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

**Ans.** It is the ability of a CPU to execute multiple threads independently at the same time but share the process resources simultaneously.

It makes the execution of the program fast and easy.

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

**Ans.**

1. Allow the program run continuously even if a part of it is blocked.
2. Utilize maximum CPU Time.
3. Improves the responsiveness of the complex application of the programs.
4. Increase the use of CPU resources and reduce costs of maintenance.
5. **What is Thread in Java?**

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

1. **What are the two way of implementing thread in Java?**
2. **Extending the thread class**

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();

     }

   }

1. **Implementing Runnable interface in Java**

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();

    }

    }

1. **What’s the difference between thread and process?**

**Ans.** Thread: It simply refers to the smallest units of the particular process.

Process: It simply refers to a program that is in execution.

1. **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.

1. **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).

Example:

synchronized(monitor)  {

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

   }

**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)  {

  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

  }