Mahindra
École Centrale
COLLEGE OF ENGINEERING

Object Oriented Programming JAVA

Dr. Prafulla Kalapatapu

Computer Science Engineering

Mahindra Ecole Centrale

prafulla.kalapatapu@mechyd.ac.in

Multi Threading

Multi threading

Mahindra École Centrale
COLLEGE OF ENGINEERING

- What is a thread?

  - A thread represents a separate/independent path of execution of a group of statements.

  - In java program, if we write a group of statements, then these statements are executed by JVM one by one. This execution is called as Thread.

  - In every java program, there is always a thread running internally.

- What is that thread?

  following program shows that thread.

Program

```
class Current {
public static void main(String []args) {
System.out.println("let us find current thread");
Thread t=Thread.currentThread();
System.out.println("current thread ="+t);
System.out.println("its name ="+t.getName());
}
}
```

Output :

Priority number(1 to 10) by default

```
let us find current thread
current thread =Thread[main,5,main]
its name =main
```

Name of thread

Thread group name

Given object belongs to which class

- Which thread always runs in java program by default

  main
- How many types of tasks in execution

  1. Single tasking

  2. Multi tasking

1. Single Tasking :

   In single tasking environment only one task executed at a time

   Ex: programs written by the student in lab

- What is the draw back in single tasking

  Keeping processor in idle state.

2. Multi tasking:

Executing several tasks simultaneously (context switching) is the concept of multi tasking

- Two types of Multitasking

  - Process Based

  - Thread Based

Process Based Multitasking:

Executing several tasks simultaneously where each task is a separate independent process.

Ex: listening Mp3 songs
downloading file from net **Simultaneously(independent of each other)**
writing java program

This type of multitasking is best suitable for OS level

Thread based multitasking:

- Executing several tasks simultaneously, where each task is separate independent part of the same program.
- This type of multitasking is best suitable for programmatical level.
- Each independent part is called Thread.

- How can we implement multithreading in java

  Java provides in-built support for multithreading by introducing a rich library.

- What is the advantage of multithreading

  To improve the performance of your application

- Applications of Multithreading

  - Server side programming
  - Video games
  - Multimedia applications

Ways to define, instantiate, start a thread

We can define thread in two ways

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

1. Extends Thread class:

Define Thread:

- Write an user defined class that extends Thread class
- Override run() method. (which is in Thread class)

What is the responsibility of run() method

It specifies the job of the thread

Why we have to override run() method

run() method is defined empty in Thread class

Ex: `public void run() {     }`

Mahindra École Centrale
COLLEGE OF ENGINEERING

Ex:

```
public  class  MyThread  extends  Thread
{
            public void run()
            {
            for(int i=0;i<10;i++)
            {
                        System.out.println("Child Thread");
            }
            }
}
```

Define the thread

Job of the thread

Instantiating the thread:

Creating an object to the user defined class that extends Thread.

Ex: MyThread mt=new MyThread();

Starting the thread

- To start the thread, call start() method on the created object
- start() method is in Thread class

What is the responsibility of start() method

```
public void start()
 {
- creating/ registering the thread in the Thread scheduler
- it calls run() method
}
```

Ex: mt.start();

Full example

```
public class MyThread extends Thread {
        public void run() {
        for(int i=0;i<10;i++)
        {
                System.out.println(“Child Thread”);
        }
        }  }
class Tdemo {
        public static void main(String args[]) {
        MyThread mt=new MyThread();
        mt.start();
        for(int i=0;i<10;i++)
        {
                System.out.println(“Main Thread”);
        }
        }  }
```

Mahindra École Centrale
COLLEGE OF ENGINEERING

Thread scheduler

Thread scheduler

Mahindra École Centrale
COLLEGE OF ENGINEERING

- What is Thread scheduler

  Thread scheduler is a program or software

- What is the responsibility of thread scheduler

  It picks up the threads from the thread pool to make them run

- Where Thread scheduler is located

  - It is located internally in JVM

  - It is JVM's vendor dependent.

Mahindra
École Centrale
COLLEGE OF ENGINEERING

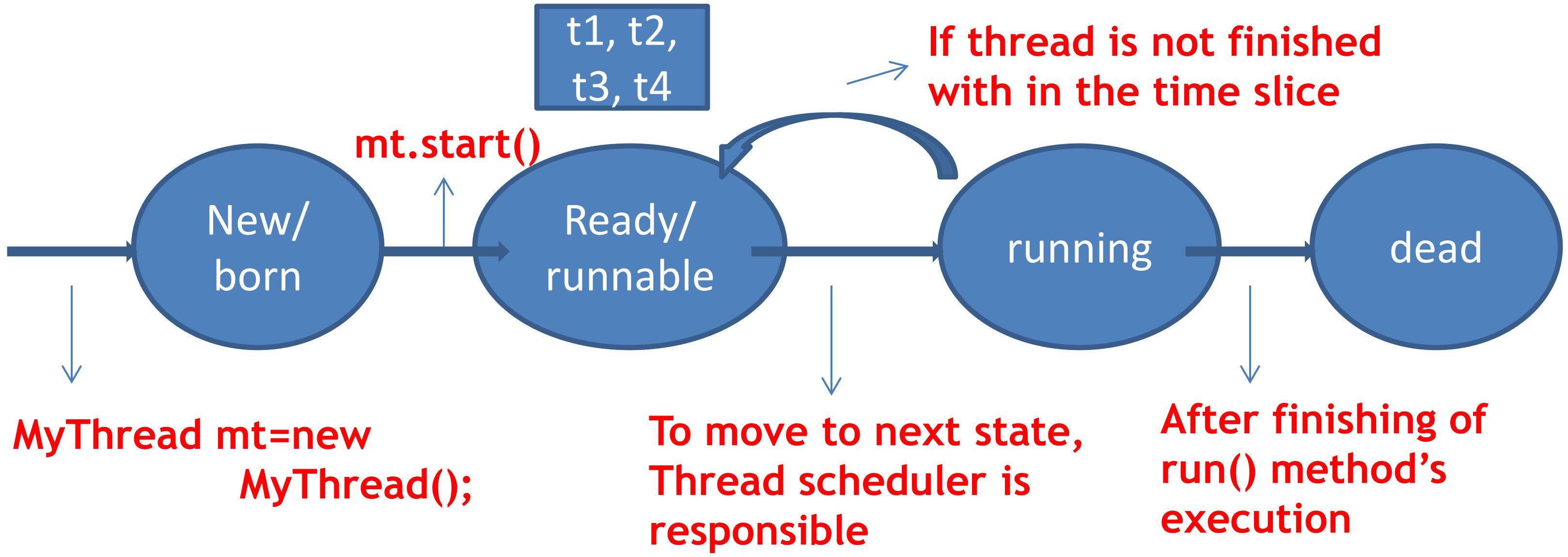
Difference between t.start() and t.run()

- In the case of t.start() new thread will be created which is responsible for the execution of the run() method

- But in the case of t.run() no new thread will be created and run() method will be executed just like normal method

Life Cycle of a Thread

There are four states in life cycle of thread


t1, t2, t3, t4
If thread is not finished with in the time slice
mt.start()
New/ born
Ready/ runnable
running
dead
MyThread mt=new MyThread();
To move to next state, Thread scheduler is responsible
After finishing of run() method's execution

Note: After starting thread, we are not allowed to start the same thread once again. Violation leads to runtime exception. "IllegalThreadStateException"

Mahindra
École Centrale
COLLEGE OF ENGINEERING

Ex:

```
MyThread mt=new MyThread();
mt.start();
mt.start();
```

$\longrightarrow$ IllegalThreadStateException