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

Ans: Multithreading means multiple thread sandis considere doneo$ the most important $ eatureso $ Java. As

thenamesuggests,itistheabilityo$aCP=toexecutemultiplethreadsindependentlyatthesametimeb ut

sharetheprocessresourcessimultaneously.Itsmainpurposeistoprovidesimultaneousexecutiono$ multiple

threadstoutilizetheCP=timeasmuchaspossible.ItisaJava$eaturewhereonecansubdividethespeci $ic

programintotwoormorethreadstomaketheexecutiono$theprogram$astandeasy.

2. What are the benefits of using Multithreading?

Ans: Therearevariousbene$itso$multithreadingasgivenbelowv

Allowtheprogramtoruncontinuouslyeveni$aparto$itisblocked.

Improveper$ormanceascomparedtotraditionalparallelprogramsthatusemultipleprocesses.

Allowstowritee$$ectiveprogramsthatutilizemaximumCP=tim<

Improvestheresponsivenesso$complexapplicationsorprograms.

Increaseuseo$CP=resourcesandreducecostso$maintenance.

Savestimeandparallelismtasks.

I$anexceptionoccursinasinglethread,itwillnota$$ectotherthreadsasthreadsareindependent.

Lessresource-intensivethanexecutingmultipleprocessesatthesametime.

3. What is Thread in Java?

Ans:Threadsarebasicallythelightweightandsmallestunito$processingthatcanbemanagedindepe ndently

byascheduler.Threadsarere$erredtoaspartso$aprocessthatsimplyletaprogramexecutee$$icientl ywith

otherpartsorthreadso$theprocessatthesametime.=singthreads,onecanper$ormcomplicatedtasks in

theeasiestway.Itisconsideredthesimplestwaytotakeadvantageo$multipleCP=savailableinamac hine.

Theysharethecommonaddressspaceandareindependento$eachother.

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

Ans: There are basically two way so $implementing threading java given below

Extending the Thread class

Example:

Mythreadisinrunningstate.

```
class MultithreadingDemo extends Thread
{
public void run()
{
System.out.println("My thread is in running state.");
}
public static void main(String args[])
{
MultithreadingDemo obj=new MultithreadingDemo();
obj.start();
}
}
```

Output:

Assignment Questions

Implementing Runnable interface in Java

Example:

My thread is in running state.

5. What's the difference between thread and process?

Ans: Thread: It simply refers to the smallest units of the particular process. It has the ability to execute different

parts (referred to as thread) 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: We can create daemon threads in java 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 the thread is a daemon, it will return true otherwise it returns false.

Example:

```
class MultithreadingDemo implements Runnable
{
public void run()
{
System.out.println("My thread is in running state.");
}
public static void main(String args[])
{
MultithreadingDemo obj=new MultithreadingDemo();
Thread Obj =new Thread(obj);
tobj.start();
}
}
```

Program to illustrate the use of setDaemon() and isDaemon() method.

```
public class DaemonThread extends Thread
{
public DaemonThread(String name){
super(name);
}
public void run()
{
// Checking whether the thread is Daemon or not
```

```
if(Thread.currentThread().isDaemon())
{
System.out.println(getName() + " is Daemon thread");
}
else
{
System.out.println(getName() + " is User thread");
}
}
```

Output:

Assignment Questions

```
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