***MultiThreding***

**1.What do you mean by Multithreding ?Why is it important?**

->Multithreading means Multiple threads and is considerd one of the most important features of Java. As the name suggest’s, It is a ability of CPU to execute a multiple threads independently at the same time but share the process resources simultaneously.Is main purpose is to provides simultaneous execution of multiple thread to utilize cpu time as much as possible .It is a java feature where one can subdivide the specific program into two or more threads to make the execution of program fast and easy.

**2.What are the benefits of using Multithreading ?**

->There are various benefits of multithreading as given below:

1. the program to run continuously even if a part of its blocked.

2.Improve performance as compared to traditional parallel programs that use multiple processes.

3.Allow to write effective programs that utilizes maximum CPU time .

4.Improves the responsiveness of complex applications or programs.

5.Increase use of CPU resources and reduce cost of maintainance.

6.Save time and parallelism task.

7.If an exception occurs in a single thread,it will not affected other threads as threads are independent.

8.Less resource-intensive than executing multiple processes same time.

**3.What is Thread in Java ?**

->Threads are basically the light weight and smallest unit of processing that can be managed independently by a scheduler.Threads are refer to as part of process that simply let a program execute efficiently with other parts or threads of the process at the same time. Using threads,one can perform complicated tasks in the easiest way .It is considerd a simplest way to take advantage of multiple CPUs available in a machine. They share the comman address space and are independent of each other .

**4.What are the Two ways of implementing thread in Java?**

->There are basically two ways of implementing thread in Java as given below:

Extending the thread class

Example:

Class Demo extends Thread {

Public void run(){

System.out.println(“My thread is in running state ”);

}

Public static vaid main(String [] args){

Demo d=new Demo();

d.start();

}

}

//Output: My thread is in running state

\*Implementing Runnable interface in Java

class Demo2 implements Runnable{

public void run(){

System.out.println(“MY Thread is in running State”);

}

}

Public static void main(String [] args){

Demo2 d2=new Demo2();

Thread t2=new Thread(d2);

t2.start();

}

}//My Thread is in running state .

**5.What’s the difference between threads and process?**

->Threads:its simply refers to the smallest unit of the particular process.It has the ability to executes the different parts(refered to as thread) of the program of 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?**

->We can create daemon threads in java using the thread class setDaemon(true).It is use to mark the current thread as daemon thread or user thread.isDaemon() method is generally check to use whether the current thread is daemon or not . If the thread is daemon , It will written true otherwise False.

Example:Program to illustrate setDaemon and isDaemon() method .

Public class DaemonThread extends Thread{

Public DaemonThread(String name){

Super(name);

}

Public void run(){

If(Thread.currentThread().isDaemon())

{

System.out.println(getName()+”Is Daemon thread”);

}

else{

System.out.println(getName()+”Is user Thread”);

}

}

public static void main(String[] args)

{

DaemonThread t1 = new DaemonThread("t1");

DaemonThread t2 = new DaemonThread("t2");

DaemonThread t3 = new DaemonThread("t3");

// Setting user thread t1 to Daemon

t1.setDaemon(true);

// starting first 2 threads

t1.start();

t2.start();

// Setting user thread t3 to Daemon

t3.setDaemon(true);

t3.start();

}

}//Output:

t1 is Daemon thread

t3 is Daemon thread

t2 is User thread

**7. What are the wait() and sleep() methods?**

Ans: wait(): As the name suggests, it is a non-static method that causes the current thread to wait and go to

sleep until some other threads call the notify () or notifyAll() method for the object’s monitor (lock). It simply

releases the lock and is mostly used for inter-thread communication. It is defined in the object class, and

should only be called from a synchronized context.

Example:

sleep(): As the name suggests, it is a static method that pauses or stops the execution of the current thread for

some specified period. It doesn’t release the lock while waiting and is mostly used to introduce pause on

execution. It is defined in thread class, and no need to call from a synchronized context.

Example:

synchronized(monitor)

{

monitor.wait(); Here Lock Is Released by Current Thread

}

synchronized(monitor)

{

Thread.sleep(1000); Here Lock Is Held by The Current Thread

//after 1000 milliseconds, the current thread will wake up, or after we call that is

interrupt() method