Multithreading Interview Questions

1. What is multitasking and what are the different types of multitasking?

- ➢ Executing several tasks simultaneously is the concept of multitasking.

  There are two types of multitasking
  - **Process Based Multitasking**
  - **Thread Based Multitasking**

<u>Process Based Multi-Tasking:</u>

- ➢ Executing several tasks simultaneously where each task is a separate independent process is called process based multitasking.
- ➢ Process based multitasking is best suitable at operating system level.

e.g:

While typing a java-program in the system we can listen an audio song in the system and at the same time we can download a file in the system, all these tasks can be performed simultaneously and independent of each other hence it is process based multitasking.

<u>Thread Based multitasking:</u>

- ➢ Executing several tasks simultaneously at the program level where each task is a separate independent part of the same program is called thread based multitasking and each independent part is called thread.

- Thread based multitasking is best suitable at programmatic level.

2. **What is the main purpose of multitasking?**

- Whether it is process based or thread based multitasking its main purpose is to improve the performance of the system by reducing the response time.

3. **Where multithreading is applicable?**

- The main important areas of multithreading are to develop multimedia graphics to develop animations to develop video games, to develop application servers.
- Compare to old languages developing multi-threaded environment is very easy because java provides support for mulita threading with its inbuilt reach api.

4. **In how many ways we can define a thread and which one is the best approach and why?**

- We can define a thread by extending Thread class and by implementing Runnable interface.
- Best way to define a thread is by implementing Runnable interface approach is recommended.
- If we use Thread class approach our class will always extend thread class hence we will miss inheritance advantage of any other class but in implementing runnable approach we can inherit other class as well and we don't miss the inheritance benefits.

5. **What is thread Scheduler?**

- It is the part of JVM it is responsible to schedule threads that is if multiple threads are waiting to get chance of execution then in which order threads will be executed is decided by thread scheduler.
- We can't expect the exact algorithm followed by thread scheduler it varies jvm to jvm hence we can't expect thread execution order and exact output hence whenever situation come to multithreading there is no guarantee of exact output but we can provide several possible outputs.

6. **Differentiate between t.start() and t.run() according to below program?**

```
public class Threads extends Thread {
	public void run() {
		for(int i=0;i<10;i++) {
			System.out.println(i);
		}
	}
	public static void main(String[] args) {
		Threads t=new Threads();
		t.run();
		t.start();
	}

}
```

- In the above program t.start() will create a new thread which is responsible for the execution of run() method but in the case of t.run() a new thread won't be created and run() method will be executed just like a normal method call.

7. **What is the importance of Thread class Start () method?**

- Thread class start() method is responsible to register the thread with thread scheduler and do all other mandatory activities, Hence without executing thread class start() method there is no chance of starting a new thread in java, Due to this thread class start method is considered as heart of multithreading.

8. **Overloading of run () method is possible or not? If possible then what happen if we call the Thread class start () method? which run() method will be called?**

- Over loading of run () method is always possible but the Thread class start () method will always invoke the no argument run () method and for the overloaded run () method we have to call it explicitly.

9. **If we don't override the run method and call the start () method then what will be the output we will get? Take the below case and explain?**

```
public class Threads1 extends Thread {

	public static void main(String[] args) {
		Threads1 t=new Threads1();
		t.start();
	}

}
```

- If we are not overriding run () method and call the start() method then Thread class run () method will be executed hence we can't get any output.

10.What will happen in the below example?

```
public class Threads1 extends Thread {
     public void start() {
          System.out.println("start method overridden");
     }
     public void run() {
          System.out.println("Waiting for execution");
     }
     public static void main(String[] args) {
          Threads1 t=new Threads1();
          t.start();
     }

}
```

- In the above example we will get the output as "`start method overridden`".
- If we override the start() method then our start() method will be executed like a normal method call and new thread won't be created.

11.Explain Thread lifecycle?

12.After starting a thread if we are trying to restart the thread then what will happen?

- After starting a thread if we try to restart the thread then we will get runtime exception saying illegal Thread state exception.

13.How we can get name of a thread?

- By using Thread.currentThread().getName() method we can get the name of a thread.

14.How we can set name of a thread?

- By using Thread.currentThread().setName() method we can set name of a thread.

15.What are Thread priorities in java?

- Every thread in java has some priority it may be default priority assigned by the jvm or the priority assigned by the programmer explicitly.
- The Min priority is 1 and the max priority is 10 and 5 is the norm priority.

16.How to get and set priority of a thread?

- By using setPriority() we can set the priority of a thread and by using getPriority() we can get the priority of a thread.

17.What is the default priority of any thread?

- The default priority only for the main thread is 5 but for all remaining threads the default priority will be inherited from parent to child.
- Whatever priority parent thread has the same priority will be there for child thread.

18.In how many ways we can prevent a thread execution?

- By using (yield, join and sleep) method we can stop a thread from execution.

19.What is the purpose of yield method?

- Yield method causes to pass current executing thread to give the chance for waiting thread of same priority, If there is no waiting thread or all waiting threads have low priority then same thread can continue its execution.
- If multiple threads are waiting with the same priority then which waiting thread will get the chance we can't expect it depends on thread scheduler.
- The thread which is yielded when it will get the chance once again it depends on thread scheduler and we can't expect exactly.

20.What is the complete prototype of yield method?

- **Public static native void yield();**

21.What is the purpose of join () method?

- If a thread wants to wait until completing some other thread then we should go for join method.
- For example if a thread t1 wants to wait until completing t2 then t1 has to call t2.join();

- If t1 executes t2.join () then immediately then t1 will be entered into waiting state until t2 completes, Once t2 completes then t1 can continue its execution.

22.Join() method throws which exception?

- Join () method throws interrupted exception which is a checked exception hence we have to handle it by using try,catch and throws keyword.

23.What will happen if the thread calls join () method on it self?

- If the thread calls join method on itself then the program will be stucked(This is something like dead lock), In this case thread has to wait infinite amount of time.

E.G:

```
Psvm() throws IE{

Thread.currentThread().join();

}
```

24.When we should use sleep() method?

- If a thread don't want to perform any operation for a particular amount of time then we should go for sleep() method.
- Every sleep method throws interrupted exception which is checked exception hence when ever we are using sleep() method compulsory we should handle interrupted exception either by try catch or by throws keyword otherwise we will get compile time error.

25.How a thread can interrupt another thread?

- A thread can interrupt a sleeping thread or waiting thread by using interrupt method of thread class.

Public void interrupt();

26.Difference between yield () join() and sleep() ?

Properties	Yield()	Join()	Sleep()
➢ **Purpose**	If a thread wants to pass its execution to give the chance for remaining threads of same priority then we should go for yield () method.	If a thread wants to wait until the execution of some other thread then we should go for join () method.	If a thread do not want to perform any operation for a particular amount of time then we should go for sleep () method.
➢ **Is it overloaded**	No	Yes	Yes
➢ **Is it final?**	No	Yes	No
➢ **Is it throws Interrupted Exception**	No	Yes	Yes
➢ **Is it native**	Yes	No	Sleep(long ms)//native Sleep(long ms,int ns)//non native

27.What is the biggest advantage of synchronized keyword?

- **Synchronized is a modifier applicable on methods and blocks but not on clsses.**
- If multiple threads are trying to operate simultaneously on the same java object then there may be a chance of data inconsistency problem.
- To overcome the problem we should go for synchronized keyword.

- If a method or a block declared as synchronized then only one thread is allowed to execute that method or block on the given object so that data inconsistency problem will be resolved.
- The main advantage of synchronized keyword is we can resolve data inconsistency problem.
- The main disadvantage of synchronized keyword is it increases waiting time of thread and creates performance issue.

28.Internal working of Synchronization?

- Internally synchronization concept is implemented by lock, Every object in java have a unique lock whenever we are using synchronized keyword then only lock concept will come into the picture.
- If a thread wants to execute synchronized method on the given object first it has to get lock of that object once thread got the lock then it is allowed to execute any synchronized method on that object.
- Once method execution completes automatically thread releases the lock.
- Acquiring and releasing lock internally takes care by jvm and programmer not responsible for this activity.
- While a thread executing synchronized method on the given object the remaining threads are not allowed to execute any synchronized method on same object but remaining threads are allowed to execute nonsynchronized method simultaneously.
- Lock concept is implemented based on object but not based on method.

29.Synchronization area and nonsynchronized area of an object.

30.When a thread need class level lock?

- If a thread wants to execute static synchronized method it required class level lock.
- Once thread got class level lock then it is allowed to execute any static synchronized method of that class.
- Once method execution completes automatically thread releases the lock.
- While a thread executing static synchronized method the remaining threads are not allowed to execute any static synchronized method of that class simultaneously but remaining threads are allowed to execute the following methods simultaneously.

- Normal static methods, Synchronized instance methods, Normal instance methods.

31.What is class level lock?

- Every class in java has a unique lock which is nothing but class level lock.

32.What is synchronized block and what is the advantage over synchronized method?

- If very few lines of code required synchronization then it is not recommended to declare entire method as synchronized we have to enclose those few lines of the code by using synchronized block.
- The main advantage of synchronized block over synchronized method is it reduces the waiting time of the thread and improves the performance of the application.

33.**How to declare synchronized block?**

- We can declare synchronized block as follows
  - **To get lock of current object:**

synchronized (this){

(If a thread got the lock of current object then it is allowed to execute this area).

}

  - **To get the lock of particular object b:**

synchronized (b) {

(If a thread got the lock of particular object (b) then it is allowed to execute this area).

}

  - **To get the class level lock:**

synchronized (Display.class) {

(If a thread got the class level lock of particular class (Display) then it is allowed to execute this area).

}

34.**In which areas lock concept is applicable?**

- Lock concept is applicable only on object types and class types but not for primitives hence we can't pass primitive type argument to synchronized block.

35.What is synchronized keyword where we can apply?

- Synchronized is a modifier applicable on methods and blocks.

36.Explain advantage of synchronized keyword?

- Synchronized keyword helps to get rid of the data inconsistency problem because if we declare a method or block synchronized then only one thread is allowed to execute that particular method or block on a given object.

37.Explain disadvantage of synchronized keyword?

- If multiple threads are trying to execute synchronized method or block on the same object then only one thread is allowed to execute at a time on the object because of lock concept and the remaining threads are in waiting state this increases our system response time and decrease the performance.

38.What is Race condition?

- If multiple threads are trying to execute simultaneously on the same object then there may be a chance of causing data inconsistency problem this is nothing but race condition.
- We can overcome this issue by using synchronized keyword.

39.What is object lock and when it is required?

40.What is class level lock and when it is required?

41.What is the difference between class level lock and object level lock?

42.While a thread is executing synchronized method on the given object is the remaining threads are allowed to execute any other synchronized method simultaneously on that object.

43.What is synchronized block?

44.How to declare synchronized block to get the lock of current object?

45.How to declare synchronized block to get class level lock?

46.What is the advantage of synchronized block over synchronized method?

47.Is a thread can acquire multiple locks simultaneously?

48.What is synchronized statement?

49.How the threads can communicate with each other?

- Two threads can communicate with each other by using wait,notify and notify all methods.
- The thread which is expecting updatation is responsible to call wait() method then immediately the thread will enter into waiting state.

- The thread which is responsible to perform updetation, after performing updetation the thread is responsible to call notify method then the waiting thread will get that notification and continue its execution with those updated items.

50.Wait(),notify() and notifyAll() method present in which class and why?

- These methods are present in object class but not in Thread class.
- Thread can call these methods on any java object that's why these are present in object class.

51.Where we can use wait(),notify() and notifyAll() methods?

- To call these methods on any object, Thread should be owner of that object that is the thread should has lock of that object that is the thread should be inside synchronized area.
- We can call these methods only from synchronized area other wise we will get runtime exception saying illegal monitor state exception.
- If a thread call wait method on any object it immediately releases the lock of that particular object and entered into waiting state.
- If a thread calls notify method on any object it releases lock of the of the object but may not immediately except wait notify and notify all there is no other method where thread releases the lock.

52.Difference between notify and notifyAll()?

Notify()	NotifyAll()
➢ We can use notify() method to give the notification for only one waiting thread.If multiple threads are waiting then only one thread will be notified and the remaining threads have to wait for further notifications. ➢ Which thread will be notified we can't expect it depends on jvm.	➢ We can use notify all to give the notification for all waiting threads of a particular object even though multiple threads notified but execution will performed one by one because threads need the lock and only one lock is available.

53.What is Deadlock?

- If two threads are waiting for each other for ever such type of infinite waiting is called deadlock.
- Synchronized keyword is the only reason for deadlock situetion.
- There are no resolution technique for deadlock but several prevention technique is available.

54.Differentiate between Deadlock and Starvation?

- Long waiting of a thread where waiting never ends is called deadlock.
- Long waiting of a thread where waiting ends at a certain point is called starvation.

55.What are demon threads?

- **The threads which are executing in the background are called demon threads.**
- **Garbage collector, Signal dispatcher, Attach listener etc.**
- **The main objective of demon thread is to provide support for non demon threads(main thread).**
- **For example if main thread runs with low memory then jvm calls garbage collector to destroy useless objects so that number of bytes of free memory will improve so that main thread will continue its execution.**

56.How can we check daemon nature of a thread?

- We can check daemon nature of a thread by using isDaemon() method of thread class.
- Public Boolean isDaemon().

57.How can we convert a normal thread into daemon thread?

- We can make a thread daemon thread by using setDaemon() method of thread class.
- Public void setDaemon(Boolean b).
- But changing daemon nature is possible before starting of a thread only.
- After starting a thread if we are trying to change the daemon nature then we will get illegal thread state exception.
- By default main thread is always nondaemon but for remaining threads daemon nature is inherited from parent to child i.e if the parent thread is daemon then automatically child thread is also daemon and if the parent thread is non daemon then automatically child thread is nondaemon.

58.**What is thread group?**

- **Based on functionality we can group threads into a single unit which is nothing but thread group.**
- **In addition to threads thread group can also contain sub thread group.**
- **The main advantage of maintaining threads in the form of thread group is we can perform common operations very easily.**

59.**How to get the thread group name for a particular thread?**

- **Thread.currentThread().getThreadGroup().getName().**

60.**Which thread group acts as root for every thread group?**

- **System is the root for every thread group.**

61.**What is thread pool?**

- Creating a new thread for every job may create performance and memory problem to overcome this we should go for thread pool.
- Thread pool is a pool of already created threads ready to do our job.
- Java 1.5 version introduces threadpool framework to implement thread pools.
- Thread pool framework also known as executor framework.
- We can create a thread pool as follows.
- ExecutorService service=Executors.newFixedThreadPool(3)
- We can submit a runnable job by using submit() method.
- service.submit(job);
- We can shutdown executor service by using shutdown()method
- Service.shutdown()

62.**What is callable and future?**

- In case of runnable job thread won't return anything after completing the job if a thread is required to return some result after execution then we should go for callable.
- Callable interface contains only one method call().
- Public object call() throws exception.
- If we submit callable object to executor then after completing the job thread returns an object of the type future i.e future object can be used to retrieve the result from callable job.

63.Differences between runnable and callable?

Runnable	Callable
➢ If a thread not required to return anything after completing the job then we should go for runnable. ➢ Runnable interface contains only one method run(). ➢ Return type is void. ➢ With in the run method there is a chance of rising checked exception so we need to handle by using try catch because we can't use throws keyword for run method. ➢ Present in lang package.	➢ If a thread required to return something after completing the job then we should go for callable. ➢ Callable interface contains only one method call(). ➢ Return type of call method is object. ➢ With in call method if any chance of rising checked exception we need not required to handle them using try and catch because call method already throws exception. ➢ Present in util.concurrant package.