**178. What is multi programming.**

1. **Multiprogramming:** Running multiple programs on a single machine is called as multi programming.

2. Different forms of multiprogramming:

a. **Multi-user**: Multiple users will use same computer.

b**. Multi-tasking**: Single user will run multiple tasks.

i) Multithreading.

3. We can have system which will assign some more tasks in its idle time, So we can increase the throughput.

4. Unix and Linux Operating System are famous for multi-user functionality. The O.S schedules the program such that users feel that their programs are executing simultaneously.

5. Multi-user machines were most called as timesharing machines.

6. Today mostly we are using multitasking and multithreading.

**179. What is Multitasking.**

1.Multitasking means running more than one tasks simultaneously for one user.

2. In actuality the C.P.U executes program alternatively rather executing it simultaneously. The C.P.U can only run single program at a time. But as it is executing the programs alternatively on a very fast pace it seems that the programs are executing in a simultaneous manner.

3. Example of multi-tasking operating system: Windows, Linux, MaxOS.

4. **Multithreading:** Multiple threads running simultaneously is called as multithreading.Threads are a part of the operating system. They are light weighted compared to actual programs and processes.

For example: In an application threads which play video, have runtime chat, have runtime advertisements and other things.

5. Main is the starting point of a program.

6. Multithreading is used in games, animation , webservers and etc.

7. Thread is the smallest part of the process that allows a program to operate more efficiently by running multiple tasks simultaneously. It is a light-weighted process.

8. We use threads to perform the complicated tasks in the background.

**180. Control Flow of a Program.**

1.One program will basically have one control flow.

2. Control flow is the sequence in which the program will be executed.

3. We can execute multiple threads simultaneously.

4. Program cannot have two main methods.

**181. Multithreading using Thread Class.**

1.Following are the classes available to achieve multithreading in java.

a. Thread class: It contains the actual mechanism for multithreading. Without this class we cannot achieve multithreading. It is mandatory.

b. Runnable Interface:

2. There are two ways to achieve multithreading:

a. Extend the thread class.

b. If you are inheriting other class then implement Runnable Interface.

3. Whenever you extends the Thread class then you must override the run() method (Signature of the method is public void run())to achieve the multithreading.

4. run() method is a starting point of a thread just like the main is a starting point of a program.

5. To start the thread, we must use the start() method. Start method is a built-in method of thread class. So to start a thread you have to create an object of the class extending the Thread class and then call the start the start() method of the Thread class.

6. In multithreading multiple control flows gets created.

7. The Thread and Main executed in a random manner but simultaneously. Sometimes the thread is executed sometimes main without a proper order. It is executed as per allocation of time. Every time the order of execution changes.

**182. Multithreading using Runnable Interface.**

1.There is only on method in an interface which is **public void run();**

2. When we implement the Runnable interface on a class then it makes that class capable of running as a thread by it cannot run by itself. To run this class, we need the object of Thread class as it have all the mechanism to achieve multithreading.

3. The object of implemented class is pass as an argument to the constructor of Thread class.

4. It is just like the Thread is horse and Runnable implemented class is a cabin that is attach to horse.

**183. Demo: Multithreading using Thread and Runnable.**

1. In case of Runnable interface implementation we have to create the Thread object and then pass the object of a class which implements the Runnable Interface.

2. Thread class’s object is mandatory to achieve multithreading.

**184. States of a Thread.**

1. There are different states of Thread.

a. New

b. ready/ runable (start())

c. Running (run())

d. Terminate

e. Waiting (wait())(Thread is trying to acquire some resources.)

f. Timed Wait (sleep()) (Intentionally delaying the Thread, you have to use sleep).

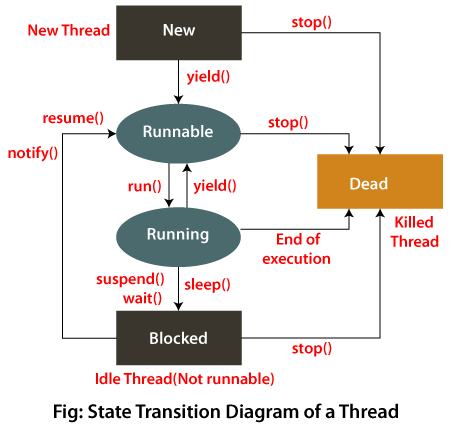
g. Blocked.

2. Once the thread started and terminated you cannot restart the thread. You have to start a new thread.

3. If two Threads (Thread1 and Thread2) are trying to access the same resource then one Thread (Thread1) should access the resource and one (Thread2) should wait. After Thread1 complete its use of resource it should notify the Thread2.

4. If Thread is waiting then Thread is in the waiting state. Once the resource gets free the Thread will be notified of it.

5. The thread is requesting a resource. It doesn’t want anyone to access that thread so it put the lock (Monitor). Thats when we can say that the Thread is in blocked state.



**185. Thread priorities.**

1. JVM maintains the ready queue or there is a component in scheduler which maintains the ready queue.

2. The C.P.U can maintain only one program at a time.

3. Scheduler decides time to be allocated to every thread in case of Round Robin fashion of time sharing.

4. Java supports priority from 1 to 10.

5. The threads can executed as per their priorites.

6. Thread.MIN\_PRIORITY = 1;

Thread.NORM \_PRIORITY = 5;

Thread.MAX \_PRIORITY = 10;

7. The thread having higher priority will get more C.P.U time or it will complete first.

8. Java has its own of multithreading environment. It can take help from Operating System.

9. JVM or JRE is having its own scheduler. Java Threads are not scheduled by Operating System but JVM.

**186. Thread Class**

1. Whenever you are learning about any class, you should know about its constructor, get-set methods, and other commonly used methods.

2. Constructor of Thread Class:

a. Thread()

b. Thread(Runnable r)

c. Thread(Runnable r, String name)

d. Thread(ThreadGroup g, String name)

e. Thread(String name)

3. We can make group of the threads. You can create object ThreadGroup class and give all the threads to it.

4. Methods of Thread Class:

a. long getId(); //Id of thread

b. String getName(); //Name of thread

c. int getPriority(); //Gets current priority of thread.

d. ThreadState getState(); //Gets state of thread.

e. ThreadGroup getThreadGroup(); // Gives the reference of the thread group to which the thread belongs.

f. void setName(String name); //To set the thread name.

g. void setPriority(int p); // To set the thread priority.

h. void setDaemon(boolean d); //To set thread as daemon set or not.

5. Daemon thread is a background thread with least priority.

6. Garbage collector in Java is daemon thread. The autosave in MS Word is daemon thread.

7. Enquiry Methods of Thread Class:

a. boolean isAlive();

b. boolean isDaemon();

c. boolean isInterrupted(); //Checks whether the thread was interrupted or not.

8. There is interrup method in java which tells the thread to stop the ongoing work or terminate.

9. The thread can be interrupted by some other thread or it can interrupt itself.

10. Instance Methods:

a. void interrupt(); // Use to interrupt the thread. Mostly done while thread is waiting or sleeping.

b. void join(); //Use to delay the termination of thread so it can terminate with other threads.

c. void join(long milis);

d. void run(); //Use to run the thread. It contains the actual functionality of thread.

e. void start(); // Use to start and get the thread in a ready queue (Runnable state).

11. Static Methods:

a. int activeCount(); //It gives the count of active thread in group.

b. Thread currentThread(); //Returns reference to the current thread.

c. void yield(); //This method is used to yield the high priority thread so that low priority thread will get resource and won’t starve.

d. void dumpStack(); //It used to get stack of method calls(sequence of method calls).

**187. Thread Methods: Constructors, sleep and Interrupt.**

1. Thread(“My Thread 1”): This constructor is used to name the thread. It is different from class name.

2. We usually set the priority like setPriority(MIN\_PRIORITY +2); We don’t give the direct values.

3. Thread.sleep() method throws RuntimeException.

4. We can interrupt the thread while it is sleeping.