**COLLECTIONS**

**What is HashMap and Map?**

Map is Interface and Hashmap is class that implements this interface.

<br /><font size=-1>

**What is the significance of ListIterator?**

**Or**

**What is the difference b/w Iterator and ListIterator?**

**Iterator :** Enables you to cycle through a collection in the forward direction only, for obtaining or removing elements

**ListIterator :** It extends Iterator, allow bidirectional traversal of list and the modification of elements

**Difference between HashMap and HashTable?** Can we make hashmap synchronized?

1. The HashMap class is roughly equivalent to Hashtable, except that it is unsynchronized and permits nulls. (***HashMap*** *allows null values as key and value whereas Hashtable doesn’t allow nulls*).  
2. HashMap does not guarantee that the order of the map will remain constant over time.  
3. HashMap is non synchronized whereas Hashtable is synchronized.  
4. Iterator in the HashMap is fail-safe while the enumerator for the Hashtable isn’t.

Note on Some Important Terms  
1)Synchronized means only one thread can modify a hash table at one point of time. Basically, it means that any thread before performing an update on a hashtable will have to acquire a lock on the object while others will wait for lock to be released.

2)Fail-safe is relevant from the context of iterators. If an iterator has been created on a collection object and some other thread tries to modify the collection object “structurally”, a concurrent modification exception will be thrown. It is possible for other threads though to invoke “set” method since it doesn’t modify the collection “structurally”. However, if prior to calling “set”, the collection has been modified structurally, “IllegalArgumentException” will be thrown.

HashMap can be synchronized by

Map m = Collections.synchronizeMap(hashMap);

**What is the difference between set and list?**

A Set stores elements in an unordered way and does not contain duplicate elements, whereas a list stores elements in an ordered way but may contain duplicate elements.

**Difference between Vector and ArrayList? What is the Vector class?**

Vector is synchronized whereas ArrayList is not. The Vector class provides the capability to implement a growable array of objects. ArrayList and Vector class both implement the List interface. Both classes are implemented using dynamically resizable arrays, providing fast random access and fast traversal. In vector the data is retrieved using the elementAt() method while in ArrayList, it is done using the get() method. ArrayList has no default size while vector has a default size of 10. when you want programs to run in multithreading environment then use concept of vector because it is synchronized. But ArrayList is not synchronized so, avoid use of it in a multithreading environment.

**What is an Iterator interface? Is Iterator a Class or Interface? What is its use?**

The Iterator is an interface, used to traverse through the elements of a Collection. It is not advisable to modify the collection itself while traversing an Iterator.

**What is the Collections API?**

The Collections API is a set of classes and interfaces that support operations on collections of objects.  
Example of classes: HashSet, HashMap, ArrayList, LinkedList, TreeSet and TreeMap.  
Example of interfaces: Collection, Set, List and Map.

**What is the List interface?**

The List interface provides support for ordered collections of objects.

**How can we access elements of a collection?**

We can access the elements of a collection using the following ways:  
1.Every collection object has get(index) method to get the element of the object. This method will return Object.  
2.Collection provide Enumeration or Iterator object so that we can get the objects of a collection one by one.

**What is the Set interface?**

The Set interface provides methods for accessing the elements of a finite mathematical set. Sets do not allow duplicate elements.

**What’s the difference between a queue and a stack?**

Stack is a data structure that is based on last-in-first-out rule (LIFO), while queues are based on First-in-first-out (FIFO) rule.

**What is the Map interface?**

The Map interface is used associate keys with values.

**What is the Properties class?**

The properties class is a subclass of Hashtable that can be read from or written to a stream. It also provides the capability to specify a set of default values to be used.

**Which implementation of the List interface provides for the fastest insertion of a new element into the middle of the list?**

a. Vector  
b. ArrayList  
c. LinkedList  
d. None of the above

ArrayList and Vector both use an array to store the elements of the list. When an element is inserted into the middle of the list the elements that follow the insertion point must be shifted to make room for the new element. The LinkedList is implemented using a doubly linked list; an insertion requires only the updating of the links at the point of insertion. Therefore, the LinkedList allows for fast insertions and deletions.

**How can we use hashset in collection interface?**

This class implements the set interface, backed by a hash table (actually a HashMap instance). It makes no guarantees as to the iteration order of the set; in particular, it does not guarantee that the order will remain constant over time. This class permits the Null element.

This class offers constant time performance for the basic operations (add, remove, contains and size), assuming the hash function disperses the elements properly among the buckets.

**What are differences between Enumeration, ArrayList, Hashtable and Collections and Collection?**

Enumeration: It is series of elements. It can be use to enumerate through the elements of a vector, keys or values of a hashtable. You can not remove elements from Enumeration.

ArrayList: It is re-sizable array implementation. Belongs to ‘List’ group in collection. It permits all elements, including null. It is not thread -safe.

Hashtable: It maps key to value. You can use non-null value for key or value. It is part of group Map in collection.

Collections: It implements Polymorphic algorithms which operate on collections.

Collection: It is the root interface in the collection hierarchy.

**What is difference between array & arraylist?**

An ArrayList is resizable, where as, an array is not. ArrayList is a part of the Collection Framework. We can store any type of objects, and we can deal with only objects. It is growable. Array is collection of similar data items. We can have array of primitives or objects. It is of fixed size. We can have multi dimensional arrays.

**Array:** can store primitive **ArrayList:** Stores object only

**Array:** fix size **ArrayList:** resizable

**Array:** can have multi dimensional

**Array:** lang                                **ArrayList:** Collection framework

**Can you limit the initial capacity of vector in java?**

Yes you can limit the initial capacity. We can construct an empty vector with specified initial capacity

public vector(int initialcapacity)

**What method should the key class of Hashmap override?**

The methods to override are equals() and hashCode().

**What is the difference between Enumeration and Iterator?**

The functionality of Enumeration interface is duplicated by the Iterator interface. Iterator has a remove() method while Enumeration doesn’t. Enumeration acts as Read-only interface, because it has the methods only to traverse and fetch the objects, where as using Iterator we can manipulate the objects also like adding and removing the objects.

So Enumeration is used when ever we want to make Collection objects as Read-only.

|  |  |
| --- | --- |
|  | [**Java Abstract Class and Interface Interview Questions**](http://www.interview-questions-java.com/java-questions/java-abstract-class-and-interface-interview-questions) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **What is the difference between Abstract class and Interface** **Or** **When should you use an abstract class, when an interface, when both?Or** **What is similarities/difference between an Abstract class and Interface?** **Or** **What is the difference between interface and an abstract class?**  1. Abstract class is a class which contain one or more abstract methods, which has to be implemented by sub classes. An abstract class can contain no abstract methods also i.e. abstract class may contain concrete methods. A Java Interface can contain only method declarations and public static final constants and doesn’t contain their implementation. The classes which implement the Interface must provide the method definition for all the methods present.  2. Abstract class definition begins with the keyword “abstract” keyword followed by Class definition. An Interface definition begins with the keyword “interface”.  3. Abstract classes are useful in a situation when some general methods should be implemented and specialization behavior should be implemented by subclasses. Interfaces are useful in a situation when all its properties need to be implemented by subclasses  4. All variables in an Interface are by default - public static final while an abstract class can have instance variables.  5. An interface is also used in situations when a class needs to extend an other class apart from the abstract class. In such situations its not possible to have multiple inheritance of classes. An interface on the other hand can be used when it is required to implement one or more interfaces. Abstract class does not support Multiple Inheritance whereas an Interface supports multiple Inheritance.  6. An Interface can only have public members whereas an abstract class can contain private as well as protected members.  7. A class implementing an interface must implement all of the methods defined in the interface, while a class extending an abstract class need not implement any of the methods defined in the abstract class.  8. The problem with an interface is, if you want to add a new feature (method) in its contract, then you MUST implement those method in all of the classes which implement that interface. However, in the case of an abstract class, the method can be simply implemented in the abstract class and the same can be called by its subclass  9. Interfaces are slow as it requires extra indirection to to find corresponding method in in the actual class. Abstract classes are fast  10.Interfaces are often used to describe the peripheral abilities of a class, and not its central identity, E.g. an Automobile class might implement the Recyclable interface, which could apply to many otherwise totally unrelated objects.  Note: There is no difference between a fully abstract class (all methods declared as abstract and all fields are public static final) and an interface.  Note: If the various objects are all of-a-kind, and share a common state and behavior, then tend towards a common base class. If all they share is a set of method signatures, then tend towards an interface.  Similarities: Neither Abstract classes nor Interface can be instantiated.  **What does it mean that a method or class is abstract?**  An abstract class cannot be instantiated. Only its subclasses can be instantiated. A class that has one or more abstract methods must be declared abstract. A subclass that does not provide an implementation for its inherited abstract methods must also be declared abstract. You indicate that a class is abstract with the abstract keyword like this:  public abstract class AbstractClass  Abstract classes may contain abstract methods. A method declared abstract is not actually implemented in the class. It exists only to be overridden in subclasses. 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 itself be declared abstract. Only the method’s prototype is provided in the class definition. Also, a final method can not be abstract and vice versa. Methods specified in an interface are implicitly abstract. . It has no body. For example,  public abstract float getInfo()  **What must a class do to implement an interface?**  The class must provide all of the methods in the interface and identify the interface in its implements clause.  **What is an abstract method?**  An abstract method is a method whose implementation is deferred to a subclass.  **What is interface? How to support multiple inhertance in Java?**  **Or**  **What is a cloneable interface and how many methods does it contain?**  An Interface are implicitly abstract and public. Interfaces with empty bodies are called marker interfaces having certain property or behavior. Examples:java.lang.Cloneable,java.io.Serializable,java.util.EventListener. An interface body can contain constant declarations, method prototype declarations, nested class declarations, and nested interface declarations.  Interfaces provide support for multiple inheritance in Java. A class that implements the interfaces is bound to implement all the methods defined in Interface. Example of Interface: public interface sampleInterface { public void functionOne();  public long CONSTANT\_ONE = 1000; }  **What is an abstract class?** **Or** **Can you make an instance of an abstract class?**  Abstract classes can contain abstract and concrete methods. Abstract classes cannot be instantiated directly i.e. we cannot call the constructor of an abstract class directly nor we can create an instance of an abstract class by using “Class.forName().newInstance()” (Here we get java.lang.InstantiationException). However, if we create an instance of a class that extends an Abstract class, compiler will initialize both the classes. Here compiler will implicitly call the constructor of the Abstract class. Any class that contain an abstract method must be declared “abstract” and abstract methods can have definitions only in child classes. By overriding and customizing the abstract methods in more than one subclass makes “Polymorphism” and through Inheritance we define body to the abstract methods. Basically an abstract class serves as a template. Abstract class must be extended/subclassed for it to be implemented. A class may be declared abstract even if it has no abstract methods. This prevents it from being instantiated. Abstract class is a class that provides some general functionality but leaves specific implementation to its inheriting classes.  Example of Abstract class:  abstract class AbstractClassExample{  protected String name; public String getname() { return name; } public abstract void function(); }  Example: Vehicle is an abstract class and Bus Truck, car etc are specific implementations  No! You cannot make an instance of an abstract class. An abstract class has to be sub-classed. If you have an abstract class and you want to use a method which has been implemented, you may need to subclass that abstract class, instantiate your subclass and then call that method.  **What is meant by “Abstract Interface”?**  Firstly, an interface is abstract. That means you cannot have any implementation in an interface. All the methods declared in an interface are abstract methods or signatures of the methods.  **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. Example of Interface:  public interface SampleInterface { public void functionOne();  public long CONSTANT\_ONE = 1000; }  **Can Abstract Class have constructors? Can interfaces have constructors?**  Abstract class’s can have a constructor, but you cannot access it through the object, since you cannot instantiate abstract class. To access the constructor create a sub class and extend the abstract class which is having the constructor.  Example public abstract class AbstractExample { public AbstractExample(){ System.out.println(”In AbstractExample()”); } }  public class Test extends AbstractExample{ public static void main(String args[]){ Test obj=new Test(); } }  **If interface & abstract class have same methods and those methods contain no implementation, which one would you prefer?**  Obviously one should ideally go for an interface, as we can only extend one class. Implementing an interface for a class is very much effective rather than extending an abstract class because we can extend some other useful class for this subclass  **THREAD**  **What are synchronized methods and synchronized statements?**  Synchronized methods are methods that are used to control access to an object. A thread only executes a synchronized method after it has acquired the lock for the method’s object or class. Synchronized statements are similar to synchronized methods. A synchronized statement can only be executed after a thread has acquired the lock for the object or class referenced in the synchronized statement.  **Can Java object be locked down for exclusive use by a given thread?**  **Or**  **What happens when a thread cannot acquire a lock on an object?**  Yes. You can lock an object by putting it in a “synchronized” block. The locked object is inaccessible to any thread other than the one that explicitly claimed it. If a thread attempts to execute a synchronized method or synchronized statement and is unable to acquire an object’s lock, it enters the waiting state until the lock becomes available.  **What’s the difference between the methods sleep() and wait()?**  The sleep method is used when the thread has to be put aside for a fixed amount of time. Ex: sleep(1000), puts the thread aside for exactly one second. The wait method is used to put the thread aside for up to the specified time. It could wait for much lesser time if it receives a notify() or notifyAll() call. Ex: wait(1000), causes a wait of up to one second. The method wait() is defined in the Object and the method sleep() is defined in the class Thread.  **What is the difference between process and thread?**  A thread is a separate path of execution in a program. A Process is a program in execution.  **What is daemon thread and which method is used to create the daemon thread?**  Daemon threads are threads with low priority and runs in the back ground doing the garbage collection operation for the java runtime system. The setDaemon() method is used to create a daemon thread. These threads run without the intervention of the user. To determine if a thread is a daemon thread, use the accessor method isDaemon()  When a standalone application is run then as long as any user threads are active the JVM cannot terminate, otherwise the JVM terminates along with any daemon threads which might be active. Thus a daemon thread is at the mercy of the runtime system. Daemon threads exist only to serve user threads.  **EXCEPTION**  **What is the difference between exception and error?**  Error’s are irrecoverable exceptions. Usually a program terminates when an error is encountered.  **What is the difference between throw and throws keywords?**  The throw keyword denotes a statement that causes an exception to be initiated. It takes the Exception object to be thrown as an argument. The exception will be caught by an enclosing try-catch block or propagated further up the calling hierarchy. The throws keyword is a modifier of a method that denotes that an exception may be thrown by the method. An exception can be rethrown.  **What class of exceptions are generated by the Java run-time system?**  The Java runtime system generates Runtime Exceptions and Errors.  **What is the base class for Error and Exception?**  Throwable  **WRAPPER CLASS**  Following table lists the primitive types and the corresponding wrapper classes:   |  |  | | --- | --- | | **Primitive** | **Wrapper** | | Boolean | java.lang.Boolean | | Byte | java.lang.Byte | | Char | java.lang.Character | | double | java.lang.Double | | Float | java.lang.Float | | Int | java.lang.Integer | | Long | java.lang.Long | | Short | java.lang.Short | | Void | java.lang.Void | |  |  | |  |  | |