Java - MultiThreading

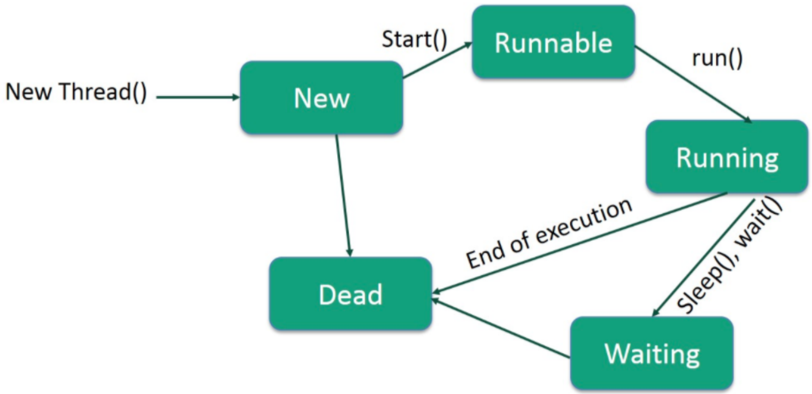
Md. Mohsin Uddin

East West University

mmuddin@ewubd.edu

July 27, 2019

Life Cycle of a Thread



Create a Thread by Implementing a Runnable Interface : Part I

```
package multithreading;

class RunnableDemo implements Runnable {
   private Thread t;
   private final String threadName;

   RunnableDemo( String name) {
      threadName = name;
      System.out.println("Creating " + threadName );
   }

   @Override
   public void run() {
      System.out.println("Running " + threadName );
      try {
         for(int i = 4; i > 0; i--) {
```

Create a Thread by Implementing a Runnable Interface : Part II

```
            System.out.println("Thread: " +
                  threadName + ", " + i);
            // Let the thread sleep for a while.
            Thread.sleep(50);
         }
      } catch (InterruptedException e) {
         System.out.println("Thread " +
                  threadName + " interrupted.");
      }
      System.out.println("Thread " +
                 threadName + " exiting.");
   }

   public void start () {
      System.out.println("Starting " +  threadName );
      if (t == null) {
         t = new Thread (this, threadName);
         t.start ();
```

Create a Thread by Implementing a Runnable Interface : Part III

```
        }
    }
}
class TestThread {

    public static void main(String args[]) {
        RunnableDemo R1 = new RunnableDemo( "Thread-1");
        R1.start();

        RunnableDemo R2 = new RunnableDemo( "Thread-2");
        R2.start();
    }
}
```

Create a Thread by Implementing a Runnable Interface : Part IV

When the above code is compiled and executed, it produces the following result:

```
Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-1
Thread: Thread-1, 4
Running Thread-2
Thread: Thread-2, 4
Thread: Thread-2, 3
Thread: Thread-1, 3
Thread: Thread-1, 2
Thread: Thread-2, 2
Thread: Thread-1, 1
Thread: Thread-2, 1
Thread Thread-1 exiting.
Thread Thread-2 exiting.
```

Create a Thread by Extending a Thread Class : Part I

```
package multithreading;

class ThreadDemo extends Thread {
   private Thread t;
   private String threadName;

   ThreadDemo( String name) {
      threadName = name;
      System.out.println("Creating " + threadName );
   }

   public void run() {
      System.out.println("Running " + threadName );
      try {
         for(int i = 4; i > 0; i--) {
            System.out.println("Thread: " +
                  threadName + ", " + i);
            // Let the thread sleep for a while.
```

Create a Thread by Extending a Thread Class : Part II

```
                Thread.sleep(50);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread " +
                     threadName + " interrupted.");
        }
        System.out.println("Thread " +
                     threadName + " exiting.");
    }

    public void start () {
        System.out.println("Starting " +  threadName );
        if (t == null) {
            t = new Thread (this, threadName);
            t.start ();
        }
    }
}
```

Create a Thread by Extending a Thread Class : Part III

```
class TestThread2 {

   public static void main(String args[]) {
      ThreadDemo T1 = new ThreadDemo( "Thread-1");
      T1.start();

      ThreadDemo T2 = new ThreadDemo( "Thread-2");
      T2.start();
   }
}
```

When the above code is compiled and executed, it produces the following result:

Create a Thread by Extending a Thread Class : Part IV

```
Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-1
Thread: Thread-1, 4
Running Thread-2
Thread: Thread-2, 4
Thread: Thread-2, 3
Thread: Thread-1, 3
Thread: Thread-1, 2
Thread: Thread-2, 2
Thread: Thread-2, 1
Thread: Thread-1, 1
Thread Thread-1 exiting.
Thread Thread-2 exiting.
```

Thread Methods : Part I

```
package multithreading;

class DisplayMessage implements Runnable {
    private String message;

    public DisplayMessage(String message) {
        this.message = message;
    }

    public void run() {
        while(true) {
            System.out.println(message);
        }
    }
}

class GuessANumber extends Thread {
```

Thread Methods : Part II

```
    private int number;
    public GuessANumber(int number) {
        this.number = number;
    }

    public void run() {
        int counter = 0;
        int guess = 0;
        do {
            guess = (int) (Math.random() * 100 + 1);
            System.out.println
                    (this.getName() + " guesses " + guess);
            counter++;
        } while(guess != number);
        System.out.println("** Correct!"
          + this.getName() + "in" + counter + "guesses.**");
    }
}
```

Thread Methods : Part III

```
public class ThreadClassDemo {

  public static void main(String [] args) {
    Runnable hello = new DisplayMessage("Hello");
    Thread thread1 = new Thread(hello);
    thread1.setDaemon(true);
    thread1.setName("hello");
    System.out.println("Starting hello thread...");
    thread1.start();

    Runnable bye = new DisplayMessage("Goodbye");
    Thread thread2 = new Thread(bye);
    thread2.setPriority(Thread.MIN_PRIORITY);
    thread2.setDaemon(true);
    thread2.setName("GoodBye");
    System.out.println("Starting goodbye thread...");
    thread2.start();

    System.out.println("Starting thread3...");
```

Thread Methods : Part IV

```
        Thread thread3 = new GuessANumber(27);
        thread3.setName("Guess1");
        thread3.start();
        try {
           thread3.join();
        } catch (InterruptedException e) {
           System.out.println("Thread interrupted.");
        }
        System.out.println("Starting thread4...");
        Thread thread4 = new GuessANumber(75);
        thread4.setName("Guess2");
        thread4.start();
        System.out.println("main() is ending...");
     }
  }
```

The join() method : Part I

```
package multithreading;
class TestJoinMethod1 extends Thread{
    public void run(){
        for(int i=1;i<=5;i++){
            try{
                Thread.sleep(500);
            }
            catch(Exception e) {
                System.out.println(e);
            }
            System.out.println
                (Thread.currentThread().getName()+"␣"+i);
        }
    }
    public static void main(String args[]){
        TestJoinMethod1 t1=new TestJoinMethod1();
        t1.setName("Thread-1");
        TestJoinMethod1 t2=new TestJoinMethod1();
        t2.setName("Thread-2");
```

The join() method : Part II

```
        TestJoinMethod1 t3=new TestJoinMethod1();
        t3.setName("Thread-3");
        t1.start();
        try{
            t1.join();
        }catch(Exception e){
            System.out.println(e);
        }

        t2.start();
        t3.start();
    }
}
```

The join() method : Part III

When the above code is compiled and executed, it produces the following result:

```
Thread-1 1
Thread-1 2
Thread-1 3
Thread-1 4
Thread-1 5
Thread-3 1
Thread-2 1
Thread-2 2
Thread-3 2
Thread-3 3
Thread-2 3
Thread-3 4
Thread-2 4
Thread-2 5
Thread-3 5
```

References

- DEITEL, Java How to Program, 11/e
- Java: the complete reference, Herbert Schildt, McGraw-Hill Education Group