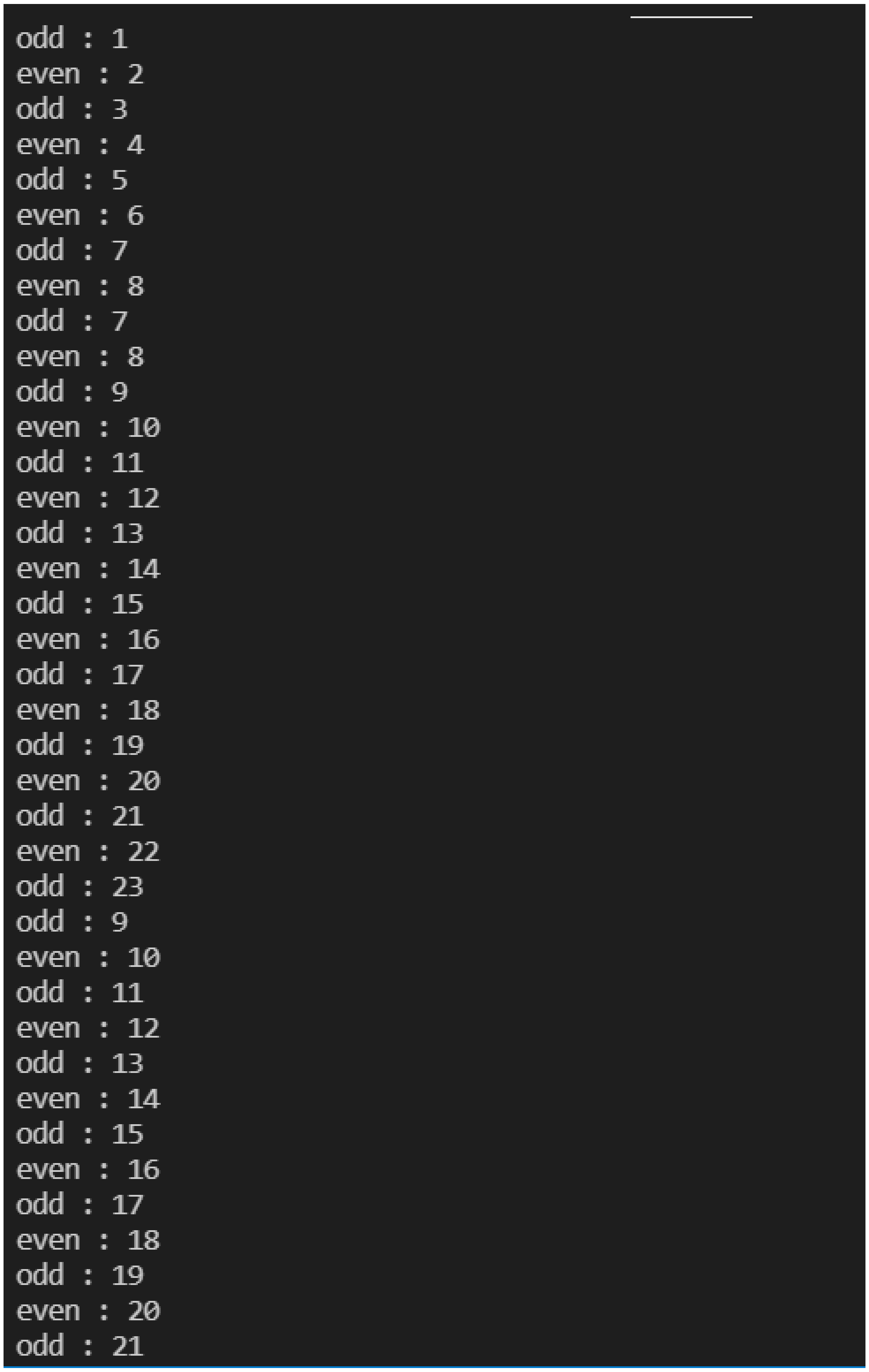
PROGRAMS BASED ON MULTITHREADING

1. Demonstrate multithreading by creating two threads, one for printing the odd numbers and the other for printing even numbers with in a given range of your choice.

Code:

```java
public class OddEvenThreadType2 {
    public static void main(String[] args) {

        Printer printer = new Printer();

        MyRunnable r1 = new MyRunnable(true, printer);// isOdd = true
        Thread t1 = new Thread(r1);
        MyRunnable r2 = new MyRunnable(false, printer);// isOdd = false
        Thread t2 = new Thread(r2);
        t1.start();
        t2.start();
    }
}

class Printer {
    private Object lock = new Object();
    private volatile boolean isOdd = false;

    public void printEven(int number) throws InterruptedException {
        synchronized (lock) {
            while (isOdd == false) {
                lock.wait();
            }
            System.out.println("even : " + number);
            isOdd = false;
            lock.notifyAll();
        }
    }

    public void printOdd(int number) throws InterruptedException {
        synchronized (lock) {
            while (isOdd == true) {
                lock.wait();
            }
            System.out.println("odd : " + number);
            isOdd = true;
            lock.notifyAll();
        }
    }
}

class MyRunnable implements Runnable {

    private boolean isOdd;
    Printer printer;

    MyRunnable(boolean isOdd, Printer printer) {
        this.isOdd = isOdd;
        this.printer = printer;
    }

    public void run() {
        int number = isOdd == true ? 1 : 2;
        while (number <= 25) {
            if (isOdd) {
                try {
                    printer.printOdd(number);
                } catch (InterruptedException e) {
                }
            } else {
                try {
                    printer.printEven(number);
                } catch (InterruptedException e) {
                }
            }
            number += 2;
        }
    }
}
```

Output:

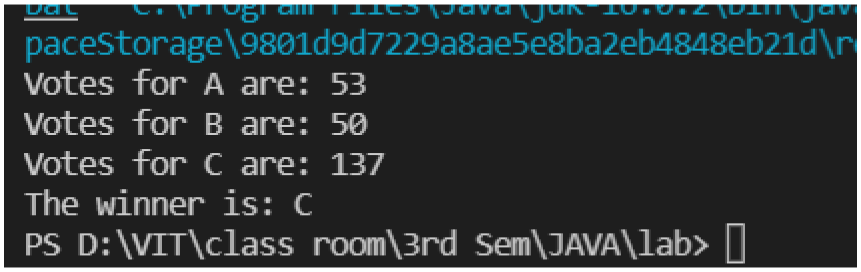

```
odd : 1
even : 2
odd : 3
even : 4
odd : 5
even : 6
odd : 7
even : 8
odd : 7
even : 8
odd : 9
even : 10
odd : 11
even : 12
odd : 13
even : 14
odd : 15
even : 16
odd : 17
even : 18
odd : 19
even : 20
odd : 21
even : 22
odd : 23
odd : 9
even : 10
odd : 11
even : 12
odd : 13
even : 14
odd : 15
even : 16
odd : 17
even : 18
odd : 19
even : 20
odd : 21
```

2. Write a program to demonstrate the knowledge of students in multithreading. Eg., Three students A, B and C of B.Tech- II year contest for the PR election. With the total strength of 240 students in II year, simulate the vote casting by generating 240 random numbers (1 for student A, 2 for B and 3 for C) and store them in an array. Create four threads to equally share the task of counting the number of votes cast for all the three candidates. Use synchronized method or synchronized block to update the three count variables. The main thread should receive the final vote count for all three contestants and hence decide the PR based on the values received.

Code:

```java
import java.util.*;

public class Voting extends Thread {
    static int total = 240, ac = 0, bc = 0, cc = 0;

    synchronized void takeVote(int val) {
        if (total > 0) {
            total--;
            if (val == 1) {
                ac++;
            } else if (val == 2) {
                bc++;
            } else if (val == 3) {
                cc++;
            }
        }
        try {
            Thread.sleep(20);
        } catch (Exception e) {
            System.out.println(e);
        }
    }

    public static void main(String args[]) {
        Voting obj = new Voting();
        Thread t1 = new Thread() {
            public void run() {
                while (total > 0) {
                    obj.takeVote(1);
                }
            }
        };

        Thread t2 = new Thread() {
            public void run() {
                while (total > 0) {
                    obj.takeVote(2);
                }
            }
        };

        Thread t3 = new Thread() {
            public void run() {
                while (total > 0) {
                    obj.takeVote(3);
                }
            }
        };

        t1.start();
        t2.start();
        t3.start();

        try {
            t1.join();
            t2.join();
            t3.join();
        } catch (Exception e) {

        }

        System.out.println("Votes for A are: " + ac);
        System.out.println("Votes for B are: " + bc);
        System.out.println("Votes for C are: " + cc);
        System.out.print("The winner is: ");

        if (ac > bc && ac > cc) {
            System.out.println("A");
        } else if (bc > ac && bc > cc) {
            System.out.println("B");
        } else if (cc > ac && cc > bc) {
            System.out.println("C");
        } else {
            System.out.println("Tie");
        }
    }
}
```

Output:



```
bat" "C:\Program Files\Java\jdk-16.0.2\bin\java
paceStorage\9801d9d7229a8ae5e8ba2eb4848eb21d\r
Votes for A are: 53
Votes for B are: 50
Votes for C are: 137
The winner is: C
PS D:\VIT\class room\3rd Sem\JAVA\lab> 
```