NUR AZYYATI BITNI ABU BAKAR | 291560

TUTORIAL 7 : THE CODE

Task1 AtomicAssignment.java

```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<String, String>();  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("Values are not equal");
            }
            try {
                    Thread.sleep( millis: 10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public static void readConfig() {  2 usages
        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() {
            @Override
            public void run() {
                for (int i = 0; i < 10000; i++) {
                    readConfig();
                    try {
                        Thread.sleep( millis: 10);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }, name: "Configuration-Thread");
        configThread.start();
        Thread[] threads = new Thread[5];
        for (int i = 0; i < threads.length; i++) {
            threads[i] = new Thread(new AtomicAssignment(), name: "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.");
    }
}
```

THE OUTPUT:

```
"C:\Program Files\Java\jdk-16.0.1\bin\ja
[main] All threads have finished.

Process finished with exit code 0
```

Task2 Deadlock.java (Output from sample coding)

```java
import java.util.Random;

public class Deadlock implements Runnable {
    private static final Object resource1 = new Object();  2 usages
    private static final Object resource2 = new Object();  2 usages
    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();
            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.");
                    }
                }
            }
        }
    }
}
```

THE OUTPUT: Output from sample coding

```
[thread-1] Trying to lock resource 2.
[thread-1] Locked resource 2.
[thread-1] Trying to lock resource 2.
[thread-1] Locked resource 2.
[thread-1] Trying to lock resource 1.
[thread-1] Locked resource 1.
[thread-1] Trying to lock resource 1.
[thread-1] Locked resource 1.
[thread-1] Trying to lock resource 2.
[thread-2] Trying to lock resource 2.
[thread-1] Locked resource 2.
[thread-1] 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-1] Locked resource 2.
[thread-1] Trying to lock resource 1.
[thread-1] Locked resource 1.
[thread-1] Trying to lock resource 1.
[thread-1] Locked resource 1.
[thread-2] Locked resource 2.
[thread-1] Trying to lock resource 2.
[thread-2] Trying to lock resource 1.
```

Task2 Deadlock.java (Sample of new modification and the new output)

```java
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();
            if (b) {
                lockResourcesInOrder();
            } else {
                lockResourcesInOrder();
            }
        }
    }

    private void lockResourcesInOrder() {  2 usages
        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.");
            }
        }
    }
}
```

THE OUTPUT: Output from new sample coding

```
[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 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.

Process finished with exit code 0
```