# Multi-Threading

1) What do you mean by Multithreading? why is it important?

Ans) Multithreading is a programming technique that allows multiple threads to execute concurrently within a single process.
Multithreading is important for several reasons, it can improve the performance of a program by allowing it to take advantage of multi-core processors & execute multiple tasks simultaneously.

2) What are benefits of using Multithreading?

Ans) - Allow the program to run continuously even if a part of it is blocked.
- Improve performance
- Improve responsiveness of complex applications
- Save time & parallelism tasks.

3) What is Thread in java?

Ans) Threads are basically the lightweight and smallest unit of processing that can be managed independently by a scheduler.

4) What are 2 ways of implementing thread?

Ans) by Extending Thread class

```
class MultithreadingDemo extends Thread {
    public void run()
    { S.o.p ("running state."); }
```

ii) Implementing Runnable interface :-

```
class MultithreadingDemo implements Runnable {
    ...
}
```

5) Difference b/w thread and process?

Ans) Thread

it simply refers to the smallest units of the particular process. It has the ability to execute different parts of the program at the same time.

Process

it simply refers to a program that is in execution i.e, an active program. A process can be handled using PCB (process control block).

6) How can we create daemon threads?

Ans) by using the thread class setDaemon(true). It is used to mark the current thread as daemon thread or user thread. isDaemon() method is generally used to check whether the current thread is daemon or not, if thread is daemon, it will return true otherwise false.

7) What are wait() & sleep() methods?

Ans) wait()

As name suggests, it is a non-static method that causes the current thread to wait & go to sleep until some other threads call the notify() method for the object's monitor (lock).

Eg

```
Synchronized (monitor) {
    monitor.wait();
}
```

Sleep()

It is a static method that pauses or stops the execution of the current threads for some specified period.

Eg

```
synchronized (monitor) {
    Thread.sleep (1000);
}
```