Weekly Assignment 1

Answer 1)

|  |  |  |
| --- | --- | --- |
| Sr No. | Process | Threads |
| 1. | When a program is in execution that is known as Process | Inside a process execution of small sequences is known as thread |
| 2. | A Process takes more time to terminate | Thread terminates quickly |
| 3. | The process is independent by nature | Threads are the subset of Processes they depend on the process |
| 4. | The process contains all the instructions so it could follow all the instruction step by step. | As threads are small and individual execution, they do not store much information about execution steps but they use process state and resource to follow the further steps. |
| 5. | The process uses a desecrate range of memory address. | Threads shear their address space. |
| 6. | Context Switching is slow in the process.  In context switching the state of execution is stored so it could restore and resumed from the last point of execution. | Threads take less time for context switching and quickly execute from the last point which has been stored. |
| 7. | In inter-process communication, it takes too much time to communication between two processes  In multiple processes, if one process is blocked the remaining process continues executing | The thread can communicate quickly.  If a single thread gets blocked its all related threads would block too. |

Answer 2)

**Thread scheduler** in java is the part of the JVM that decides which thread should run.There is no guarantee that which runnable thread will be chosen to run by the thread scheduler.Only one thread at a time can run in a single process. The thread scheduler mainly uses preemptive or time slicing scheduling to schedule the threads.

Answer 3)

Synchronization in java is the capability to control the access of multiple threads to any shared resource.Java Synchronization is better option where we want to allow only one thread to access the shared resource.

Achieving Synchronization in multithreading:-

 Just a single specific thread can get to the resource at a given purpose of time.

class Multithread

{

public void printCount()

{

try

{

for(int i = 5; i > 0; i--)

{

System.out.println("Counter --- " + i );

}

} catch (Exception e)

{

System.out.println("Thread interrupted.");

}

}

}

class Thread extends Multithread

{

private Thread t;

private String threadName;

Multithread MT;

Thread( String name, Multithread mt)

{

threadName = name;

MT= mt;

}

public void run()

{

synchronized(MT)

{

MT.printCount();

}

System.out.println("Thread " + threadName + " exiting.");

}

public void start ()

{

System.out.println("Starting " + threadName );

if (t == null)

{

t = new Thread (this, threadName);

t.start ();

}

}

}

public class TestThread

{

public static void main(String args[])

{

Multithread MT = new Multithread();

Thread T = new Thread( "Thread - 1 ", MT);

Thread T1 = new Thread( "Thread - 2 ", MT);

T.start();

T1.start(); // wait for threads to end

try

{

T.join();

T1.join();

} catch ( Exception e)

{

System.out.println("Interrupted");

}

}

}

Answer 4)

Daemon thread is a low priority thread that runs in background to perform tasks such as garbage collection. **Daemon thread in java** is a service provider thread that provides services to the user thread. Its life depend on the mercy of user threads i.e. when all the user threads dies, JVM terminates this thread automatically.

**Properties:**

* They can not prevent the JVM from exiting when all the user threads finish their execution.
* JVM terminates itself when all user threads finish their execution
* If JVM finds running daemon thread, it terminates the thread and after that shutdown itself. JVM does not care whether Daemon thread is running or not.
* It is an utmost low priority thread.