STIWK3014 REAL TIME PROGRAMMING
Tutorial / Exercise 7: Atomic and Deadlocks

Tasks 1:
Run the coding below and screenshot the outcome.

```java
public class AtomicAssignment implements Runnable {
        private static final SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd hh:mm:
            ss:SSS");
        private static Map<String, String> configuration = new HashMap<String, String>();

        public void run() {
                for (int i = 0; i < 10000; i++) {
                        Map<String, String> currConfig = configuration;
                        String value1 = currConfig.get("key-1");
                        String value2 = currConfig.get("key-2");
                        String value3 = currConfig.get("key-3");
                        if (!(value1.equals(value2) && value2.equals(value3))) {
                                throw new IllegalStateException("Values are not equal.");
                        }
                        try {
                                Thread.sleep(10);
                        } catch (InterruptedException e) {
                                e.printStackTrace();
                        }
                }
        }

        public static void readConfig() {
                Map<String, String> newConfig = new HashMap<String, String>();
                Date now = new Date();
                newConfig.put("key-1", sdf.format(now));
                newConfig.put("key-2", sdf.format(now));
                newConfig.put("key-3", sdf.format(now));
                configuration = newConfig;
        }

        public static void main(String[] args) throws InterruptedException {
                readConfig();
                Thread configThread = new Thread(new Runnable() {
                        public void run() {
                                for (int i = 0; i < 10000; i++) {
                                        readConfig();
                                        try {
                                                Thread.sleep(10);
                                        } catch (InterruptedException e) {
                                                e.printStackTrace();
                                        }
                                }
                        }
                }, "configuration-thread");
                configThread.start();
                Thread[] threads = new Thread[5];
                for (int i = 0; i < threads.length; i++) {
                        threads[i] = new Thread(new AtomicAssignment(), "thread-" + i);
                        threads[i].start();
                }
                for (int i = 0; i < threads.length; i++) {
                        threads[i].join();
                }
                configThread.join();
                System.out.println("[" + Thread.currentThread().getName() + "] All threads
                    have finished.");
        }
}
```

```java
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

public class AtomicAssignment implements Runnable {
    private static final SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss");  3 usages
    private static Map<String, String> configuration = new HashMap<>();  2 usages

    public void run() {
        for (int i = 0; i < 10000; i++) {
            Map<String, String> currConfig = configuration;
            String value1 = currConfig.get("key-1");
            String value2 = currConfig.get("key-2");
            String value3 = currConfig.get("key-3");
            if (! (value1.equals(value2) && value2.equals(value3))) {
                throw new IllegalStateException("Value are not equal.");
            }
            try {
                Thread.sleep( millis: 10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
```

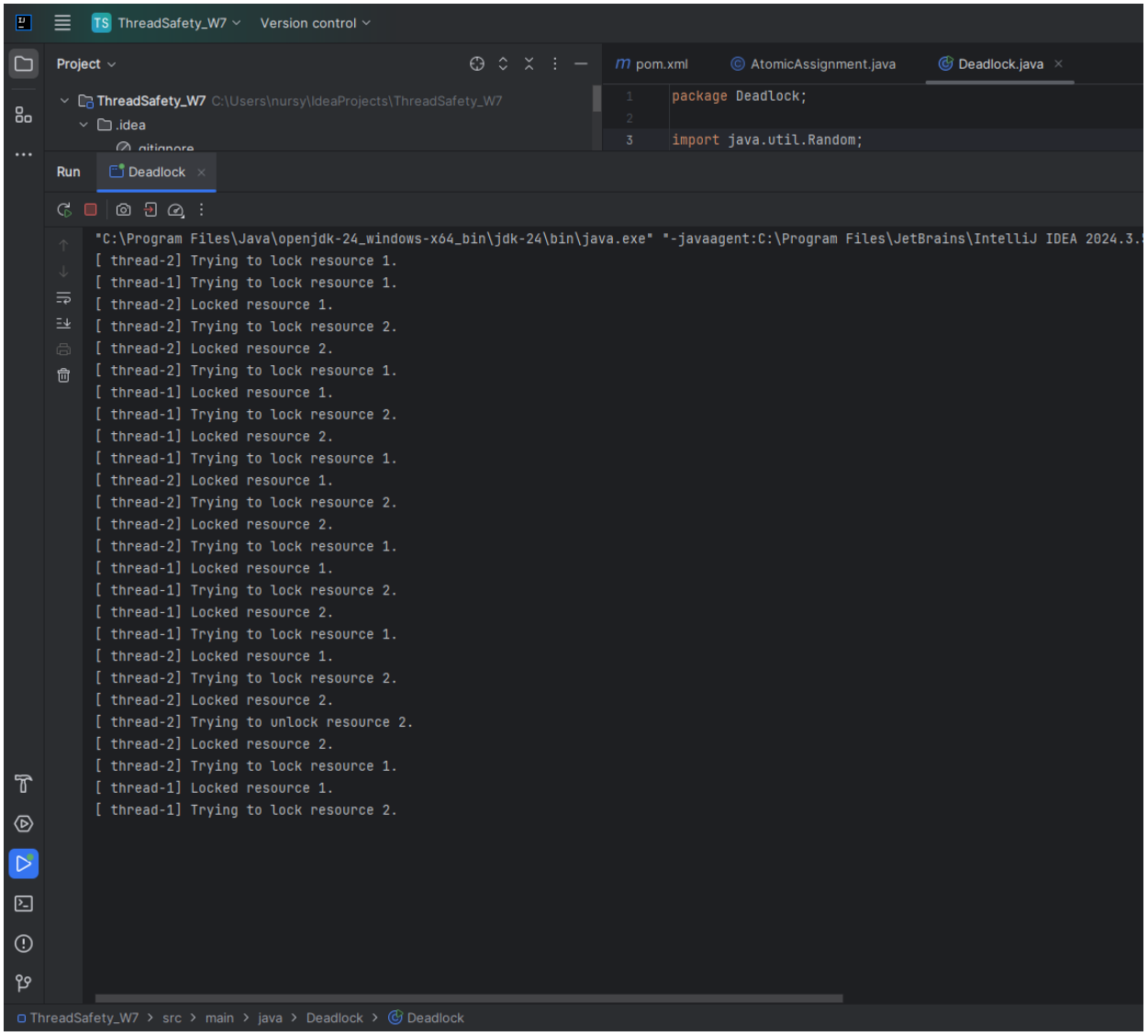
```
"C:\Program Files\Java\openjdk-24_windows-x64_bin\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.5\lib\idea_rt.jar=56067" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.
(main) All threads have finished.

Process finished with exit code 0
```

```
"C:\Program Files\Java\openjdk-24_windows-x64_bin\jdk-24\bin\j
(main) All threads have finished.

Process finished with exit code 0
```

Tasks 2:

- Run the coding below and screenshot the outcome. For the submission (Rename: Output from sample coding)
- Modify and implement some kind of thread monitoring to avoid the deadlock situation and attach the new modification of sample coding with the new outcome. For the submission (Rename: Sample of new modification and the new output).

```
public class Deadlock implements Runnable {
        private static final Object resource1 = new Object();
        private static final Object resource2 = new Object();
        private final Random random = new Random(System.currentTimeMillis());

        public static void main(String[] args) {
                Thread myThread1 = new Thread(new Deadlock(), "thread-1");
                Thread myThread2 = new Thread(new Deadlock(), "thread-2");
                myThread1.start();
                myThread2.start();
        }

        public void run() {
                for (int i = 0; i < 10000; i++) {
                        boolean b = random.nextBoolean();
                        if (b) {
                                System.out.println("[" + Thread.currentThread().getName() + ↩
                                     "] Trying to lock resource 1.");
                                synchronized (resource1) {
                                        System.out.println("[" + Thread.currentThread(). ↩
                                            getName() + "] Locked resource 1.");
                                        System.out.println("[" + Thread.currentThread(). ↩
                                            getName() + "] Trying to lock resource 2.");
                                        synchronized (resource2) {
                                                System.out.println("[" + Thread. ↩
                                                    currentThread().getName() + "] Locked  ↩
                                                    resource 2.");
                                        }
                                }
                        } else {
                                System.out.println("[" + Thread.currentThread().getName() + ↩
                                     "] Trying to lock resource 2.");
                                synchronized (resource2) {
                                        System.out.println("[" + Thread.currentThread(). ↩
                                            getName() + "] Locked resource 2.");
                                        System.out.println("[" + Thread.currentThread(). ↩
                                            getName() + "] Trying to lock resource 1.");
                                        synchronized (resource1) {
                                                System.out.println("[" + Thread. ↩
                                                    currentThread().getName() + "] Locked  ↩
                                                    resource 1.");
                                        }
                                }
                        }
                }
        }
}
```


ThreadSafety_W7
Version control
Project
ThreadSafety_W7 C:\Users\nursy\IdeaProjects\ThreadSafety_W7
.idea
pom.xml
AtomicAssignment.java
Deadlock.java
package Deadlock;
import java.util.Random;
Run
Deadlock
"C:\Program Files\Java\openjdk-24_windows-x64_bin\jdk-24\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.
[thread-2] Trying to lock resource 1.
[thread-1] Trying to lock resource 1.
[thread-2] Locked resource 1.
[thread-2] Trying to lock resource 2.
[thread-2] Locked resource 2.
[thread-2] Trying to lock resource 1.
[thread-1] Locked resource 1.
[thread-1] Trying to lock resource 2.
[thread-1] Locked resource 2.
[thread-1] Trying to lock resource 1.
[thread-2] Locked resource 1.
[thread-2] Trying to lock resource 2.
[thread-2] Locked resource 2.
[thread-2] Trying to lock resource 1.
[thread-1] Locked resource 1.
[thread-1] Trying to lock resource 2.
[thread-1] Locked resource 2.
[thread-1] Trying to lock resource 1.
[thread-2] Locked resource 1.
[thread-2] Trying to lock resource 2.
[thread-2] Locked resource 2.
[thread-2] Trying to unlock resource 2.
[thread-2] Locked resource 2.
[thread-2] Trying to lock resource 1.
[thread-1] Locked resource 1.
[thread-1] Trying to lock resource 2.
ThreadSafety_W7 > src > main > java > Deadlock > Deadlock

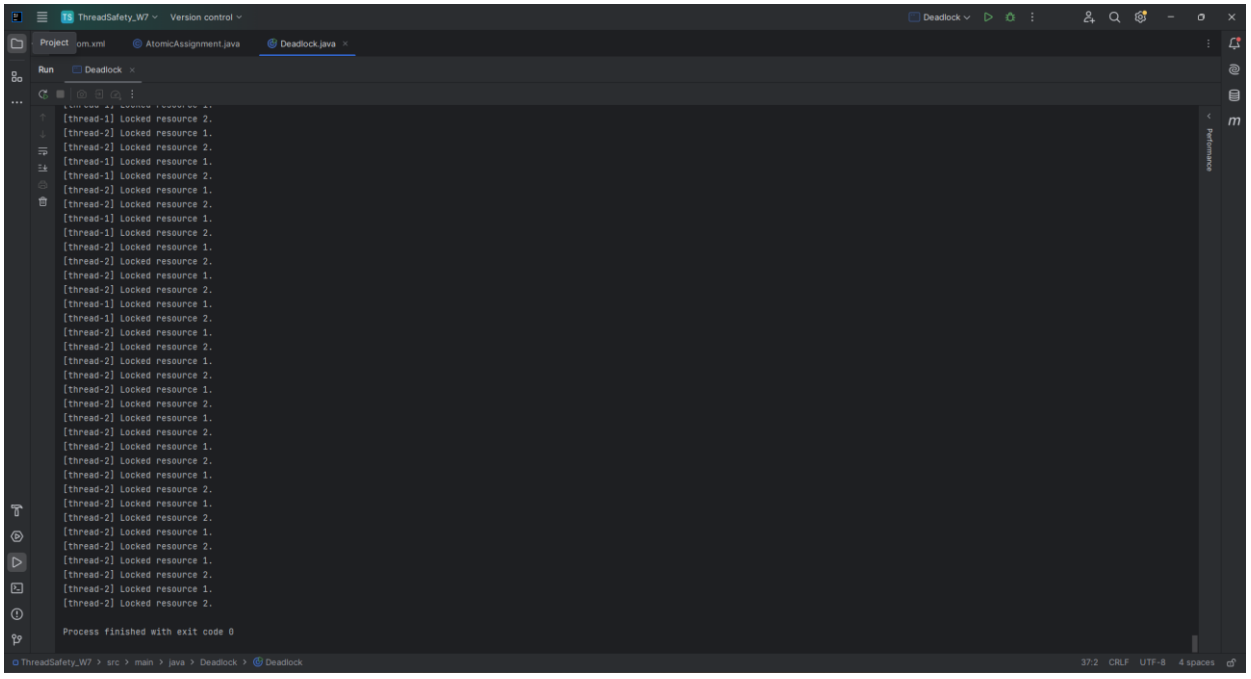
New modification and new output

ThreadSafety_W7 Version control

pom.xml AtomicAssignment.java Deadlock.java

```java
package Deadlock;

import java.util.Random;

public class Deadlock implements Runnable {
    private static final Object resource1 = new Object();  1 usage
    private static final Object resource2 = new Object();  1 usage
    private final Random random = new Random(System.currentTimeMillis());  1 usage

    public static void main(String[] args) {
        Thread myThread1 = new Thread(new Deadlock(),  name: "thread-1");
        Thread myThread2 = new Thread(new Deadlock(),  name: "thread-2");
        myThread1.start();
        myThread2.start();
    }
    public void run() {
        for (int i = 0; i < 10000; i++) {
            boolean b = random.nextBoolean();

            Object lock1 = resource1;
            Object lock2 = resource2;

            synchronized (lock1) {
                System.out.println("[" + Thread.currentThread().getName() + "] Locked resource 1.");
                synchronized (lock2) {
                    System.out.println("[" + Thread.currentThread().getName() + "] Locked resource 2.");
                }
            }

            try {
                Thread.sleep( millis: 1);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

ThreadSafety_W7 > src > main > java > Deadlock > Deadlock > run



Plagiarism

No mark will be given for plagiarism activities.

Submission:

Platform: 1. Online Learning - Sample Coding & Output in PdF form
2. GitHub – Upload the file and attach your GitHub link repositories.

Date: 30 April 2025 (Wednesday, before 12.30 noon)