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.