1)Write a java program to perform sum of upper triangle elements?
Input:
```
arr={ {1,2,3}
 {4,5,6},
 {7,8,9} };
```

sol:

```java
public class Test {
    public static void main(String[] args) {
        int[][] arr = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        int sum = 0;
        for (int i = 0; i < arr.length; i++) {
            for (int j = i; j < arr[i].length; j++) {
                sum=sum + arr[i][j];
            }
        }
        System.out.println("Sum of upper triangle elements: " + sum);
    }
}
```

2)Write a java program to perform lower triangle elements ?
Input:
```
arr={ {1,2,3}
 {4,5,6},
 {7,8,9} };
```

sol:

```java
public class Test {
    public static void main(String[] args) {
        int[][] arr = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        int sum = 0;
        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j <= i; j++) {
                sum=sum + arr[i][j];
            }
        }
        System.out.println("Sum of lower triangle elements: " + sum);
    }
}
```

3)Write a Java program to create a thread by extending the Thread class and another
by implementing the Runnable interface. Make sure each thread prints a message indicating
which method was used to create it.

sol:

```
class myThread extends Thread {
	public void run() {
		System.out.println("Thread by extendingThread class");
		System.out.println();
	}
}
class myRunnable implements Runnable {
	public void run() {
		System.out.println("Thread by implementing Runnable Interface");
	}
}

public class Test{
	public static void main(String[] args) {
		myThread mt=new myThread();
		mt.start();
		myRunnable mr=new myRunnable();
		Thread t=new Thread(mr);
		t.start();
	}
}
```

4)Write a Java program that creates two threads. The main thread should wait for the
first thread to complete using the join() method before starting the second thread.
Each thread should print its name and a message indicating that it has started and finished.
Expected Output:
The main thread should wait for the first thread to finish before starting the second thread.

sol:

```
class FirstThread extends Thread {
    public void run() {
        System.out.println(Thread.currentThread().getName() + " has started.");
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(Thread.currentThread().getName() + " has finished.");
    }
}

class SecondThread extends Thread {
    public void run() {
        System.out.println(Thread.currentThread().getName() + " has started.");
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(Thread.currentThread().getName() + " has finished.");
    }
}

public class Test {
    public static void main(String[] args) {
        FirstThread t1 = new FirstThread();
        t1.setName("First Thread");

        SecondThread t2 = new SecondThread();
        t2.setName("Second Thread");

        t1.start();

        try {
            t1.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        t2.start();
    }
}
```

5)Write a Java program that creates a thread using the Thread class.
The thread should print "Thread is going to sleep" and then sleep for 3 seconds.

If the thread is interrupted while sleeping, it should catch the
InterruptedException
and print "Thread was interrupted".
Expected Output:
The thread should print the message indicating it is going to sleep.
If interrupted, it should print the message indicating it was interrupted

sol:

```
class SleepThread extends Thread {
    public void run() {
        System.out.println("Thread is going to sleep");

        try {

            Thread.sleep(3000);
        } catch (InterruptedException e) {
            System.out.println("Thread was interrupted");
        }

        System.out.println("Thread has finished");
    }
}

public class Test {
    public static void main(String[] args) {
        SleepThread t = new SleepThread();
        t.start();

        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        t.interrupt();
    }
}
```