

Java

Assignment 4



Prepared by:

Name of Student : Chaitanya Dalvi

Roll No: 19

Batch: 2023-27

Dept. of CSE

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 1

Title:WAP to create two threads by extending Thread class. Print your “first name” in thread-1 six times and Print your “last name” in thread-2 seven times.

Code:

```
class Thread1 extends Thread {
    public void run() {
        try {
            for (int i = 0; i < 6; i++) {
                System.out.println("Chaitanya");
                Thread.sleep(500);
            }
        } catch (InterruptedException e) {
            System.out.println("Class1 thread interrupted");
        }
    }
}

class Thread2 extends Thread {
    public void run() {
        try {
            for (int i = 0; i < 7; i++) {
                System.out.println("Dalvi");
                Thread.sleep(500);
            }
        } catch (InterruptedException e) {
            System.out.println("Class2 thread interrupted");
        }
    }
}

public class ExtendThread {
    public static void main(String[] args) {
        Thread1 t1 = new Thread1();
        Thread2 t2 = new Thread2();
        System.out.println("Begin");
        t1.start();
        t2.start();
        System.out.println("End");
    }
}
```

Output: (screenshot)

```
> TERMINAL
cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac ExtendThread.java && java ExtendThread
/Users/chaitanyadalvi/.zshrc:2: /opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/us
r/bin:/bin:/usr/sbin:/sbin:/var/run/com.apple.security.cryptextd/codex.system/bootstrap/usr/local/bin:/var/run/com.app
le.security.cryptextd/codex.system/bootstrap/usr/bin:/var/run/com.apple.security.cryptextd/codex.system/bootstrap/usr/a
ppleinternal/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Too
lbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Toolbo
x/scripts:/Users/chaitanyadalvi/.npm-global/bin:/usr/local/mysql-9.0.1-macos14-arm64 not found
● chaitanyadalvi@Mac assignment4 % cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac ExtendThread.java &&
java ExtendThread
Begin
Chaitanya
Dalvi
End
Chaitanya
Dalvi
Chaitanya
Dalvi
Chaitanya
Dalvi
Chaitanya
Dalvi
Chaitanya
Dalvi
Dalvi
○ chaitanyadalvi@Mac assignment4 %
```

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 2

Title:WAP to create three threads by implementing Runnable interface. Print “Java” in

thread-1 five times and Print your “Python” in thread-2 eight times. Print your “C++” in

thread-3 seven times.

```
class Thread1 implements Runnable {  
    public void run() {  
        try {  
            for (int i = 0; i < 5; i++) {  
                System.out.println("Java");  
                Thread.sleep(500);  
            }  
        } catch (InterruptedException e) {  
            System.out.println("Class1 thread interrupted");  
        }  
    }  
}  
  
class Thread2 implements Runnable {  
    public void run() {  
        try {  
            for (int i = 0; i < 8; i++) {  
                System.out.println("Python");  
                Thread.sleep(500);  
            }  
        } catch (InterruptedException e) {  
            System.out.println("Class2 thread interrupted");  
        }  
    }  
}  
  
class Thread3 implements Runnable {  
    public void run() {  
        try {  
            for (int i = 0; i < 7; i++) {  
                System.out.println("C++");  
                Thread.sleep(500);  
            }  
        } catch (InterruptedException e) {
```


Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 3

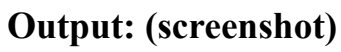
Title:WAP to create two threads. For creating thread-1, implement Runnable interface. For

creating thread-2, extend thread class. Print name of thread in thread-1 five times and

Print name of thread in thread-2 six times. Use the concept of anonymous class to implement Runnable interface.

```
class Thread2 extends Thread {  
    public void run() {  
        try {  
            for (int i = 0; i < 6; i++) {  
  
System.out.println(Thread.currentThread().getName());  
                Thread.sleep(500);  
            }  
        } catch (InterruptedException e) {  
            System.out.println("Class2 thread interrupted");  
        }  
    }  
}  
  
public class AnonymousThread {  
    public static void main(String[] args) {  
        Thread t1 = new Thread(new Runnable() {  
            public void run() {  
                try {  
                    for (int i = 0; i < 5; i++) {  
  
System.out.println(Thread.currentThread().getName());  
                        Thread.sleep(500);  
                    }  
                } catch (InterruptedException e) {  
                    System.out.println("Class1 thread  
interrupted");  
                }  
            }  
        });  
        Thread t2 = new Thread();  
        t1.start();  
    }  
}
```

Output: (screenshot)



Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 4

Title: Write a Java program to create and start multiple threads that increment a shared

counter variable concurrently. (You should get correct result.) (Hint: use the concept of

synchronized method).

```
class CounterEx2 {  
    int count;  
  
    public synchronized void increment() {  
        count++;  
    }  
}
```

```
class Class1W0 implements Runnable {  
    CounterEx2 c;  
  
    Class1W0(CounterEx2 c) {  
        this.c = c;  
    }  
  
    public void run() {  
        for (int i = 0; i < 1000; i++) {  
            c.increment();  
        }  
    }  
}
```

```
class Class2W0 implements Runnable {  
    CounterEx2 c;  
  
    Class2W0(CounterEx2 c) {  
        this.c = c;  
    }  
  
    public void run() {  
        for (int i = 0; i < 1000; i++) {  
            c.increment();  
        }  
    }  
}
```



```

}

public class SynchronizedThreads {
    public static void main(String[] args) {
        CounterEx2 c = new CounterEx2();
        Class1W0 obj1 = new Class1W0(c);
        Class2W0 obj2 = new Class2W0(c);
        Thread t1 = new Thread(obj1);
        Thread t2 = new Thread(obj2);
        t1.start();
        t2.start();
        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted");
        }
        System.out.println(c.count);
    }
}

```

Output: (screenshot)



```

▼ TERMINAL
cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac SynchronizedThreads.java && java SynchronizedThreads
/Users/chaitanyadalvi/.zshrc:2: /opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/
usr/bin:/bin:/usr/sbin:/sbin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:/var/run/com
.apple.security.cryptexd/codex.system/bootstrap/usr/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap
/usr/appleinternal/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/Users/chaitanyadalvi/Library/Application Support/JetBr
ains/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/Users/chaitanyadalvi/Library/Application Support/JetBra
ins/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/usr/local/mysql-9.0.1-macos14-arm64 not found
● chaitanyadalvi@Mac assignment4 % cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac SynchronizedThreads.
java && java SynchronizedThreads
2000
○ chaitanyadalvi@Mac assignment4 %

```

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 5

Title: Write a Java program to demonstrate the concept of synchronized blocks.

```
class Counter {
    private int count = 0;

    public void increment() {
        synchronized (this) {
            count++;
        }
    }

    public int getCount() {
        return count;
    }
}

class MyThread extends Thread {
    Counter counter;

    public MyThread(Counter counter) {
        this.counter = counter;
    }

    public void run() {
        for (int i = 0; i < 1000; i++) {
            counter.increment();
        }
    }
}

public class SynchronizedBlock {
    public static void main(String[] args) {
        Counter counter = new Counter();
        MyThread t1 = new MyThread(counter);
        MyThread t2 = new MyThread(counter);
        t1.start();
        t2.start();
        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Final count: " + counter.getCount());
    }
}
```

```
}  
}
```

Output: (screenshot)



```
✓ TERMINAL  
cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac SynchronizedBlock.java && java SynchronizedBlock  
/Users/chaitanyadalvi/.zshrc:2: /opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/appleinternal/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/usr/local/mysql-9.0.1-macos14-arm64 not found  
● chaitanyadalvi@Mac assignment4 % cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac SynchronizedBlock.java && java SynchronizedBlock  
Final count: 2000  
○ chaitanyadalvi@Mac assignment4 %
```

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 6

Title: Write a Java program to create a producer-consumer scenario using the wait() and

notify() methods for thread synchronization.

```
class Q {
    int n;
    boolean isValueSet = false;

    synchronized int consume() {
        while (!isValueSet) {
            try {
                wait();
            } catch (InterruptedException e) {
                System.out.println("Wait interrupted");
            }
        }
        System.out.println("Get: " + n);
        System.out.println();
        isValueSet = false;
        notify();
        return n;
    }

    synchronized void produce(int n) {
        while (isValueSet) {
            try {
                wait();
            } catch (InterruptedException e) {
                System.out.println("Wait interrupted");
            }
        }
        this.n = n;
        isValueSet = true;
        System.out.println("Put: " + n);
        notify();
    }
}

class Producer implements Runnable {
    Q q;
```

```

    Producer(Q q) {
        this.q = q;
    }

    public void run() {
        int i = 0;
        while (true) {
            q.produce(i++);
            try {
                Thread.sleep(200);
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted");
            }
        }
    }
}

class Consumer implements Runnable {
    Q q;

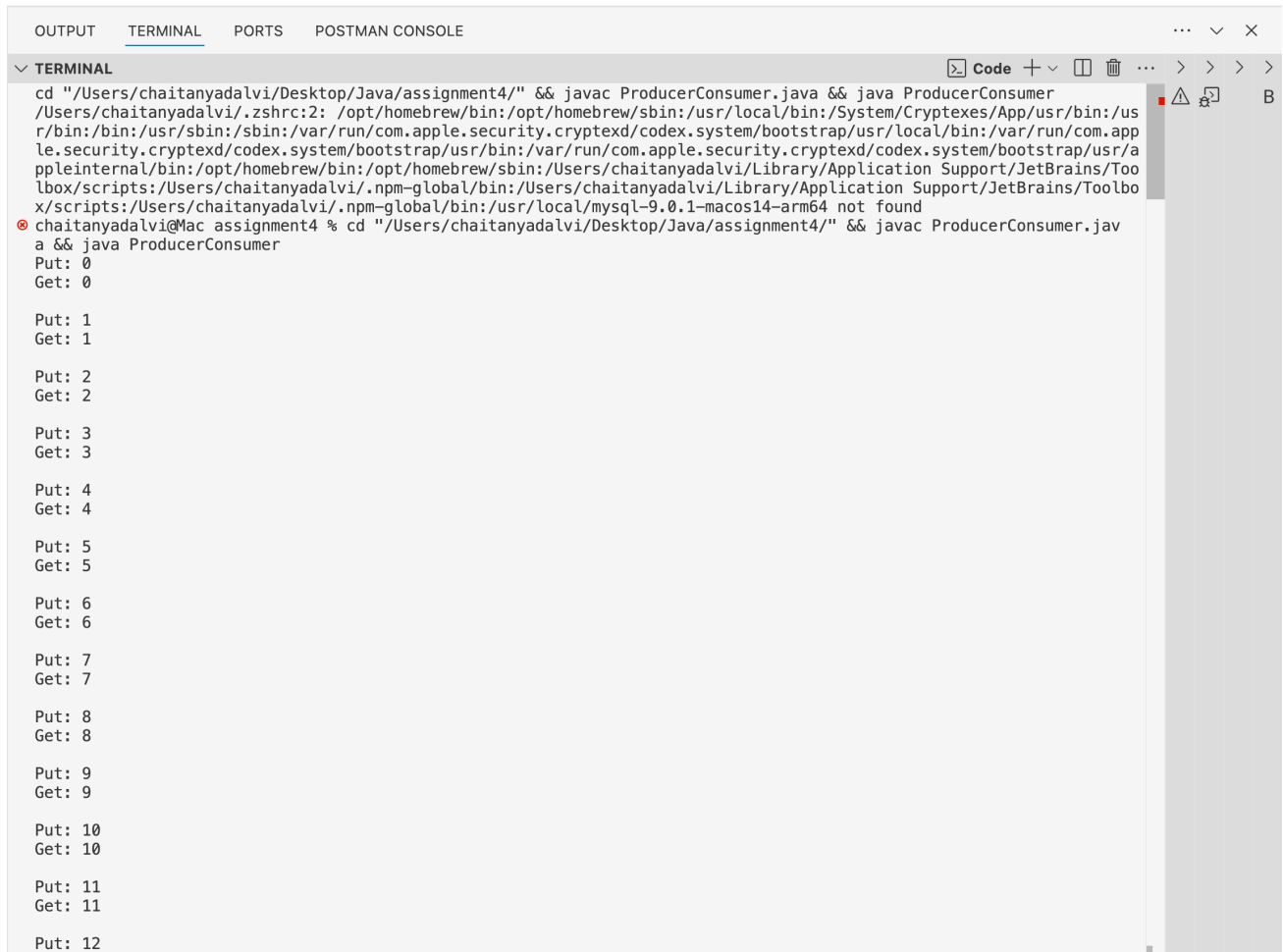
    Consumer(Q q) {
        this.q = q;
    }

    public void run() {
        while (true) {
            q.consume();
            try {
                Thread.sleep(200);
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted");
            }
        }
    }
}

public class ProducerConsumer {
    public static void main(String[] args) {
        Q q = new Q();
        Producer p = new Producer(q);
        Consumer c = new Consumer(q);
        Thread t1 = new Thread(p);
        Thread t2 = new Thread(c);
        t1.start();
        t2.start();
    }
}

```

Output: (screenshot)



```
cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac ProducerConsumer.java && java ProducerConsumer
/Users/chaitanyadalvi/.zshrc:2: /opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/appleinternal/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/usr/local/mysql-9.0.1-macos14-arm64 not found
chaitanyadalvi@Mac assignment4 % cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac ProducerConsumer.java && java ProducerConsumer
Put: 0
Get: 0

Put: 1
Get: 1

Put: 2
Get: 2

Put: 3
Get: 3

Put: 4
Get: 4

Put: 5
Get: 5

Put: 6
Get: 6

Put: 7
Get: 7

Put: 8
Get: 8

Put: 9
Get: 9

Put: 10
Get: 10

Put: 11
Get: 11

Put: 12
```

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 7

WAP to demonstrate deadlock among multiple threads.

```
public class DeadLockEx {
    public static void main(String[] args) {
        final String food = "Food";
        final String water = "Water";

        Thread t1 = new Thread(() -> {
            synchronized (food) {
                System.out.println("Thread 1: locked resource 1
(food)");

                try {
                    Thread.sleep(100);
                } catch (Exception e) {
                }

                synchronized (water) {
                    System.out.println("Thread 1: locked resource 2
(water)");
                }
            }
        });

        Thread t2 = new Thread(() -> {
            synchronized (water) {
                System.out.println("Thread 2: locked resource 2
(water)");

                try {
                    Thread.sleep(100);
                } catch (Exception e) {
                }

                synchronized (food) {
                    System.out.println("Thread 2: locked resource 1
(food)");
                }
            }
        });

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

Output: (screenshot)



```
cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac DeadLockEx.java && java DeadLockEx
/Users/chaitanyadalvi/.zshrc:2: /opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/appleinternal/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/usr/local/mysql-9.0.1-macos14-arm64 not found
chaitanyadalvi@Mac assignment4 % cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac DeadLockEx.java && java DeadLockEx
Thread 1: locked resource 1 (food)
Thread 2: locked resource 2 (water)
```


Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 8

Title:WAP to demonstrate the working of join() method.

```
class CounterEx2 {
    int count;
    public synchronized void increment() {
        count++;
    }
}

class Class1W0 implements Runnable {
    CounterEx2 c;
    Class1W0(CounterEx2 c) {
        this.c = c;
    }
    public void run() {
        for(int i=0; i<1000; i++) {
            c.increment();
        }
    }
}

class Class2W0 implements Runnable {
    CounterEx2 c;
    Class2W0(CounterEx2 c) {
        this.c = c;
    }
    public void run() {
        for(int i=0; i<1000; i++) {
            c.increment();
        }
    }
}

public class JoinEx {
    public static void main(String[] args) throws
    InterruptedException {
        CounterEx2 c = new CounterEx2();
        Class1W0 obj1 = new Class1W0(c);
        Class2W0 obj2 = new Class2W0(c);
        Thread t1 = new Thread(obj1);
        Thread t2 = new Thread(obj2);
        t1.start();
        t2.start();
    }
}
```

```
        t1.join();
        t2.join();
        System.out.println(c.count);
    }
}
```

Output: (screenshot)



```
▼ TERMINAL Code + - [ ] [ ] ... > > > >
cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac JoinEx.java && java JoinEx
/Users/chaitanyadalvi/.zshrc:2: /opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/
usr/bin:/bin:/usr/sbin:/sbin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:/var/run/com
.apple.security.cryptexd/codex.system/bootstrap/usr/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap
/usr/appleinternal/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/Users/chaitanyadalvi/Library/Application Support/JetBr
ains/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/Users/chaitanyadalvi/Library/Application Support/JetBra
ins/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/usr/local/mysql-9.0.1-macos14-arm64 not found
● chaitanyadalvi@Mac assignment4 % cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac JoinEx.java && java
JoinEx
2000
○ chaitanyadalvi@Mac assignment4 %
```

Name of Student: Chaitanya Dalvi

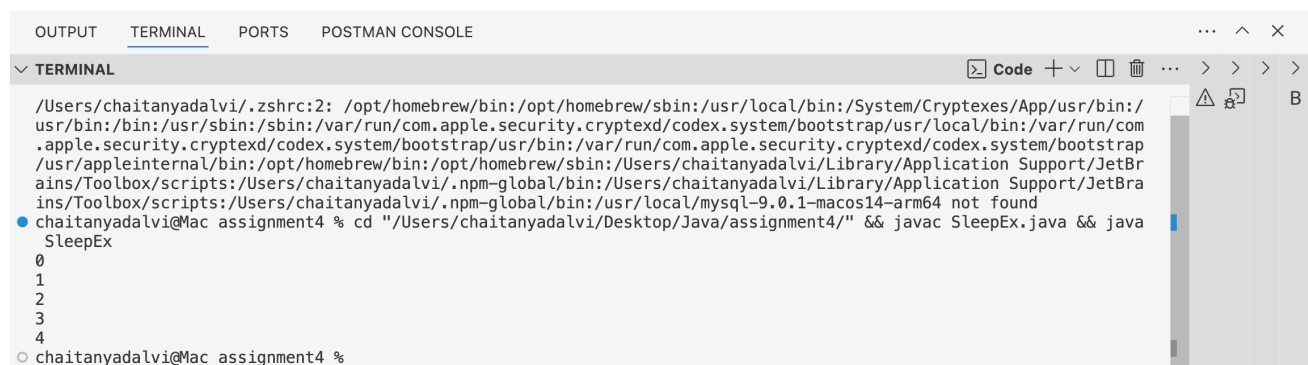
Roll Number: 19

Experiment No: 9

Title:WAP to demonstrate the working of Thread.sleep() method.

```
public class SleepEx {  
    public static void main(String[] args) {  
        for (int i = 0; i < 5; i++) {  
            try {  
                Thread.sleep(1000);  
                System.out.println(i);  
            } catch (InterruptedException e) {  
                System.out.println("Thread interrupted");  
            }  
        }  
    }  
}
```

Output: (screenshot)



```
OUTPUT  TERMINAL  PORTS  POSTMAN CONSOLE  
▼ TERMINAL  
/Users/chaitanyadalvi/.zshrc:2: /opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/appleinternal/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/usr/local/mysql-9.0.1-macos14-arm64 not found  
● chaitanyadalvi@Mac assignment4 % cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac SleepEx.java && java SleepEx  
0  
1  
2  
3  
4  
○ chaitanyadalvi@Mac assignment4 %
```

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 10

Title: WAP to demonstrate URL class.

```
public class URLEx {  
  
    public static void main(String[] args) {  
        try {  
            java.net.URL url = new java.net.URL("https://  
www.google.com:80/index.html");  
            System.out.println("Protocol: " + url.getProtocol());  
            System.out.println("Host: " + url.getHost());  
            System.out.println("Port: " + url.getPort());  
            System.out.println("File: " + url.getFile());  
        } catch (java.net.MalformedURLException e) {  
            System.out.println("Malformed URL");  
        }  
    }  
}
```

Output: (screenshot)



```
cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac URLEx.java && java URLEx  
/Users/chaitanyadalvi/.zshrc:2: /opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/appleinternal/bin:/opt/homebrew/bin:/opt/homebrew/sbin:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Toolbox/scripts:/Users/chaitanyadalvi/.npm-global/bin:/usr/local/mysql-9.0.1-macos14-arm64 not found  
● chaitanyadalvi@Mac assignment4 % cd "/Users/chaitanyadalvi/Desktop/Java/assignment4/" && javac URLEx.java && java URLEx  
Protocol: https  
Host: www.google.com  
Port: 80  
File: /index.html  
○ chaitanyadalvi@Mac assignment4 %
```

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 11

Title:WAP to create chat application using the concept of connection oriented approach using TCP protocol.

TCPclient.java

```
import java.net.*;
import java.util.Scanner;
import java.io.*;

public class TCPClient {
    public static void main(String[] args) throws Exception {
        Socket s = new Socket("localhost", 5555);
        DataInputStream din = new
DataInputStream(s.getInputStream());
        DataOutputStream dout = new
DataOutputStream(s.getOutputStream());
        String str = "", str2 = "";
        Scanner sc = new Scanner(System.in);
        while (!str.equals("stop")) {
            str = sc.nextLine();
            dout.writeUTF(str);
            dout.flush();

            str2 = din.readUTF();
            System.out.println("Server says: " + str2);
        }
        din.close();
        dout.close();
        sc.close();
        s.close();
    }
}
```

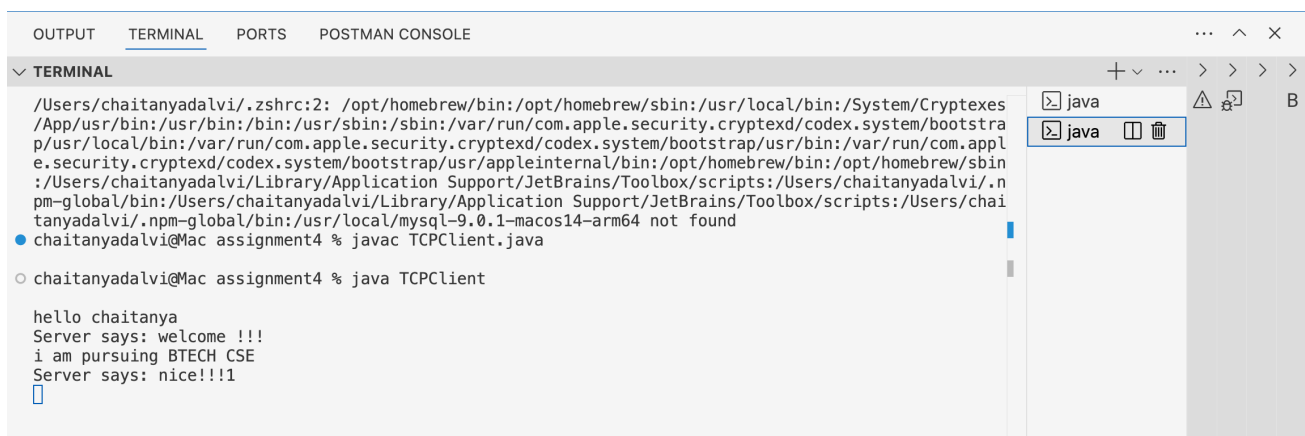
TCPserver.java

```
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class TCPServer {
    public static void main(String[] args) throws Exception {
        ServerSocket ss = new ServerSocket(5555);
        Socket s = ss.accept();
        String str = "", str2 = "";
        Scanner sc = new Scanner(System.in);
        DataInputStream din = new
DataInputStream(s.getInputStream());
        DataOutputStream dout = new
DataOutputStream(s.getOutputStream());
        while (!str.equals("stop")) {
            str2 = din.readUTF();
            System.out.println("Client says: " + str2);

            str = sc.nextLine();
            dout.writeUTF(str);
            dout.flush();
        }
        din.close();
        dout.close();
        sc.close();
        ss.close();
    }
}
```

Output: (screenshot)



OUTPUT

TERMINAL

PORTS

POSTMAN CONSOLE

▼ TERMINAL

+/...>>>>

java

java

B

```
/Users/chaitanyadalvi/.zshrc:2: /opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/System/Cryptexes
/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/var/run/com.apple.security.cryptexd/codex.system/bootstra
p/usr/local/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/bin:/var/run/com.appl
e.security.cryptexd/codex.system/bootstrap/usr/appleinternal/bin:/opt/homebrew/bin:/opt/homebrew/sbin
:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Toolbox/scripts:/Users/chaitanyadalvi/.n
pm-global/bin:/Users/chaitanyadalvi/Library/Application Support/JetBrains/Toolbox/scripts:/Users/chai
tanyadalvi/.npm-global/bin:/usr/local/mysql-9.0.1-macos14-arm64 not found
● chaitanyadalvi@Mac assignment4 % javac TCPServer.java

○ chaitanyadalvi@Mac assignment4 % java TCPServer

Client says: hello chaitanya
welcome !!!
Client says: i am pursuing BTECH CSE
nice!!!1
□
```

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 12

Title: WAP to create a UDP client application and UDP server application. Client will send a

number to server. Server has to calculate the cube of a number sent back to the client.

UDPClient.java

```
import java.net.*;

public class UDPClient {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket();
        int n = 8;
        String s1 = n + "";
        byte b1[] = s1.getBytes();
        InetAddress ia = InetAddress.getLocalHost();
        DatagramPacket dp1 = new DatagramPacket(b1, b1.length, ia,
5555);
        ds.send(dp1);

        byte b2[] = new byte[1024];
        DatagramPacket dp2 = new DatagramPacket(b2, b2.length);
        ds.receive(dp2);
        String s2 = new String(dp2.getData(), 0, dp2.getLength());
        System.out.println("Result: " + s2);
        ds.close();
    }
}
```

UDPServer.java

```
import java.net.*;

public class UDPServer {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket(5555);
        byte b1[] = new byte[1024];
        DatagramPacket dp1 = new DatagramPacket(b1, 0, b1.length);
        ds.receive(dp1);
        String s2 = new String(dp1.getData(), 0, dp1.getLength());
        System.out.println(s2);
        int num = Integer.parseInt(s2);
```

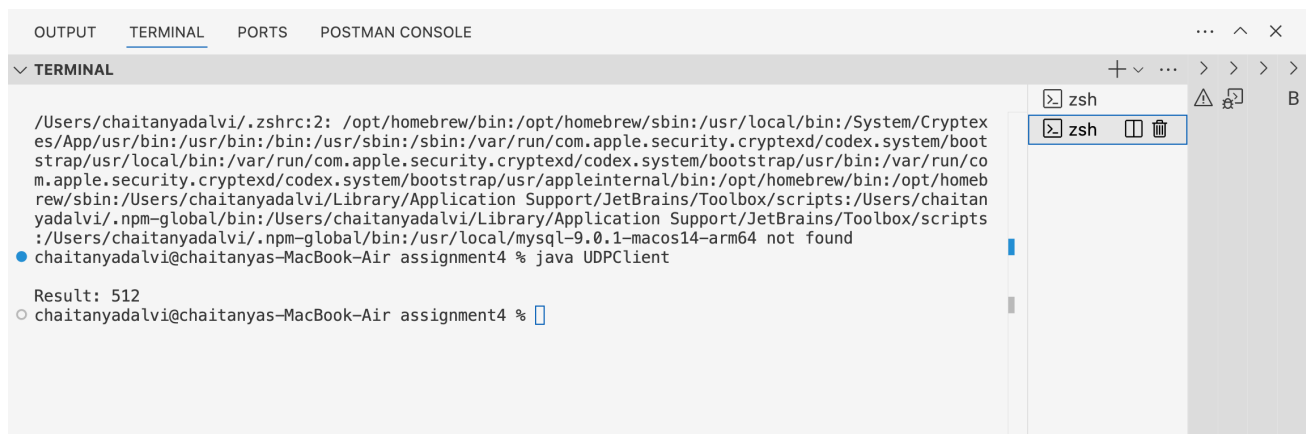


```

        int res = num * num * num;
        byte b2[] = String.valueOf(res).getBytes();
        InetAddress ia1 = InetAddress.getLocalHost();
        DatagramPacket dp2 = new DatagramPacket(b2, b2.length, ia1,
dp1.getPort());
        ds.send(dp2);
        ds.close();
    }
}

```

Output: (screenshot)



Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 13

Title: Differentiate between process and thread?

Process:

- 1. Independent Execution:** Each process runs independently with its own memory space. Processes do not share data directly.
- 2. Heavyweight:** Processes are more resource-intensive. Starting or switching between processes takes more time and resources.
- 3. Inter-process Communication (IPC):** Communication between processes requires special mechanisms like sockets, files, or pipes, making it slower and more complex.

Thread:

- 1. Shared Memory:** Multiple threads within the same process share the same memory space, making communication between them faster and more efficient.
- 2. Lightweight:** Threads are lighter than processes, meaning they use fewer resources and can be created or switched faster.
- 3. Concurrency:** Threads allow multiple parts of the same program to run concurrently (in parallel), improving efficiency, especially for tasks like handling multiple users in a web server.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 14

Title: Differentiate between Process based multitasking and thread based multitasking

1. Process-based multitasking:

- **What is it?** Running multiple programs (processes) at the same time.
- **Example:** Running a browser, music player, and text editor simultaneously.
- **Key point:** Each process has its own memory space, making them independent but resource-heavy.

2. Thread-based multitasking:

- **What is it?** Running multiple threads within the same program.
- **Example in Java:** In a game, one thread handles graphics while another handles user input.
- **Key point:** Threads share the same memory space within a single process, making it more efficient and lightweight compared to processes.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 15

Title: What do you understand by inter-thread communication?

Inter-thread communication in Java is a way for threads to talk to each other and coordinate their actions. It allows one thread to send signals or data to another thread, ensuring proper synchronization between them.

For example, if one thread is producing data and another is consuming it, inter-thread communication ensures the consumer waits until the producer has provided the data. Java provides methods like **wait()**, **notify()**, and **notifyAll()** to handle this.

In short, it's a mechanism for threads to cooperate with each other without causing conflicts or inconsistencies

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 16

Title: What is multithreading. What are the advantages of multithreading?

Multithreading is a feature in Java that allows multiple parts of a program (threads) to run simultaneously. A **thread** is the smallest unit of a program that can execute independently. In multithreading, several threads run concurrently within a single program, enabling it to perform multiple tasks at the same time.

For example, in a game, one thread can handle the player's actions while another handles the background music.

Advantages of Multithreading in Java:

- 1. Improved Performance:** By running multiple threads simultaneously, programs can perform tasks faster, especially on multi-core processors.
- 2. Efficient CPU Usage:** It makes better use of CPU resources by not letting the CPU sit idle while waiting for one task to finish.
- 3. Concurrent Execution:** Multiple tasks (like downloading a file while playing a video) can happen at the same time, making the application more responsive.
- 4. Simpler Program Design:** Multithreading allows developers to split complex tasks into smaller, independent threads, which makes programs easier to manage.
- 5. Better User Experience:** It keeps programs responsive. For example, a user can still interact with the program (like scrolling) while other tasks (like data processing) run in the background.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 17

Title: What is the purpose of `join()` method?

The `join()` method in Java is used to make one thread wait until another thread finishes its execution. When you call `join()` on a thread, it pauses the current thread until the thread you called `join()` on completes its task.

Example:

If Thread A calls `Thread B.join()`, then Thread A will wait until Thread B finishes before continuing.

Purpose:

- To ensure that a thread completes before moving on with the rest of the program.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 18

Title: What is the difference between wait() and sleep() method?

wait():

- Belongs to **Object** class.
- It is used for **thread communication** (i.e., makes the current thread release the lock and wait until another thread calls `notify()` or `notifyAll()` on the same object).
- The thread will **release the lock** on the object.

sleep():

- Belongs to **Thread** class.
- It is used to **pause the current thread** for a specified time.
- The thread does **not release any locks** while sleeping.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 19

Title: Is it possible to start a thread twice?

No, in Java, you cannot start a thread twice.

Once a thread has been started using the `start ()` method, it goes from the "new" state to "running." After it finishes its task, it cannot be started again. If you try to call `start ()` on the same thread object again, Java will throw an `IllegalThreadStateException`. You would need to create a new thread object if you want to run the task again.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 20

Title: Can we call the `run()` method instead of `start()`? What would be the result of calling `run()` method instead of `start()` method?

In Java, calling the `run ()` method directly executes the code in that method on the current thread. This means it runs synchronously and blocks the current thread until the `run ()` method completes.

On the other hand, calling the `start ()` method creates a new thread and then calls the `run ()` method within that new thread. This allows the main program to continue running while the new thread executes.

Result of Calling `run ()` vs. `start ()`:

- **Calling `run ()`:** The task runs in the same thread. No new thread is created. It blocks until completion.
- **Calling `start ()`:** A new thread is created. The task runs concurrently, allowing the main thread to keep executing.

In summary, use `start ()` for parallel execution and `run ()` for sequential execution within the same thread.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 21

Title: What is the purpose of the Synchronized method and Synchronized block?

In Java, the purpose of **synchronized methods** and **synchronized blocks** is to control access to a resource by multiple threads, preventing conflicts and ensuring data integrity.

Synchronized Method:

- A method marked with the **synchronized** keyword can only be executed by one thread at a time for a given object.
- This prevents multiple threads from modifying the same data simultaneously.

Synchronized Block:

- A synchronized block allows more fine-grained control over which part of the code is synchronized.
- It lets you specify the object to lock on, allowing other threads to run in sections of code that don't need synchronization.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 22

Title: What is the difference between `notify()` and `notifyAll()`?

In Java, both `notify()` and `notifyAll()` are methods used in thread synchronization, specifically with objects that are waiting for a condition to be met.

`notify()`

- **Wakes up one waiting thread.** If multiple threads are waiting on the same object, only one of them will be chosen to proceed.
- It's more efficient when you know that only one thread should continue.

`notifyAll()`

- **Wakes up all waiting threads.** All threads waiting on the object's monitor will be notified and can compete to acquire the lock.
- Use this when multiple threads might need to react to the condition, ensuring that all are given a chance.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 23

Title: Explain deadlock? How can deadlock be avoided?

What is Deadlock?

A **deadlock** is a situation in programming where two or more threads are blocked forever, each waiting for the other to release a resource. In simpler terms, it's like two people who each have a key to the other's door; neither can get out because they're both waiting for the other to unlock their door.

How to Avoid Deadlock in Java

- 1. Resource Ordering:** Always acquire resources in a specific order. For example, if two threads need two locks, make sure they always lock them in the same order.
- 2. Timeouts:** Use timeouts when trying to acquire a lock. If a thread can't get the lock in a certain amount of time, it gives up and can retry later.
- 3. Lock Hierarchies:** Establish a hierarchy of locks and ensure threads always acquire locks in order of hierarchy.
- 4. Avoid Nested Locks:** Try to avoid holding multiple locks at the same time. If you must, use the same locking order for all threads.
- 5. Use Higher-level Concurrency Utilities:** Java provides utilities like `java.util.concurrent` that can help manage threads and resources more safely than using low-level locks.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 24

Title: What is Thread Scheduler in java?

The **Thread Scheduler** in Java is a part of the Java Virtual Machine (JVM) that manages the execution of threads. Its main job is to determine which thread runs at any given time and for how long.

Key Points:

- **Multithreading:** Java supports running multiple threads concurrently, and the thread scheduler ensures that these threads get CPU time.
- **Priority:** Threads can have different priorities, which can influence the order in which they run. Higher-priority threads are generally scheduled to run before lower-priority ones.
- **Preemptive Scheduling:** The scheduler can interrupt a running thread to give CPU time to another thread, ensuring fair use of resources.
- **Non-Preemptive Scheduling:** A thread will continue to run until it finishes or voluntarily yields control.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 25

Title: What is race-condition?

A **race condition** in Java occurs when two or more threads try to access and modify shared data at the same time, leading to unpredictable results.

For example, if one thread is updating a variable while another thread reads it, the second thread might get an inconsistent or outdated value. This happens because the operations are not properly synchronized, meaning they don't happen in a controlled order. To avoid race conditions, you can use synchronization techniques like **synchronized blocks** or **locks**.

Name of Student: Chaitanya Dalvi

Roll Number: 19

Experiment No: 26

Title: What are the states in the lifecycle of a Thread?

In Java, a thread goes through several states in its lifecycle:

1. **New:** The thread is created but not yet started. It's in this state after you create a `Thread` object.
2. **Runnable:** The thread is ready to run and waiting for the CPU to schedule it. This state can be reached from both the New state and when the thread is running.
3. **Blocked:** The thread is waiting to acquire a lock to enter a synchronized block or method. It cannot proceed until it gets the lock.
4. **Waiting:** The thread is waiting indefinitely for another thread to perform a specific action (like `wait()`, `join()`, or `LockSupport.park()`).
5. **Timed Waiting:** Similar to the Waiting state, but it will wait for a specified amount of time (like `sleep(milliseconds)`, `wait(milliseconds)`, or `join(milliseconds)`).
6. **Terminated:** The thread has completed its execution (either naturally or due to an error). Once in this state, the thread cannot be restarted.