**Multithreading**

Multithreading is a process of executing multiple thread simultaneously.

Thread is just a lightweight sub-process or the smallest unit or processing.

Multithreading and Multiprocessing both are used to achieve Multitasking.

**Multitasking**

Process-based Multitasking (Multiprocessing)

Thread-based Multitasking (Multithreading)

**Process-based Multitasking (Multiprocessing)**

Each process has separate memory area.

It is heavyweight.

**Thread-based Multitasking (Multithreading)**

Threads are shared same memory address space.

It is Lightweight.

**Which is better Multitasking or Multiprocessing**

Multitasking Thread based is better because context switching and intern thread communication is fast.

**What is Thread in Java?**

Thread is lightweight sub-process of task.

main() method is thread.

**Types of Thread**

Two type of thread User Threads and Daemon Thread.

Life Cycle of a Thread

**How to create a user thread in Java or Two ways to create a Thread**

Using extending Thread Class and Implementing Runnable Interface.

It is better to extend the Thread class because it will provide so many method to perform operations on thread.

And Runnable interface having only one method which run().

**Which is better extend or implementing?**

When we need to do multiple thing to extended go with implementation.

But if one class to do the go with extending.

**How to perform multiple tasks by multiple threads?**

Just create multiple Threads for one task and call start() method.

**Thread Scheduler**

Thread Scheduler is to decide which thread have to run or execute on the basis of priority of threads

Priority scales having 1 to 10 and if two thread having same priority then Time of Arrival of Thread is consider.

**Sleeping a thread**

Thread.sleep() method is used to halt the working of the thread for some given time.

**Can we start a thread twice?**

No we cant start a thread twice.

**What happens if we call the run() method instead of start() method?**

If we call directly run() method then it will treat as normal object method.

**Joining a thread**

Join() method is used to wait until the other thread have finish the execution.

When the join() method is invoke the current thread will stop the execution and goes into the wait state.

**Naming a thread** – we can set the name of thread using setName() method.

**Priority of a thread** – There is 1 to 10 of priority of thread which will decide thread scheduler by JVM.

**Daemon Thread**

Daemon thread is provide the service to user thread for background supporting task.

Its life depend on user thread.

JVM terminate daemon thread automatically after all threads will dies.

There is two methods for daemon thread

setDaemon() – is use to mark a current thread or user thread is demon thread.

isDaemon() – is use to check current thread is daemon.

**Java Thread Pool**

Java Thread Pool is represent the group of worker threads which will waiting for execution and reusability.

**Garbage collection**

In Java Garbage means unreferenced objects.

Garbage collection is the process of clean the unused object or unreferenced object in heap memory for reusing.

JVM will take care of this Garbage collection process for us.

**gc() method is use to invoke the garbage collection to perform cleanup processing.**

We can done object to unreferenced by below logics

1) By nulling a reference:

Employee e=new Employee();  
e=null;

2) By assigning a reference to another:  
Employee e1=new Employee();  
Employee e2=new Employee();  
e1=e2;//now the first object referred by e1 is available for garbage collection

3) By anonymous object:  
new Employee();

**Finalized() method**

It will invoke before the garbage collecting.

This method is use the cleanup process.

**Java Runtime Class – java.lang.Runtime**

This class is used to interaction with Java Runtime Enviormnewt using some method like getRuntime(), freeMemory(), totalMemory() etc.

Synchronization with synchronized method

Synchronized block

Static synchronization

Deadlock

Inter-thread communication