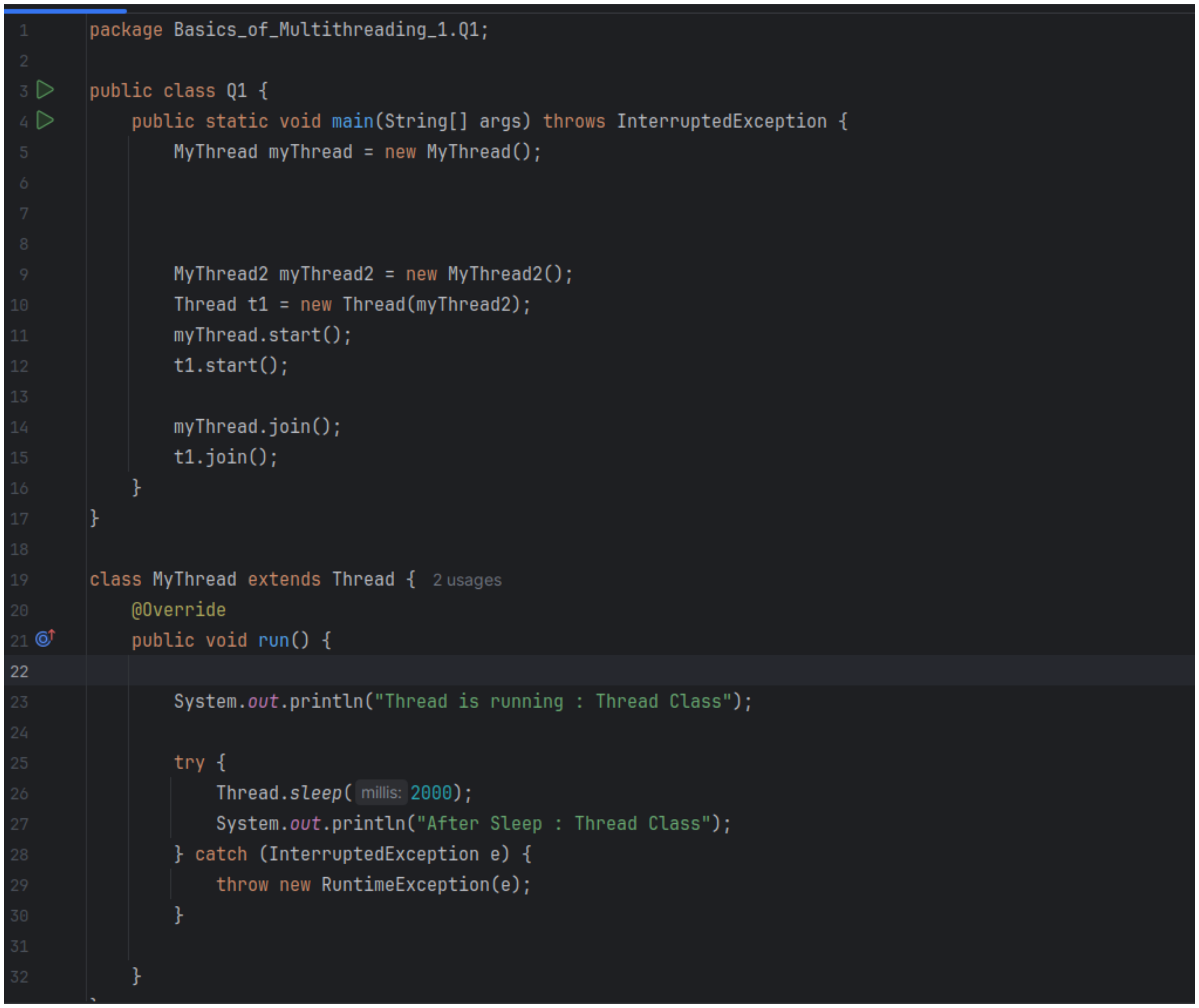
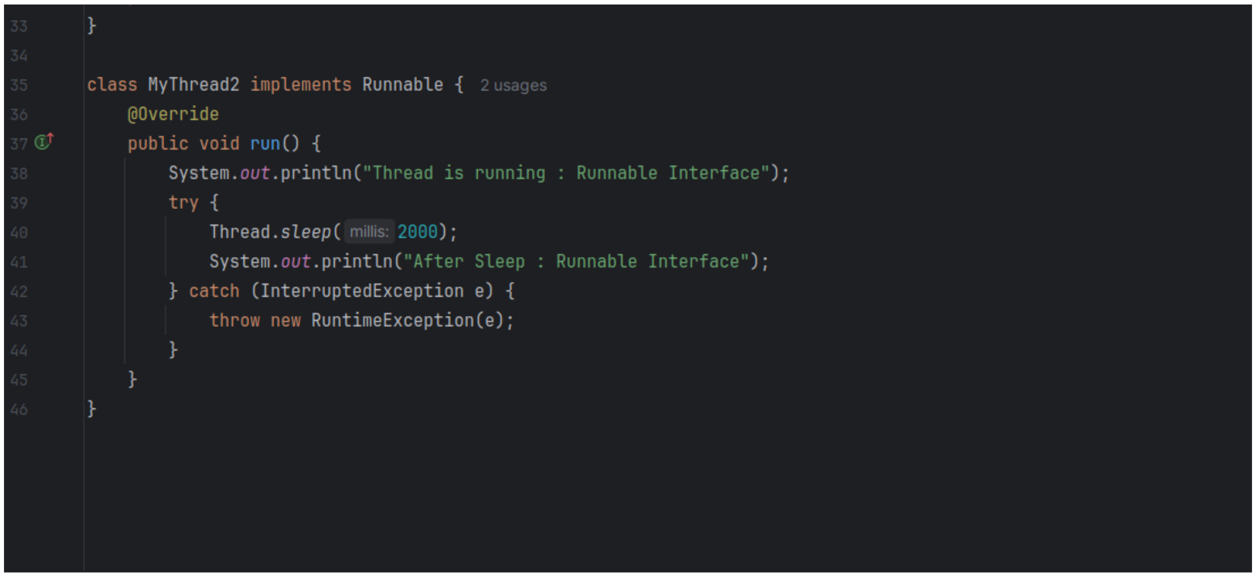
# Assignment - Basics of Multithreading 1

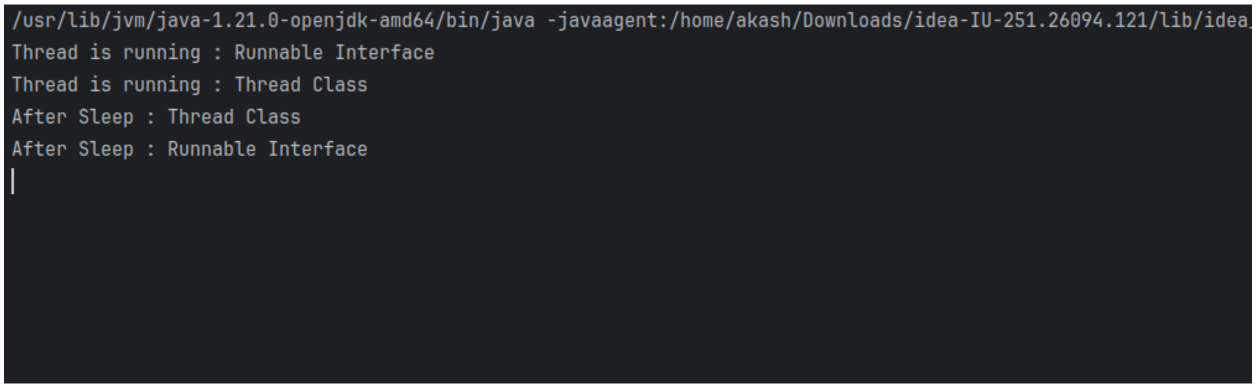
Q1)Create and Run a Thread using Runnable Interface and Thread class and show usage of sleep and join methods in the created threads.



```java
package Basics_of_Multithreading_1.Q1;

public class Q1 {
    public static void main(String[] args) throws InterruptedException {
        MyThread myThread = new MyThread();



        MyThread2 myThread2 = new MyThread2();
        Thread t1 = new Thread(myThread2);
        myThread.start();
        t1.start();

        myThread.join();
        t1.join();
    }
}

class MyThread extends Thread {  2 usages
    @Override
    public void run() {

        System.out.println("Thread is running : Thread Class");

        try {
            Thread.sleep( millis: 2000);
            System.out.println("After Sleep : Thread Class");
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }

    }
```

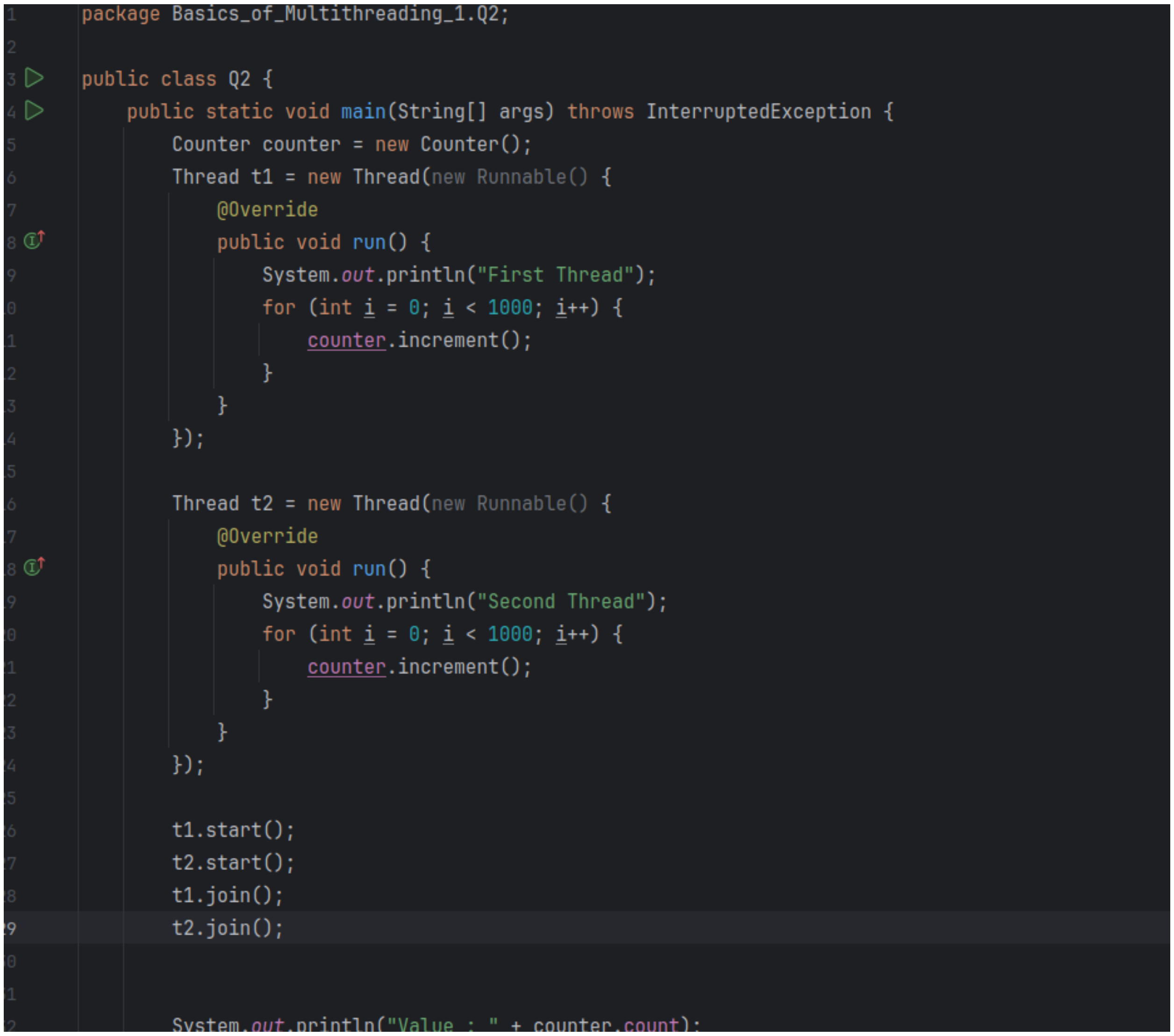


```java
}

class MyThread2 implements Runnable {  2 usages
    @Override
    public void run() {
        System.out.println("Thread is running : Runnable Interface");
        try {
            Thread.sleep( millis: 2000);
            System.out.println("After Sleep : Runnable Interface");
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
}
```

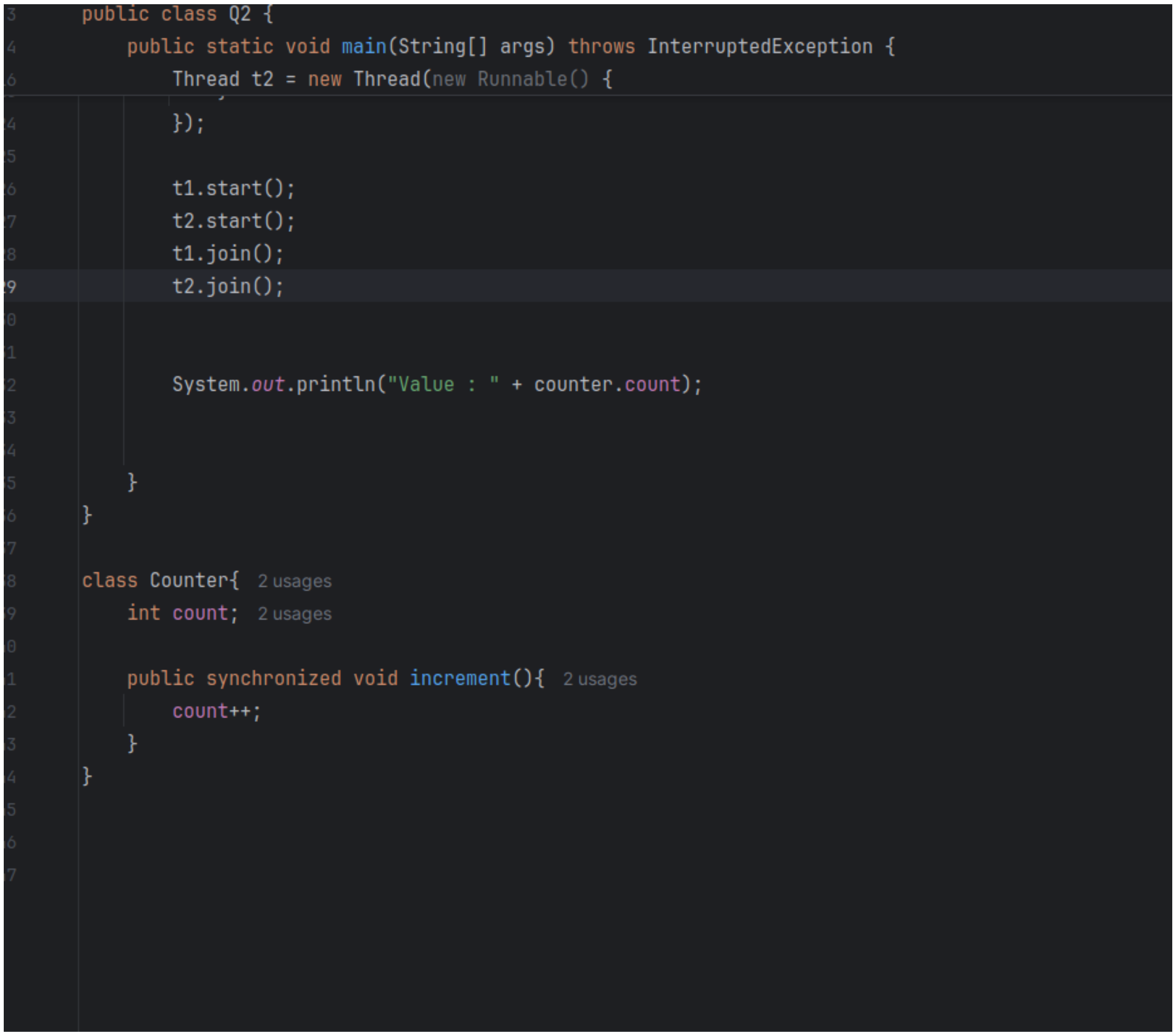


```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/idea_
Thread is running : Runnable Interface
Thread is running : Thread Class
After Sleep : Thread Class
After Sleep : Runnable Interface
```

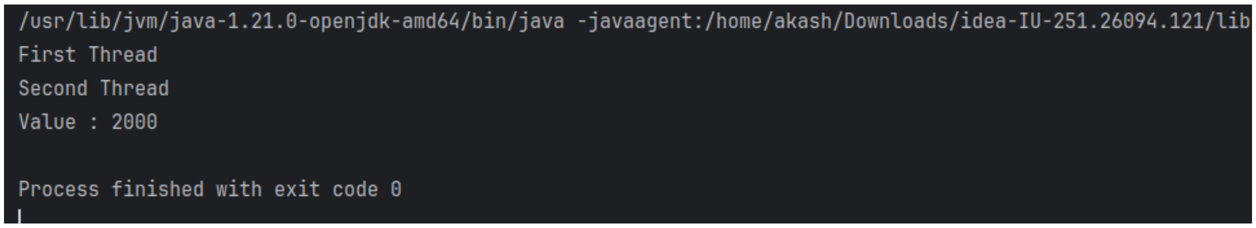
Q2)Use Synchronize method and synchronize block to enable synchronization between multiple threads trying to access method at same time.



```java
package Basics_of_Multithreading_1.Q2;

public class Q2 {
    public static void main(String[] args) throws InterruptedException {
        Counter counter = new Counter();
        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("First Thread");
                for (int i = 0; i < 1000; i++) {
                    counter.increment();
                }
            }
        });

        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("Second Thread");
                for (int i = 0; i < 1000; i++) {
                    counter.increment();
                }
            }
        });

        t1.start();
        t2.start();
        t1.join();
        t2.join();


        System.out.println("Value : " + counter.count);
```

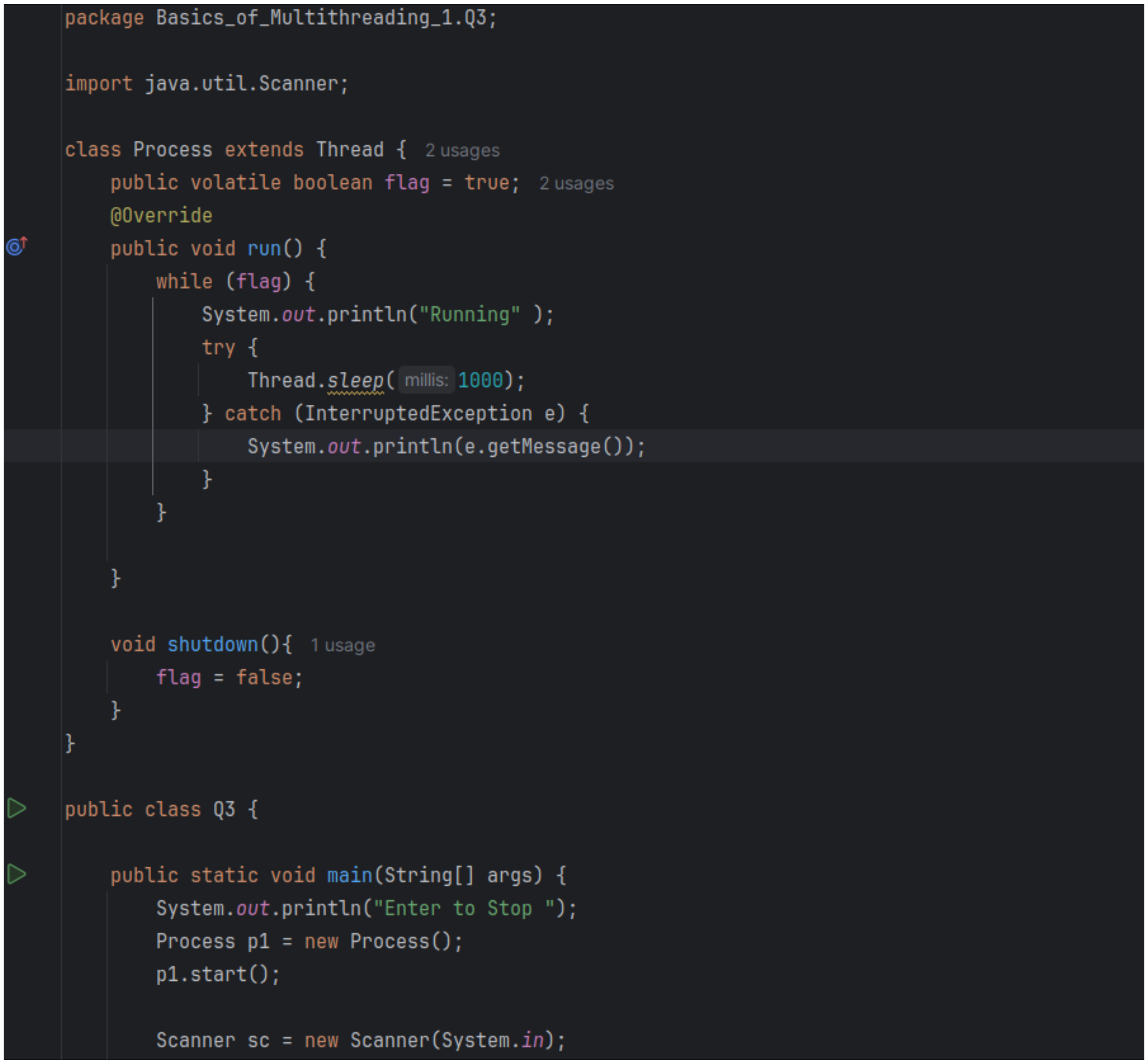


```java
public class Q2 {
    public static void main(String[] args) throws InterruptedException {
        Thread t2 = new Thread(new Runnable() {
        });

        t1.start();
        t2.start();
        t1.join();
        t2.join();


        System.out.println("Value : " + counter.count);


    }
}

class Counter{  2 usages
    int count;  2 usages

    public synchronized void increment(){  2 usages
        count++;
    }
}
```

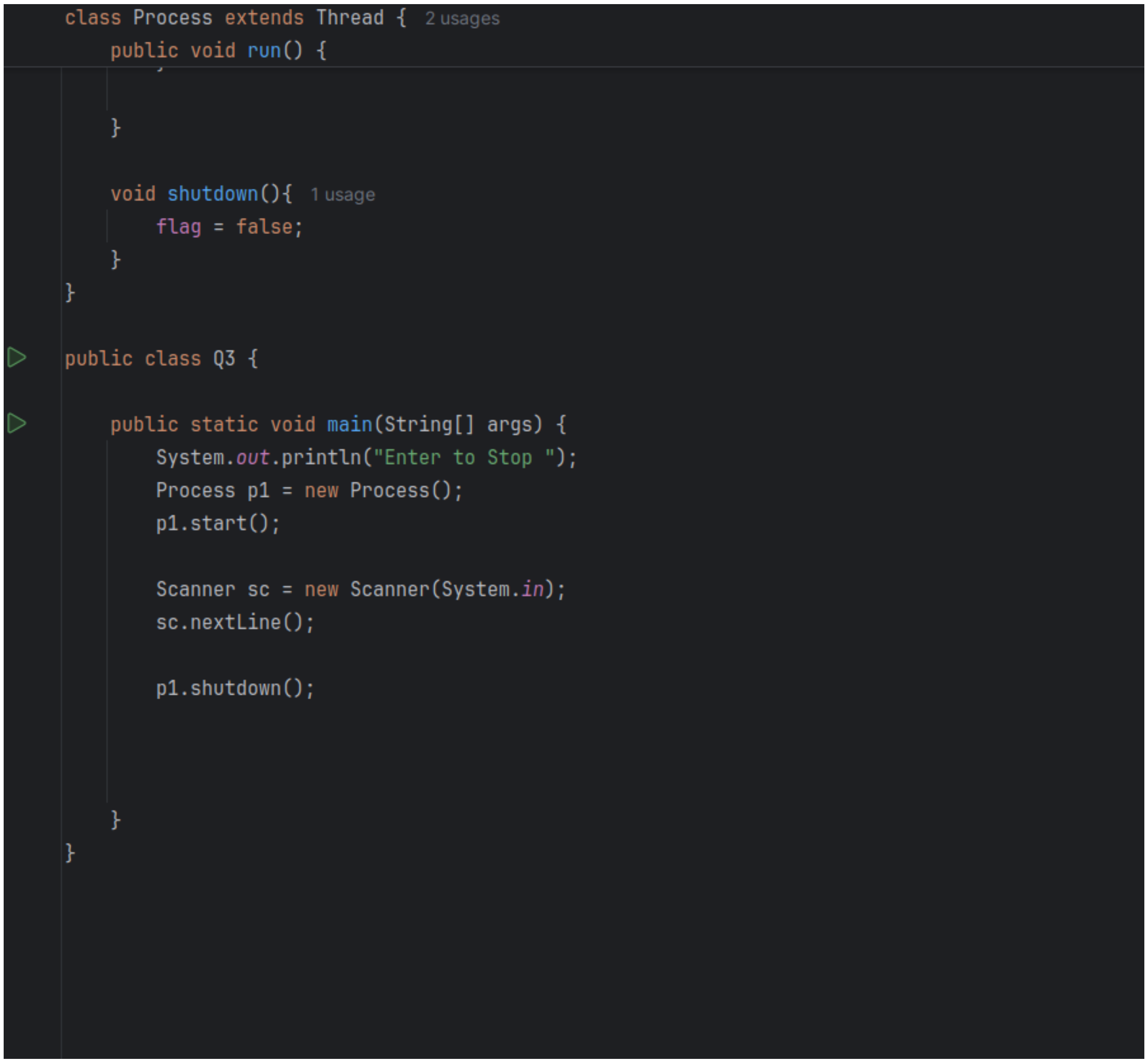


```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib
First Thread
Second Thread
Value : 2000

Process finished with exit code 0
```

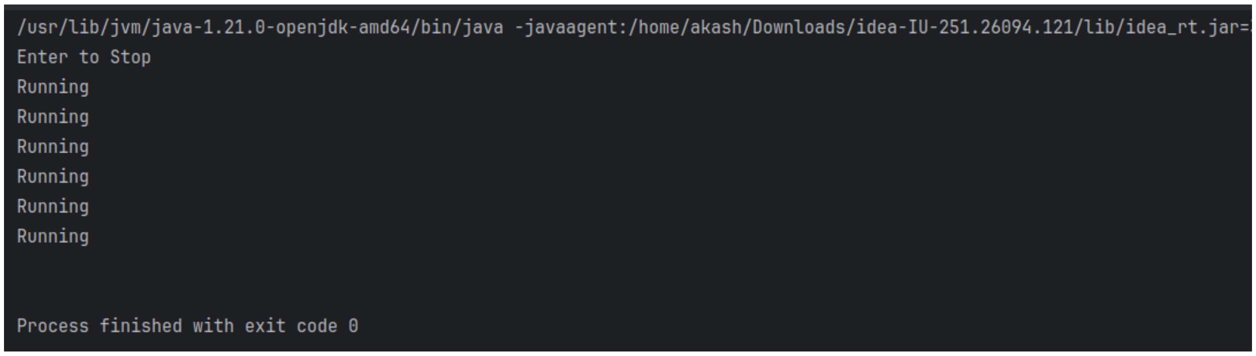
Q3)WAP to showcase the usage of volatile in java.



```
package Basics_of_Multithreading_1.Q3;

import java.util.Scanner;

class Process extends Thread {  2 usages
    public volatile boolean flag = true;  2 usages
    @Override
    public void run() {
        while (flag) {
            System.out.println("Running" );
            try {
                Thread.sleep( millis: 1000);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }

    }

    void shutdown(){  1 usage
        flag = false;
    }
}

public class Q3 {

    public static void main(String[] args) {
        System.out.println("Enter to Stop ");
        Process p1 = new Process();
        p1.start();

        Scanner sc = new Scanner(System.in);
```

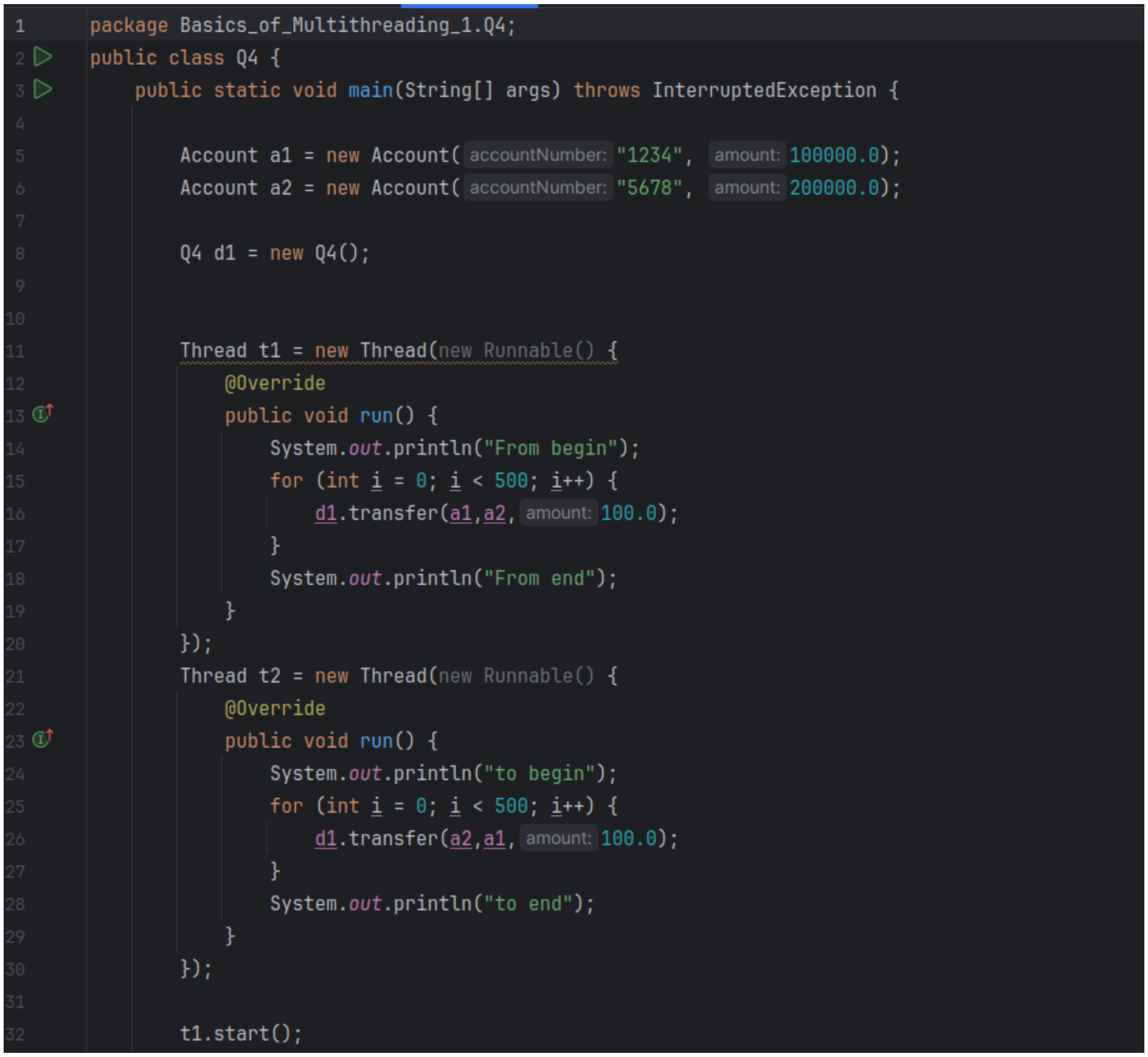


```
class Process extends Thread {  2 usages
    public void run() {

    }

    void shutdown(){  1 usage
        flag = false;
    }
}

public class Q3 {

    public static void main(String[] args) {
        System.out.println("Enter to Stop ");
        Process p1 = new Process();
        p1.start();

        Scanner sc = new Scanner(System.in);
        sc.nextLine();

        p1.shutdown();



    }
}
```

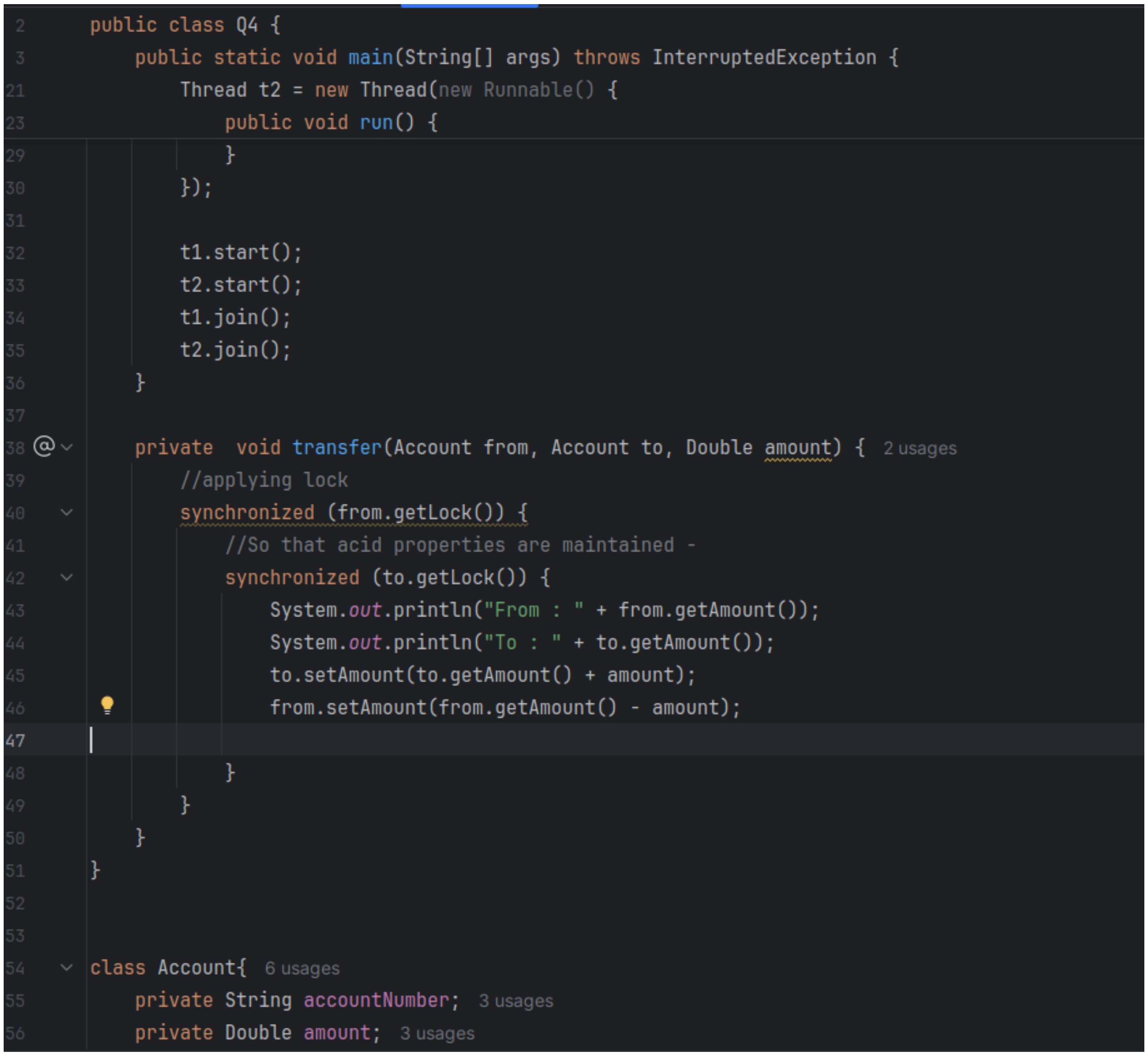


```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/idea_rt.jar=
Enter to Stop
Running
Running
Running
Running
Running
Running


Process finished with exit code 0
```

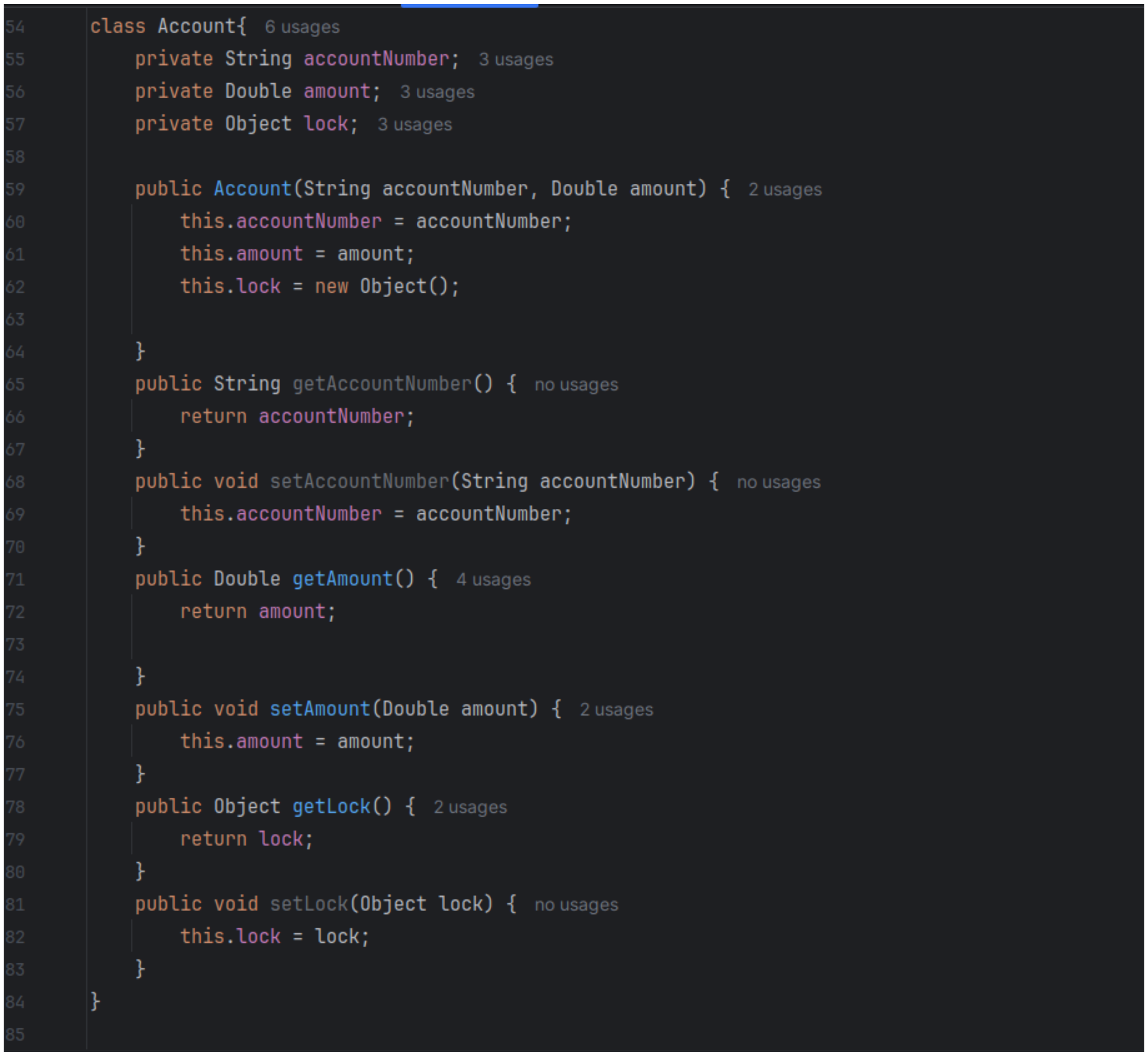
Q4)Write a code to simulate a deadlock in java



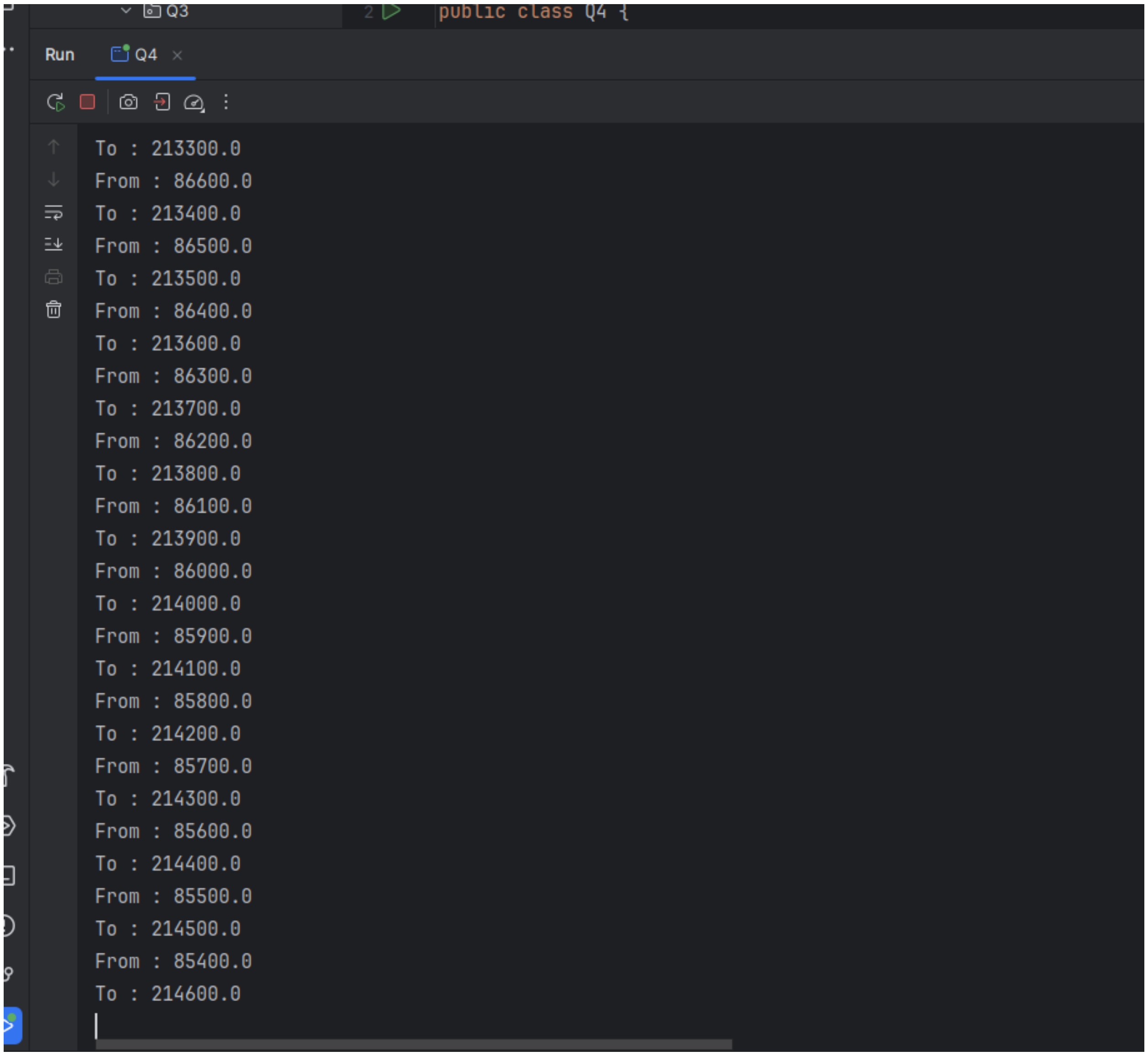
```java
package Basics_of_Multithreading_1.Q4;
public class Q4 {
    public static void main(String[] args) throws InterruptedException {

        Account a1 = new Account( accountNumber: "1234",  amount: 100000.0);
        Account a2 = new Account( accountNumber: "5678",  amount: 200000.0);

        Q4 d1 = new Q4();


        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("From begin");
                for (int i = 0; i < 500; i++) {
                    d1.transfer(a1,a2, amount: 100.0);
                }
                System.out.println("From end");
            }
        });
        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run() {
                System.out.println("to begin");
                for (int i = 0; i < 500; i++) {
                    d1.transfer(a2,a1, amount: 100.0);
                }
                System.out.println("to end");
            }
        });

        t1.start();
```



```java
public class Q4 {
    public static void main(String[] args) throws InterruptedException {
        Thread t2 = new Thread(new Runnable() {
            public void run() {
            }
        });

        t1.start();
        t2.start();
        t1.join();
        t2.join();
    }

    private  void transfer(Account from, Account to, Double amount) {  2 usages
        //applying lock
        synchronized (from.getLock()) {
            //So that acid properties are maintained -
            synchronized (to.getLock()) {
                System.out.println("From : " + from.getAmount());
                System.out.println("To : " + to.getAmount());
                to.setAmount(to.getAmount() + amount);
                from.setAmount(from.getAmount() - amount);

            }
        }
    }
}


class Account{  6 usages
    private String accountNumber;  3 usages
    private Double amount;  3 usages
```



```java
class Account{  6 usages
    private String accountNumber;  3 usages
    private Double amount;  3 usages
    private Object lock;  3 usages

    public Account(String accountNumber, Double amount) {  2 usages
        this.accountNumber = accountNumber;
        this.amount = amount;
        this.lock = new Object();

    }
    public String getAccountNumber() {  no usages
        return accountNumber;
    }
    public void setAccountNumber(String accountNumber) {  no usages
        this.accountNumber = accountNumber;
    }
    public Double getAmount() {  4 usages
        return amount;

    }
    public void setAmount(Double amount) {  2 usages
        this.amount = amount;
    }
    public Object getLock() {  2 usages
        return lock;
    }
    public void setLock(Object lock) {  no usages
        this.lock = lock;
    }
}
```



```
public class Q4 {
```

```
To : 213300.0
From : 86600.0
To : 213400.0
From : 86500.0
To : 213500.0
From : 86400.0
To : 213600.0
From : 86300.0
To : 213700.0
From : 86200.0
To : 213800.0
From : 86100.0
To : 213900.0
From : 86000.0
To : 214000.0
From : 85900.0
To : 214100.0
From : 85800.0
To : 214200.0
From : 85700.0
To : 214300.0
From : 85600.0
To : 214400.0
From : 85500.0
To : 214500.0
From : 85400.0
To : 214600.0
```