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 MAX_LENGTH = 50; 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?

The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declared *protected*, however methods and fields in a interface cannot be declared *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 *m1* method, 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 2^{16} - 1 (i.e. 0 to 65535.)

82. What is the range of the short type?

The range of the *short* type is -(2^{15}) to 2^{15} - 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 while*statement 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.

109. What is the difference between an if statement and a switch statement?

The *if* statement is used to select among two alternatives. It uses a boolean expression to decide which alternative should be executed. The switch statement is used to select among multiple alternatives. It uses an int expression to determine which alternative should be executed.

110. What are the practical benefits, if any, of importing a specific class rather than an entire package (e.g. import java.net.* versus import java.net.Socket)?

It makes no difference in the generated class files since only the classes that are actually used are referenced by the generated class file. There is another practical benefit to importing single classes, and this arises when two (or more) packages have classes with the same name. Take *java.util.Timer* and javax.swing.Timer, for example. If I *import java.util.** and *javax.swing.** and then try to use "Timer", I get an error while compiling (the class name is ambiguous between both packages). Let's say what you really wanted was the *javax.swing.Timer* class, and the only classes you plan on using in *java.util* are Collection and HashMap. In this case, some people will prefer to *import java.util.Collection* and *import java.util.HashMap* instead of importing *java.util.**. This will now allow them to use Timer, Collection, HashMap, and other *javax.swing* classes without using fully qualified class names in.

111. Can a method be overloaded based on different return type but same argument type ?

No, because the methods can be called without using their return type in which case there is ambiquity for the compiler.

112. What happens to a static variable that is defined within a method of a class ?

Can't do it. You'll get a compilation error.

113. How many static initializers can you have ?

As many as you want, but the static initializers and class variable initializers are executed in textual order and may not refer to class variables declared in the class whose declarations appear textually after the use, even though these class variables are in scope.

114. What is the difference between method overriding and overloading?

Overriding is a method with the same name and arguments as in a parent, whereas overloading is the same method name but different arguments

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 *continue* statement 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(;;);*

121. To what value is a variable of the String type automatically initialized?

The default value of an String type is null.

122. What is the difference between a field variable and a local variable?

A field variable is a variable that is declared as a member of a class. A local variable is a variable that is declared local to a method.

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

this() is used to invoke a constructor of the same class. *super()* is used to invoke a superclass constructor.

124. What does it mean that a class or member is final?

A final class cannot be inherited. A final method cannot be overridden in a subclass. A final field cannot be changed after it's initialized, and it must include an initializer statement where it's declared.

125. What does it mean that a method or class is abstract?

An abstract class cannot be instantiated. Abstract methods may only be included in abstract classes. However, an abstract class is not required to have any abstract methods, though most of them do. Each subclass of an abstract class must override the abstract methods of its superclasses or it also should be declared abstract.

126. What is a transient variable?

Transient variable is a variable that may not be serialized.

127. How does Java handle integer overflows and underflows?

It uses those low order bytes of the result that can fit into the size of the type allowed by the operation.

128. What is the difference between the >> and >>> operators?

The >> operator carries the sign bit when shifting right. The >>> zero-fills bits that have been shifted out.

129. Is sizeof a keyword?

The sizeof operator is not a keyword.