Enterprise Architecture – 4232

Lab sheet 02 Task 01

1. **Create multiththreadapp**

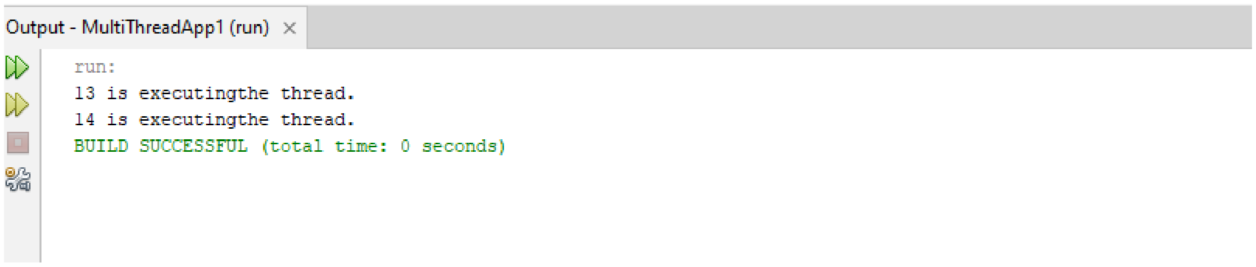
```
package multithreadapp1;

public class MultiThreadApp1 {
   public static void main(String[] args) {
   }
   }
```

Create simplethread class

```
package multithreadapp1;
public class SimpleThread extends Thread {
   @Override
public void run() {
System.out.println(Thread.currentThread().getId() + " is executingthe thread.");
}
public static void main(String[] args) {
SimpleThread thread1 = new SimpleThread();
SimpleThread thread2 = new SimpleThread();
thread1.start(); // Starts thread1
thread2.start(); // Starts thread2
}}
```

Output



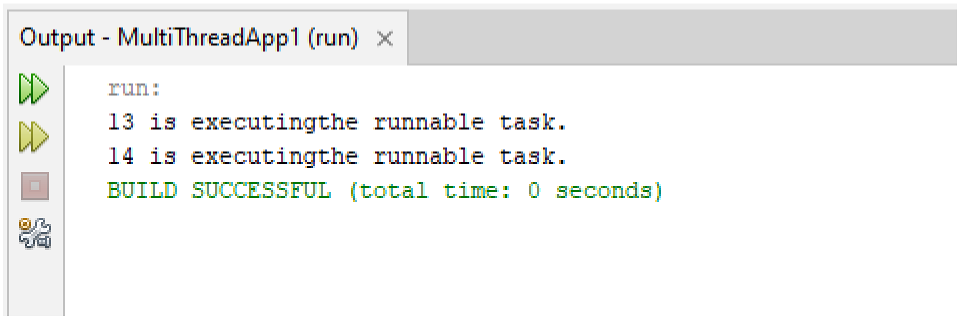
```
run:
13 is executingthe thread.
14 is executingthe thread.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Task 02

2.create runnable task class

```
public class RunnableTask implements Runnable{
@Override
public void run() {
System.out.println(Thread.currentThread().getId() + " is executingthe 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
}
}
```

Output



```
run:
13 is executingthe runnable task.
14 is executingthe runnable task.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Task 03

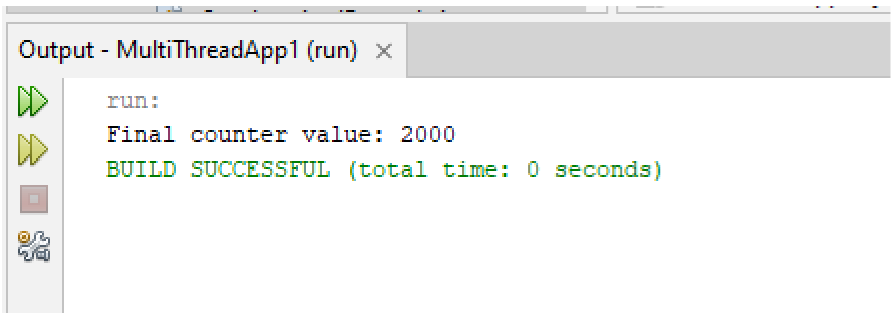
3.counter class

```
public 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;
}}
```

SynchronizedExample class

```
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());
}}
```

Output



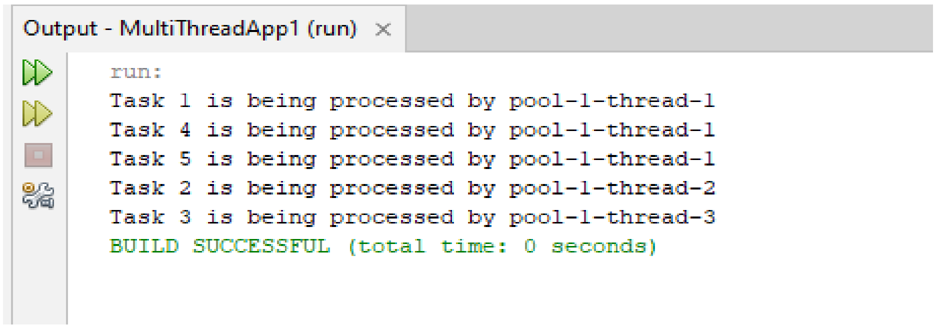
```
run:
Final counter value: 2000
BUILD SUCCESSFUL (total time: 0 seconds)
```

Task 04

4. create ThreadPoolExample.java. class

```
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();
}}
```

Output



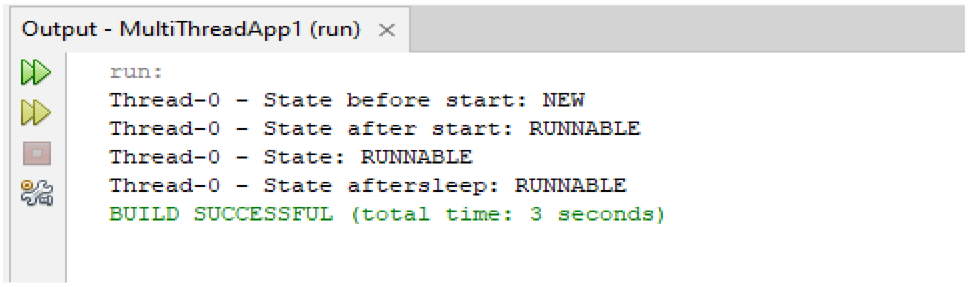
```
run:
Task 1 is being processed by pool-1-thread-1
Task 4 is being processed by pool-1-thread-1
Task 5 is being processed by pool-1-thread-1
Task 2 is being processed by pool-1-thread-2
Task 3 is being processed by pool-1-thread-3
BUILD SUCCESSFUL (total time: 0 seconds)
```

Task 05

5.Create ThreadLifecycleExample class

```
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 aftersleep: " +
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());
}}
```

Output



```
run:
Thread-0 - State before start: NEW
Thread-0 - State after start: RUNNABLE
Thread-0 - State: RUNNABLE
Thread-0 - State aftersleep: RUNNABLE
BUILD SUCCESSFUL (total time: 3 seconds)
```