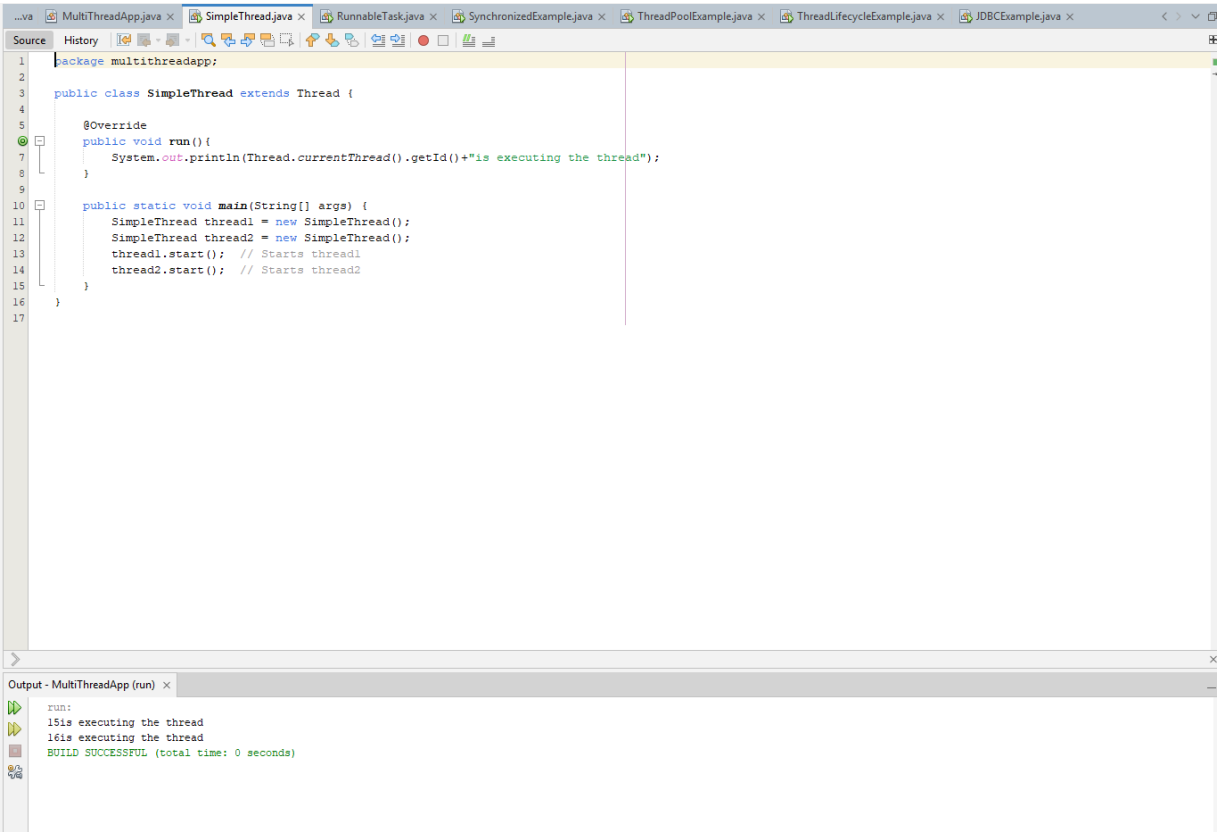
GAM/IT/2022/F/0064 – P. K. D. H. Madushani

Lab Sheet : Multi-threaded Java Application

1. Create a Simple Thread Class

Code:

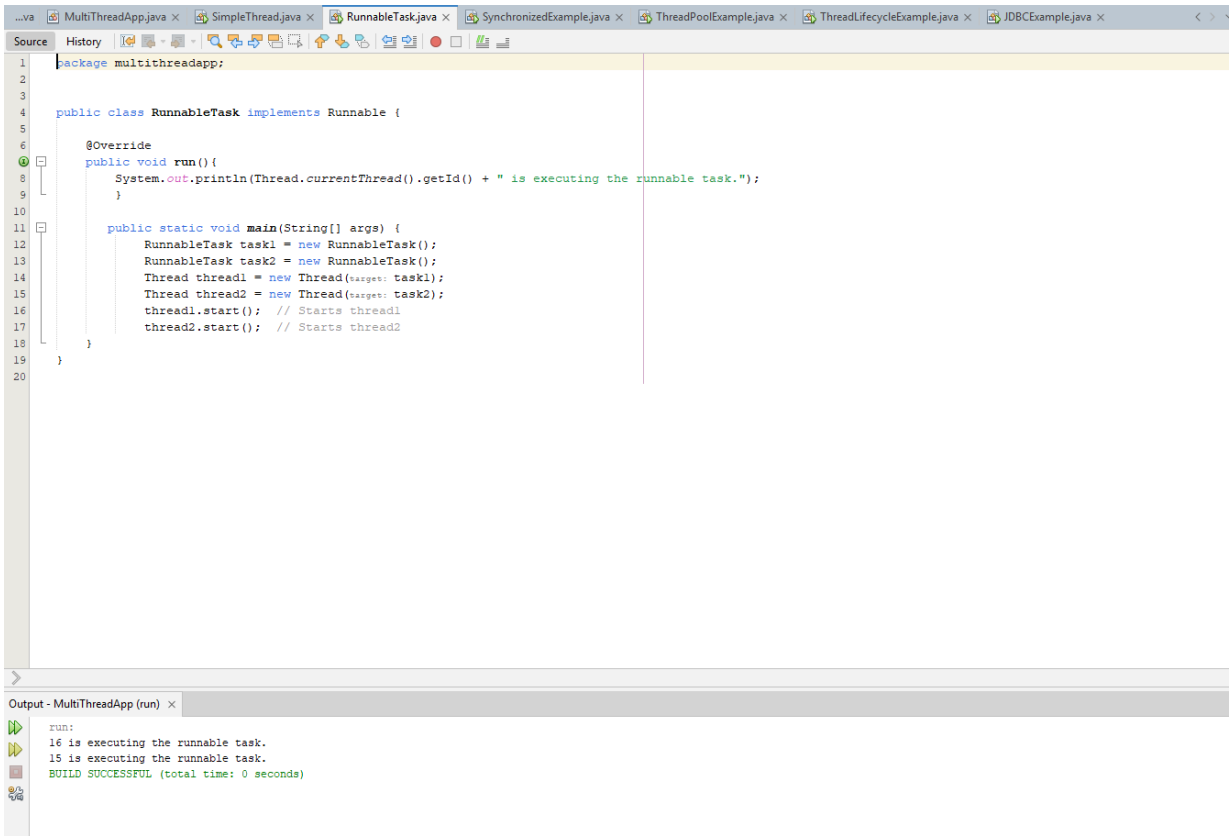
```
public class SimpleThread extends Thread {
@Override
public void run() {
        System.out.println(Thread.currentThread().getId() + " is executing the thread.");
}
public static void main(String[] args) {
        SimpleThread thread1 = new SimpleThread();
        SimpleThread thread2 = new SimpleThread();
        thread1.start();  // Starts thread1
        thread2.start();  // Starts thread2
}
}
```



2. Create a Runnable Class

Code:

```
public class RunnableTask implements Runnable {
@Override
public void run() {
System.out.println(Thread.currentThread().getId() + " is executing the runnable task.");
}
public static void main(String[] args) {
        RunnableTask task1 = new RunnableTask();
        RunnableTask task2 = new RunnableTask();
        Thread thread1 = new Thread(task1);
        Thread thread2 = new Thread(task2);
        thread1.start();  // Starts thread1
        thread2.start();  // Starts thread2
}
}
```



3. Synchronizing Shared Resources

Code:

```
class Counter {
private int count = 0;
// Synchronized method to ensure thread-safe access to the counter
public synchronized void increment() {
count++;
}
public int getCount() {
return count;
}
}
public class SynchronizedExample extends Thread {
	private Counter counter;
public SynchronizedExample(Counter counter) {
	this.counter = counter;
}
@Override
public void run() {
	for (int i = 0; i < 1000; i++) {
	counter.increment();
}
}
public static void main(String[] args) throws InterruptedException {
	Counter counter = new Counter();
// Create and start multiple threads
	Thread thread1 = new SynchronizedExample(counter);
	Thread thread2 = new SynchronizedExample(counter);
	thread1.start();
```

```
        thread2.start();
    // Wait for threads to finish
        thread1.join();
        thread2.join();
        System.out.println("Final counter value: " + counter.getCount());
    }
}
```

...va | MultiThreadApp.java | SimpleThread.java | RunnableTask.java | SynchronizedExample.java | ThreadPoolExample.java | ThreadLifecycleExample.java | JDBCExample.java

Source | History

```
1  package multithreadapp;
2
3  class Counter {
4          private int count = 0;
5  // Synchronized method to ensure thread-safe access to the counter
6  public synchronized void increment() {
7  count++;
8  }
9  public int getCount() {
10 return count;
11 }
12 }
13 public class SynchronizedExample extends Thread {
14         private Counter counter;
15
16 public SynchronizedExample(Counter counter) {
17 this.counter = counter;
18 }
19
20 @Override
21 public void run() {
22 for (int i = 0; i < 1000; i++) {
23 counter.increment();
24 }
25 }
26 public static void main(String[] args) throws InterruptedException {
27         Counter counter = new Counter();
28
29         // Create and start multiple threads
30         Thread thread1 = new SynchronizedExample(counter);
31         Thread thread2 = new SynchronizedExample(counter);
32         thread1.start();
33         thread2.start();
34         // Wait for threads to finish
35         thread1.join();
36         thread2.join();
37         System.out.println("Final counter value: " + counter.getCount());
38         }
39 }
40
```

multithreadapp.Counter

Output - MultiThreadApp (run)

```
run:
Final counter value: 2000
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. Using ExecutorService for Thread Pooling

Code:

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
class Task implements Runnable {
private int taskId;
public Task(int taskId) {
        this.taskId = taskId;
@Override
public void run() {
        System.out.println("Task " + taskId + " is being processed by " +
Thread.currentThread().getName());
}
}
public class ThreadPoolExample {
public static void main(String[] args) {
// Create a thread pool with 3 threads
        ExecutorService executorService = Executors.newFixedThreadPool(3);
// Submit tasks to the pool
for (int i = 1; i <= 5; i++) {
executorService.submit(new Task(i));
}
// Shutdown the thread pool
executorService.shutdown();
}
}
```

```java
package multithreadapp;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class Task implements Runnable {
private int taskId;

public Task(int taskId) {
this.taskId = taskId;
}

@Override
public void run() {
System.out.println("Task " + taskId + " is being processed by " + Thread.currentThread().getName());
}
}

public class ThreadPoolExample {
    public static void main(String[] args) {
// Create a thread pool with 3 threads
ExecutorService executorService = Executors.newFixedThreadPool(nThreads: 3);
// Submit tasks to the pool
for (int i = 1; i <= 5; i++) {
executorService.submit(new Task(taskId: i));
}
// Shutdown the thread pool
executorService.shutdown();
}
}
```

```
run:
Task 3 is being processed by pool-1-thread-3
Task 2 is being processed by pool-1-thread-2
Task 4 is being processed by pool-1-thread-3
Task 1 is being processed by pool-1-thread-1
Task 5 is being processed by pool-1-thread-3
BUILD SUCCESSFUL (total time: 0 seconds)
```

5. Thread Lifecycle Example

Code:

```
public class ThreadLifecycleExample extends Thread {
@Override
public void run() {
        System.out.println(Thread.currentThread().getName() + " - State: " +
Thread.currentThread().getState());
try {
Thread.sleep(2000); // Simulate waiting state
}
catch (InterruptedException e) {
e.printStackTrace();
}
```

```
System.out.println(Thread.currentThread().getName() + " - State after sleep: " +
Thread.currentThread().getState());

}

public static void main(String[] args) {

        ThreadLifecycleExample thread = new ThreadLifecycleExample();

        System.out.println(thread.getName() + " - State before start: " + thread.getState());

        thread.start();  // Start the thread

        System.out.println(thread.getName() + " - State after start: " + thread.getState());

}

}
```

```
...va  MultiThreadApp.java  SimpleThread.java  RunnableTask.java  SynchronizedExample.java  ThreadPoolExample.java  ThreadLifecycleExample.java  JDBCExample.java
Source  History
1   package multithreadapp;
2
3   public class ThreadLifecycleExample  extends Thread {
4   @Override
6   public void run() {
6   System.out.println(Thread.currentThread().getName() + " - State: " + Thread.currentThread().getState());
7
8   try {
9   Thread.sleep(millis: 2000); // Simulate waiting state
10  } catch (InterruptedException e) {
12  e.printStackTrace();
12  }
13  System.out.println(Thread.currentThread().getName() + " - State after sleep: " + Thread.currentThread().getState());
14  }
15
16  public static void main(String[] args) {
17  ThreadLifecycleExample thread = new ThreadLifecycleExample();
18  System.out.println(thread.getName() + " - State before start: " + thread.getState());
19  thread.start();  // Start the thread
21  System.out.println(thread.getName() + " - State after start: " + thread.getState());
21  }
22  }
23
24
multithreadapp.ThreadLifecycleExample  main
Output - MultiThreadApp (run)
run:
Thread-0 - State before start: NEW
Thread-0 - State after start: RUNNABLE
Thread-0 - State: RUNNABLE
Thread-0 - State after sleep: RUNNABLE
BUILD SUCCESSFUL (total time: 2 seconds)
```