Spring Chitkara University Live [SCUL] 📚⚽

YEAH !

Multithreading in JAVA

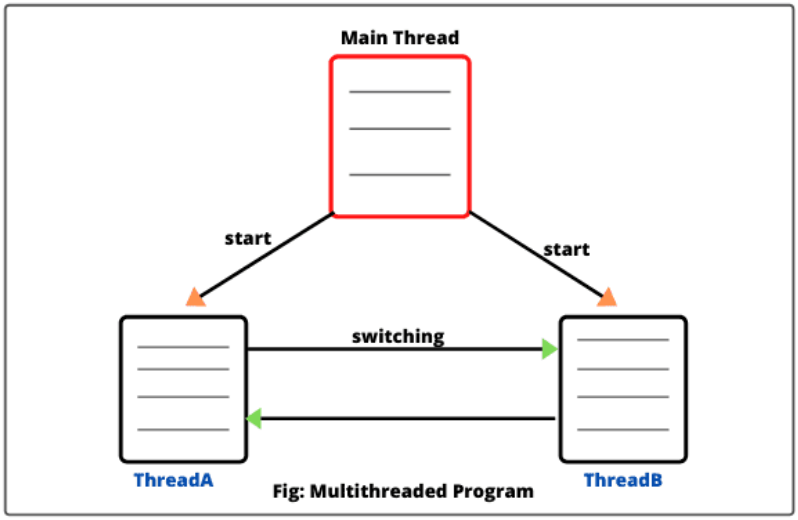


Fig: Multithreaded Program

What is a Thread?

A thread is a path of execution within a process. A process can contain multiple threads.

Why Multithreading?

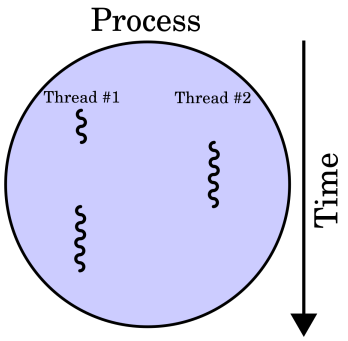
A thread is also known as lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc. More advantages of multithreading are discussed below

Process vs Thread?

The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces.

Threads are not independent of one another like processes are, and as a result threads share with other threads their code section, data section, and OS

resources (like open files and signals). But, like process, a thread has its own program counter (PC), register set, and stack space.



Advantages of Thread over Process

1. Responsiveness: If the process is divided into multiple threads, if one thread completes its execution, then its output can be immediately returned.

2. Faster context switch: Context switch time between threads is lower compared to process context switch. Process context switching requires more overhead from the CPU.

3. Effective utilization of multiprocessor system: If we have multiple threads in a single process, then we can schedule multiple threads on multiple processor. This will make process execution faster.

4. Resource sharing: Resources like code, data, and files can be shared among all threads within a process.

Note: stack and registers can't be shared among the threads. Each thread has its own stack and registers.

5. Communication: Communication between multiple threads is easier, as the threads shares common address space. while in process we have to follow some specific communication technique for communication between two process.

6. Enhanced throughput of the system: If a process is divided into multiple threads, and each thread function is considered as one job, then the number of

jobs completed per unit of time is increased, thus increasing the throughput of the system.

Types of Threads

There are two types of threads.

User Level Thread

Kernel Level Thread

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

Threads can be created by using two mechanisms :

1. Extending the Thread class
2. Implementing the Runnable Interface

Thread creation by extending the Thread class
We create a class that extends the java.lang.Thread class. This class overrides the run() method available in the Thread class. A thread begins its life inside run() method. We create an object of our new class and call start() method to start the execution of a thread. Start() invokes the run() method on the Thread object.

Thread creation by implementing the Runnable Interface
We create a new class which implements java.lang.Runnable interface and override run() method. Then we instantiate a Thread object and call start() method on this object.

Thread Class vs Runnable Interface

Spring Chitkara University Live [SCUL] 📚⚽

1. If we extend the Thread class, our class cannot extend any other class because Java doesn't support multiple inheritance. But, if we implement the Runnable interface, our class can still extend other base classes.
2. We can achieve basic functionality of a thread by extending Thread class because it provides some inbuilt methods like yield(), interrupt() etc. that are not available in Runnable interface.
3. Using runnable will give you an object that can be shared amongst multiple threads.

Concurrency : multiple processes scheduled one after other . Still at one instance of time , only one process is running in the processor .

Parallelism : multiple processes run at same time . Number of threads = No. of CPU * No of cores in one CPU .

Learning Resource : https://www.baeldung.com/java-concurrency

HOW TO CREATE THREAD ?

1. Extending java.lang.thread
2. Implement Runnable Interface

Understand that you can not use Multithreading in dependent code ie WHICH PART OF YOUR CODE CAN BE MULTITHREADED ?

Ques: Print the sum of first n natural numbers . Can we use multithreading in this scenario ?

Note : We can use multithreading for calling the function with different values .The function call operation will happen separately by different threads .

Spring Chitkara University Live [SCUL] 📚⚽

Examples are CPU intensive tasks like : pow (200000,3333333330 and pow(298088098,55555888) . These can be done in parallel using multithreading .

Ques: We have to Connect our app to db . Then we insert some data in db .
Can we use multithreading in this scenario ?

Ques: Insert ten objects in the database . All objects are separate and not dependent . We can use one different thread for each object . Can we use multithreading in this scenario ?

Daemon thread is background thread .

Lets jump to code

Create a public class CustomThread . Extend the Thread class . Please override the run function of the Thread class .We want the the data about the currentThread.().getName() in the run function . Now in main , call the run function .

Is this Multithreading ?
NO , lets see why.

Now lets call the start function instead of run .

When we do thread.start() , it creates a new callstack . Main thread is also running and Thread0 is also running .**Is this Multithreading ?**
Yes , lets see why.

A CPU can have multiple cores. A processor with two cores is called a dual-core processor; with four cores, a quad-core; six cores, hexa-core; eight cores, octa-core

If there is only one core in your PC , can you in any case achieve MT ?
NO , because 2 parallel threads cant run .

Can I create and start infinite threads ?

```
for(int i=0;i<10;i++){
    CustomThread th = new CustomThread();
    th.start();
}
```

NO , lets understand the concept of threadpool waiting .

```
group.add(this);
```

Spring Chitkara University Live [SCUL] 📚⚽

Lets find out where is the definition of the start function . Go to parent Thread class we are extending and lets see all the functions here . (Command + Fn+ F12)

Can we start a thread multiple times ?
thread.start();
thread.start();
Because Thread status changes and thus it gives exception .

```
Exception in thread "main" java.lang.IllegalThreadStateException
        at java.base/java.lang.Thread.start(Thread.java:791)
        at CustomThread.main(CustomThread.java:13)
```

Now , lets have a look at the Thread parent class and see how it is written .

Ques : Can we modify the threadStatus in our own custom thread class using overriding or any other way ?
NO , threadStatus is a private variable and there are no getters and setters provided by Java .

In the next class we will see the Second way of creating a thread by implementing the Runnable interface .

REFERENCES :

Thread Lifecycle https://www.geeksforgeeks.org/lifecycle-and-states-of-a-thread-in-java/
Inner class and Static Class https://www.geeksforgeeks.org/static-class-in-java/
Main Thread https://www.geeksforgeeks.org/main-thread-java/
Achieving Concurrency
https://www.geeksforgeeks.org/java-concurrency-yield-sleep-and-join-methods/
Inter Thread Communication
https://www.geeksforgeeks.org/inter-thread-communication-java/

Note : All code files , notes and download links have been uploaded to Github .
Please make sure to download Java in your systems , IntelliJ as IDE and Maven in your systems as build automation tool .