ECAP615

Programming in Java

Harjinder Kaur

Assistant Professor

Learning Outcomes

After this lecture, you will be able to

- learn the basic concept of Multithreading.
- understand the different ways of implementing Multithreading.

Multithreading

- Before we talk about multithreading, let's discuss threads.
- A thread is a light-weight smallest part of a process that can run concurrently with the other parts(other threads) of the same process.

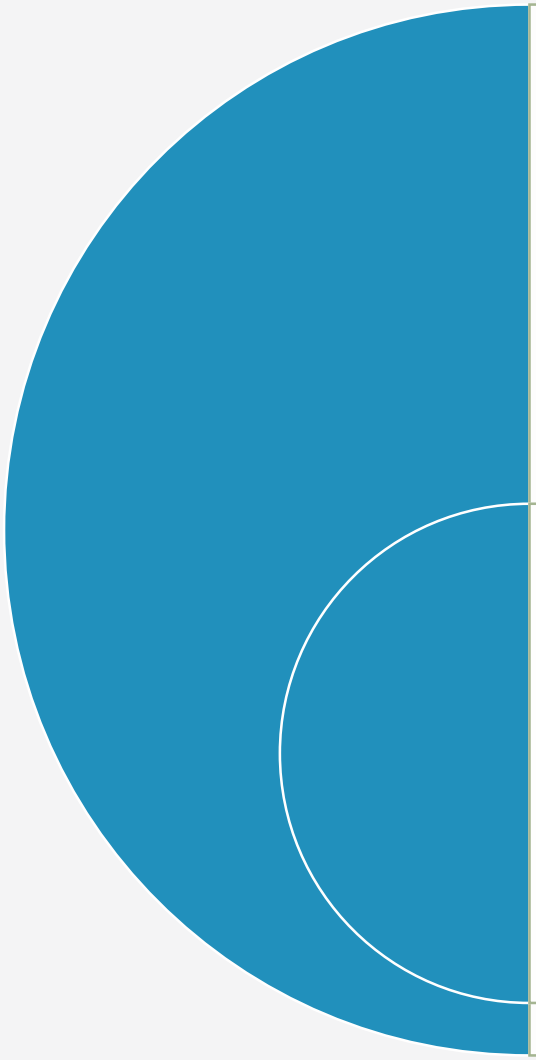
Multithreading

- Threads are independent because they all have a separate path of execution that's the reason if an exception occurs in one thread, it doesn't affect the execution of other threads.
- All threads of a process share a common memory.
- The process of executing multiple threads simultaneously is known as multithreading.

Multithreading

- Multithreading in Java is a process of executing two or more threads simultaneously to maximum utilization of CPU.
- Java's multithreading system is built upon the Thread class, its methods, and its companion interface, Runnable.
- Each thread runs parallel to each other.
- Multiple threads don't allocate separate memory areas, hence they save memory.

Advantages



The users are not blocked because threads are independent, and we can perform multiple operations at times.

As such, the threads are independent, the other threads won't get affected if one thread meets an exception.

Creation of Threads

Threads can be created by using two mechanisms :

- Extending the Thread class.
- Implementing the Runnable Interface.

Methods of Thread class.

- getName(): It is used for Obtaining a thread's name.
- getPriority(): Obtain a thread's priority.
- isAlive(): Determine if a thread is still running.
- join(): Wait for a thread to terminate.
- run(): Entry point for the thread.
- sleep(): suspend a thread for a period of time.
- start(): start a thread by calling its run() method.

Thread creation by extending the Thread class

- We create a class that extends the java.lang.Thread class.
- This class overrides the run() method available in the Thread class.
- A thread begins its life inside run() method.
- We create an object of our new class and call start() method to start the execution of a thread.
- Start() invokes the run() method on the Thread object.

Program

```
// Java code for thread creation by extending
// the Thread class
class MultithreadingDemo extends Thread {
        public void run()
        {
                try {
// Displaying the thread that is running
System.out.println("Thread " + Thread.currentThread().getId()+ " is
running");
                }
catch (Exception e) {
```

Continue...

Program

Continue…

```
// Throwing an exception
System.out.println("Exception is
caught");
            } } }
// Main Class
public class Multithread {
public static void main(String[] args)
      {
            int n = 8; // Number of
threads
            for (int i = 0; i < n; i++) {
                  MultithreadingDemo
object
                        = new
MultithreadingDemo();
                  object.start();
            }
      }
}
```

Thread creation by implementing the Runnable Interface

- We create a new class which implements java.lang.
- Runnable interface and override run() method.
- Then we instantiate a Thread object and call start() method on this object.

Program

```
// Java code for thread creation by
implementing

// the Runnable Interface

class MultithreadingDemo1 implements
Runnable {

        public void run()

        {

                try {

// Displaying the thread that is running

System.out.println(             "Thread " +
Thread.currentThread().getId()+ " is running");

                }
```

Continue...

Program

Continue…

```
catch (Exception e) {
// Throwing an exception

        System.out.println("Exception is
caught");
                } } }

// Main Class
class Multithread {
        public static void main(String[] args)
        {
                int n = 8; // Number of threads
                for (int i = 0; i < n; i++) {
                        Thread object
                                = new Thread(new
MultithreadingDemo());
                        object.start();
                } } }
```


That’s all for now…