EXTERNAL PRACTICAL

Java Programming Lab - ETIT 353

Aim

Write an application that executes two threads. One thread displays "An" every 1000 milliseconds and other displays "B" every 3000 milliseconds. Create the threads by extending the Thread class.

Write an application that displays deadlock between threads.

syeda reeha quasar
14114802719

EXPERIMENT – 1

Aim:

Write an application that executes two threads. One thread displays "An" every 1000 milliseconds and other displays "B" every 3000 milliseconds. Create the threads by extending the Thread class.

Theory:

Class : A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

Threads : Threads allows a program to operate more efficiently by doing multiple things at the same time. Threads can be used to perform complicated tasks in the background without interrupting the main program.

Multithreading : Multithreading in Java is a process of executing multiple threads simultaneously. A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking. However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process. Java Multithreading is mostly used in games, animation, etc.

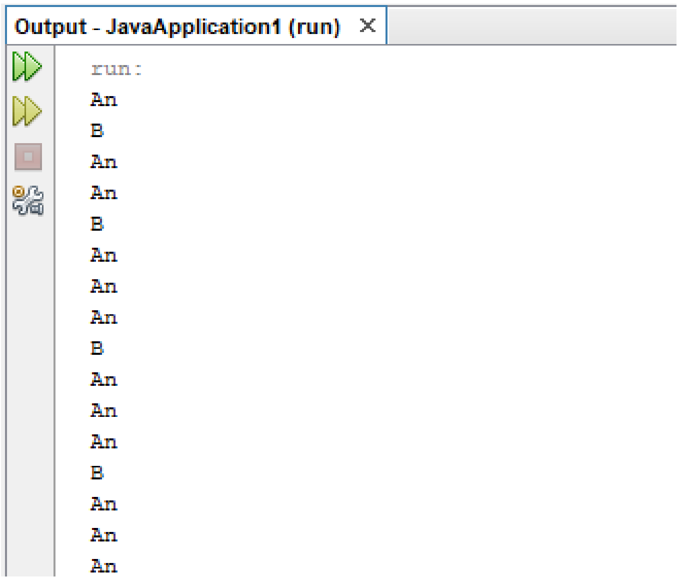
Thread.sleep() : Thread.sleep() method can be used to pause the execution of current thread for specified time in milliseconds. The argument value for

milliseconds can’t be negative, else it throws IllegalArgumentException.

Source Code:

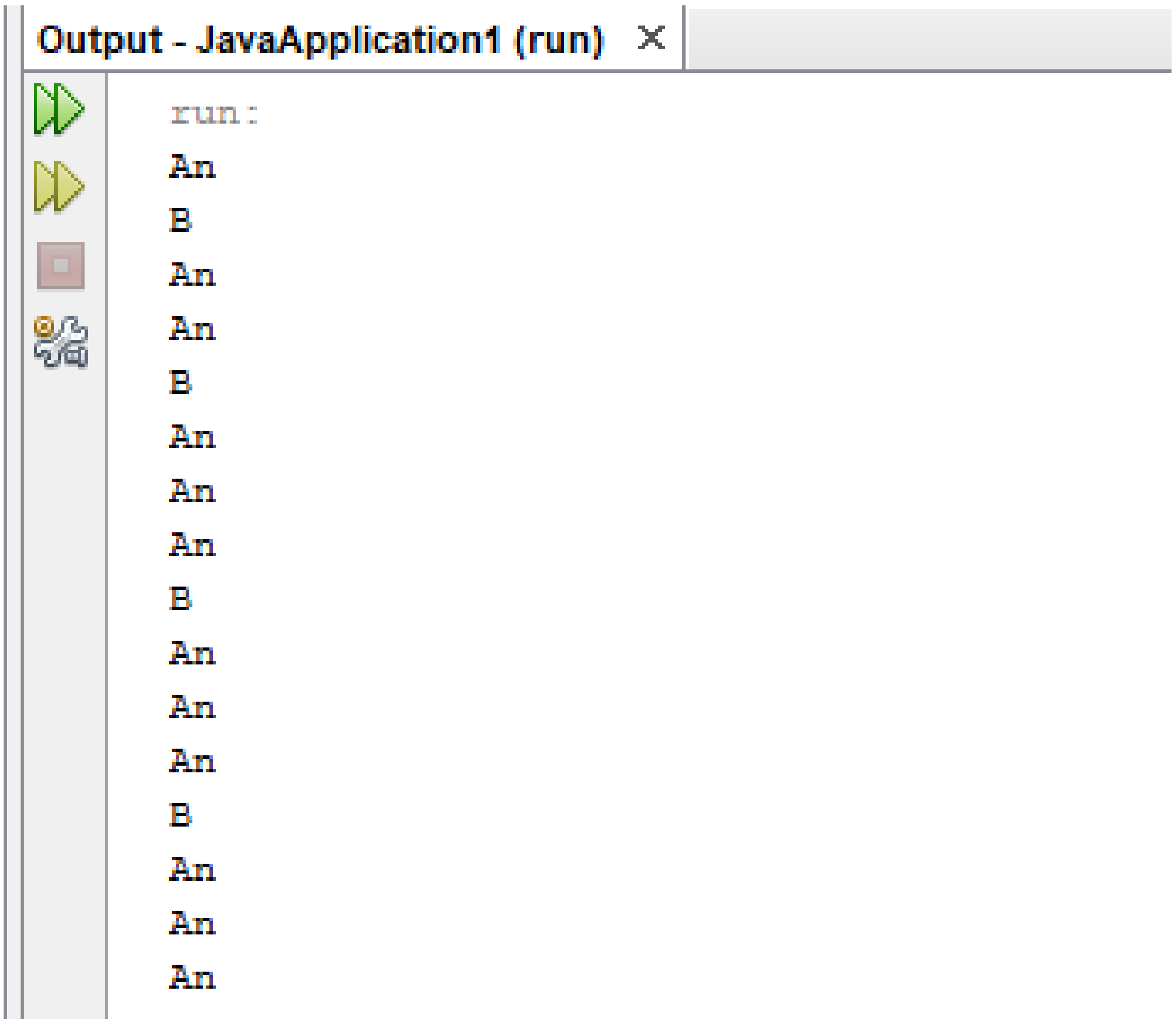
```
package javaapplication1;

/**
 *
 * @author reeha
 */

class A extends Thread {

    public void run() {
        for (;;) {
            System.out.println("An");
            try {
                Thread.currentThread().sleep(1000);
            }

            catch (InterruptedException e) {
            }
        }
    }
}

class B extends Thread {
    public void run() {
        for (;;) {
            System.out.println("B");
            try {
                Thread.currentThread().sleep(3000);
            } catch (InterruptedException e) {
            }
        }
    }
}

public class threading {
    public static void main(String args[]) {

        A obj1 = new A();
        B obj2 = new B();
        obj1.start();
        obj2.start();
    }
}
```

Output:



Output - JavaApplication1 (run)

```
run:
An
B
An
An
B
An
An
An
B
An
An
An
B
An
An
An
```



```
run:
An
B
An
An
B
An
An
An
B
An
An
An
B
An
An
An
```

EXPERIMENT – 2

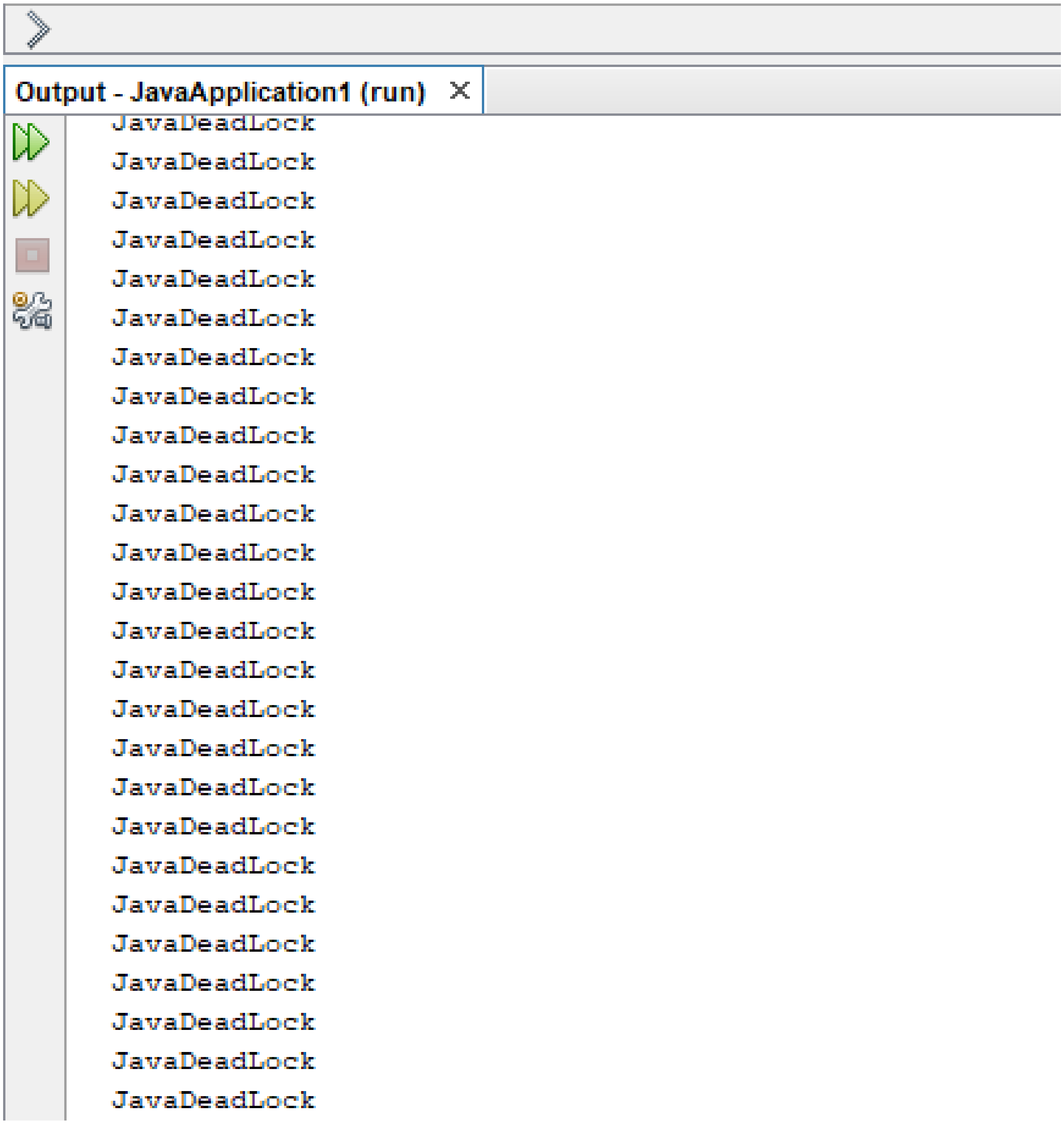
Aim:

Write an application that displays deadlock between threads.

Source Code:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package javaapplication1;

/**
 *
 * @author reeha
 */
public class MyDeadlock {

    String str1 = "Java";
    String str2 = "DeadLock";

    Thread trd1 = new Thread("My Thread 1") {
        public void run() {
            while (true) {
                synchronized (str1) {
                    synchronized (str2) {
                        System.out.println(str1 + str2);
                }
            }
        }
    }
};

Thread trd2 = new Thread("My Thread 2") {
    public void run() {
        while (true) {
            synchronized (str2) {
                synchronized (str1) {
                    System.out.println(str2 + str1);
                }
            }
        }
    }
};

public static void main(String a[]) {
    MyDeadlock obj = new MyDeadlock();
    obj.trd1.start();
    obj.trd2.start();
}
}
```

Output:



```
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
JavaDeadLock
```