1. What is Java?

Java is an **object oriented programming language which follows the concepts of OOPS like Abstraction, polymorphism, inheritance and encapsulation.**

1. What are the features of Java?

* Simple
* Object Oriented
* Platform Independent
* Portable
* Robust
* Simple: It doesn’t have the concepts of operator overloading and explicit pointers. No need to remove the unreferenced objects because there is Automatic Garbage Collection in java.
* Object Oriented: It follows the concepts of OOPS like APIE(abstraction, polymorphism, inheritance and encapsulation).
* Platform Independent: The byte code makes java platform independent. As it can be run on any platform. (WORA)
* Portable: The byte code can be easily carried to any platform.
* Robust: Java is strong in terms of memory management, automatic garbage collection, no explicit pointers all these make Java Robust.

1. Difference between JDK, JRE and JVM ?

JVM (Java Virtual Machine) : It is an abstract machine which loads, verifies and executes the code, provides the run time environment for the byte code to be executed. It directly comes in contact with OS but the code within the JVM never interacts with it which makes it more secured.

JRE (Java Run Time Environment): It is the implementation of the JVM. It has got all the libraries + all the files used by JVM at run time. It provides the run time environment.

JDK (Java Development Kit): It provides the utilities for the compilation of the code and includes development tools. It consists of the JRE.

1. What is the Java memory Model?

* Class Area: It consists of the static variables.
* Heap: The objects created at the run time are stored in heap.
* Stack: It holds the primitive variables, partial results and local variables.
* Program Counter: It consists of the address of the current instruction being executed.
* Native Methods: it consists of the methods of other programming languages like C.

1. What are the types of variables in JAVA?

* Local Variables: These variables are declared within a method and their scope is also within the method. They don’t have any default values.
* Static Variables: The variable declared as static are known as static variables. It cannot be local.
* Instance Variables: The variables declared outside a method but inside a class are known as instance variables. They do have default values like int (0), string (null) etc.

1. What is class and object?

Class is a blueprint or a template from which objects are collected. That is why objects are known as instances (result) of class.

Object is a real world entity which has state(properties), behavior(functionality) and an uniqueiD.

1. What are different OOPS concepts?

* Abstraction
* Polymorphism
* Encapsulation
* Inheritance

1. What is abstraction?

Abstraction: Abstraction is the **concept** of hiding the implementation and showing only the essential features of the object. Abstraction mainly comes into picture for future flexibility when the developer is not sure of the implementation.

A class that is declared as abstract is known as **abstract class**. It needs to be extended and its method implemented. It cannot be instantiated. A method that is declared as abstract and does not have implementation is known as abstract method.

1. What is Encapsulation?

Encapsulation is the **concept** of restricting the access of data members from one class to another by giving the private keyword and providing access through the getters and setters.

1. What is the difference between abstraction and encapsulation?

* **Encapsulation**is a **concept** by which you restrict the access to some of the object's components, as well as binding the data and methods operating on the data.
* **Abstraction** is the **concept** to define an object that can represent abstract entities which can work, change state and communicate with other entities.
* **Abstraction** is implemented in Java using interface and abstract class while **Encapsulation** is implemented using private and protected access modifier.

1. What is inheritance?

Inheritance is the **concept** of acquisition of properties and behavior from one class to another. It is mainly used for code reusability. It is an IS-A relationship. The extends keyword is used to extend once class to another. A class that is inherited is called a super class. The new class is called a subclass.

1. What is polymorphism?

Polymorphism is the **concept** of an object’s ability to take on many forms. There are two types of polymorphism in java: compile time polymorphism and runtime polymorphism.

1. What is method overloading and overriding?

Method overloading is the **concept** of a class having multiple methods of same name but vary with type and number of arguments. It enhances the readability. It is performed within a class.

Method overriding is the **concept** of implementing a method in the sub class which is already provided in the super class. It should compulsorily be in an IS-A relation. The subclass method should have the same method name and the parameters.

1. What is static in JAVA?

Static is a **keyword** which specifies that the variables, keywords will have common properties to all the instances of the class, rather than depending on the object. It can be applied to a variable, method, block or a nested class. It is mainly used for memory management.

1. What is final, finally, finalize?

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **final** | **finally** | **finalize** |
| 1) | Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed. | Finally is used to place important code, it will be executed whether exception is handled or not. | Finalize is used to perform clean up processing just before object is garbage collected. |
| 2) | Final is a keyword. | Finally is a block. | Finalize is a method. |

1. Abstract vs interface ?

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| 1) Abstract class can **have abstract and non-abstract** methods. | Interface can have **only abstract** methods. |
| 2) Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple inheritance**. |
| 3) Abstract class **can have final, non-final, static and non-static variables**. | Interface has **only static and final variables**. |
| 4) Abstract class **can have static methods, main method and constructor**. | Interface **can't have static methods, main method or constructor**. |
| 5) Abstract class **can provide the implementation of interface**. | Interface **can't provide the implementation of abstract class**. |
| 6) The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| 7) **Example:** public abstract class Shape{ public abstract void draw(); } | **Example:** public interface Drawable{ void draw(); } |

1. What is static and dynamic binding?

The **concept** of Establishing a connection between method body and method call is known as binding. There are two types of binding.

* Static Binding
* Dynamic binding

Static binding: The **concept** of determining the object at the compile time then it is known as early binding or static binding. When a class has final, static or private method is also known as static binding.

Dynamic binding: The **concept** of determining the type of the object at the run time it is known as dynamic binding.

18)Which one you choose between abstract and interface?

Use an abstract class if you have some functionality that you want it's subclasses to have. For instance, if you have a set of functions that you want all of the base abstract class's subclasses to have.

Use an interface if you just want a general contract on behavior/functionality. If you have a function or object that you want to take in a set of different objects, use an interface. Then you can change out the object that is passed in, without changing the method or object that is taking it.

|  |  |
| --- | --- |
|  | Abstract classes are used to group a number of concrete classes under one entity.  For example, take the abstract class Animal. Animal is not something concrete. it's a family of, well, animals. but they all share certain aspectes, for example, each has a speak() option (well, except fish and sort). but each one implements it differently. this way you can override just the methods which are not the same, for example sleep() or breath() are common (again, fish are differnet :) ).  Interfaces on the other hand are more direct definition of an 'action'. That's why most (if not all) the interfaces in Java ends with 'able' (Comprable, Serializable...) By implementing the interface, you're telling other programmers or who ever uses your code that this class can do this and this. A dog, for example, is not, Animable. |

19) Why do you create abstract classes in application development?

For flexibility of the code in the future and also when the developer is not sure of the implementation. The real world abstract things can be considered to be abstract.

20)Why do you create interfaces in application development?

When multiple classes have similar functionality but vary in the implementation of that particular functionality then interfaces come into picture.

21)What are different types of access modifiers?

Public: Accessible everywhere.

Private: Accessible within the class.

Protected: Accessible within the package and outside the package only through inheritance.

Default: Accessible only within the package.

22) What is call by value?

Passing a variable by value is that the value held in the variable that is passed as an argument is copied into the parameters that are defined in the method header. That is why changes made to the variable within the method had no effect on the variable that was passed.

when objects are passed, the object itself is passed. No copy is made.

23) What is string pool?

String pool is a pool of strings stored in java heap memory. String is possible only because of the immutability concept of the string. String pools help in saving lot of space for java Runtime. When we use double quotes to create a String, it first looks for String with same value in the String pool, if found it just returns the reference else it creates a new String in the pool and then returns the reference.

4) Why string is immutable?

Majority of the data is represented in the form of strings. As a result, there are high chances of strings pointing to the same reference. If a string changes, it will affect the other string which is being referenced to the same pointer and there are chances that it might become unreferenced which gets deleted by the automatic garbage collector. This is the reason why strings are made immutable.

25) How do you make a class immutable?

By declaring a class final.

By declaring the properties of the class as final.

By not providing the setters.

26)StringBuffer vs StringBuilder ?

|  |  |  |
| --- | --- | --- |
| **No.** | **StringBuffer** | **StringBuilder** |
| 1) | StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously. | StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously. |
| 2) | StringBuffer is *less efficient* than StringBuilder. | StringBuilder is *more efficient* than StringBuffer |

27) == vs Equals?

== method checks for the reference of the string.

Equals method checks for the actual content of the string.

28)What is exception handling and how do you achieve it?

Exception handling is a problem that occurs during the execution of the program. It disrupts the normal flow of the program and the application terminates abnormally, which is not recommended so these exceptions are to be handled. It is achieved through try, catch and throw blocks.

29)How do you make sure a code must be executed even if exception happens?

By placing the code in the finally block.

30)What code you normally write in finally block?

Most important statements like closing connections and statements are written in finally block. It should always be followed by try or catch block. The finally block gets executed even the exception is handled or not.

31)What are checked vs unchecked exceptions?

Checked exceptions are exceptions that occur at the compile time. The extend only the throwable class. Example: File not found exception.

Unchecked exceptions are exceptions that occur at the run time. For example: - array out of bounds exception.

32)How do you create custom exceptions?

Using the throw keyword, we can explicitly throw exceptions either checked or unchecked exceptions. It is mainly used to throw custom exceptions.

33)How does exception propagation works?

An exception is first thrown from the top of the stack and if it is not caught, it drops down the call stack to the previous method, if not caught there, the exception again drops down to the previous method, and so on until they are caught or until they reach the very bottom of the call stack. This is called exception propagation.

34)exception vs error?

|  |  |
| --- | --- |
| Errors | Exceptions |
| Errors in java are of type java.lang.Error. | Exceptions in java are of type java.lang.Exception. |
| All errors in java are unchecked type. | Exceptions include both checked as well as unchecked type. |
| Errors happen at run time. They will not be known to compiler. | Checked exceptions are known to compiler where as unchecked exceptions are not known to compiler because they occur at run time. |
| It is impossible to recover from errors. | You can recover from exceptions by handling them through try-catch blocks. |
| Errors are mostly caused by the environment in which application is running. | Exceptions are mainly caused by the application itself. |
| Examples : java.lang.StackOverflowError, java.lang.OutOfMemoryError | Examples : Checked Exceptions : SQLException, IOException Unchecked Exceptions : ArrayIndexOutOfBoundException, ClassCastException, NullPointerException |

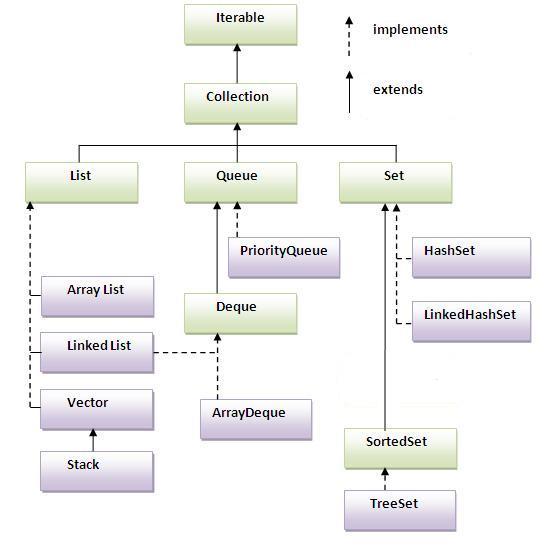
35)Inner classes?

Writing a class within another is allowed in Java. The class written within is called the **inner class**, and the class that holds the inner class is called the **outer class**.

36)What are anonymous classes?

Anonymous classes enable you to make your code more concise. They enable you to declare and instantiate a class at the same time. They are like local classes except that they do not have a name. Use them if you need to use a local class only once.

37) JAVA Collection framework tree ?

`

38) ArrayList vs LinkedList

|  |  |
| --- | --- |
| **ArrayList** | **LinkedList** |
| 1) ArrayList internally uses **dynamic array** to store the elements. | LinkedList internally uses **doubly linked list** to store the elements. |
| 2) Manipulation with ArrayList is **slow** because it internally uses array. If any element is removed from the array, all the bits are shifted in memory. | Manipulation with LinkedList is **faster** than ArrayList because it uses doubly linked list so no bit shifting is required in memory. |
| 3) ArrayList class can **act as a list** only because it implements List only. | LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces. |
| 4) ArrayList is **better for storing and accessing** data. | LinkedList is **better for manipulating** data |

39) List vs Set

1) List is an ordered collection it maintains the insertion order, which means upon displaying the list content it will display the elements in the same order in which they got inserted into the list.

Set is an unordered collection, it doesn’t maintain any order. There are few implementations of Set which maintains the order such as LinkedHashSet (It maintains the elements in insertion order).

2) List allows duplicates while Set doesn’t allow duplicate elements. All the elements of a Set should be unique if you try to insert the duplicate element in Set it would replace the existing value.

3) List implementations: **[ArrayList](http://beginnersbook.com/2013/12/java-arraylist/" \t "_blank" \o "ArrayList in java with example programs – Collections Framework)**, [**LinkedList**](http://beginnersbook.com/2013/12/linkedlist-in-java-with-example/) etc.

Set implementations: **[HashSet](http://beginnersbook.com/2013/12/hashset-class-in-java-with-example/" \t "_blank" \o "HashSet Class in Java with example)**, **[LinkedHashSet](http://beginnersbook.com/2013/12/linkedhashset-class-in-java-with-example/" \t "_blank" \o "LinkedHashSet Class in Java with Example)**, **[TreeSet](http://beginnersbook.com/2013/12/treeset-class-in-java-with-example/" \t "_blank" \o "TreeSet Class in Java with example)** etc.

4) List allows any number of null values. Set can have only a single null value at most.

5) **[ListIterator](http://beginnersbook.com/2014/06/listiterator-in-java-with-examples/" \t "_blank" \o "ListIterator in Java with examples)** can be used to traverse a List in both the directions(forward and backward) However it cannot be used to traverse a Set. We can use [**Iterator**](http://beginnersbook.com/2014/06/java-iterator-with-examples/) (It works with List too) to traverse a Set.

6) List interface has one legacy class called [**Vector**](http://beginnersbook.com/2013/12/vector-in-java/)whereas Set interface does not have any legacy class.

40) Which one you prefer beween ArrayList vs LinkedList ?

LinkedList<E> allows for constant-time insertions or removals using iterators, but only sequential access of elements. In other words, you can walk the list forwards or backwards, but finding a position in the list takes time proportional to the size of the list. Javadoc says "operations that index into the list will traverse the list from the beginning or the end, whichever is closer", so those methods are O(n/4) on average, though O(1) for index = 0.

ArrayList<E>, on the other hand, allow fast random read access, so you can grab any element in constant time. But adding or removing from anywhere but the end requires shifting all the latter elements over, either to make an opening or fill the gap. Also, if you add more elements than the capacity of the underlying array, a new array (1.5 times the size) is allocated, and the old array is copied to the new one, so adding to an ArrayList is O(n) in the worst case but constant on average.

41) How do you sort collection of elements in JAVA ?

Collections.sort can be called with a custom comparator. And that comparator can be implemented to allow sorting in different sort orders.

42) What is Collections class in JAVA ?

**Collections** class provides static methods for sorting the elements of collection. If collection elements are of Set type, we can use TreeSet. But We cannot sort the elements of List. Collections class provides methods for sorting the elements of List type elements.

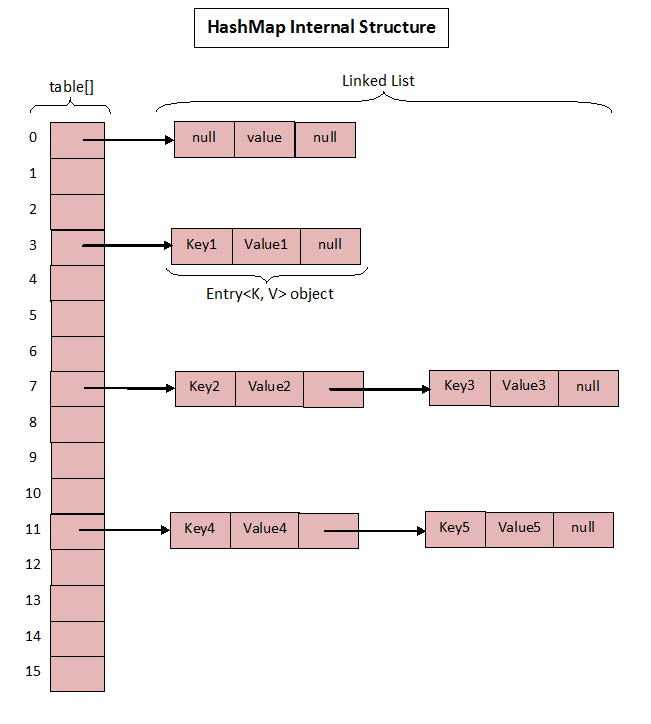
43) HashTable vs HashMap

|  |  |
| --- | --- |
| **HashMap** | **Hashtable** |
| 1) HashMap is **non synchronized**. It is not-thread safe and can't be shared between many threads without proper synchronization code. | Hashtable is **synchronized**. It is thread-safe and can be shared with many threads. |
| 2) HashMap **allows one null key and multiple null values**. | Hashtable **doesn't allow any null key or value**. |
| 3) HashMap is a **new class introduced in JDK 1.2**. | Hashtable is a **legacy class**. |
| 4) HashMap is **fast**. | Hashtable is **slow**. |
| 5) We can make the HashMap as synchronized by calling this code Map m = Collections.synchronizedMap(hashMap); | Hashtable is internally synchronized and can't be unsynchronized. |
| 6) HashMap is **traversed by Iterator**. | Hashtable is **traversed by Enumerator and Iterator**. |
| 7) Iterator in HashMap is **fail-fast**. | Enumerator in Hashtable is **not fail-fast**. |
| 8) HashMap inherits **AbstractMap** class. | Hashtable inherits **Dictionary** class. |

44) What is HashMap

* A HashMap contains values based on the key. It implements the Map interface and extends AbstractMap class.
* It contains only unique elements.
* It may have one null key and multiple null values.
* It maintains no order.

45) How does HashMap Works

HashMap stores the data in the form of key-value pairs. Each key-value pair is stored in an object of Entry<K, V>class. Entry<K, V> class is the static inner class of HashMap which is defined like below.

### How put() method works?

Step 1 : First checks whether the key is null or not. If the key is null, it calls putForNullKey() method. table[0] is always reserved for null key. Because, hash code of null is 0.

Step 2 : If the key is not null, then it calculates the hash code of the key by calling hash() method.

Step 3 : Calls indexFor() method by passing the hash code calculated in step 2 and length of the table[] array. This method returns index in table[] array for the specified key-value pair.

Step 4 : After getting the index, it checks all keys present in the linked list at that index ( or bucket). If the key is already present in the linked list, it replaces the old value with new value.

Step 5 : If the key is not present in the linked list, it appends the specified key-value pair at the end of the linked list.

### How get() method Works?

Step 1 : First checks whether specified key is null or not. If the key is null, it calls getForNullKey() method.

Step 2 : If the key is not null, hash code of the specified key is calculated.

Step 3 : indexFor() method is used to find out the index of the specified key in the table[] array.

Step 4 : After getting index, it will iterate though linked list at that position and checks for the key using equals() method. If the key is found, it returns the value associated with it. otherwise returns null.

### 46) What is HashMap collison

There are chances that the **hashFunction** returns same index for 2 different keys.  This is what we call a **collision**.

**Solution:**

1. *Separate Chaining:* Have an array of lists. If there is more than one key hashed to the same index, append to the list at that index in the array. A decent hash function guarantees that the keys are evenly distributed across the indices. **This is what Java uses.**
2. *2-Choice:*It is similar to Separate Chaining, just that there are 2 hash functions instead of 1. Get the indices from both the hash function. Compare the sizes of the list at these indices and append the new key to the smaller list. This can be extended to k-Choice.
3. *Cuckoo hashing:*You have 2 hash functions. Let the first hash function return **x**. If vacant, put the key **K** there, if not, try the second hash function and let's say it returns**y**, put the key there if vacant. If not, kick out the key **L** at **y**and store **K**there. Run both the hash functions on**L** and try the other index ignoring the index at which it previously it used. Repeat the process. If you cannot find a spot for it after 15 retries, put it in the overflow table.

47) What is ConcurrentHashMap

A hash table supporting full concurrency of retrievals and high expected concurrency for updates. This class obeys the same functional specification as [Hashtable](https://docs.oracle.com/javase/8/docs/api/java/util/Hashtable.html" \o "class in java.util), and includes versions of methods corresponding to each method of Hashtable. However, even though all operations are thread-safe, retrieval operations do not entail locking, and there is not any support for locking the entire table in a way that prevents all access. This class is fully interoperable with Hashtable in programs that rely on its thread safety but not on its synchronization details.

48) Why do you think Strings makes appropriate keys for HashMap ?

 An immutable key will always keep the same hashCode() value, and the hashing function will find the correct bucket ( = index in the hashMap's array) again.

49) What kind of classes are good for HashMap keys ?

Immutable classes

50) What is Iterator vs ListIterator?

1) Iterator is used for traversing List and Set both.

We can use ListIterator to traverse List only, we cannot traverse Set using ListIterator.

2) We can traverse in only forward direction using Iterator.

Using ListIterator, we can traverse a List in both the directions (forward and Backward).

3) We cannot obtain indexes while using Iterator

We can obtain indexes at any point of time while traversing a list using ListIterator. The methods nextIndex() and previousIndex() are used for this purpose.

4) We cannot add element to collection while traversing it using Iterator, it throws ConcurrentModificationException when you try to do it.

We can add element at any point of time while traversing a list using ListIterator.

5) We cannot replace the existing element value when using Iterator.

By using set(E e) method of ListIterator we can replace the last element returned by next() or previous() methods.

6) Methods of Iterator:

* hasNext()
* next()
* remove()

Methods of ListIterator:

* add(E e)
* hasNext()
* hasPrevious()
* next()
* nextIndex()
* previous()
* previousIndex()
* remove()
* set(E e)

51) Comparable vs Comparator interfaces

|  |  |
| --- | --- |
| **Comparable** | **Comparator** |
| 1) Comparable provides **single sorting sequence**. In other words, we can sort the collection on the basis of single element such as id or name or price etc. | Comparator provides **multiple sorting sequence**. In other words, we can sort the collection on the basis of multiple elements such as id, name and price etc. |
| 2) Comparable **affects the original class** i.e. actual class is modified. | Comparator **doesn't affect the original class**i.e. actual class is not modified. |
| 3) Comparable provides **compareTo() method**to sort elements. | Comparator provides **compare() method** to sort elements. |
| 4) Comparable is found in **java.lang** package. | Comparator is found in **java.util** package. |
| 5) We can sort the list elements of Comparable type by **Collections.sort(List)** method. | We can sort the list elements of Comparator type by **Collections.sort(List,Comparator)** method. |

52) Treeset vs TreeMap

1) Major difference between TreeSet and TreeMap is that TreeSet implements Set interface while TreeMap implements Map interface in Java.

2) TreeSet stores only one object while TreeMap uses two objects called key and Value. Objects in TreeSet are sorted while keys in TreeMap remain in sorted Order.

3) Duplicate objects are not allowed in TreeSet but duplicates values are allowed in TreeMap.

54) Why do you need override Hash code and equals

Every object is placed in Hash bucket depending on the hashcode they have. It is not necessary that every different object must have different hashcode. **hashcode is used to narrow the search result.**When we try to insert any key in HashMap first it checks whether any other object present with same hashcode and if yes then it checks for the equals() method. If two objects are same then HashMap will not add that key instead it will replace the old value by new one.

55) What is multi-threading?

**Multithreading in java** is a concept of executing multiple threads simultaneously.

Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

But we use multithreading than multiprocessing because threads share a common memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

56) How do you create Threads?

There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

|  |  |
| --- | --- |
| Thread Class  Thread class provide constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface.  Runnable interface:   |  | | --- | | The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. Runnable interface have only one method named run().  57) How do you synchronize your code?  Synchronization in java is the capability to control the access of multiple threads to any shared resource.  There are two types of thread synchronization mutual exclusive and inter-thread communication.   1. Mutual Exclusive    1. Synchronized method.    2. Synchronized block.    3. static synchronization. 2. Cooperation (Inter-thread communication in java)   58. What is volatile?  volatile is used to indicate that a **variable's value will be modified by different threads**.  Declaring a volatile Java variable means:   * The value of this variable will **never be cached thread-locally**: all reads and writes will go straight to "main memory"; * Access to the variable **acts as though it is enclosed in a**synchronized**block**, synchronized on itself.   59) What is race condition?  A race condition occurs when two or more threads can access shared data and they try to change it at the same time. Because the thread scheduling algorithm can swap between threads at any time, you don't know the order in which the threads will attempt to access the shared data. Therefore, the result of the change in data is dependent on the thread scheduling algorithm, i.e. both threads are "racing" to access/change the data.  60) What is dead lock?  Deadlock in java is a part of multithreading. Deadlock can occur in a situation when a thread is waiting for an object lock, that is acquired by another thread and second thread is waiting for an object lock that is acquired by first thread. Since, both threads are waiting for each other to release the lock, the condition is called deadlock.  Deadlock in java  61) What is Thread Local ?  The ThreadLocal class in Java enables you to create variables that can only be read and written by the same thread. Thus, even if two threads are executing the same code, and the code has a reference to aThreadLocal variable, then the two threads cannot see each other's ThreadLocal variables.  62) What is Thread Pool ?  A thread pool is a group of threads initially created that waits for jobs and executes them. The idea is to have the threads always existing, so that we won't have to pay overhead time for creating them every time. They are appropriate when we know there's a stream of jobs to process, even though there could be some time when there are no jobs.  63) How do you use Executor framework?  Executors framework (java.util.concurrent.Executor), released with the JDK 5 in package java.util.concurrent is used to run the Runnable objects without creating new threads every time and mostly re-using the already created threads.  64) What is starvation?  **Starvation** occurs when a thread is continually denied access to resources and as a result it is unable to make progress. This usually happens when greedy threads consume shared resources for long periods of time. When this happens for extended periods of time, the thread not getting enough CPU time or access to the resource will not be able to make enough progress leading to [**thread starvation**](http://docs.oracle.com/javase/tutorial/essential/concurrency/starvelive.html).  One of the likely causes of thread starvation is incorrect thread priorities among different threads or thread groups.  Another possible cause could be use of non-terminating loops (infinite loops) or waiting excessive amount of time on specific resources while holding on to critical locks required by other threads.  65) Synchronized methods vs Synchronized blocks?  The key difference is this: if you declare a method to be synchronized, then the entire body of the method becomes synchronized; if you use the synchronized block, however, then you can surround just the "critical section" of the method in the synchronized block, while leaving the rest of the method out of the block.  If the entire method is part of the critical section, then there effectively is no difference. If that is not the case, then you should use a synchronized block around just the critical section. The more statements you have in a synchronized block, the less overall parallelism you get, so you want to keep those to the minimum.  66) What is serialization and externalization?  **Serialization in java** is a mechanism of writing the state of an object into a byte stream. It is mainly used in Hibernate, RMI, JPA, EJB, JMS technologies. The reverse operation of serialization is called *deserialization*. The String class and all the wrapper classes implements *java.io.Serializable* interface by default.  Java class can be saved into byte stream using externalization. To get complete control on class persistence as byte stream, java provides Externalization. *java.io.Externalizable* interface declares two method i.e *writeExternal and readExternal.* A class which needs to be externalize, must implement Externalizable interface and has to define two method readExternal and writeExternal.  67) How do you avoid a variable particiapting in serialization ?  Using the transient keyword we can avoid a variable particiapting in serialization.  68)what is marker interface?  Marker interface is used as a tag to inform a message to the Java compiler so that it can add special behaviour to the class implementing it. Java marker interface has no members in it. Consider java.io.Serializable marker interface. It does not have any members defined it it. When a Java class is to be serialized, you should intimate the Java compiler in some way that there is a possibility of serializing this java class. In this scenario, marker interfaces are used. The java class which may be serialized has to implement the java.io.Serializable marker interface. In such way, we are intimating the java compiler.  69)What is enum in java?  An *enum type* is a special data type that enables for a variable to be a set of predefined constants. The variable must be equal to one of the values that have been predefined for it. Common examples include compass directions (values of NORTH, SOUTH, EAST, and WEST) and the days of the week.  Because they are constants, the names of an enum type's fields are in uppercase letters. | |
|  |
|  |