Task 1

```
package Week_07;

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

public class AtomicAssignment implements Runnable{
    private static final SimpleDateFormat sdf = new SimpleDateFormat("yyyy-
MM-dd hh:mm:ss:SS");
    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.");
    }



}
```

output :

"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.5\lib\idea_rt.jar=56964" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath C:\Users\User\IdeaProjects\STIWK3014\SimpleThreading\target\classes Week_07.AtomicAssignment

[main] All threads have finished.

Process finished with exit code 0

Task 2

```
package Week_07;

import java.util.Random;

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");
                }
                synchronized (resource2) {
                    System.out.println("[" +
Thread.currentThread().getName() + "] Trying to lock resource 2");
                    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");
                    }
                }
            }
        }
    }
}
```

output:

[thread-2] Locked resource 2

[thread-2] 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-2] Locked resource 1

[thread-2] Trying to lock resource 2

[thread-2] Locked resource 2

[thread-2] 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-2] Locked resource 1

[thread-2] Trying to lock resource 2

[thread-2] Locked resource 2

[thread-2] Trying to lock resource 2

[thread-2] Locked resource 2

[thread-2] 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-2] Locked resource 1

[thread-2] Trying to lock resource 2

[thread-2] Locked resource 2

[thread-2] Trying to lock resource 1

[thread-2] Locked resource 1

[thread-2] 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 2

[thread-2] Locked resource 2

[thread-2] Trying to lock resource 1

[thread-2] Locked resource 1

[thread-2] Trying to lock resource 1)

[thread-2] Locked resource 1

[thread-2] Trying to lock resource 2

[thread-2] Locked resource 2

Process finished with exit code 0