**MultiThreading-Assignment**

Question 1: -**What do you mean by Multithreading? Why is it important?**

Ans: -Multithreading in Java is a process of executing multiple threads simultaneously . A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking . Its main purpose is to provide simultaneous execution of multiple threads to utilize the CPU time as much as posible.It is the important Java feature where one can subdivide the specific program into two or more threads to make the execution of the program easy and fast.

Question 2: -**What are the benefits of using Multithreading?**

Ans: - There are various benefits of multithreading as given below.

1.Allow the program to run continuously even if a part of it is blocked.

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

3.Allow to write effective code that utilises minimum CPU time.

4.Improves the responsiveness of complex applications or programs.

5.Saves time and parallelism tasks.

6.Less resource-intensive than executing multiple processes at the same time.

Question 3: -**What is Thread in Java?**

Ans: - A thread is a lightweight sub-process, the smallest unit of processing.Thread is a line of execution in a program. Thread in java helps us to achieve multiprogramming where a program or process can operate more efficiently by executing more than one instruction at a time. A thread in java also helps a complicated or larger task to operate in the background without interrupting the main program.

Question 4: -**What are the two ways of implementing thread in Java?**

Ans: -There are basically two ways of implementing thread in java as given below.

1.**Extending Thread class.**

**package com.practical.PWSkill;**

**class thread1 extends Thread {**

**@Override**

**public void run() {**

**try {**

**for (int i = 0; i < 10; i++) {**

**System.*out*.println("hello first thread");**

**Thread.*sleep*(2000);**

**}**

**} catch (InterruptedException e) {**

**System.*out*.println(e.getMessage());**

**}**

**}**

**}**

**class thread2 extends Thread {**

**@Override**

**public void run() {**

**for (int i = 0; i < 10; i++) {**

**System.*out*.println("hello second thread");**

**}**

**}**

**}**

**public class thread\_extends {**

**public static void main(String[] args) {**

**System.*out*.println("main thread");**

**String thread\_name = Thread.*currentThread*().getName();**

**System.*out*.println(thread\_name);**

**System.*out*.println(Thread.*currentThread*().getPriority());**

**System.*out*.println("after changing......");**

**Thread t = Thread.*currentThread*();**

**t.setName("this is my thread");**

**t.setPriority(1);**

**String changename\_thread = Thread.*currentThread*().getName();**

**System.*out*.println(changename\_thread);**

**System.*out*.println("after changing priority:" + Thread.*currentThread*().getPriority());**

**System.*out*.println(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>");**

**thread1 t1 = new thread1();**

**t1.start();**

**thread2 t2 = new thread2();**

**t2.start();**

**}**

**}**

1.**Implementing Runnable Interface.**

**package com.practical.PWSkill;**

**class first\_thread implements Runnable {**

**@Override**

**public void run() {**

**for (int i = 0; i <= 10; i++) {**

**System.*out*.println("hello thread1 ..........");**

**}**

**}**

**}**

**class second\_thread implements Runnable {**

**public void run() {**

**for (int i = 0; i <= 10; i++) {**

**System.*out*.println("hello thread2 ..........");**

**}**

**}**

**}**

**public class thread\_impliment{**

**public static void main(String[] args) {**

**System.*out*.println("main thread....");**

**first\_thread t1 = new first\_thread();**

**second\_thread t2 = new second\_thread();**

**Thread t11 = new Thread(t1);**

**Thread t22 = new Thread(t2);**

**t11.start();**

**t22.start();**

**}**

**}**

Question 5: -**What's the difference between thread and process?**

Ans : -

| **S.NO** | **Process** | **Thread** |
| --- | --- | --- |
| **1.** | Process means any program is in execution. | Thread means a segment of a process. |
| **2.** | The process takes more time to terminate and takes more time for creation. | The thread takes less time to terminate and takes less time for creation. |
| **3.** | The process is called the heavyweight process. | A Thread is lightweight as each thread in a process shares code, data, and resources. |
| **4.** | It also takes more time for context switching. | It takes less time for context switching. |

Question 6: -**How can we create daemon threads?**

Ans: -We can create daemon threads in java using Thread class setDaemon(True).It is used to mark the current thread as daemon thread or user thread. isDaemon() method generally used to check whether the current thread is daemon or not.If the thread is a Daemon ,it will return true otherwise return false.

**Example: -**

**package com.practical.PWSkill;**

**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");**

**}**

**}**

**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();**

**}**

**}**

Question 7: -**What are the wait() and sleep() methods?**

Ans : -

Wait(): -As we know ,it is a non-static method that causes the current thread to wait and go to sleep until at that time other thread 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 the object class and should only be called from a synchronized context.

Example: -

synchronized(monitor)

{

monitor.wait();

Here Lock Is Released by Current Thread

}

Sleep(): -As we know ,it is static method that pauses or stops the execution of current thread for some specific task.It does not releases the lock while waiting and is mostly used to introduce pauses on execution.It is defined is thread class and no need to call from synchronized context.

Example: -

synchronized(this)

{

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

}