Mahindra École Centrale
COLLEGE OF ENGINEERING

Object Oriented Programming JAVA

Dr. Prafulla Kalapatapu
Computer Science Engineering
Mahindra Ecole Centrale
prafulla.kalapatapu@mechyd.ac.in

Mahindra École Centrale
COLLEGE OF ENGINEERING

Multi Threading

2. By implementing runnable interface

Mahindra École Centrale
COLLEGE OF ENGINEERING

- What is runnable interface

  It supports in-built multi threading

- How many methods it contain

  Only one

  ```
  public void run();
  ```

- Steps to implement from a Runnable interface

  - Write an user defined class that implements from runnable interface
  - Override run() method. i.e to define job of thread

Define, instantiate and start by implementing runnable interface

Mahindra École Centrale
COLLEGE OF ENGINEERING

Define a Thread:

Ex:

```
class MyRunnable implements Runnable      {
        public void run()  {
                System.out.println("child thread");
        }  }
```

Define the thread

Job of the thread

Instantiation of Thread:

```
MyRunnable mr=new MyRunnable();
mr.start();
Thread t1=new Thread(mr);
```

Error (start() method is in Thread class)

Starting of Thread:

```
t1.start();
```

It calls run() method, which is in target runnable class ex: MyRunnable class

- Which approach is best to define a Thread

  - Among the 2 approaches of defining a Thread, implements runnable interface is always recommended.

  - In first approach, as our class already extending thread class, there is no chance of extending any other. Hence we are missing Inheritance concept and so it is not recommended to use

Thread class constructors

```
Thread t=new Thread();
Thread t=new Thread(Runnable r);
Thread t=new Thread(String name);
Thread t=new Thread(Runnable r, String name);
Thread t=new Thread(ThreadGroup g, Runnable r);
Thread t=new Thread(ThreadGroup g, Runnable r, String name);
Thread t=new Thread(ThreadGroup g, Runnable r, String name, long stacksize);
```

Getting and setting the name of Thread

```
public final String getName();
public final void setName();
```

Thread Priority

Mahindra École Centrale
COLLEGE OF ENGINEERING

- Range of Thread Priority is 1 to 10.

  Thread.MAX_PRIORITY is 10
  Thread.NORM_PRIORITY is 5
  Thread.MIN_PRIORITY is 1

- NOTE:

  - Thread Priority is used by Thread Scheduler
  - Thread Priorities concept may not be supported by some platform

Default priority :

- For main() method it is always 5, but we can set with setPriority() method

- But for all other threads, thread priority will be inherited from the parent thread.

- setPriority() method argument should range between 1 to 10, if not Run-time Exception raises saying “IllegalArgumentException”.

Methods to prevent a thread from execution

There are 3 methods to prevent thread from execution

(i) yield()

(ii) join()

(iii) sleep()

(i) yield() :

- It temporarily pauses the execution and gives the chance to waiting thread of the same priority
- If there is no waiting thread or all the waiting threads having low priority then the same thread will get chance once again for execution
- Note: the behaviour of yield() method may not be supported by platform properly

(ii) join() :

- If a thread wants to wait until completing some other threads, then we should go for join() method
- join() method throws checked exception (InterruptedException)

(iii) sleep() :

- If a thread don't want to perform any activity for a particular amount of time. Then we should go for sleep() method.
- sleep() method throws checked exception (InterruptedException)

Interrupting a Thread

A thread can interrupt another thread (waiting or sleeping thread) is by using interrupt() method of the Thread class

Comparison

Property	yield()	join()	sleep()
Is static	Yes	No	Yes
Is final	No	Yes	No
Is native	Yes	No	Yes
Is overloaded	No	Yes	Yes
Is it throws InterruptedException	No	Yes	Yes

Synchronization

Mahindra École Centrale
COLLEGE OF ENGINEERING

- To perform synchronization we have keyword called synchronized
  - It can be for blocks
  - it can be for methods

- synchronized keyword is used to avoid data inconsistency problem

- Limitation of synchronization
  Only one thread executes at a time.

- To apply synchronization we need lock and we can put lock using synchronized keyword

Types of locks :

- There are two locks
  - object level lock
  - class level lock

- If we use synchronized keyword for the instance method, we can get the object level lock.

- If we use synchronized keyword for the static method, we can get the class level lock.

Synchronized Block

- If very few lines of code requires synchronization then it is not recommended to declare entire method as synchronized. We have to declare those few lines of code inside synchronized block.

- Syntax:

```
synchronized( object name)
{
        ---
        ---
}
```

Daemon Thread

- A daemon thread is a thread that executes continuously.
- A daemon threads are service providers for other threads, it generally provides background processing

  Ex: Garbage Collector

- To make t as Daemon thread we can use

  `t.setDaemon();`

- To know if a thread is daemon or not

  `boolean x=t.isDaemon();`

Inter Thread Communication

Mahindra École Centrale
COLLEGE OF ENGINEERING

- Two threads can communicate with each other by using wait(), notify(), notifyall() methods
- All these three threads, are to be called from synchronized block or area. Violation leads to Runtime Exception "IllegalMonitorStateException"
- When ever a thread calls wait(), it releases lock and enters into the waiting state.

Method	Is lock released
yield()	No
sleep()	No
join()	No
wait()	Yes
notify()	Yes
notifyall()	Yes