THREAD CREATION & SYNCHRONIZATION

Prepared by

Dr. Rajesh Kumar Ojha
Asst. Prof., CSE, Silicon University

THREAD CREATION

There are two ways to create a thread:

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

Thread class:

- Thread class provide constructors and methods to create and perform operations on a thread.
- Thread class extends Object class and implements Runnable interface.

Commonly used Constructors of Thread class:

- Thread()
- Thread(String name)
- Thread(Runnable r)
- Thread(Runnable r,String name)

Starting a thread:

start() method of Thread class is used to start a newly created thread. It performs following tasks: A new thread starts(with new callstack).

- The thread moves from New state to the Runnable state.
- When the thread gets a chance to execute, its target run() method will run.

By extending Thread class:

```
class Multi extends Thread{
public void run(){
System.out.println("thread is running...");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
 }
}
```

Output: thread is running...

By implementing the Runnable interface:

```
class Multi3 implements Runnable{
public void run(){
System.out.println("thread is running...");
}
public static void main(String args[]){
Multi3 m1=new Multi3();
Thread t1 =new Thread(m1);
t1.start();
 }
}
```

Output: thread is running...

sleep() method

- The sleep() method of Thread class is used to sleep a thread for the specified amount of time.
- The Thread class provides two methods for sleeping a thread:

➢ public static void **sleep(long miliseconds)**throws InterruptedException

➢ public static void **sleep(long miliseconds, int nanos)**throws InterruptedException

Example of sleep()

```
class TestSleepMethod1 extends Thread{
 public void run(){
  for(int i=1;i<5;i++){
    try{
Thread.sleep(500);
}catch(InterruptedException e){System.out.println(e);}
    System.out.println(i);
  }
 }
 public static void main(String args[]){
  TestSleepMethod1 t1=new TestSleepMethod1();
  TestSleepMethod1 t2=new TestSleepMethod1();
  t1.start();
  t2.start();
 }
}
```

O/P: 1 1 2 2 3 3 4 4

THREAD SYNCRONIZATION

- Synchronization in java is the capability *to control the access of multiple threads to any shared resource*.
- Java Synchronization is better option where we want to allow only one thread to access the shared resource.
- The synchronization is mainly used to
  - To prevent thread interference.
  - To prevent consistency problem.

Types of Synchronization

There are two types of synchronization

- Process Synchronization
- Thread Synchronization

Thread Synchronization

- There are two types of thread synchronization mutual exclusive and inter-thread communication.
- Mutual Exclusive
  - Synchronized method.
  - Synchronized block.
  - static synchronization.
- Cooperation (Inter-thread communication in java)

Mutual Exclusive

Mutual Exclusive helps keep threads from interfering with one another while sharing data. This can be done by three ways in java:

- by synchronized method
- by synchronized block
- by static synchronization

problem without Synchronization

```
class Table{
void printTable(int n){//method not synchronized
for(int i=1;i<=5;i++){
System.out.println(n*i);
 try{
Thread.sleep(400);
}catch(Exception e){
System.out.println(e);
}
}
}  }
```

```
class MyThread1 extends Thread{
Table t;
MyThread1(Table t){
this.t=t;
}
 public void run(){
 t.printTable(5);
 }
}
class MyThread2 extends Thread{
Table t;
MyThread2(Table t){
this.t=t;
}
public void run(){
 t.printTable(100);
 }
}
class TestSynchronization1{
public static void main(String args[]){
//only one object
Table obj = new Table();
MyThread1 t1=new MyThread1(obj);
MyThread2 t2=new MyThread2(obj);
t1.start();
 t2.start();
 }
}
```

Output: 5

100

10

200

15

300

20

400

25

500

Java synchronized method

- If you declare any method as synchronized, it is known as synchronized method.
- Synchronized method is used to lock an object for any shared resource.
- When a thread invokes a synchronized method, it automatically acquires the lock for that object and releases it when the thread completes its task.

```
//example of java synchronized method
class Table{
synchronized void printTable(int n){//synchronized method
for(int i=1;i<=5;i++){
System.out.println(n*i);
try{
Thread.sleep(400);
}catch(Exception e){
System.out.println(e);
}
}
 }
}
```

```
class MyThread1 extends Thread{
Table t;
MyThread1(Table t){
this.t=t;
}
 public void run(){
 t.printTable(5);
 }
}
class MyThread2 extends Thread{
Table t;
MyThread2(Table t){
this.t=t;
}
public void run(){
 t.printTable(100);
 }
}
public class TestSynchronization1{
public static void main(String args[]){
//only one object
Table obj = new Table();
MyThread1 t1=new MyThread1(obj);
MyThread2 t2=new MyThread2(obj);
t1.start();
 t2.start();
 }
}
```

Output: 5

10

15

20

25

100

200

300

400

500

Creating Multiple Threads

```
class NewThread implements Runnable {
  String name; // name of thread
  Thread t;

  NewThread(String threadname) {
    name = threadname;
    t = new Thread(this, name);
    System.out.println("New thread: " + t);
    t.start(); // Start the thread
  }
```

```
  public void run() {
    try {
       for(int i = 5; i > 0; i--) {
         System.out.println(name + ": " + i);
         Thread.sleep(1000);
       }
    } catch (InterruptedException e) {
       System.out.println(name + "Interrupted");
    }
    System.out.println(name + " exiting.");
  }
}

class MultiThreadDemo {
  public static void main(String args[]) {
    new NewThread("One"); // start threads
    new NewThread("Two");
    new NewThread("Three");
```

```
    try {
      // wait for other threads to end
      Thread.sleep(10000);
    } catch (InterruptedException e) {
      System.out.println("Main thread Interrupted");
    }

    System.out.println("Main thread exiting.");
  }
}
```

```
New thread: Thread[One,5,main]
New thread: Thread[Two,5,main]
New thread: Thread[Three,5,main]
One: 5
Two: 5
Three: 5
One: 4
```

```
Two: 4
Three: 4
One: 3
Three: 3
Two: 3
One: 2
Three: 2
Two: 2
One: 1
Three: 1
Two: 1
One exiting.
Two exiting.
Three exiting.
Main thread exiting.
```

Thank you