**Question 18:**

**Program to print the power of two number using Threading and use of synchronized**

**Code**

class ThreadTest {

    int count = 1;

    public synchronized void printPower(int n) {

        int temp = 1;

        for (count = 1; count <= 5; count++) {

            temp = temp \* n;

            System.out.println("power of " + n + " ^ " + count + " is " + temp);

        }

    }

}

public class Test {

    public static void main(String[] args) {

        ThreadTest t1 = new ThreadTest();

        ThreadTest t2 = new ThreadTest();

        Runnable r1 = new Runnable() {

            public void run() {

                t1.printPower(2);

            }

        };

        Runnable r2 = new Runnable() {

            public void run() {

                t2.printPower(10);

            }

        };

        Thread th1 = new Thread(r1);

        Thread th2 = new Thread(r2);

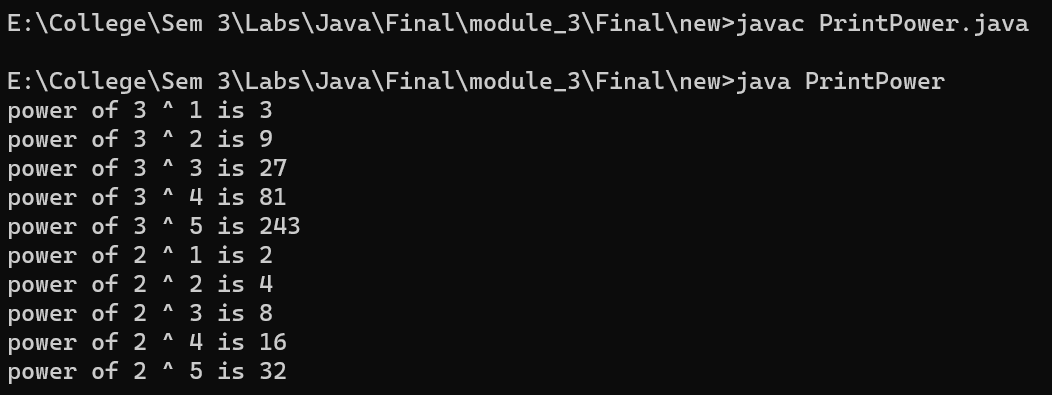
        th1.start();

        th2.start();

    }

};

**Output**

****

**Question 19:**

**Print Number in Sequence using two class Odd and Even (Use wait and notify)**

**Code**

class Manage {

    boolean odd;

    int count = 1;

    int MAX = 20;

}

class PrintOdd implements Runnable {

    Manage m;

    public PrintOdd(Manage m) {

        this.m = m;

    }

    @Override

    public void run() {

        synchronized (m) {

            while (m.count < m.MAX) {

                while (!m.odd) {

                    try {

                        m.wait();

                    } catch (InterruptedException e) {

                        e.printStackTrace();

                    }

                }

                System.out.println("Odd Thread :" + m.count);

                m.count++;

                m.odd = false;

                m.notify();

            }

        }

    }

}

class PrintEven implements Runnable {

    Manage m;

    public PrintEven(Manage m) {

        this.m = m;

    }

    @Override

    public void run() {

        synchronized (m) {

            while (m.count < m.MAX) {

                while (m.odd) {

                    try {

                        m.wait();

                    } catch (InterruptedException e) {

                        e.printStackTrace();

                    }

                }

                System.out.println("Even thread :" + m.count);

                m.count++;

                m.odd = true;

                m.notify();

            }

        }

    }

}

public class PrintNumber {

    public static void main(String[] args) {

        Manage m = new Manage();

        m.odd = true;

        Thread t1 = new Thread(new PrintEven(m));

        Thread t2 = new Thread(new PrintOdd(m));

        t1.start();

        t2.start();

        try {

            t1.join();

            t2.join();

        } catch (InterruptedException e) {

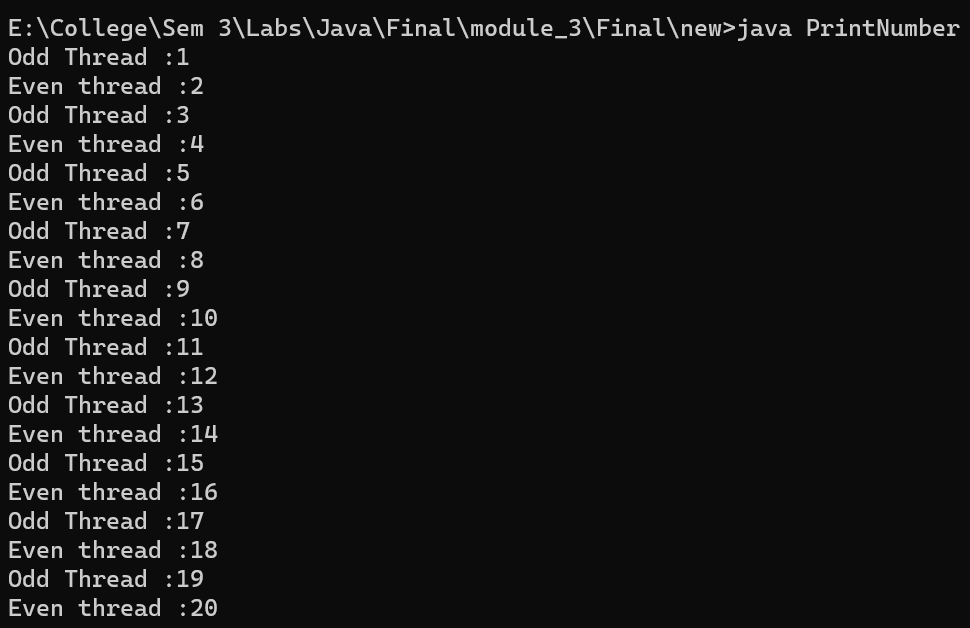
            e.printStackTrace();

        }

    }

}

**Output**

****

**Question 20:**

**Print tick tok in sequence manner (use wait and notify)**

**Code**

class Manage {

    boolean tick;

    int count = 1;

    int MAX = 20;

}

class PrintTick implements Runnable {

    Manage m;

    public PrintTick(Manage m) {

        this.m = m;

    }

    @Override

    public void run() {

        synchronized (m) {

            while (m.count < m.MAX) {

                while (!m.tick) {

                    try {

                        m.wait();

                    } catch (InterruptedException e) {

                        e.printStackTrace();

                    }

                }

               System.out.println("Tick");

                m.count++;

                m.tick = false;

                m.notify();

            }

        }

    }

}

class PrintTok implements Runnable {

    Manage m;

    public PrintTok(Manage m) {

        this.m = m;

    }

    @Override

    public void run() {

        synchronized (m) {

            while (m.count < m.MAX) {

                while (m.tick) {

                    try {

                        m.wait();

                    } catch (InterruptedException e) {

                        e.printStackTrace();

                    }

                }

               System.out.println("Tok");

                m.count++;

                m.tick = true;

                m.notify();

            }

        }

    }

}

public class PrintTickTok {

    public static void main(String[] args) {

        Manage m = new Manage();

        m.tick = true;

        Thread t1 = new Thread(new PrintTok(m));

        Thread t2 = new Thread(new PrintTick(m));

        t1.start();

        t2.start();

        try {

            t1.join();

            t2.join();

        } catch (InterruptedException e) {

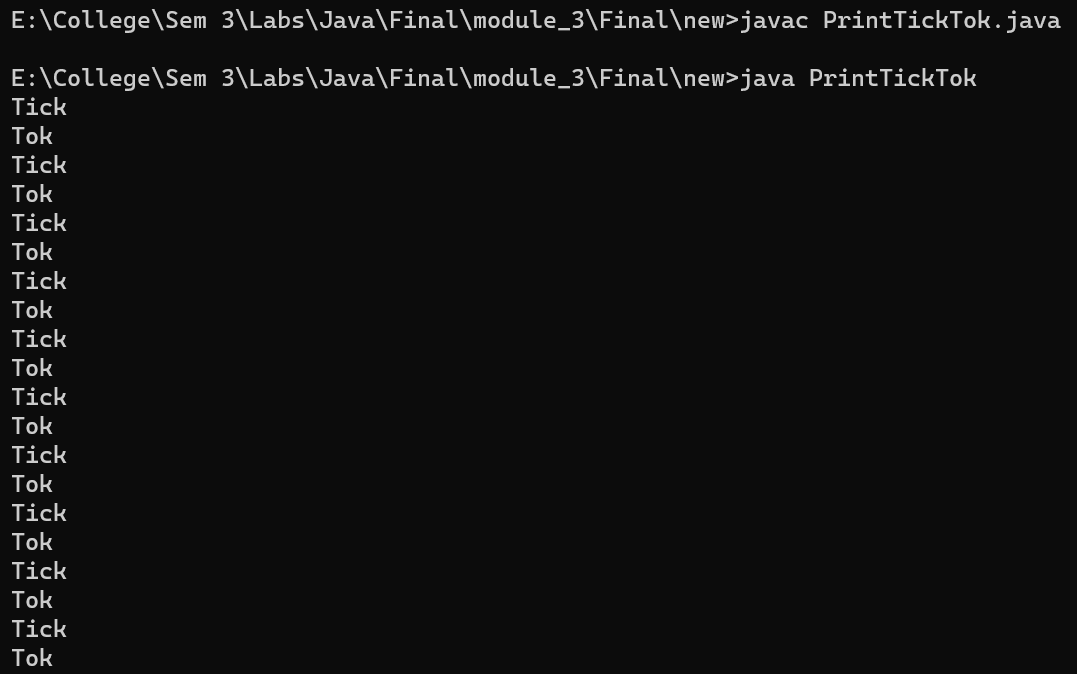
            e.printStackTrace();

        }

    }

}

**Output**

****