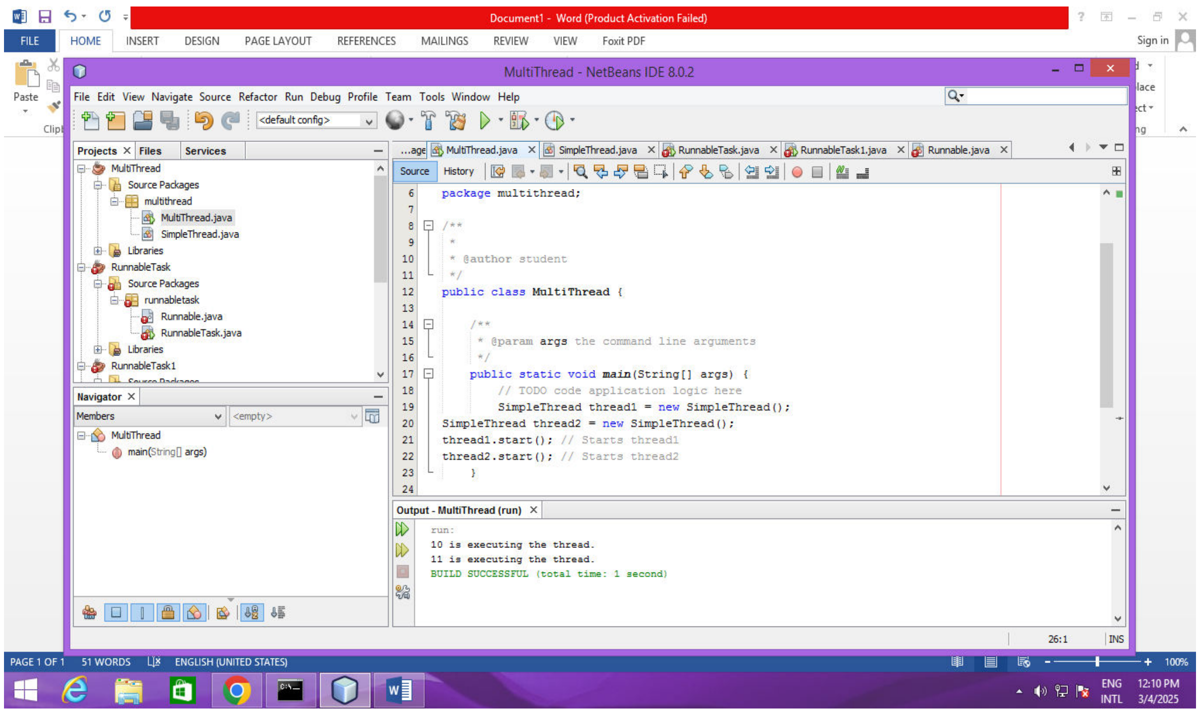
GAM/IT/2022/F/0033

LAB2-TASK 01

```
public class SimpleThread extends Thread {


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();

thread2.start();

   }

   }
```

OUTPUT



```
package multithread;

/**
 *
 * @author student
 */
public class MultiThread {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        SimpleThread thread1 = new SimpleThread();
SimpleThread thread2 = new SimpleThread();
thread1.start(); // Starts thread1
thread2.start(); // Starts thread2
    }
```

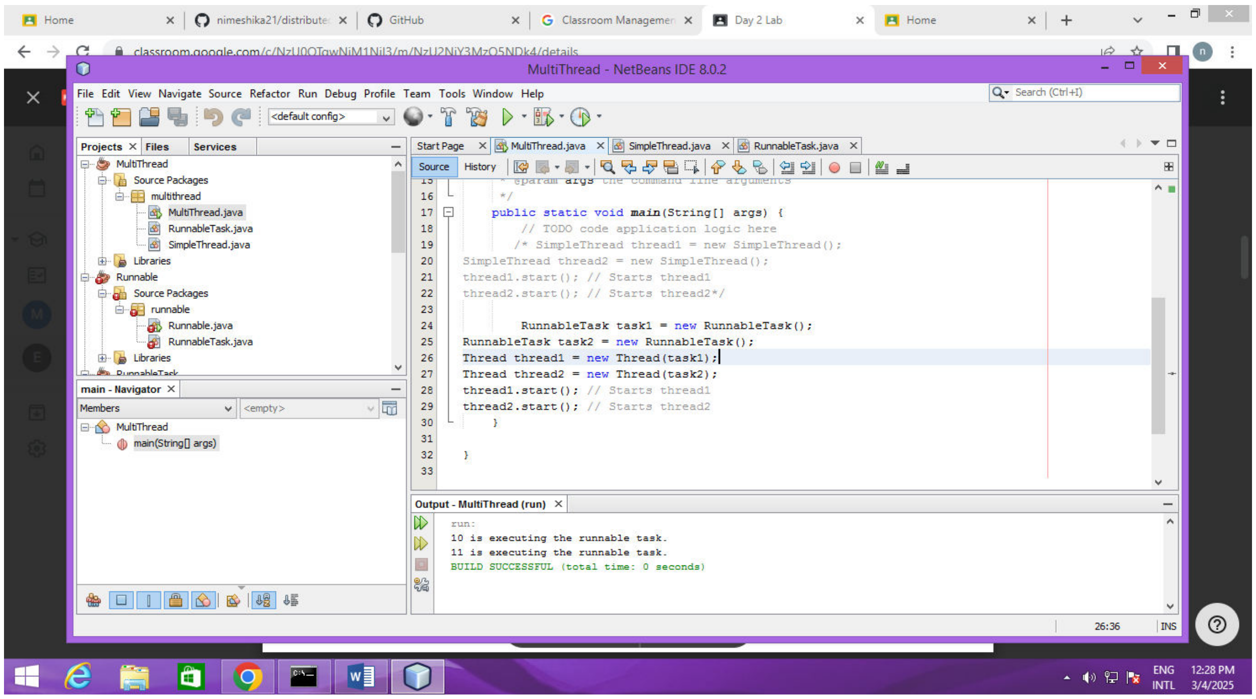
```
run:
10 is executing the thread.
11 is executing the thread.
BUILD SUCCESSFUL (total time: 1 second)
```

Document1 - Word (Product Activation Failed)
MultiThread - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
MultiThread
Source Packages
multithread
MultiThread.java
SimpleThread.java
Libraries
RunnableTask
runnabletask
Runnable.java
RunnableTask.java
RunnableTask1
Navigator
MultiThread
main(String[] args)
MultiThread.java SimpleThread.java RunnableTask.java RunnableTask1.java Runnable.java
Output - MultiThread (run)
PAGE 1 OF 1 51 WORDS ENGLISH (UNITED STATES)
12:10 PM 3/4/2025

2) Lab02- TASK 02

```
public class RunnableTask implements Runnable {

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();

thread2.start();

   }

   }
```

OUTPUT



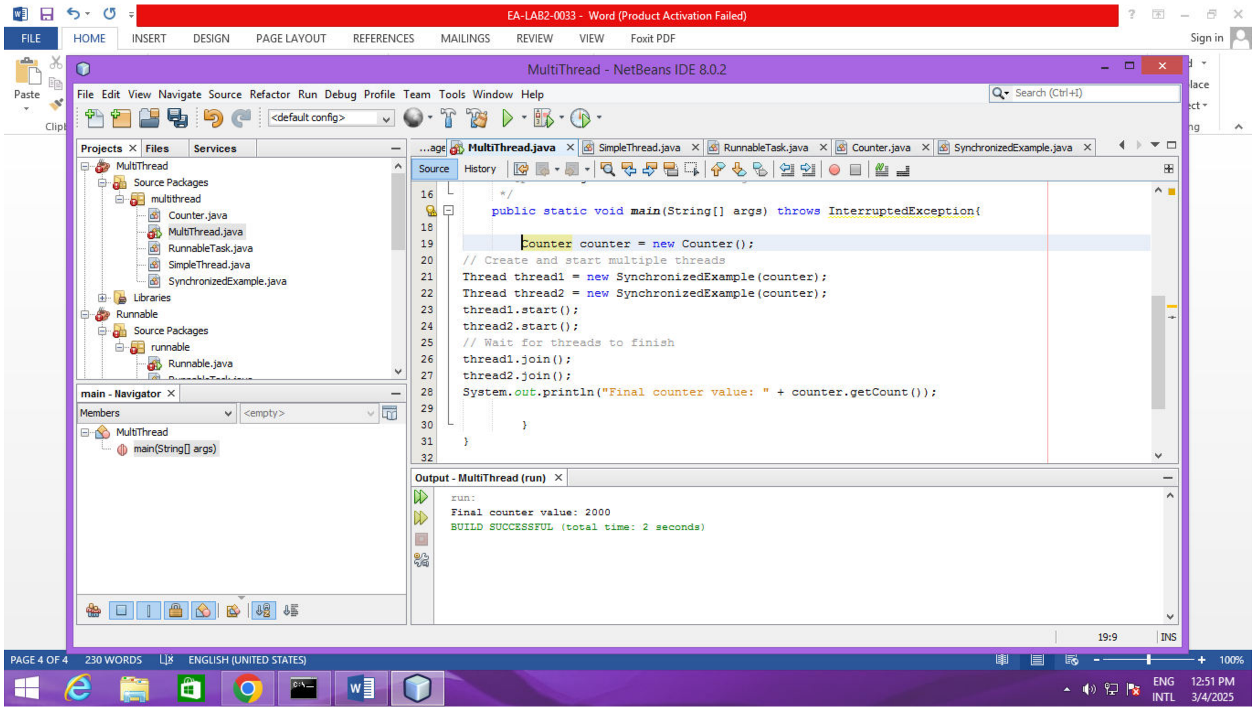
```
         * @param args the command line arguments
         */
        public static void main(String[] args) {
            // TODO code application logic here
           /* SimpleThread thread1 = new SimpleThread();
SimpleThread thread2 = new SimpleThread();
thread1.start(); // Starts thread1
thread2.start(); // Starts thread2*/

            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
        }

}
```

```
run:
10 is executing the runnable task.
11 is executing the runnable task.
BUILD SUCCESSFUL (total time: 0 seconds)
```

3) LAB03-TASK-03

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

thread1.join();
thread2.join();
System.out.println("Final counter value: " + counter.getCount());  } }
```

OUTPUT



```
    */
    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 - MultiThread (run)

```
run:
Final counter value: 2000
BUILD SUCCESSFUL (total time: 2 seconds)
```

4) Lab02-task-04

package multithread;

importjava.util.concurrent.ExecutorService;

importjava.util.concurrent.Executors;

class Task implements Runnable {

privateinttaskId;

public Task(inttaskId) {

this.taskId = taskId;

}

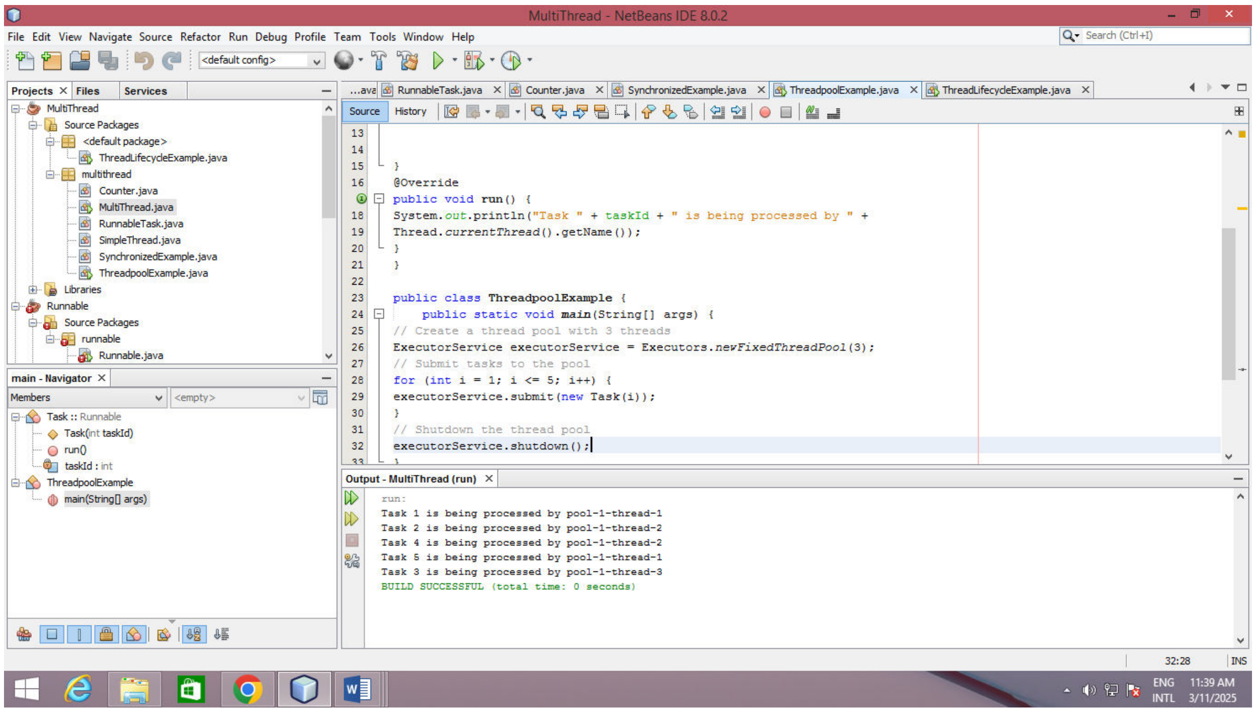
@Override

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
ExecutorServiceexecutorService = Executors.newFixedThreadPool(3);
// Submit tasks to the pool
for (inti = 1; i<= 5; i++) {
executorService.submit(new Task(i));
}
// Shutdown the thread pool
executorService.shutdown();
}
}
```

OUTPUT



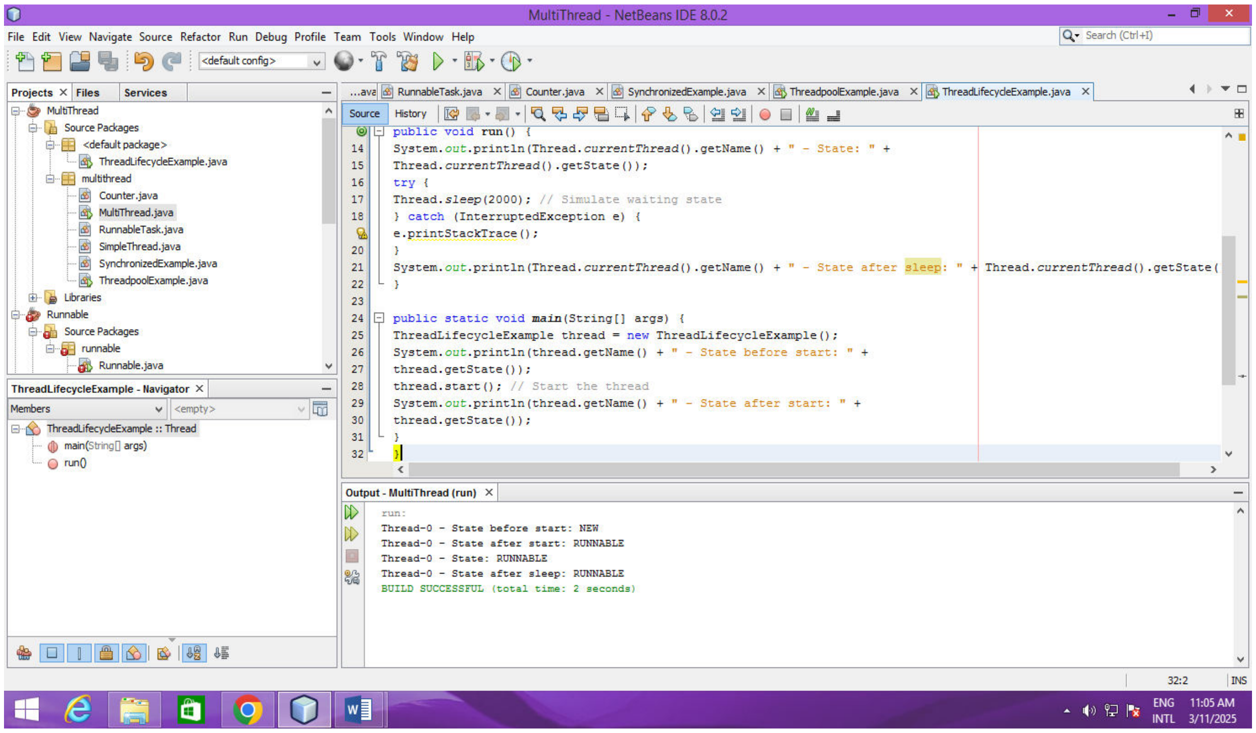
```
    }
@Override
public void run() {
System.out.println("Task " + taskId + " is being processed by " +
Thread.currentThread().getName());
}
}

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();
```

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

5) Lab02- TASK 05

```
public class ThreadLifecycleExample extends Thread{
   @Override
public void run() {
System.out.println(Thread.currentThread().getName() + " - State: " +
Thread.currentThread().getState());
try {
Thread.sleep(2000);
} 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();
System.out.println(thread.getName() + " - State after start: " +
thread.getState());
}
}
```

OUTPUT



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

Output - MultiThread (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)
```