List.	Vector can use both Enumeration and Iterator for traversing.

37. What is the difference between equals() and == in Java?

Equals() method is defined in Object class in Java and used for checking equality of two objects defined by business logic.

"==" or equality operator in Java is a binary operator provided by Java programming language and used to compare primitives and objects. public boolean equals(Object o) is the method provided by the Object class. The default implementation uses == operator to compare two objects. For example: method can be overridden like String class. equals() method is used to compare the values of two objects.

38. What is the constructor?

The constructor can be defined as the special type of method that is used to initialize the state of an object. It is invoked when the class is instantiated, and the memory is allocated for the object. Every time, an object is created using the new keyword, the default constructor of the class is called. The name of the constructor must be similar to the class name. The constructor must not have an explicit return type.

39. How many types of constructors are used in Java?

Based on the parameters passed in the constructors, there are two types of constructors in Java.

Default Constructor: default constructor is the one which does not accept any value. The default constructor is mainly used to initialize the instance variable with the default values. It can also be used for performing some useful task on object creation. A default constructor is invoked implicitly by the compiler if there is no constructor defined in the class.

Parameterized Constructor: The parameterized constructor is the one which can initialize the instance variables with the given values. In other words, we can say that the constructors which can accept the arguments are called parameterized constructors.

40. What is the purpose of a default constructor?

The purpose of the default constructor is to assign the default value to the objects. The java compiler creates a default constructor implicitly if there is no constructor in the class.

```
class Student3{
int id;
String name;

void display(){System.out.println(id+" "+name);}

public static void main(String args[]){
Student3 s1=new Student3();
Student3 s2=new Student3();
s1.display();
s2.display();
}
}
```

Output:

```
0 null
0 null
```

41. Does constructor return any value?

Ans: yes, The constructor implicitly returns the current instance of the class (You can't use an explicit return type with the constructor)

42. What are the differences between the constructors and methods?

Java Constructor	Java Method
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor name must be same as the class name.	The method name may or may not be same as class name.

43. What is the output of the following Java program?

```
public class Test
{
   Test(int a, int b)
   {
      System.out.println("a = "+a+" b = "+b);
   }
   Test(int a, float b)
   {
      System.out.println("a = "+a+" b = "+b);
   }
   public static void main (String args[])
   {
      byte a = 10;
      byte b = 15;
```

```
        Test test = new Test(a,b);
    }
}
```

The **output** of the following program is:

a = 10 b = 15

44. What is the output of the following Java program?

```
class Test
{
    int test_a, test_b;
    Test(int a, int b)
    {
    test_a = a;
    test_b = b;
    }
    public static void main (String args[])
    {
        Test test = new Test();
        System.out.println(test.test_a+" "+test.test_b);
    }
}
```

There is a compiler error in the program because there is a call to the default constructor in the main method which is not present in the class. However, there is only one parameterized constructor in the class Test. Therefore, no default constructor is invoked by the constructor implicitly.

45. What is the static variable?

The static variable is used to refer to the common property of all objects (that is not unique for each object), e.g., The company name of employees, college name of students, etc. Static variable gets memory only once in the class area at the time of

class loading. Using a static variable makes your program more memory efficient (it saves memory). Static variable belongs to the class rather than the object.

//Program of static variable


```
class Student8{
  int rollno;
  String name;
  static String college ="ITS";

  Student8(int r,String n){
  rollno = r;
  name = n;
  }
 void display (){System.out.println(rollno+" "+name+" "+college);}

 public static void main(String args[]){
 Student8 s1 = new Student8(111,"Karan");
 Student8 s2 = new Student8(222,"Aryan");

 s1.display();
 s2.display();
 }
}
```

Output: 111 Karan ITS

222 Aryan ITS

46. Why is the main method static?

Because the object is not required to call the static method. If we make the main method non-static, JVM will have to create its object first and then call main() method which will lead to the extra memory allocation.

47. Can we override the static methods?

No, we can't override static methods.

48. Can we make constructors static?

As we know that the static context (method, block, or variable) belongs to the class, not the object. Since Constructors are invoked only when the object is created, there is no sense to make the constructors static. However, if you try to do so, the compiler will show the compiler error.

49. Describe in brief OOPs concepts?

OOPs is an abbreviation for Object-Oriented Programming Language which entirely deals with an object. The main aim of OOPs is to implement real-world entities such as objects, classes, inheritance, polymorphism, and so on. Simula is considered the first object-oriented programming language. The most popular programming languages are Java, Python, PHP, C++, and many others.

The following are the OOPs concepts that are included in Java:

Class

Object

Abstraction

Encapsulation

Inheritance

Polymorphism

50. What is Abstraction?

In simple words, abstraction can be defined as hiding unnecessary data and showing or executing necessary data. In technical terms, abstraction can be defined as hiding internal processes and showing only the functionality.

51. Define encapsulation?

Binding data and code together into a single unit are called encapsulation. The capsule is the best example of encapsulation.

52. What is Inheritance in Java?

When an object of child class has the ability to acquire the properties of a parent class then it is called inheritance. It is mainly used to acquire runtime polymorphism and also it provides code reusability.

53. What is Polymorphism in Java?

Polymorphism in Java provides a way to perform one task in different possible ways. To achieve polymorphism in Java we use method overloading and method overriding. For example, the shape is the task and various possible ways in shapes are triangles, rectangle, circle, and so on.

54. What are the advantages of OOPs?

The following are the advantages of OOPs, and they are:

- OOPs provides data hiding
- OOPs makes development and maintenance easier when compared to a procedural programming language.
- OOPs has the ability to stimulate real-world entities more effectively.

55. What is this keyword in java?

The this keyword is a reference variable that refers to the current object. There are the various uses of this keyword in Java. It can be used to refer to current class properties such as instance methods, variable, constructors, etc. It can also be passed as an argument into the methods or constructors. It can also be returned from the method as the current class instance.

56. What are the main uses of this keyword?

There are the following uses of this keyword.

- this can be used to refer to the current class instance variable.
- this can be used to invoke current class method (implicitly)
- this() can be used to invoke the current class constructor.
- this can be passed as an argument in the method call.
- this can be passed as an argument in the constructor call.
- this can be used to return the current class instance from the method.

57. How to sort an array in Java?

"import java. util. Arrays;

Arrays. sort(array);"

58. What is a method in Java?

A method is a block of code that only runs when it is called. You can pass data, known as parameters, into a method. Methods are used to perform certain actions, and they are also known as functions.

59. What is a string in Java?

String is a sequence of characters, for e.g. “Hello” is a string of 5 characters. In java, string is an immutable object which means it is constant and cannot be changed once it has been created.

60. What is the Inheritance?

Inheritance is a mechanism by which one object acquires all the properties and behavior of another object of another class. It is used for Code Reusability and Method Overriding. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also. Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

There are five types of inheritance in Java.

- Single-level inheritance
- Multi-level inheritance
- Multiple Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

Multiple inheritance is not supported in Java through class.

61. Why is Inheritance used in Java?

There are various advantages of using inheritance in Java that is given below.

Inheritance provides code reusability. The derived class does not need to redefine the method of base class unless it needs to provide the specific implementation of the method.

Runtime polymorphism cannot be achieved without using inheritance.

We can simulate the inheritance of classes with the real-time objects which makes OOPs more realistic.

Inheritance provides data hiding. The base class can hide some data from the derived class by making it private.

Method overriding cannot be achieved without inheritance. By method overriding, we can give a specific implementation of some basic method contained by the base class.

62. Which class is the superclass for all the classes?

The object class is the superclass of all other classes in Java.

63. Why is multiple inheritance not supported in java?

To reduce the complexity and simplify the language, multiple inheritance is not supported in java. Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since the compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have the same method or different, there will be a compile time error.

```
class A{

void msg(){System.out.println("Hello");}

}

class B{

void msg(){System.out.println("Welcome");}

}

class C extends A,B{//suppose if it were


 Public Static void main(String args[]){

   C obj=new C();

   obj.msg();//Now which msg() method would be invoked?

}

}
```

Output:

Compile Time Error

64. Why does Java not support pointers?

The pointer is a variable that refers to the memory address. They are not used in Java because they are unsafe(unsecured) and complex to understand.

65. What is super in java?

The super keyword in Java is a reference variable that is used to refer to the immediate parent class object. Whenever you create the instance of the subclass, an instance of the parent class is created implicitly which is referred by super reference variable. The super() is called in the class constructor implicitly by the compiler if there is no super or this.

66. What are the main uses of the super keyword?

There are the following uses of super keyword.

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.

67. What are the differences between this and super keyword?

There are the following differences between this and super keyword.

- The super keyword always points to the parent class contexts whereas this keyword always points to the current class context.
- The super keyword is primarily used for initializing the base class variables within the derived class constructor whereas this keyword primarily used to differentiate between local and instance variables when passed in the class constructor.
- The super and this must be the first statement inside constructor otherwise the compiler will throw an error.

68. What is method overloading?

Method overloading is the polymorphism technique which allows us to create multiple methods with the same name but different signature. We can achieve method overloading in two ways.

By Changing the number of arguments

By Changing the data type of arguments

Method overloading increases the readability of the program. Method overloading is performed to figure out the program quickly.

69. Why is method overloading not possible by changing the return type in java?

In Java, method overloading is not possible by changing the return type of the program due to avoid the ambiguity.

```
class Adder{
static int add(int a,int b){return a+b;}
static double add(int a,int b){return a+b;}
}
class TestOverloading3{
public static void main(String[] args){
System.out.println(Adder.add(11,11));//ambiguity
}}
```


Output:

Compile Time Error: method add(int, int) is already defined in class Adder

70. Can we overload the main() method?

Yes, we can have any number of main methods in a Java program by using method overloading.

71. What is method overriding:

If a subclass provides a specific implementation of a method that is already provided by its parent class, it is known as Method Overriding. It is used for runtime polymorphism and to implement the interface methods.

Rules for Method overriding

- The method must have the same name as in the parent class.
- The method must have the same signature as in the parent class.

- Two classes must have an IS-A relationship between them.

72. Can we override the static method?

No, you can't override the static method because they are the part of the class, not the object.

73. Why can we not override static method?

It is because the static method is the part of the class, and it is bound with class whereas instance method is bound with the object, and static gets memory in class area, and instance gets memory in a heap.

74. Difference between method Overloading and Overriding.

Method Overloading	Method Overriding
1) Method overloading increases the readability of the program.	Method overriding provides the specific implementation of the method that is already provided by its superclass.
2) Method overloading occurs within the class.	Method overriding occurs in two classes that have IS-A relationship between them.
3) In this case, the parameters must be different.	In this case, the parameters must be the same.

75. Can we override the private methods?

No, we cannot override the private methods because the scope of private methods is limited to the class and we cannot access them outside of the class.

76. What is the final variable?

In Java, the final variable is used to restrict the user from updating it. If we initialize the final variable, we can't change its value. In other words, we can say that the final variable once assigned to a value, can never be changed after that. The final variable which is not assigned to any value can only be assigned through the class constructor.

77. Can we declare a constructor as final?

The constructor can never be declared as final because it is never inherited. Constructors are not ordinary methods; therefore, there is no sense to declare constructors as final. However, if you try to do so, The compiler will throw an error.

78. Can we declare an interface as final?

No, we cannot declare an interface as final because the interface must be implemented by some class to provide its definition. Therefore, there is no sense to make an interface final. However, if you try to do so, the compiler will show an error.

79. What is the difference between the final method and abstract method?

The main difference between the final method and abstract method is that the abstract method cannot be final as we need to override them in the subclass to give its definition.

80.What is the difference between compile-time polymorphism and runtime polymorphism?

SN	compile-time polymorphism	Runtime polymorphism
1	In compile-time polymorphism, call to a method is resolved at compile-time.	In runtime polymorphism, call to an overridden method is resolved at runtime.
2	It is also known as static binding, early binding, or overloading.	It is also known as dynamic binding, late binding, overriding, or dynamic method dispatch.
3	Overloading is a way to achieve compile-time polymorphism in which, we can define multiple methods or constructors with different signatures.	Overriding is a way to achieve runtime polymorphism in which, we can redefine some particular method or variable in the derived class. By using overriding, we can give some specific implementation to the base class properties in the derived class.
4	It provides fast execution because the type of an object is determined at compile-time.	It provides slower execution as compare to compile-time because the type of an object is determined at run-time.
5	Compile-time polymorphism provides less flexibility because all the things are resolved at compile-time.	Run-time polymorphism provides more flexibility because all the things are resolved at runtime.

81. What is Runtime Polymorphism?

Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

```
class Bike{
  void run(){System.out.println("running");}
}
class Splendor extends Bike{
  void run(){System.out.println("running safely with 60km");}
  public static void main(String args[]){
    Bike b = new Splendor();//upcasting
    b.run();
  }
}
```

Output:

running safely with 60km.

82. What is the difference between static binding and dynamic binding?

In case of the static binding, the type of the object is determined at compile-time whereas, in the dynamic binding, the type of the object is determined at runtime.

Static Binding

```
class Dog{
 private void eat(){System.out.println("dog is eating...");}

 public static void main(String args[]){
  Dog d1=new Dog();
  d1.eat();
```

```
 }
}
```

Dynamic Binding

```
class Animal{
 void eat(){System.out.println("animal is eating...");}
}


class Dog extends Animal{
 void eat(){System.out.println("dog is eating...");}


 public static void main(String args[]){
  Animal a=new Dog();
  a.eat();
 }
}
```

83. What is the abstraction?

Abstraction is a process of hiding the implementation details and showing only functionality to the user. It displays just the essential things to the user and hides the internal information, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery. Abstraction enables you to focus on what the object does instead of how it does it. Abstraction lets you focus on what the object does instead of how it does it.

In Java, there are two ways to achieve the abstraction.

Abstract Class

Interface

84. What is the difference between abstraction and encapsulation?

Abstraction hides the implementation details whereas encapsulation wraps code and data into a single unit.

85. What is the abstract class?

A class that is declared as abstract is known as an abstract class. It needs to be extended and its method implemented. It cannot be instantiated. It can have abstract methods, non-abstract methods, constructors, and static methods. It can also have the final methods which will force the subclass not to change the body of the method. Consider the following example.

```
abstract class Bike{

  abstract void run();

}

class Honda4 extends Bike{

void run(){System.out.println("running safely");}

public static void main(String args[]){

 Bike obj = new Honda4();

 obj.run();

}

}
```


Output

```
running safely
```

86. What is the interface?

The interface is a blueprint for a class that has static constants and abstract methods. It can be used to achieve full abstraction and multiple inheritance. It is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. Java Interface also represents the IS-A relationship. It cannot be instantiated just like the abstract class. However, we need to implement it to define its methods. Since Java 8, we can have the default, static, and private methods in an interface.

87. What are the differences between abstract class and interface?

Abstract class	Interface
An abstract class can have a method body (non-abstract methods).	The interface has only abstract methods.
An abstract class can have instance variables.	An interface cannot have instance variables.
An abstract class can have the constructor.	The interface cannot have the constructor.
An abstract class can have static methods.	The interface cannot have static methods.
You can extend one abstract class.	You can implement multiple interfaces.
The abstract class **can provide the implementation of the interface**.	The Interface **can't provide the implementation of the abstract class**.
The **abstract keyword** is used to declare an abstract class.	The **interface keyword** is used to declare an interface.

An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
An **abstract class** can be extended using keyword **extends**	An **interface class** can be implemented using keyword **implements**
A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.

Example:
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

88. What are the advantages of Encapsulation in Java?

There are the following advantages of Encapsulation in Java?

- By providing only the setter or getter method, you can make the class read-only or write-only. In other words, you can skip the getter or setter methods.

- It provides you the control over the data. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods.
- It is a way to achieve data hiding in Java because other class will not be able to access the data through the private data members.

89. What is the package?

A package is a group of similar type of classes, interfaces, and sub-packages. It provides access protection and removes naming collision. The packages in Java can be categorized into two forms, inbuilt package, and user-defined package. There are many built-in packages such as Java, lang, awt, javax, swing, net, io, util, sql, etc. Consider the following example to create a package in Java.

```
//save as Simple.java

package mypack;

public class Simple{

 public static void main(String args[]){

   System.out.println("Welcome to package");

  }

}
```

90. How can we access some class in another class in Java?

There are two ways to access a class in another class.

By using the fully qualified name: To access a class in a different package, either we must use the fully qualified name of that class, or we must import the package containing that class.

By using the relative path, We can use the path of the class that is related to the package that contains our class. It can be the same or subpackage.

91. How many types of exception can occur in a Java program?

There are mainly two types of exceptions: checked and unchecked. Here, an error is considered as the unchecked exception. According to Oracle, there are three types of exceptions:

Checked Exception: Checked exceptions are the one which are checked at compile-time. For example, SQLException, ClassNotFoundException, etc.

Unchecked Exception: Unchecked exceptions are the one which are handled at runtime because they can not be checked at compile-time. For example, ArithmaticException, NullPointerException, ArrayIndexOutOfBoundsException, etc.

Error: Error cause the program to exit since they are not recoverable. For Example, OutOfMemoryError, AssertionError, etc.

92. What is Exception Handling?

Exception Handling is a mechanism that is used to handle runtime errors. It is used primarily to handle checked exceptions. Exception handling maintains the normal flow of the program. There are mainly two types of exceptions: checked and unchecked. Here, the error is considered as the unchecked exception.

93. What is the difference between Checked Exception and Unchecked Exception?

1) Checked Exception

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions, e.g., IOException, SQLException, etc. Checked exceptions are checked at compile-time.

2) Unchecked Exception

The classes that extend RuntimeException are known as unchecked exceptions, e.g., ArithmeticException, NullPointerException, etc. Unchecked exceptions are not checked at compile-time.

94. What is the difference between throw and throws?

throw keyword	throws keyword
1) The **throw** keyword is used to throw an exception explicitly.	The **throws** keyword is used to declare an exception.
2) The checked exceptions cannot be propagated with throw only.	The checked exception can be propagated with throws
3) The **throw** keyword is followed by an instance.	The **throws** keyword is followed by class.
4) The **throw** keyword is used within the method.	The **throws** keyword is used with the method signature.
5) You cannot throw multiple exceptions.	You can declare multiple exceptions, e.g., public void method()throws IOException, SQLException.

95. What is the meaning of immutable regarding String?

The simple meaning of immutable is unmodifiable or unchangeable. In Java, String is immutable, i.e., once string object has been created, its value can't be changed. Consider the following example for better understanding.

```
class Testimmutablestring{
 public static void main(String args[]){
   String s="Sachin";
   s.concat(" Tendulkar");//concat() method appends the string at the end
   System.out.println(s);//will print Sachin because strings are immutable objects
 }
}
```


Output:

Sachin

96. What are the differences between String and StringBuffer?

No.	String	StringBuffer
1)	The String class is immutable.	The StringBuffer class is mutable.
2)	The String is slow and consumes more memory when you concat too many strings because every time it creates a new instance.	The StringBuffer is fast and consumes less memory when you cancat strings.
3)	The String class overrides the equals() method of Object class. So you can compare the contents of two strings by equals() method.	The StringBuffer class doesn't override the equals() method of Object class.

97. What are the differences between StringBuffer and StringBuilder?

No.	StringBuffer	StringBuilder
1)	StringBuffer is *synchronized*, i.e., thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	StringBuilder is *non-synchronized*,i.e., not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
2)	StringBuffer is *less efficient* than StringBuilder.	StringBuilder is *more efficient* than StringBuffer.

98. Write a Java program to count the number of words present in a string?

Program:

```
 public class Test
{
   public static void main (String args[])
   {
      String s = "Sharma is a good player and he is so punctual";
      String words[] = s.split(" ");
```

```
        System.out.println("The Number of words present in the string are :
"+words.length);
    }
}
```

Output

The Number of words present in the string are : 10

99. What are the advantages of Java inner classes?

There are two types of advantages of Java inner classes.

Nested classes represent a special type of relationship that is it can access all the members (data members and methods) of the outer class including private.

Nested classes are used to develop a more readable and maintainable code because it logically groups classes and interfaces in one place only.

Code Optimization: It requires less code to write.

100. What is Garbage Collection?

Garbage collection is a process of reclaiming the unused runtime objects. It is performed for memory management. In other words, we can say that It is the process of removing unused objects from the memory to free up space and make this space available for Java Virtual Machine. Due to garbage collection java gives 0 as output to a variable whose value is not set, i.e., the variable has been defined but not initialized. For this purpose, we were using free() function in the C language and delete() in C++. In Java, it is performed automatically. So, java provides better memory management.

101. What is the difference between final, finally and finalize?

No.	final	finally	finalize
1)	Final is used to apply restrictions on class, method, and variable. The final class can't be inherited, final method can't be overridden, and final variable value can't be changed.	Finally is used to place important code, it will be executed whether an exception is handled or not.	Finalize is used to perform clean up processing just before an object is garbage collected.
2)	Final is a keyword.	Finally is a block.	Finalize is a method.

102. What is the purpose of the Runtime class?

Java Runtime class is used to interact with a java runtime environment. Java Runtime class provides methods to execute a process, invoke GC, get total and free memory, etc. There is only one instance of java.lang.Runtime class is available for one java application. The Runtime.getRuntime() method returns the singleton instance of Runtime class.

103. What is multithreading?

Multithreading is a process of executing multiple threads simultaneously. Multithreading is used to obtain the multitasking. It consumes less memory and gives the fast and efficient performance. Its main advantages are:

Threads share the same address space.

The thread is lightweight.

The cost of communication between the processes is low.

104. What is the thread?

A thread is a lightweight subprocess. It is a separate path of execution because each thread runs in a different stack frame. A process may contain multiple threads. Threads share the process resources, but still, they execute independently.

105. Differentiate between process and thread?

There are the following differences between the process and thread.

- A Program in the execution is called the process whereas; A thread is a subset of the process
- Processes are independent whereas threads are the subset of process.

- Process have different address space in memory, while threads contain a shared address space.
- Context switching is faster between the threads as compared to processes.
- Inter-process communication is slower and expensive than inter-thread communication.
- Any change in Parent process doesn't affect the child process whereas changes in parent thread can affect the child thread.

106. What are the advantages of multithreading?

Multithreading programming has the following advantages:

- Multithreading allows an application/program to be always reactive for input, even already running with some background tasks
- Multithreading allows the faster execution of tasks, as threads execute independently.
- Multithreading provides better utilization of cache memory as threads share the common memory resources.

107. What are the states in the lifecycle of a Thread?

A thread can have one of the following states during its lifetime:

New: In this state, a Thread class object is created using a new operator, but the thread is not alive. Thread doesn't start until we call the start() method.

Runnable: In this state, the thread is ready to run after calling the start() method. However, the thread is not yet selected by the thread scheduler.

Running: In this state, the thread scheduler picks the thread from the ready state, and the thread is running.

Waiting/Blocked: In this state, a thread is not running but still alive, or it is waiting for the other thread to finish.

Dead/Terminated: A thread is in terminated or dead state when the run() method exits.

108. What does join () method?

The join() method waits for a thread to die. In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task. Join method is overloaded in Thread class in the following ways.

public void join()throws InterruptedException

public void join(long milliseconds)throws InterruptedException

The End.