**Section-End Project - Lesson 3**

## **ThreadsGroup, View all threads, ThreadPriority**

In this exercise, you are going to learn how to display information on a ThreadGroup, how to set a thread priority, and so on.

1. **Display threads of a ThreadGroup**
2. **Display all threads in the system**
3. **Set thread priority**

### **(3.1) Display threads of a ThreadGroup**

1. Start Spring Tools Suite IDE if you have not done so yet.  
   1. Create a new Java project

* Select **File->New Java Project (Alt+Shift+N)**. The **New Java Project** dialog box appears.
* For the Project Name field, type in **ThreadGroupTest**as project name.
* Choose ‘Use default JRE (11….). Make sure you have set your default JRE set to version 11 and the appropriate compliance settings as well. Deselect ‘Create module-info.java file’ if it is selected. Click Next.Click Finish.
* Observe that **ThreadGroupTest**project appears
* Right click on the project and select **File**->**New Class.**
* Enter in a package name of your choosing or use the default. Type in **ThreadGroupTest** as the class name. The main method stub should also be created.
* The IDE generated **ThreadGroupTest.java** is displayed in the source editor window of STS IDE.

2.  Modify the IDE generated **ThreadGroupTest.java**as shown in Code-3.11 below.  Study the code by paying special attention to the bold fonted parts.

|  |
| --- |
| public class ThreadGroupTest {         public static void main (String[] args) {           **// Start three threads first.  They should belong         // to a same ThreadsGroup.         new SimpleThread("Boston").start();         new SimpleThread("New York").start();         new SimpleThread("Seoul").start();**         **// Get ThreadGroup of the current thread and display         // the number of active threads that belong to the         // ThreadGroup.         ThreadGroup group = Thread.currentThread().getThreadGroup();**         System.out.println("Number of active threads in this thread group = "                             + group.activeCount());                 // Display the names of the threads in the current         // ThreadGroup.         Thread[] tarray = new Thread[10];         int actualSize = group.enumerate(tarray);         for (int i=0; i<actualSize;i++){             System.out.println("Thread " + tarray[i].getName()                                + " in thread group " + group.getName());         }             } } |

Code-3.11: ThreadGroupTest.java  
  
3. Write **SimpleThread.java**as shown in Code-3.12 below.

|  |
| --- |
| public class SimpleThread extends Thread {         public SimpleThread(String str) {         super(str);     }         public void run() {         for (int i = 0; i < 5; i++) {             // System.out.format("%d %s%n", i, getName());             try {                 sleep((long)(Math.random() \* 1000));             } catch (InterruptedException e) {}         }         System.out.format("DONE! %s%n", getName());     } } |

Code-3.12: SimpleThread.java  
  
4. Build and run the project

* Right click **ThreadGroupTest**project and select **Run**.
* Observe the result in the **Output**window. (Figure-3.13 below)

|  |
| --- |
| Number of active threads in this thread group = 4 Thread main in thread group main Thread Boston in thread group main Thread New York in thread group main Thread Seoul in thread group main DONE! Seoul DONE! New York DONE! Boston |

Figure-3.13: Result of running ThreadGroupTest application

5. For your own exercise, do the following. Build and run the application.

* Modify **ThreadGroupTest.java** to create another (4th) **SimpleThread**instance using your capital city of your country.

### **(3.2) Display all threads in the system**

1. Create a new Java project

* Select **File->New Java Project (Alt+Shift+N)**. The **New Java Project** dialog box appears.
* For the Project Name field, type in **DisplayAllThreads**as project name.
* Choose ‘Use default JRE (11….). Make sure you have set your default JRE set to version 11 and the appropriate compliance settings as well. Deselect ‘Create module-info.java file’ if it is selected. Click Next.Click Finish.
* Observe that **DisplayAllThreads**project appears
* Right click on the project and select **File**->**New Class.**
* Enter in a package name of your choosing or use the default. Type in **DisplayAllThreads**as the class name. The main method stub should also be created.
* The IDE generated **DisplayAllThreads.java** is displayed in the source editor window of STS IDE.

2. Modify the IDE generated **DisplayAllThreads.java**as shown in Code-3.21 below.  Study the code by paying special attention to the bold fonted parts.

|  |
| --- |
| public class DisplayAllThreads {         public static void main(String[] args) {                 // Start three threads first.  They should belong         // to a same ThreadsGroup.         new SimpleThread("Boston").start();         new SimpleThread("New York").start();         new SimpleThread("Seoul").start();                 **Thread[] tarray = findAllThreads();**                 for (int i=0; i<tarray.length;i++){             System.out.println("Thread " + tarray[i].getName()             + " in thread group " + tarray[i].getThreadGroup().getName());         }             }         // Create an array of all threads in the system.  **public static Thread[] findAllThreads() {         ThreadGroup group = Thread.currentThread().getThreadGroup();                 ThreadGroup topGroup = group;                 while (group != null) {             topGroup = group;             group = group.getParent();         }                 int estimatedSize = topGroup.activeCount() \* 2;         Thread[] slackList = new Thread[estimatedSize];                 int actualSize = topGroup.enumerate(slackList);                 Thread[] list = new Thread[actualSize];         System.arraycopy(slackList, 0, list, 0, actualSize);                 return list;     }** } |

Code-3.21: DisplayAllThreads.java  
  
3. Write SimpleThread.java as shown in Code-3.22 below.

|  |
| --- |
| public class SimpleThread extends Thread {         public SimpleThread(String str) {         super(str);     }         public void run() {         for (int i = 0; i < 5; i++) {             // System.out.format("%d %s%n", i, getName());             try {                 sleep((long)(Math.random() \* 1000));             } catch (InterruptedException e) {}         }         System.out.format("DONE! %s%n", getName());     } } |

Code-3.22: SimpleThread.java  
  
4. Build and run the project

* Right click **DisplayAllThreads**project and select **Run**.
* Observe the result in the **Output**window. (Figure-3.23 below)

|  |
| --- |
| Thread Reference Handler in thread group system Thread Finalizer in thread group system Thread Signal Dispatcher in thread group system Thread main in thread group main Thread Boston in thread group main Thread New York in thread group main Thread Seoul in thread group main DONE! New York DONE! Seoul DONE! Boston |

Figure-1.23: Result of running DisplayAllThreads application  
  
  
5. For your own exercise, do the following. Build and run the application.

* Modify **DisplayAllThreads.java** to create another (4th) **SimpleThread**instance using your capital city of your country.

### **(3.3) Set thread priority**

1. Create a new Java project

* Select **File->New Java Project (Alt+Shift+N)**. The **New Java Project** dialog box appears.
* For the Project Name field, type in **ThreadsPriority**as project name.
* Choose ‘Use default JRE (11….). Make sure you have set your default JRE set to version 11 and the appropriate compliance settings as well. Deselect ‘Create module-info.java file’ if it is selected. Click Next.Click Finish.
* Observe that **ThreadsPriority**project appears
* Right click on the project and select **File**->**New Class.**
* Enter in a package name of your choosing or use the default. Type in **ThreadsPriority**as the class name. The main method stub should also be created.
* The IDE generated **ThreadsPriority .java** is displayed in the source editor window of STS IDE.

2. Modify the IDE generated **ThreadsPriority.java**as shown in Code-3.31 below.  Study the code by paying special attention to the bold fonted parts.

|  |
| --- |
| public class ThreadsPriority {         public static void main(String[] args) {                 Thread t1 = new SimpleThread("Boston");         t1.start();   **// Set the thread priority to 10(highest)         t1.setPriority(10);**                 Thread t2 = new SimpleThread("New York");         t2.start(); **// Set the thread priority to 5         t2.setPriority(5);**                 Thread t3 = new SimpleThread("Seoul");         t3.start();   **// Set the thread priority to 1         t3.setPriority(1);**             } } |

Code-3.31: ThreadsPriority.java  
  
3. Write SimpleThread.java as shown in Code-3.32 below.

|  |
| --- |
| public class SimpleThread extends Thread {         public SimpleThread(String str) {         super(str);     }         public void run() {         for (int i = 0; i < 10; i++) {             System.out.println(i + " " + getName()                                  + " Priority = " + getPriority());         }         System.out.println("Done! " + getName());     } } |

Code-3.32: SimpleThread.java  
  
4. Build and run the project

* Right click **ThreadsPriority**project and select **Run**.
* Observe the result in the **Output**window. (Figure-3.33 below)

|  |
| --- |
| 0 Boston Priority = 10 0 Seoul Priority = 1 0 New York Priority = 5 1 Boston Priority = 10 1 Seoul Priority = 1 1 New York Priority = 5 2 Boston Priority = 10 2 Seoul Priority = 1 3 Boston Priority = 10 2 New York Priority = 5 4 Boston Priority = 10 3 New York Priority = 5 5 Boston Priority = 10 6 Boston Priority = 10 7 Boston Priority = 10 8 Boston Priority = 10 9 Boston Priority = 10 Done! Boston 4 New York Priority = 5 5 New York Priority = 5 6 New York Priority = 5 7 New York Priority = 5 8 New York Priority = 5 9 New York Priority = 5 Done! New York 3 Seoul Priority = 1 4 Seoul Priority = 1 5 Seoul Priority = 1 6 Seoul Priority = 1 7 Seoul Priority = 1 8 Seoul Priority = 1 9 Seoul Priority = 1 Done! Seoul |

Figure-3.33: Result of running ThreadsPriority application

### **Summary**

In this exercise, you have learned how to retrieve information on a ThreadGroup.