**Multithreading in Java**

**Why multithreading?**

Before I begin the discussion about multithreading, I’ll say a few words about threads. A thread is the smallest part of a process that can run concurrently with the other parts (other threads) of the same process. Threads are independent because they all have separate path of execution. That is the reason of why an exception doesn’t affect the execution of other threads if it occurs. All threads of a process share the common memory. **The process of executing multiple threads simultaneously is known as multithreading.**

**The main purpose of multithreading is to provide simultaneous execution of two or more parts of a program to maximum utilize the CPU time. A multithreaded program contains two or more parts that can run concurrently. Each of those parts is called thread/**

**Threads are lightweight sub-processes. They share the common memory space. In a multithreaded environment programs that are benefited from multithreading, utilize the maximum CPU time so that the idle time can be kept to minimum.**

**A thread can be in one of the following states:**

**NEW – A thread created but not yet started;**

**RUNNABLE – A thread executing its task in the JVM;**

**BLOCKED – A thread waiting for a monitor lock;**

**WAITING – A thread that is waiting for an unknown period of time for another thread to perform a particular task;**

**TIMED\_WAITING – A thread that is waiting for another thread to perform its task for a specified period;**

**TERMINATED – A thread that has exited is in this state.**

**Multitasking vs Multithreading vs Multiprocessing vs parallel processing**

**For somebody who has never used JAVA, those terms might be confusing though they are used frequently in a multithreading discussion. Here is a short description for every term.**

* Multitasking: Ability to execute more than one task at the same time.
* Multithreading: Process of executing multiple threads simultaneously also known as Thread-based multitasking.
* Multiprocessing: Same as multitasking, however in multiprocessing more than one CPU are involved. On the other hand, multitasking uses one CPU.
* Parallel Processing: It refers to the usage of multiple CPUs in a single computer system

**Creating a thread in Java**

There are two simple ways to create a thread in Java:

1. By extending Thread class;
2. By implementing Runnable interface

Multithreading introduces asynchronous behavior to the programs. If a thread is writing some data another thread may be reading the same data at that time. This may bring inconsistency. When two or more threads need access to a shared resource there should be some way that the resource will be used only by one resource at a time. The process to achieve this is called synchronization. To implement the synchronous behavior, Java has synchronized methods. Once a thread is inside a synchronized method, no other thread can call any other synchronized method on the same object. All the other threads then wait until the first thread come out of the synchronized block.