**Java**

**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?**

No a class cannot be defined as static. Only a method, a variable or a block of code can be declared as static.

**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. How to define an Abstract class?**

A class containing abstract method is called Abstract class. An Abstract class can't be instantiated. Example of Abstract class: abstract class testAbstractClass { protected String myString; public String getMyString() { return myString; } public abstract string anyAbstractFunction(); }

**38. What is NullPointerException? and how to handle it?**

When an object is not initialized, the default value is null. When the following things happen, the NullPointerException? is thrown: --Calling the instance method of a null object. --Accessing or modifying the field of a null object. --Taking the length of a null as if it were an array. --Accessing or modifying the slots of null as if it were an array. --Throwing null as if it were a Throwable value. The NullPointerException? is a runtime exception. The best practice is to catch such exception even if it is not required by language design.

**39. An application needs to load a library before it starts to run, how to code??**

One option is to use a static block to load a library before anything is called. For example, class Test { static { System.loadLibrary("path-to-library-file"); } .... } When you call new Test(), the static block will be called first before any initialization happens. Note that the static block position may matter.

**40. What's the difference between an interface and an abstract class?**

An abstract class may contain code in method bodies, which is not allowed in an interface. With abstract classes, you have to inherit your class from it and Java does not allow multiple inheritance. On the other hand, you can implement multiple interfaces in your class.

**41. What do you understand by Synchronization?**

Synchronization is a process of controlling the access of shared resources by the multiple threads in such a manner that only one thread can access one resource at a time. In non synchronized multithreaded application, it is possible for one thread to modify a shared object while another thread is in the process of using or updating the object's value. Synchronization prevents such type of data corruption. E.g. Synchronizing a function: public synchronized void Method1 () { *Appropriate method-related code. } E.g. Synchronizing a block of code inside a function: public myFunction (){ synchronized (this) {* Synchronized code here. } }

**42. How to define an Interface?**

In Java Interface defines the methods but does not implement them. Interface can include constants. A class that implements the interfaces is bound to implement all the methods defined in Interface. Emaple of Interface:

public interface sampleInterface { public void functionOne();

public long CONSTANT\_ONE = 1000; }

**43. What is Collection API?**

Answer: The Collection API is a set of classes and interfaces that support operation on collections of objects. These classes and interfaces are more flexible, more powerful, and more regular than the vectors, arrays, and hashtables if effectively replaces. Example of classes: HashSet?, HashMap?, ArrayList?, LinkedList?, TreeSet? and TreeMap?. Example of interfaces: Collection, Set, List and Map.

**44. What is similarities/difference between an Abstract class and Interface? Differences are as follows: Interfaces provide a form of multiple inheritance. A class can extend only one other class. Interfaces are limited to public methods and constants with no implementation. Abstract classes can have a partial implementation, protected parts, static methods, etc. A Class may implement several interfaces. But in case of abstract class, a class may extend only one abstract class. Interfaces are slow as it requires extra indirection to to find corresponding method in in the actual class. Abstract classes are fast. Similarities:**

Neither Abstract classes or Interface can be instantiated.

**45. If a class is located in a package, what do you need to change in the OS environment to be able to use it?**

You need to add a directory or a jar file that contains the package directories to the CLASSPATH environment variable. Let's say a class Employee belongs to a package com.xyz.hr; and is located in the file c:\dev\com\xyz\hr\Employee.java. In this case, you'd need to add c:\dev to the variable CLASSPATH. If this class contains the method main(), you could test it from a command prompt window as follows: c:\>java com.xyz.hr.Employee

**46. What is synchronization and why is it important?**

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 causes dirty data and leads to significant errors.

**47. What is multi-threading?**

Multi-threading means various threads that run in a system.

**48. 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.

**50. 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.

**51. What is polymorphism?**

Polymorphism means "having many forms". It allows methods (may be variables) to be written that needn't be concerned about the specifics of the objects they will be applied to. That is, the method can be specified at a higher level of abstraction and can be counted on to work even on objects of un-conceived classes. **52. 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.

**53. What is the Java Virtual Machine? The Java Virtual Machine is a software that can be ported onto various hardware-based platforms.**

**54. What is the Java API?**

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets.

**55. What is the package?**

The package is a Java namespace or part of Java libraries. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages.

**56. Describe the principles of OOPS. There are three main principals of oops which are called Polymorphism, Inheritance and Encapsulation.** 57. Explain the Inheritance principle.

Inheritance is the process by which one object acquires the properties of another object.

**58. Explain the Encapsulation principle.**

Encapsulation is a process of binding or wrapping the data and the codes that operates on the data into a single entity. This keeps the data safe from outside interface and misuse. One way to think about encapsulation is as a protective wrapper that prevents code and data from being arbitrarily accessed by other code defined outside the wrapper. **59. Explain the Polymorphism principle.**

The meaning of Polymorphism is something like one name many forms. Polymorphism enables one entity to be used as as general category for different types of actions. The specific action is determined by the exact nature of the situation. The concept of polymorphism can be explained as "one interface, multiple methods".

**60. Explain the different forms of Polymorphism.**

From a practical programming viewpoint, polymorphism exists in three distinct forms in Java: Method overloading Method overriding through inheritance Method overriding through the Java interface

**61. What are Access Specifiers available in Java? ccess specifiers are keywords that determines the type of access to the member of a class. These are: Public Protected Private Defaults**

**62. Explain what is an object?**

An object is a combination of messages and data. Objects can receive and send messages and use messages to interact with each other. The messages contain information that is to be passed to the recipient object.

**63.What is the difference between a constructor and a method?**

A constructor is a member function of a class that is used to create objects of that class. It has the same name as the class itself, has no return type, and is invoked using the new operator.

A method is an ordinary member function of a class. It has its own name, a return type (which may be void), and is invoked using the dot operator.

**64. Describe synchronization in respect to multithreading.**

With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources.

Without synchonization, it is possible for one thread to modify a shared variable while another thread is in the process of using or updating same shared variable. This usually leads to significant errors.

**65. What is an abstract class?**

Abstract class must be extended/subclassed (to be useful). It serves as a template. A class that is abstract may not be instantiated (ie. you may not call its constructor), abstract class may contain static data.

Any class with an abstract method is automatically abstract itself, and must be declared as such. A class may be declared abstract even if it has no abstract methods. This prevents it from being instantiated.

**66. What is the difference between an Interface and an Abstract class?**

An abstract class can have instance methods that implement a default behavior. An Interface can only declare constants and instance methods, but cannot implement default behavior and all methods are implicitly abstract.

An interface has all public members and no implementation. An abstract class is a class which may have the usual flavors of class members (private, protected, etc.), but has some abstract methods.