**1) What is the difference between an Interface and an Abstract class?**

In Interface, multiple inheritance is allowed. There is no instance properties and no constructors. Interface is slow in terms of performan

### 2) How do you ensure that N threads can access N resources with out deadlock?

- You have to impose an forced order on the locks and unlock.

- Prefer Immutable types.

- Copying data instead of sharing data.

-Use synchronized code blocks

### 3) What happens if an exception is thrown in finally block?

Exceptions in the finally block will hide / consume any return values or exceptions from the try...catch blocks. The finally block will **not** be completed beyond the point where the exception is thrown. If the finally block was executing during the handling of an earlier exception then that first exception is lost.

4) **What is the purpose of overriding finalize() method?**

The finalize() method should be overridden for an object to include the clean up code or to dispose of the system resources that should to be done before the object is garbage collected.

**5) What happens if an uncaught exception is thrown from during the execution of the finalize() method of an object?**

The exception will be ignored and the garbage collection (finalization) of that object terminates

**6) What are the differente types of Garbage Collectors ?**

JVM has five types of *GC* implementations:

* **Serial Garbage Collector** : it basically works with a single thread. As a result**, this GC implementation freezes all application threads when it runs**. Therefore, it's not a good idea to use it in multi-threaded applications
* **Parallel Garbage Collector**: It's the default GC of the JVM, it uses multiple threads for managing heap space, but it also freezes other application threads while performing GC
* **CMS Garbage Collector :** it uses multiple garbage collector threads for garbage collection. It can afford to share processor resources with the garbage collector while the application is running.
* **G1 Garbage Collector :** it is designed for applications running on multi-processor machines with large memory space. The G1 collector partitions the heap into a set of equal-sized heap regions, each a contiguous range of virtual memory
* **Z Garbage Collector :** ZGC performs all expensive work concurrently, without stopping the execution of application threads for more than 10 ms, which makes it suitable for applications that require low latency. Similar to G1, Z Garbage Collector partitions the heap, except that heap regions can have different sizes.

**7) List vs Array :**

- The array is faster in case of access to an element while List is faster in case of adding/deleting an element from the collection.

- An ArrayList can be efficiently sorted.

- List Can contain elements of different types, array not.

- Arrays are great for numerical operations; lists cannot directly handle math operations.

**8) Why would you use a synchronized block vs. synchronized method?**

Synchronized blocks place locks for shorter periods than synchronized methods.

**9) Explain the usage of the keyword transient?**

This keyword indicates that the value of this member variable does not have to be serialized with the object.

**10) What's the main difference between a Vector and an ArrayList :**

Java Vector class is internally synchronized and ArrayList is not.

**11) What's the difference between the methods sleep() and wait():**

**S**leep(1000); puts thread aside for exactly one second. Wait(1000), causes a wait of up to one second

The major difference is sleep() is actually meant for short pause because it doesn't release lock, while wait() is meant for conditional wait and that's why it release lock. Wait is used for inter-thread communication while sleep is used to introduce pause on execution, generally.

• wait(): until call notify(), notifyAll() from object

• sleep(): until at least time expire or call interrupt().

**12) What is the difference between a Set, a Map, a List, an Hashmap and hashtable in Java?**

Set stores objects whereas Map maps keys to values.

Set does not have duplicates. Map does not have dup keys but dup values are possible.

List in Java provides ordered and indexed collection which may contain duplicates. List's contract maintains insertion order of element. Set is an unordered collection,

List allows duplicates while Set doesn't allow duplicates.

Hashtable is synchronized, better for multi-thread applications. On the other hand, it is less efficient in case of a single thread application.So Hashmap is faster in general. HashTable does not accept null either for keys or for values whereas HashMap class allows only one key with null value and as many null values as possible. LinkedHashMap, his derived class is good for sorting data according to its key.

**13) Difference between StringBuffer and StringBuilder**

StringBuffer is synchronized while StringBuilder is non-synchronized, but StringBuilder is more efficient than StringBuffer.

**14) Mutable VS immutable**

Mutable objects have fields that can be changed after construction, immutable objects have no fields that can be changed after the object is created.

To made and object immutable, the class must be declared **final, m**ust not have setters and all fields must be initialized at the beginning and declared **private** and **final**.

## Benefits of Immutable Classes in Java

* Immutable objects are by default [thread safe](http://javarevisited.blogspot.com/2012/01/how-to-write-thread-safe-code-in-java.html), can be shared without synchronization in concurrent environment.
* Immutable object boost performance of Java application by reducing [synchronization](http://java67.blogspot.com/2013/01/difference-between-synchronized-block-vs-method-java-example.html) in code.
* They are excellent keys for Map and Set.
* The immutable classes are adapted to the representation of abstract data types.
* Another important benefit of Immutable objects is **reusability**

**15) Aggregation and Composition**

Java

Between composition and aggregation, a small difference exists. It is simply if **Demo** can exist without **Test**, it is known as **aggregation** and if **Demo** cannot exist without **Test** object, it is known as **composition**. Composition is a STRONGER relationship whereas Aggregation is a WEAKER relationship. Composition depicts dependency between objects but Aggregation depicts related objects can exist independently.

**16) What is encapsulation and why is it important in Object Oriented programming?**

Encapsulation is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. That means binding object state(fields) and behaviour(methods) together

importances:

1) It improves maintainability and flexibility and re-usability.

2) The fields of a class can be made read-only or write-only

3) A class can have total control over what is stored in its fields.

4) The users of a class do not know how the class stores its data.

**17) SQL: Difference between DELETE, TRUNCATE and DROP.**

TRUNCATE (DDL) is like DELETE (DML) without the WHERE clause.

TRUNCATE can not run on a table with health constraints and on a replicate table or database.DELETE can be if the integrity constraints are respected.

TRUNCATE does not erase the data contained in the pages, but marks them as reusable. DELETE physically deletes the data.

Data deleted by a DELETE is recoverable; Data deleted by a TRUNCATE are not recoverable.

The TRUNCATE resets the auto-increment counter to 0, not the DELETE.

DROP removes the table and all rows from the database. all indexes, privileges, and triggers will also be removed. No ROLLBACK is possible.

**18) Essential difference between Callable and Runnable interface in Java**

Even though both [Callable](http://www.java67.com/2013/01/difference-between-callable-and-runnable-java.html) and [Runnable](http://www.java67.com/2012/08/what-is-thread-and-runnable-in-java.html) interface are used to encapsulate task supposed to be executed in parallel by another thread, there is two key difference between Callable and Runnable interface:  
  
1. Callable can return result  
2. Callable can throw checked Exception.

3. In order to use Callable, you need to override the call() method while in order to use the Runnable interface you need to override the run() method in Java.

**19) Hibernate Get() vs Load()**

Get() use eager method , it return null, If no row is available in the session cache or the database for the given identifier whereas load() use lazy method, it throws object not found exception.

**20) what is Fail Fast and Fail Safe**

The **Java** Collection supports two types of iterators; **Fail Fast and Fail Safe**. These iterators are very useful in exception handling. The **Fail fast** iterator aborts the operation as soon it exposes failures and stops the entire operation. Comparatively, **Fail Safe** iterator doesn't abort the operation in case of a **failure**.

**Fail**-**safe** iterators allow modifications of a collection while iterating over it. These iterators don't throw any Exception if a collection is modified while iterating over it. They use copy of original collection to traverse over the elements of the collection.

**21) What is optimistic locking and pessimistic locking?**

**Optimistic locking**:Updating the object does not require you to lock up the database resource. A record is locked only when changes are committed to the database. The lock is obtained only after the transaction has processed.

**Pessimistic locking** : a record is locked while it is edited.

**22) What are the principles of OOP.**

Inheritance , Abstraction, Encapsulation, Polymorphism, Aggregation, Composition, Association

**23) Can we override/overload static and private methods?**

Static and Private methods can’t be overridden but can be overloaded.

**24) Difference between Stack and Heap memory in Java.**

The main difference between heap and stack is that stack memory is used to store local variables and function call while heap memory stores objects in Java.

Variables stored in stack is only visible to the owner thread while objects created in the heap are visible to all threads. Stack memory is a private memory of a Java Thread while heap memory is shared among all threads.

**25) Is Java primitive data type stored on stack or heap?**

Primitive types declared locally will be on the stack while primitive types that are defined as part of an object instance are stored on the heap.

Local variables are stored on stack while instance and static variables are stored on the heap.

**26) Does Java garbage collector cleans both heap and stack memory?**

GC sweeps heap memory only. Usually, stack memory is collected automatically when the execution path reaches the end of the scope.

**27) The difference between hashcode and equals and their relation?**

**Hashcode** (prints the reference number) is a referencing equality while **equals** is a logical equality. If two objects are equal then they should have the same hashcode and if two objects are not equal then they may or may not have same hash code.

You need to override equals() and hashcode() methods of a class whose objects you want to use as Key in a hashmap. This is required because hashmap uses these 2 methods to retrieve the stored values.

**28) Some design patern you know:**

Adapter pattern, Decorator pattern, Proxy pattern, Template pattern, Visitor pattern, Composite pattern, Abstract factory pattern, Prototype pattern, Flyweight

**29) Difference between DOM and SAX parser in Java?**

DOM parser loads the whole XML into memory to create a tree based DOM model which helps it quickly locate nodes and make a change in the structure of XML while SAX parser is an event based parser and doesn't load the whole XML into memory. Due to this reason DOM is faster than SAX but require more memory and not suitable to parse large XML files

**Others questions:**

**- Is local variable thread safe in java?** YES

- **Is Instance variables thread safe in Java?** NO

**-Can a constructor call/invoke a static method?** YES

**- Can a constructor invoke a non-static/instance method?** YES

**- What are the access modifiers that cannot be applied to a constructor?**

final ,static, abstract and synchronized

**- Can we declare static members in a method?** NO

- **Can we write switch case using a double variable?** NO

- **Should the nested interface must be public?** YES

- **Difference between Serializable and Externalizable?** Serializable is a marker interface as it has defined no members while Externalizable is not as it has two methods. Externalizable extends Serializable interface.

**- Can we overrid a private method?** NO, since it is not visible from any other class.

**-** The overriding method must not throw new or broader checked exceptions. Only new unchecked exception is possible. If super class method doesn’t have throws clause, then it can be overridden with only unchecked type of exceptions.