**MULTITHREADING**

**Multithreading in Java**

* Executing multiple threads simultaneously.
* Thread is **smallest** **unit** of processing.
* Used for implementing multitasking.

**Advantages of Multithreading**

* It can perform multiple tasks together.

**Types of Threads**

* **User thread:-**
  + **High** priority thread.
  + JVM exits only after it is completed.
* **Daemon thread:-**
  + **Low** priority thread.
  + Ends as soon as user thread terminates.
  + Used in background for garbage collection like processes.

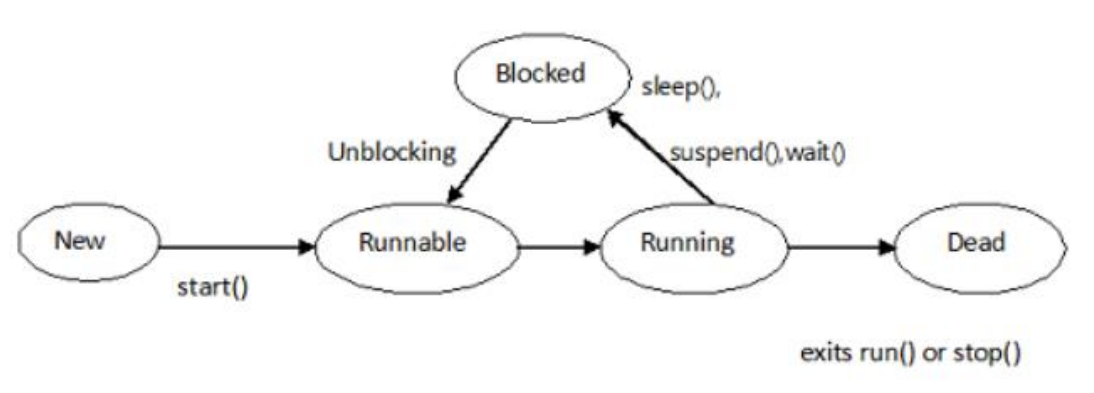
**Thread Class**



**Thread Method**



**Thread Life Cycle**



**start() Method**

* Creates 2 thread.
* **Current thread** is the current thread.
* Other thread **helps current thread** in executing its ***run()*** method.
* Thread moves from **new** state to **runnable** state.
* ***run()*** method is part of ***start()*** method, so ***run()*** is by **default** executed.

**run() Method**

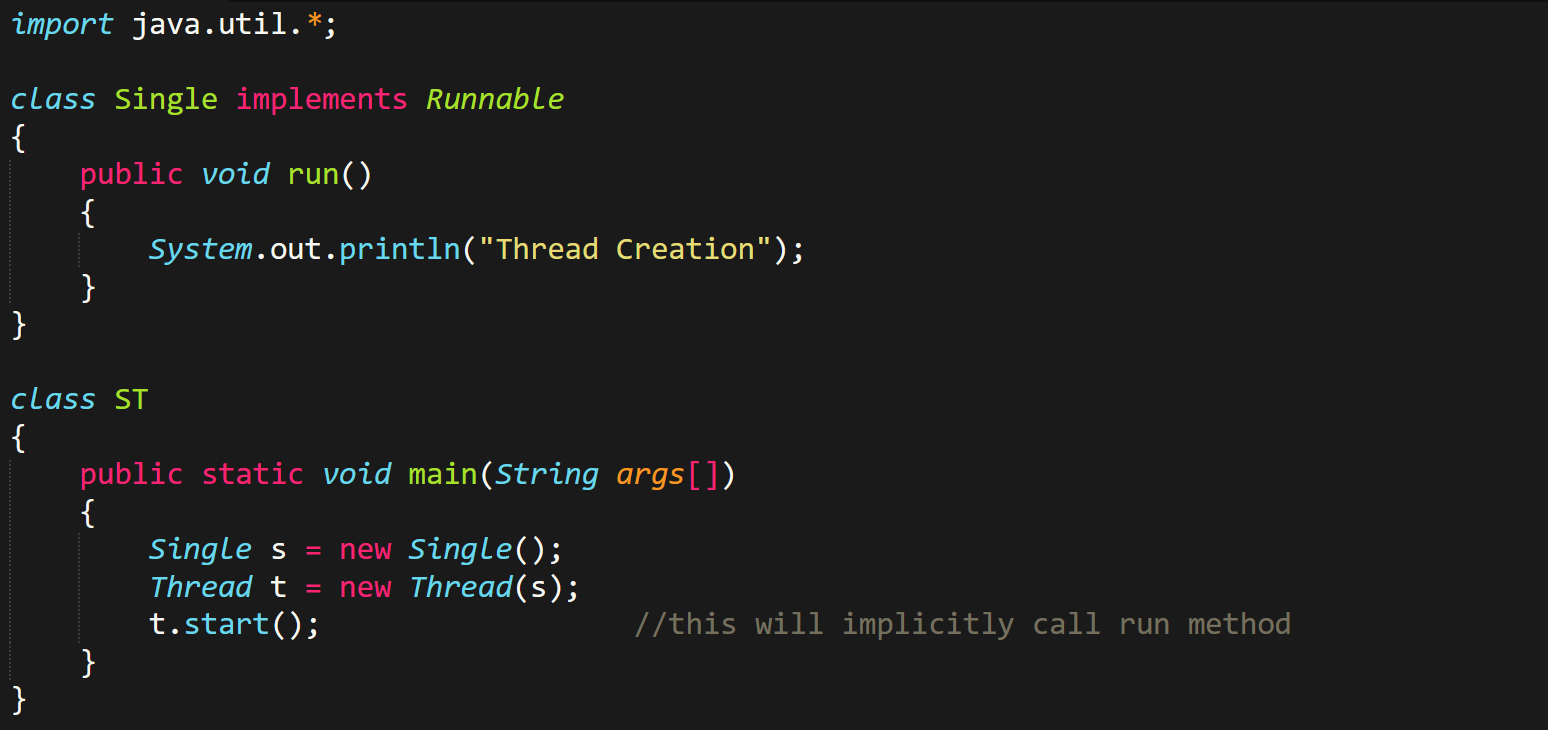
* Runs only if thread was constructed using a runnable object.
* ***run()*** method can be called multiple times.
* It can be called either by calling ***start()*** method or ***run()*** separately.
* Calling ***run()*** individually may cause problems.

**Creating Threads in Java**

* By extending (inheriting) classes



* By creating runnable surface (interface):



**suspend() Method**

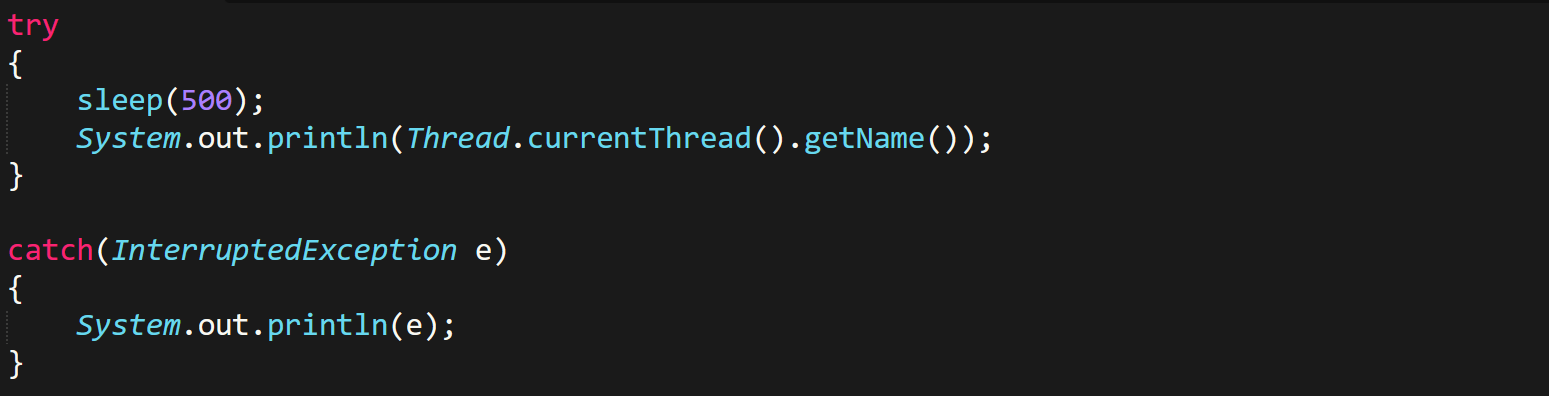


* ***suspend()*** basically works as a **restart button**.
* We use **milliseconds** as unit of time for ***sleep()***.

**stop() Method**

* Once you use ***stop()*** method, a **can’t** use ***start()*** for that thread.

**Usual Way of Writing Methods**



**join() Method**

* **Waits** for a thread to die.
* Lines written after writing ***join()*** method are executed only after that thread is dead.

**Why Exception Handling in Multithreading**

* Using exceptions is **advised** in multithreading.
* Exception handling causes threads to **not** crash due to exception in one thread.
* Its ***InturreptedException*** used for multithreadings.