Contents

[1.What are the principle concepts of OOPS? 3](#_Toc180166654)

[2.What is Abstraction? 3](#_Toc180166655)

[3.What is Encapsulation? 3](#_Toc180166656)

[4.What is the difference between abstraction and encapsulation? 3](#_Toc180166657)

[5.What is Inheritance? 3](#_Toc180166658)

[6.What is Polymorphism? 4](#_Toc180166659)

[7.How does Java implement polymorphism? 4](#_Toc180166660)

[8.Explain the different forms of Polymorphism. 4](#_Toc180166661)

[9.What is runtime polymorphism or dynamic method dispatch? 5](#_Toc180166662)

[10.What is Dynamic Binding? 5](#_Toc180166663)

[11.What is method overloading? 5](#_Toc180166664)

[12.What is method overriding? 6](#_Toc180166665)

[13.What are the differences between method overloading and method overriding? 6](#_Toc180166666)

[14.Can overloaded methods be override too? 7](#_Toc180166667)

[15.Is it possible to override the main method? 7](#_Toc180166668)

[16.How to invoke a superclass version of an Overridden method? 7](#_Toc180166669)

[17.What is super? 7](#_Toc180166670)

[18.How do you prevent a method from being overridden? 8](#_Toc180166671)

[19.What is an Interface? 8](#_Toc180166672)

[21.Can we create an object for an interface? 8](#_Toc180166673)

[22.Do interfaces have member variables? 9](#_Toc180166674)

[23.What modifiers are allowed for methods in an Interface? 9](#_Toc180166675)

[24.What is a marker interface? 9](#_Toc180166676)

[25.What is an abstract class? 9](#_Toc180166677)

[27.What are the differences between Interface and Abstract class? 10](#_Toc180166678)

[28.When should I use abstract classes and when should I use interfaces? 11](#_Toc180166679)

[29.When you declare a method as abstract, can other nonabstract methods access it? 11](#_Toc180166680)

[30.Can there be an abstract class with no abstract methods in it? 11](#_Toc180166681)

[32.How does the Java default constructor be provided? 12](#_Toc180166682)

[33.Can constructor be inherited? 12](#_Toc180166683)

[34.What are the differences between Contructors and Methods? 12](#_Toc180166684)

[35.How are this() and super() used with constructors? 13](#_Toc180166685)

[36.What are the differences between Class Methods and Instance Methods? 13](#_Toc180166686)

[37.How are this() and super() used with constructors? 14](#_Toc180166687)

[38.What are Access Specifiers? 14](#_Toc180166688)

[39.What are Access Specifiers available in Java? 14](#_Toc180166689)

[40.What is final modifier? 15](#_Toc180166690)

[42.What is static block? 16](#_Toc180166691)

[43.What are static variables? 16](#_Toc180166692)

[44.What is the difference between static and non-static variables? 16](#_Toc180166693)

# 1.What are the principle concepts of OOPS?

There are four principle concepts upon which object oriented design and programming rest. They are:

* Abstraction
* Polymorphism
* Inheritance
* Encapsulation (i.e. easily remembered as A-PIE).

## 2.What is Abstraction?

Abstraction refers to the act of representing essential features without including the background details or explanations.

## 3.What is Encapsulation?

Encapsulation is a technique used for hiding the properties and behaviors of an object and allowing outside access only as appropriate. It prevents other objects from directly altering or accessing the properties or methods of the encapsulated object.

## 4.What is the difference between abstraction and encapsulation?

* **Abstraction**
* **Encapsulation**

**is the deliverables of Abstraction. Encapsulation barely talks about grouping up your abstraction to suit the developer needs. solves the problem in the design side while Encapsulation is the Implementation.**

## 5.What is Inheritance?

* + Inheritance is the process by which objects of one class acquire the properties of objects of another class.
  + A class is inherited is called a superclass.
  + The class does the inheriting is called a subclass.
  + Inheritance is done by using the keyword extends.
  + The two most common reasons to use inheritance are:

* + - To promote code reuse
    - To use polymorphism

## 6.What is Polymorphism?

Polymorphism is briefly described as "one interface, many implementations." Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts - specifically, to allow an entity such as a variable, a function, or an object to have more than one form.

## 7.How does Java implement polymorphism?

(Inheritance, Overloading and Overriding are used to achieve Polymorphism in java).

Polymorphism manifests itself in Java in the form of multiple methods having the same name.

* + - In some cases, multiple methods have the same name, but different formal argument lists (overloaded methods).
    - In other cases, multiple methods have the same name, same return type, and same formal argument list (overridden methods).

## 8.Explain the different forms of Polymorphism.

There are two types of polymorphism one is Compile time polymorphism and the other is run time polymorphism. Compile time polymorphism is method overloading. Runtime time polymorphism is done using inheritance and interface.  
Note: *From a practical programming viewpoint, polymorphism manifests itself in three distinct forms in Java:*

* + - Method overloading
    - Method overriding through inheritance
    - Method overriding through the Java interface

## 9.What is runtime polymorphism or dynamic method dispatch?

In Java, 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 iscalled 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.

## 10.What is Dynamic Binding?

## Dynamic binding links a procedure call to its corresponding code at runtime, enabling polymorphism and inheritance in programming. 11.What is method overloading?

Method Overloading means to have two or more methods with same name in the same class with different arguments. The benefit of method overloading is that it allows you to implement methods that support the same operation but differ by argument number or type.  
Note:

* + - Overloaded methods MUST change the argument list
    - Overloaded methods CAN change the return type
    - Overloaded methods CAN change the access modifier
    - Overloaded methods CAN declare new or broader checked exceptions
    - A method can be overloaded in the same class or in a subclass

## 12.What is method overriding?

Method overriding occurs when sub class declares a method that has the same type arguments as a method declared by one of its superclass. The key benefit of overriding is the ability to define behavior that’s specific to a particular subclass type.  
Note:

* + - The overriding method cannot have a more restrictive access modifier than the method being overridden (Ex: You can’t override a method marked public and make it protected).
    - You cannot override a method marked ‘final’
    - You cannot override a method marked ‘static’

## 13.What are the differences between method overloading and method overriding?

|  |  |  |
| --- | --- | --- |
|  | **Overloaded Method** | **Overridden Method** |
| **Arguments** | Must change | Must not change |
| **Return type** | Can change | Can’t change except for covariant returns |
| **Exceptions** | Can change | Can reduce or eliminate. Must not throw new or broader checked exceptions |
| **Access** | Can change | Must not make more restrictive (can be less restrictive) |
| **Invocation** | Reference type determines which overloaded version is selected. Happens at compile time. | Object type determines which method is selected. Happens at runtime. |

## 14.Can overloaded methods be override too?

Yes, derived classes still can override the overloaded methods. Polymorphism can still happen. Compiler will not binding the method calls since it is overloaded, because it might be overridden now or in the future.

## 15.Is it possible to override the main method?

NO, because main is a static method. A static method can't be overridden in Java.

## 16.How to invoke a superclass version of an Overridden method?

To invoke a superclass method that has been overridden in a subclass, you must either call the method directly through a superclass instance, or use the super prefix in the subclass itself. From the point of the view of the subclass, the super prefix provides an explicit reference to the superclass' implementation of the method.

**// From subclass**

**super.overriddenMethod();**

## 17.What is super?

**super**

* + - ***You can only go back one level.***
    - ***In the constructor, if you use super(), it must be the very first code, and you cannot access any***

**is a keyword which is used to access the method or member variables from the superclass. If a method hides one of the member variables in its superclass, the method can refer to the hidden variable through the use of the super keyword. In the same way, if a method overrides one of the methods in its superclass, the method can invoke the overridden method through the use of the super keyword.   
Note: this.xxx variables or methods to compute its parameters.**

## 18.How do you prevent a method from being overridden?

**To prevent a specific method from being overridden in a subclass, use the final modifier on the method declaration, which means "this is the final implementation of this method", the end of its inheritance hierarchy.**

**public final void exampleMethod() {  
　 　　　　　　　　　　　　　　　　　　　　　　// Method statements  
　　　　　　　　　　　　　　　　　　　　　　　　}**

## 19.What is an Interface?

**An interface is a description of a set of methods that conforming implementing classes must have.  
Note:**

* + - ***You can’t mark an interface as final.***
    - ***Interface variables must be static.***
    - ***An Interface cannot extend anything but another interfaces.***

**20.**

**You can’t instantiate an interface directly, but you can instantiate a class that implements an interface.**

## 21.Can we create an object for an interface?

**Yes, it is always necessary to create an object implementation for an interface. Interfaces cannot be instantiated in their own right, so you must write a class that implements the interface and fulfill all the methods defined in it.**

## 22.Do interfaces have member variables?

**Interfaces may have member variables, but these are implicitly**

**public, static, and final- in other words, interfaces can declare only constants, not instance variables that are available to all implementations and may be used as key references for method arguments for example.**

## 23.What modifiers are allowed for methods in an Interface?

**Only**

**public and abstract modifiers are allowed for methods in interfaces.**

## 24.What is a marker interface?

**Marker interfaces are those which do not declare any required methods, but signify their compatibility with certain operations. The**

**java.io.Serializable interface and Cloneable are typical marker interfaces. These do not contain any methods, but classes must implement this interface in order to be serialized and de-serialized.**

## 25.What is an abstract class?

**Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation.   
Note:**

* + - ***If even a single method is abstract, the whole class must be declared abstract.***
    - ***Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods.***
    - ***You can’t mark a class as both abstract and final.***

**26**.Can we instantiate an abstract class?

**An abstract class can never be instantiated. Its sole purpose is to be extended (subclassed).**

## 27.What are the differences between Interface and Abstract class?

|  |  |
| --- | --- |
| **Abstract Class** | **Interfaces** |
| An abstract class can provide complete, default code and/or just the details that have to be overridden. | An interface cannot provide any code at all,just the signature. |
| In case of abstract class, a class may extend only one abstract class. | A Class may implement several interfaces. |
| An abstract class can have non-abstract methods. | All methods of an Interface are abstract. |
| An abstract class can have instance variables. | An Interface cannot have instance variables. |
| An abstract class can have any visibility: public, private, protected. | An Interface visibility must be public (or) none. |
| If we add a new method to an abstract class then we have the option of providing default implementation and therefore all the existing code might work properly. | If we add a new method to an Interface then we have to track down all the implementations of the interface and define implementation for the new method. |
| An abstract class can contain constructors . | An Interface cannot contain constructors . |
| Abstract classes are fast. | Interfaces are slow as it requires extra indirection to find corresponding method in the actual class. |

## 28.When should I use abstract classes and when should I use interfaces?

**Use Interfaces when…**

**Use Abstract Class when…**

* + **If various implementations are of the same kind and use common behavior or status then abstract class is better to use.**
  + **When you want to provide a generalized form of abstraction and leave the implementation task with the inheriting subclass.**
  + **Abstract classes are an excellent way to create planned inheritance hierarchies. They're also a good choice for nonleap classes in class hierarchies.**

## 29.When you declare a method as abstract, can other nonabstract methods access it?

**Yes, other nonabstract methods can access a method that you declare as abstract.**

## 30.Can there be an abstract class with no abstract methods in it?

**Yes, there can be an abstract class without abstract methods.**

**31.**

* + - **A constructor is a special method whose task is to initialize the object of its class.**
    - **It is special because its name is the same as the class name.**
    - **They do not have return types, not even void and therefore they cannot return values.**
    - **They cannot be inherited, though a derived class can call the base class constructor.**
    - **Constructor is invoked whenever an object of its associated class is created.**

## 32.How does the Java default constructor be provided?

**If a class defined by the code does not have any constructor, compiler will automatically provide one no-parameter-constructor (default-constructor) for the class in the byte code. The access modifier (public/private/etc.) of the default constructor is the same as the class itself.**

## 33.Can constructor be inherited?

**No, constructor cannot be inherited, though a derived class can call the base class constructor.**

## 34.What are the differences between Constructors and Methods?

|  |  |  |
| --- | --- | --- |
|  | **Constructors** | **Methods** |
| **Purpose** | Create an instance of a class | Group Java statements |
| **Modifiers** | Cannot be *abstract, final, native, static*, or*synchronized* | Can be *abstract, final, native, static*, or*synchronized* |
| **Return Type** | No return type, not even void | void or a valid return type |
| **Name** | Same name as the class (first letter is capitalized by convention) -- usually a noun | Any name except the class. Method names begin with a lowercase letter by convention -- usually the name of an action |
| ***this*** | Refers to another constructor in the same class. If used, it must be the first line of the constructor | Refers to an instance of the owning class. Cannot be used by static methods. |
| ***super*** | Calls the constructor of the parent class. If used, must be the first line of the constructor | Calls an overridden method in the parent class |
| **Inheritance** | Constructors are not inherited | Methods are inherited |

## 35.How are this() and super() used with constructors?

* + - * **Constructors use *this* to refer to another constructor in the same class with a different parameter list.**
      * **Constructors use *super* to invoke the superclass's constructor. If a constructor uses *super*, it must use it in the first line; otherwise, the compiler will complain.**

## 36.What are the differences between Class Methods and Instance Methods?

|  |  |
| --- | --- |
| **Class Methods** | **Instance Methods** |
| Class methods are methods which are declared as static. The method can be called without creating an instance of the class | Instance methods on the other hand require an instance of the class to exist before they can be called, so an instance of a class needs to be created by using the new keyword. Instance methods operate on specific instances of classes. |
| Class methods can only operate on class members and not on instance members as class methods are unaware of instance members. | Instance methods of the class can also not be called from within a class method unless they are being called on an instance of that class. |
| Class methods are methods which are declared as static. The method can be called without creating an　 instance of the class. | Instance methods are not declared as static. |

## 37.How are this() and super() used with constructors?

* + - * + **Constructors use *this* to refer to another constructor in the same class with a different parameter list.**
        + **Constructors use *super* to invoke the superclass's constructor. If a constructor uses super, it must use it in the first line; otherwise, the compiler will complain.**

## 38.What are Access Specifiers?

**One of the techniques in object-oriented programming is *encapsulation*. It concerns the hiding of data in a class and making this class available only through methods. Java allows you to control access to classes, methods, and fields via so-called *access specifiers*..**

## 39.What are Access Specifiers available in Java?

**Java offers four access specifiers, listed below in decreasing accessibility:**

**Public**

**Protected**

**Default(no specifier)-**

**Private**

**- *private* methods and fields can only be accessed within the same class to which the methods and fields belong. *private* methods and fields are not visible within subclasses and are not inherited by subclasses. If you do not set access to specific level, then such a class, method, or field will be accessible from inside the same package to which the class, method, or field belongs, but not from outside this package. - *protected* methods and fields can only be accessed within the same class to which the methods and fields belong, within its subclasses, and within classes of the same package. - *public* classes, methods, and fields can be accessed from everywhere.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Situation** | **public** | **protected** | **default** | **private** |
| Accessible to class  　from same package? | yes | yes | yes | no |
| Accessible to class  　from different package? | yes | no, *unless it is a subclass* | no | no |

## 40.What is final modifier?

**The　final　modifier keyword makes that the programmer cannot change the value anymore. The actual meaning depends on whether it is applied to a class, a variable, or a method.**

***final***

***final***

***final***

***Methods- A final method cannot be overridden by subclasses. Variables- A final variable cannot be changed once it is initialized.*Classes- A final class cannot have subclasses.**

**There are two reasons for marking a method as final:**

**Disallowing subclasses to change the meaning of the method.**

**Increasing efficiency by allowing the compiler to turn calls to the method into inline Java code.**

## 42.What is static block?

**Static block which exactly executed exactly once when the class is first loaded into JVM. Before going to the main method the static block will execute.**

## 43.What are static variables?

**Variables that have only one copy per class are known as static variables. They are not attached to a particular instance of a class but rather belong to a class as a whole. They are declared by using the static keyword as a modifier.**

**static type varIdentifier;**

**where, the name of the variable is varIdentifier and its data type is specified by type.  
Note: Static variables that are not explicitly initialized in the code are automatically initialized with a default value. The default value depends on the data type of the variables.**

## 44.What is the difference between static and non-static variables?

**A static variable is associated with the class as a whole rather than with specific instances of a class. Non-static variables take on unique values with each object instance.**

**45.**

**Methods declared with the keyword static as modifier are called static methods or class methods. They are so called because they affect a class as a whole, not a particular instance of the class. Static methods are always invoked without reference to a particular instance of a class.  
Note:The use of a static method suffers from the following restrictions:**

***A static method can only call other static methods.***

***A static method must only access static data.***

***A static method cannot reference to the current object using keywords super or this.***

**1. What is the most important feature of Java?**

**Java is a platform independent language.**

**2. What do you mean by platform independence?**

**Platform independence means that we can write and compile the java code in one platform (eg Windows) and can execute the class in any other supported platform eg (Linux,Solaris,etc).**

**3. What is a JVM?**

**JVM is Java Virtual Machine which is a run time environment for the compiled java class files.**

**4. Are JVM's platform independent?**

**JVM's are not platform independent. JVM's are platform specific run time implementation provided by the vendor.**

**5. What is the difference between a JDK and a JVM?**

**JDK is Java Development Kit which is for development purpose and it includes execution environment also. But JVM is purely a run time environment and hence you will not be able to compile your source files using a JVM.**

**6. What is a pointer and does Java support pointers?**

**Pointer is a reference handle to a memory location. Improper handling of pointers leads to memory leaks and reliability issues hence Java doesn't support the usage of pointers.**

**7. What is the base class of all classes?**

**java.lang.Object**

**8. Does Java support multiple inheritance?**

**Java doesn't support multiple inheritance.**

**9. Is Java a pure object oriented language?**

**Java uses primitive data types and hence is not a pure object oriented language.**

**10. Are arrays primitive data types?**

**In Java, Arrays are objects.**

**11. What is difference between Path and Classpath?**

**Path and Classpath are operating system level environment variales. Path is used define where the system can find the executables(.exe) files and classpath is used to specify the location .class files.**

**12. What are local variables?**

**Local varaiables are those which are declared within a block of code like methods. Local variables should be initialised before accessing them.**

**13. What are instance variables?**

**Instance variables are those which are defined at the class level. Instance variables need not be initialized before using them as they are automatically initialized to their default values.**

**14. How to define a constant variable in Java?**

**The variable should be declared as**

**static and final. So only one copy of the variable exists for all instances of the class and the value can't be changed also.static final int PI = 2.14; is an example for constant.**

**15. Should a main() method be compulsorily declared in all java classes?**

**No not required.**

**main() method should be defined only if the source class is a java application.**

**16. What is the return type of the main() method?**

**Main()**

**method doesn't return anything hence declared void.**

**17. Why is the main() method declared static?**

**main()**

**method is called by the JVM even before the instantiation of the class hence it is declared as static.**

**18. What is the arguement of main() method?**

**main()**

**method accepts an array of String object as arguement.**

**19. Can a main() method be overloaded?**

**Yes. You can have any number of**

**main() methods with different method signature and implementation in the class.**

**20. Can a main() method be declared final?**

**Yes. Any inheriting class will not be able to have it's own default**

**main() method.**

**21. Does the order of public and static declaration matter in main() method?**

**No. It doesn't matter but**

**void should always come before main().**

**22. Can a source file contain more than one class declaration?**

**Yes a single source file can contain any number of Class declarations but only one of the class can be declared as**

**public.**

**23. What is a package?**

**Package is a collection of related classes and interfaces. package declaration should be first statement in a java class.**

**24. Which package is imported by default?**

**java.lang package**

**is imported by default even without a package declaration.**

**25. Can a class declared as private be accessed outside it's package?**

**Not possible.**

**26. Can a class be declared as protected?**

**A class can't be declared as**

**protected. only methods can be declared as protected.**

**27. What is the access scope of a protected method?**

**A**

**protected method can be accessed by the classes within the same package or by the subclasses of the class in any package.**

**28. What is the purpose of declaring a variable as final?**

**A**

**final variable's value can't be changed. final variables should be initialized before using them.**

**29. What is the impact of declaring a method as final?**

**A method declared as**

**final can't be overridden. A sub-class can't have the same method signature with a different implementation.**

**30. I don't want my class to be inherited by any other class. What should i do?**

**You should declared your class as**

**final. But you can't define your class as final, if it is an abstract class. A class declared as final can't be extended by any other class.**

**31. Can you give few examples of final classes defined in Java API?**

**java.lang.String, java.lang.Math**

**are final classes.**

**32. How is final different from finally and finalize()?**

**final**

**is a modifier which can be applied to a class or a method or a variable. final class can't be inherited, final method can't be overridden and final variable can't be changed.   
  
finally is an exception handling code section which gets executed whether an exception is raised or not by the try block code segment.   
finalize() is a method of Object class which will be executed by the JVM just before garbage collecting object to give a final chance for resource releasing activity.**

**33. Can a class be declared as static?**

**We can not declare top level class as static, but only inner class can be declared static.**

**public class Test**

**{**

**static class InnerClass**

**{**

**public static void InnerMethod()**

**{ System.out.println("Static Inner Class!"); }**

**}**

**public static void main(String args[])**

**{**

**Test.InnerClass.InnerMethod();**

**}**

**}**

**//output: Static Inner Class!**

**34. When will you define a method as static?**

**When a method needs to be accessed even before the creation of the object of the class then we should declare the method as**

**static.**

**35. What are the restriction imposed on a static method or a static block of code?**

**A static method should not refer to instance variables without creating an instance and cannot use "this" operator to refer the instance.**

**36. I want to print "Hello" even before main() is executed. How will you acheive that?**

**Print the statement inside a static block of code. Static blocks get executed when the class gets loaded into the memory and even before the creation of an object. Hence it will be executed before the**

**main() method. And it will be executed only once.**

**37. What is the importance of static variable?**

**static variables are class level variables where all objects of the class refer to the same variable. If one object changes the value then the change gets reflected in all the objects.**

**38. Can we declare a static variable inside a method?**

**Static varaibles are class level variables and they can't be declared inside a method. If declared, the class will not compile.**

**39. What is an Abstract Class and what is it's purpose?**

**A Class which doesn't provide complete implementation is defined as an abstract class. Abstract classes enforce abstraction.**

**40. Can a abstract class be declared final?**

**Not possible. An abstract class without being inherited is of no use and hence will result in compile time error.**

**41. What is use of a abstract variable?**

**Variables can't be declared as abstract. only classes and methods can be declared as**

**abstract.**

**42. Can you create an object of an abstract class?**

**Not possible. Abstract classes can't be instantiated.**

**43. Can a abstract class be defined without any abstract methods?**

**Yes it's possible. This is basically to avoid instance creation of the class.**

**44. Class C implements Interface I containing method m1 and m2 declarations. Class C has provided implementation for method m2. Can i create an object of Class C?**

**No not possible.**

**Class C should provide implementation for all the methods in the Interface I. Since Class C didn't provide implementation for m1method, it has to be declared as abstract. Abstract classes can't be instantiated.**

**45. Can a method inside a Interface be declared as final?**

**No not possible. Doing so will result in compilation error.**

**public and abstract are the only applicable modifiers for method declaration in an interface.**

**46. Can an Interface implement another Interface?**

**Intefaces doesn't provide implementation hence a interface cannot implement another interface.**

**47. Can an Interface extend another Interface?**

**Yes an Interface can inherit another Interface, for that matter an Interface can extend more than one Interface.**

**48. Can a Class extend more than one Class?**

**Not possible. A Class can extend only one class but can implement any number of Interfaces.**

**49. Why is an Interface be able to extend more than one Interface but a Class can't extend more than one Class?**

**Basically Java doesn't allow multiple inheritance, so a Class is restricted to extend only one Class. But an Interface is a pure abstraction model and doesn't have inheritance hierarchy like classes(do remember that the base class of all classes is Object). So an Interface is allowed to extend more than one Interface.**

**50. Can an Interface be final?**

**Not possible. Doing so so will result in compilation error.**

**51. Can a class be defined inside an Interface?**

**Yes it's possible.**

**52. Can an Interface be defined inside a class?**

**Yes it's possible.**

**53. What is a Marker Interface?**

**An Interface which doesn't have any declaration inside but still enforces a mechanism.**

**54. Which object oriented Concept is achieved by using overloading and overriding?**

**Polymorphism.**

**55. Why does Java not support operator overloading?**

**Operator overloading makes the code very difficult to read and maintain. To maintain code simplicity, Java doesn't support operator overloading.**

**56. Can we define private and protected modifiers for variables in interfaces?**

**No.**

**57. What is Externalizable?**

**Externalizable is an Interface that extends Serializable Interface. And sends data into Streams in Compressed Format. It has two methods,**

**writeExternal(ObjectOuput out) and readExternal(ObjectInput in)**

**58. What modifiers are allowed for methods in an Interface?**

**Only**

**public and abstract modifiers are allowed for methods in interfaces.**

**59. What is a local, member and a class variable?**

**Variables declared within a method are "local" variables.**

**Variables declared within the class i.e not within any methods are "member" variables (global variables).**

**Variables declared within the class i.e not within any methods and are defined as "static" are class variables.**

**60. What is an abstract method?**

**An abstract method is a method whose implementation is deferred to a subclass.**

**61. What value does read() return when it has reached the end of a file?**

**The**

**read() method returns -1 when it has reached the end of a file.**

**62. Can a Byte object be cast to a double value?**

**No, an object cannot be cast to a primitive value.**

**63. What is the difference between a static and a non-static inner class?**

**A non-static inner class may have object instances that are associated with instances of the class's outer class. A static inner class does not have any object instances.**

**64. What is an object's lock and which object's have locks?**

**An object's lock is a mechanism that is used by multiple threads to obtain synchronized access to the object. A thread may execute a synchronized method of an object only after it has acquired the object's lock. All objects and classes have locks. A class's lock is acquired on the class's Class object.**

**65. What is the % operator?**

**It is referred to as the modulo or remainder operator. It returns the remainder of dividing the first operand by the second operand.**

**66. When can an object reference be cast to an interface reference?**

**An object reference be cast to an interface reference when the object implements the referenced interface.**

**67. Which class is extended by all other classes?**

**The Object class is extended by all other classes.**

**68. Which non-Unicode letter characters may be used as the first character of an identifier?**

**The non-Unicode letter characters**

**$ and \_ may appear as the first character of an identifier**

**69. What restrictions are placed on method overloading?**

**Two methods may not have the same name and argument list but different return types.**

**70. What is casting?**

**There are two types of casting, casting between primitive numeric types and casting between object references. Casting between numeric types is used to convert larger values, such as double values, to smaller values, such as byte values. Casting between object references is used to refer to an object by a compatible class, interface, or array type reference.**

**71. What is the return type of a program's main() method?**

**void.**

**72. If a variable is declared as private, where may the variable be accessed?**

**A private variable may only be accessed within the class in which it is declared.**

**73. What do you understand by private, protected and public?**

**These are accessibility modifiers.**

**Private is the most restrictive, while public is the least restrictive. There is no real difference between protected and the default type (also known as package protected) within the context of the same package, however the protected keyword allows visibility to a derived class in a different package.**

**74. What is Downcasting ?**

**Downcasting is the casting from a general to a more specific type, i.e. casting down the hierarchy**

**75. What modifiers may be used with an inner class that is a member of an outer class?**

**A (non-local) inner class may be declared as public, protected, private, static, final, or abstract.**

**76. How many bits are used to represent Unicode, ASCII, UTF-16, and UTF-8 characters?**

**Unicode requires 16 bits and ASCII require 7 bits Although the ASCII character set uses only 7 bits, it is usually represented as 8 bits.**

**UTF-8 represents characters using 8, 16, and 18 bit patterns.**

**UTF-16 uses 16-bit and larger bit patterns.**

**77. What restrictions are placed on the location of a package statement within a source code file?**

**A package statement must appear as the first line in a source code file (excluding blank lines and comments).**

**78. What is a native method?**

**A native method is a method that is implemented in a language other than Java.**

**79. What are order of precedence and associativity, and how are they used?**

**Order of precedence determines the order in which operators are evaluated in expressions. Associatity determines whether an expression is evaluated left-to-right or right-to-left.**

**80. Can an anonymous class be declared as implementing an interface and extending a class?**

**An anonymous class may implement an interface or extend a superclass, but may not be declared to do both.**

**81. What is the range of the char type?**

**The range of the**

**char type is 0 to 216 - 1 (i.e. 0 to 65535.)**

**82. What is the range of the short type?**

**The range of the**

**short type is -(215) to 215 - 1. (i.e. -32,768 to 32,767)**

**83. Why isn't there operator overloading?**

**Because C++ has proven by example that operator overloading makes code almost impossible to maintain.**

**84. What does it mean that a method or field is "static"?**

**Static variables and methods are instantiated only once per class. In other words they are class variables, not instance variables. If you change the value of a static variable in a particular object, the value of that variable changes for all instances of that class. Static methods can be referenced with the name of the class rather than the name of a particular object of the class (though that works too). That's how library methods like**

**System.out.println() work. out is a static field in the java.lang.System class.**

**85. Is null a keyword?**

**The null value is not a keyword.**

**86. Which characters may be used as the second character of an identifier, but not as the first character of an identifier?**

**The digits 0 through 9 may not be used as the first character of an identifier but they may be used after the first character of an identifier.**

**87. Is the ternary operator written x : y ? z or x ? y : z ?**

**It is written**

**x ? y : z.**

**88. How is rounding performed under integer division?**

**The fractional part of the result is truncated. This is known as rounding toward zero.**

**89. If a class is declared without any access modifiers, where may the class be accessed?**

**A class that is declared without any access modifiers is said to have package access. This means that the class can only be accessed by other classes and interfaces that are defined within the same package.**

**90. Does a class inherit the constructors of its superclass?**

**A class does not inherit constructors from any of its superclasses.**

**91. Name the eight primitive Java types.**

**The eight primitive types are byte, char, short, int, long, float, double, and boolean.**

**92. What restrictions are placed on the values of each case of a switch statement?**

**During compilation, the values of each case of a**

**switch statement must evaluate to a value that can be promoted to an int value.**

**93. What is the difference between a while statement and a do while statement?**

**A**

**while statement checks at the beginning of a loop to see whether the next loop iteration should occur. A do while statement checks at the end of a loop to see whether the next iteration of a loop should occur. The do whilestatement will always execute the body of a loop at least once.**

**94. What modifiers can be used with a local inner class?**

**A local inner class may be**

**final or abstract.**

**95. When does the compiler supply a default constructor for a class?**

**The compiler supplies a default constructor for a class if no other constructors are provided.**

**96. If a method is declared as protected, where may the method be accessed?**

**A protected method may only be accessed by classes or interfaces of the same package or by subclasses of the class in which it is declared.**

**97. What are the legal operands of the instanceof operator?**

**The left operand is an object reference or null value and the right operand is a class, interface, or array type.**

**98. Are true and false keywords?**

**The values true and false are not keywords.**

**99. What happens when you add a double value to a String?**

**The result is a String object.**

**100. What is the diffrence between inner class and nested class?**

**When a class is defined within a scope od another class, then it becomes inner class. If the access modifier of the inner class is static, then it becomes nested class.**

**101. Can an abstract class be final?**

**An abstract class may not be declared as**

**final.**

**102. What is numeric promotion?**

**Numeric promotion is the conversion of a smaller numeric type to a larger numeric type, so that integer and floating-point operations may take place. In numerical promotion, byte, char, and short values are converted to int values. The int values are also converted to long values, if necessary. The long and float values are converted to double values, as required.**

**103. What is the difference between a public and a non-public class?**

**A public class may be accessed outside of its package. A non-public class may not be accessed outside of its package.**

**104. To what value is a variable of the boolean type automatically initialized?**

**The default value of the boolean type is false.**

**105. What is the difference between the prefix and postfix forms of the ++ operator?**

**The prefix form performs the increment operation and returns the value of the increment operation. The postfix form returns the current value all of the expression and then performs the increment operation on that value.**

**106. What restrictions are placed on method overriding?**

**Overridden methods must have the same name, argument list, and return type. The overriding method may not limit the access of the method it overrides. The overriding method may not throw any exceptions that may not be thrown by the overridden method.**

**107. What is a Java package and how is it used?**

**A Java package is a naming context for classes and interfaces. A package is used to create a separate name space for groups of classes and interfaces. Packages are also used to organize related classes and interfaces into a single API unit and to control accessibility to these classes and interfaces.**

**108. What modifiers may be used with a top-level class?**

**A top-level class may be public, abstract, or final.**

**115. What is constructor chaining and how is it achieved in Java ?**

**A child object constructor always first needs to construct its parent (which in turn calls its parent constructor.). In Java it is done via an implicit call to the no-args constructor as the first statement.**

**116. What is the difference between the Boolean & operator and the && operator?**

**If an expression involving the Boolean & operator is evaluated, both operands are evaluated. Then the & operator is applied to the operand. When an expression involving the && operator is evaluated, the first operand is evaluated. If the first operand returns a value of true then the second operand is evaluated. The && operator is then applied to the first and second operands. If the first operand evaluates to false, the evaluation of the second operand is skipped.**

**117. Which Java operator is right associative?**

**The = operator is right associative.**

**118. Can a double value be cast to a byte?**

**Yes, a double value can be cast to a byte.**

**119. What is the difference between a break statement and a continue statement?**

**A**

**break statement results in the termination of the statement to which it applies (switch, for, do, or while). A continuestatement is used to end the current loop iteration and return control to the loop statement.**

**120. Can a for statement loop indefinitely?**

**Yes, a for statement can loop indefinitely. For example, consider the following:**

**for(;;);**

* + - * + **Can there be an abstract class with no abstract methods in it?**
        + **Can an Interface be final?**
        + **Can an Interface have an inner class?**

**- Yes. - No**

* + - * + **public interface abc**
        + **{**
        + **static int i=0; void dd();**
        + **class a1**
        + **{**
        + **a1()**
        + **{**
        + **int j;**
        + **System.out.println("inside");**
        + **};**
        + **public static void main(String a1[])**
        + **{**
        + **System.out.println("in interfia");**
        + **}**
        + **}**
        + **}**
        + **Can we define private and protected modifiers for variables in interfaces?**
        + **What is Externalizable?**
        + **What modifiers are allowed for methods in an Interface?**
        + **What is a local, member and a class variable?**
        + **What are the different identifier states of a Thread?**
        + **What are some alternatives to inheritance?**
        + **Why isn’t there operator overloading?**
        + **What does it mean that a method or field is "static"?**
        + **How do I convert a numeric IP address like 192.18.97.39 into a hostname like java.sun.com?**

**- Static variables and methods are instantiated only once per class. In other words they are class variables, not instance variables. If you change the value of a static variable in a particular object, the value of that variable changes for all instances of that class. Static methods can be referenced with the name of the class rather than the name of a particular object of the class (though that works too). That’s how library methods like System.out.println() work. out is a static field in the java.lang.System class. - Because C++ has proven by example that operator overloading makes code almost impossible to maintain. In fact there very nearly wasn’t even method overloading in Java, but it was thought that this was too useful for some very basic methods like print(). Note that some of the classes like DataOutputStream have unoverloaded methods like writeInt() and writeByte(). - Delegation is an alternative to inheritance. Delegation means that you include an instance of another class as an instance variable, and forward messages to the instance. It is often safer than inheritance because it forces you to think about each message you forward, because the instance is of a known class, rather than a new class, and because it doesn’t force you to accept all the methods of the super class: you can provide only the methods that really make sense. On the other hand, it makes you write more code, and it is harder to re-use (because it is not a subclass). - The different identifiers of a Thread are: R - Running or runnable thread, S - Suspended thread, CW - Thread waiting on a condition variable, MW - Thread waiting on a monitor lock, MS - Thread suspended waiting on a monitor lock - Variables declared within a method are "local" variables. Variables declared within the class i.e not within any methods are "member" variables (global variables). Variables declared within the class i.e not within any methods and are defined as "static" are class variables - Only public and abstract modifiers are allowed for methods in interfaces. - Externalizable is an Interface that extends Serializable Interface. And sends data into Streams in Compressed Format. It has two methods, writeExternal(ObjectOuput out) and readExternal(ObjectInput in)**

* + - * + **String hostname = InetAddress.getByName("192.18.97.39").getHostName();**
        + **Difference between JRE/JVM/JDK?**
        + **Why do threads block on I/O?**
        + **What is synchronization and why is it important?**
        + **Is null a keyword?**
        + **Which characters may be used as the second character of an identifier,but not as the first character of an identifier?**
        + **What modifiers may be used with an inner class that is a member of an outer class?**
        + **How many bits are used to represent Unicode, ASCII, UTF-16, and UTF-8 characters?**
        + **What are wrapped classes?**
        + **What restrictions are placed on the location of a package statement within a source code file?**
        + **What is the difference between preemptive scheduling and time slicing?**
        + **What is a native method?**
        + **What are order of precedence and associativity, and how are they used?**
        + **What is the catch or declare rule for method declarations?**
        + **Can an anonymous class be declared as implementing an interface and extending a class?**
        + **What is the range of the char type?**

**- The range of the char type is 0 to 2^16 - 1. - An anonymous class may implement an interface or extend a superclass, but may not be declared to do both. - If a checked exception may be thrown within the body of a method, the method must either catch the exception or declare it in its throws clause. - Order of precedence determines the order in which operators are evaluated in expressions. Associatity determines whether an expression is evaluated left-to-right or right-to-left - A native method is a method that is implemented in a language other than Java. - Under preemptive scheduling, the highest priority task executes until it enters the waiting or dead states or a higher priority task comes into existence. Under time slicing, a task executes for a predefined slice of time and then reenters the pool of ready tasks. The scheduler then determines which task should execute next, based on priority and other factors. - A package statement must appear as the first line in a source code file (excluding blank lines and comments). - Wrapped classes are classes that allow primitive types to be accessed as objects. - Unicode requires 16 bits and ASCII require 7 bits. Although the ASCII character set uses only 7 bits, it is usually represented as 8 bits. UTF-8 represents characters using 8, 16, and 18 bit patterns. UTF-16 uses 16-bit and larger bit patterns. - A (non-local) inner class may be declared as public, protected, private, static, final, or abstract. - The digits 0 through 9 may not be used as the first character of an identifier but they may be used after the first character of an identifier. - The null value is not a keyword. - With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object’s value. This often leads to significant errors. - Threads block on i/o (that is enters the waiting state) so that other threads may execute while the I/O operation is performed.**

**- No - Yes What are static methods?**

**1.What are the uses of final method?**

**What is Constructor?**

* + **You see that something in your design will change frequently.**
  + **If various implementations only share method signatures then it is better to use Interfaces.**
  + **you need some classes to use some methods which you don't want to be included in the class, then you go for the interface, which makes it easy to just implement and make use of the methods defined in the interface.**