Threads

- Threads and Processes
- new Thread
- Role of OS
- Main methods
- Examples

Processes and Threads

Processes and threads are independent sequences of execution.

The main difference is that threads (of the same process) run in a shared memory space, while processes run in separate memory spaces.

Thread vs Process

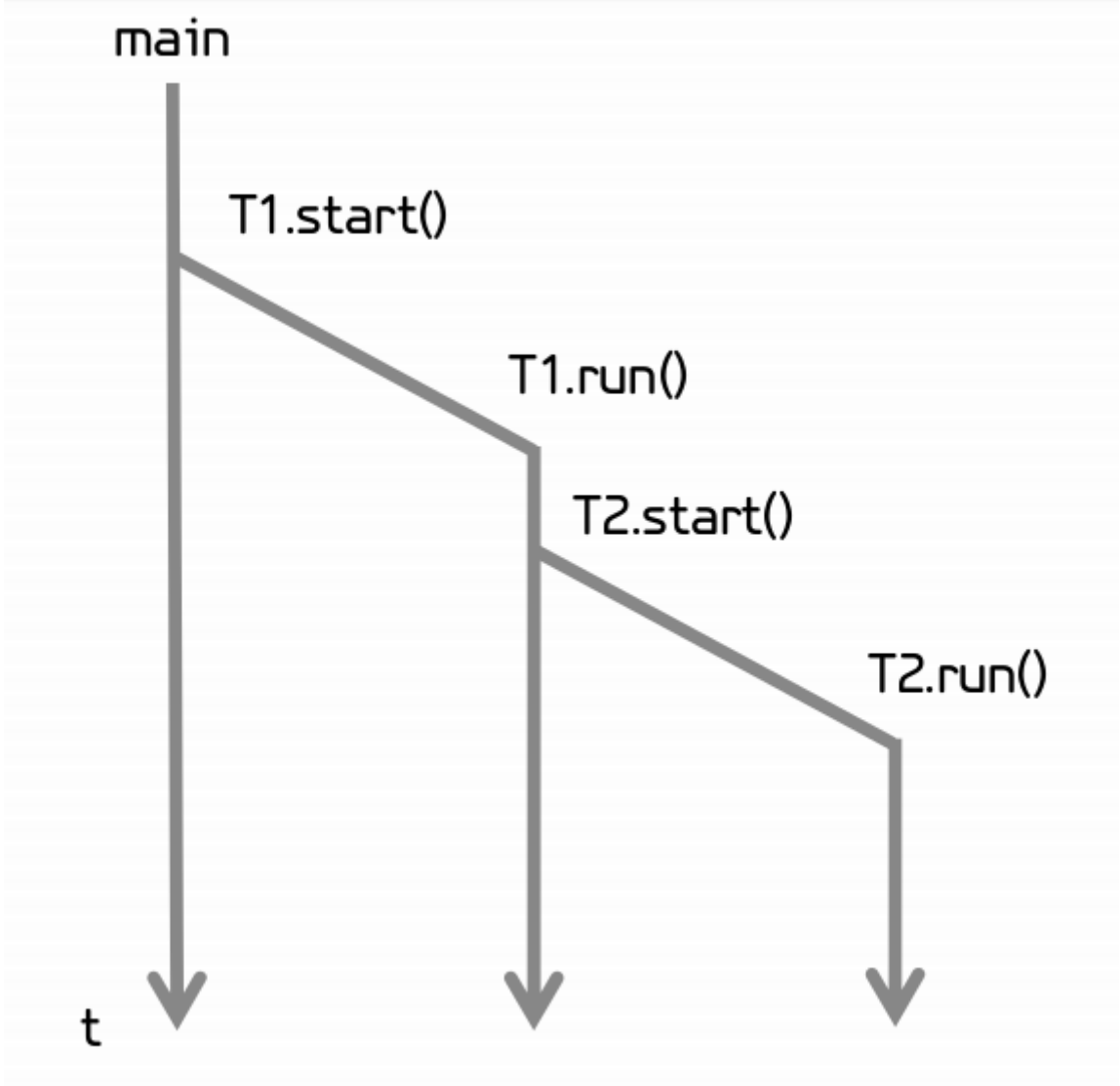
- A program in execution is often referred as process. A thread is a subset(part) of the process.
- A process may consists of multiple threads. And must have at least one thread.
- A process has its own address space. A thread uses the process’s address space and share it with the other threads of that process.
- A thread can communicate with other thread (of the same process) directly. A process can communicate with other process by using inter-process communication.
- New threads are easily created. However the creation of new processes is creation of a new JVM.

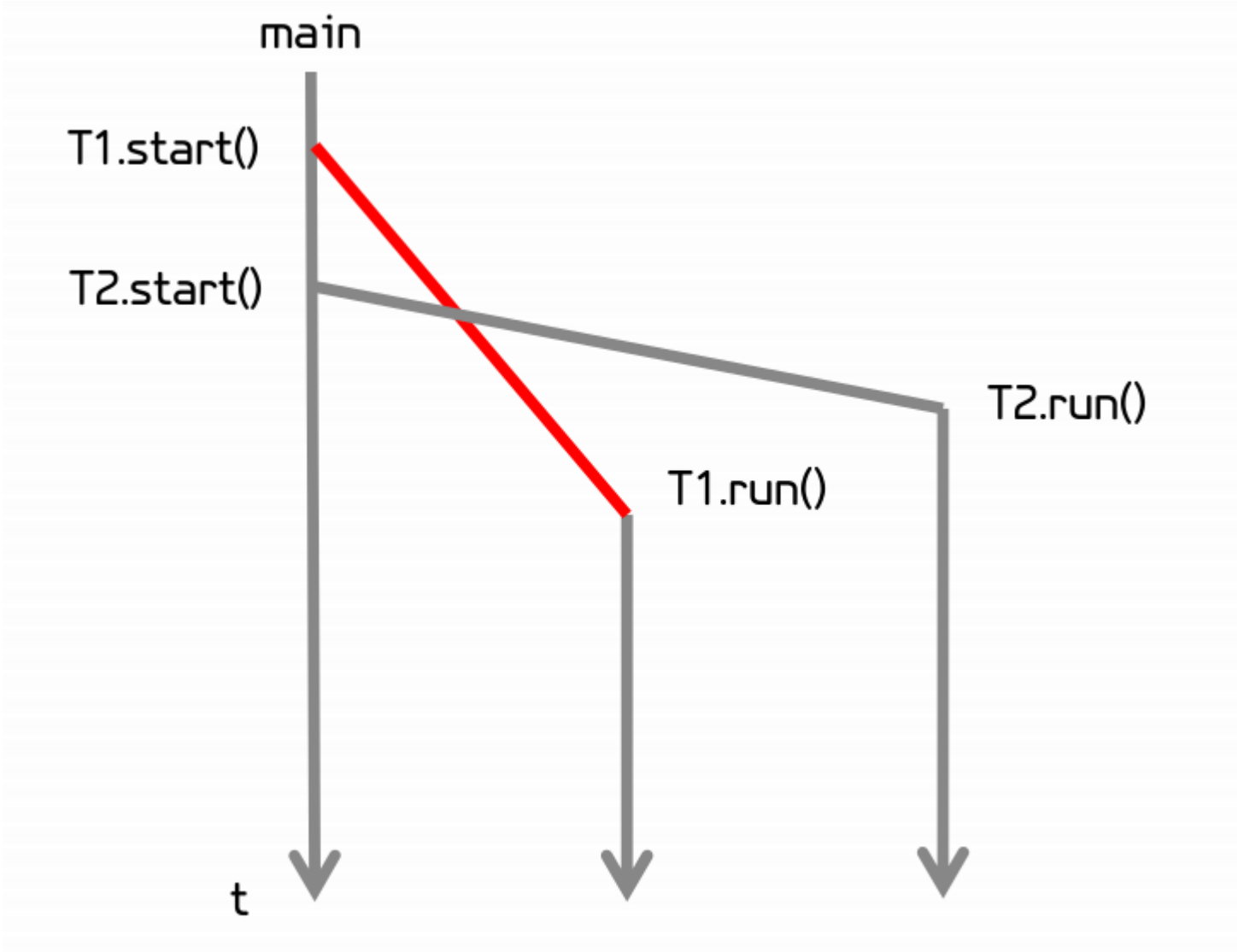
java.lang.Thread

```
public class Thread implements Runnable {
      public synchronized void start() {…}
      public void run() {
...
}
```

Thread is an object. It has methods start() and run().

After the call of method start(), method run() will be executed in a new thread (new sequence of execution).


main
T1.start()
T1.run()
T2.start()
T2.run()
t


main
T1.start()
T2.start()
T2.run()
T1.run()
t

To create a new Thread

```
class MyThreadSubclass extends Thread {

     @Override
     public void run() {…}

}


Thread thread = new MyThreadSubclass();

thread.start();


OR


Thread thread = new Thread(() -> {...});

thread.start();
```

Operating System

- Creation of new threads
- Threads scheduling (Context switch)
- Notification API

Main methods

public static native Thread currentThread()

public static native void sleep(long millis) throws InterruptedException

public void interrupt()

public boolean isInterrupted()

public final void setPriority(int newPriority)

public final int getPriority()

public final synchronized void setName(String name)

public final String getName()

public final void join() throws InterruptedException

public final void setDaemon(boolean on)

public final boolean isDaemon()

public State getState()

java.lang.Thread.State

```
public enum State {
    NEW,
    RUNNABLE,
    BLOCKED,
    WAITING,
    TIMED_WAITING,
    TERMINATED;
}
```