9. Demonstrate the concept of Threads to achieve multitasking program using Thread class and Runnable Interface (separately) for the below scenario:

> Create an Array of 9 numbers. And create three Threads to split the task evenly among the three threads. And each thread has to add up and report the answer to the main thread where the main thread waits for the 3 threads and computes the summation of all the three threads. Note: Assign names to the threads as well.

Using the Thread class:

```
import java.util.Arrays;


public class Main {
  public static void main(String[] args) throws InterruptedException {
    int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    int chunkSize = numbers.length / 3;


    Thread thread1 = new Thread(new SumThread(numbers, 0, chunkSize, "Thread 1"));
    Thread thread2 = new Thread(new SumThread(numbers, chunkSize, 2 * chunkSize, "Thread 2"));
    Thread thread3 = new Thread(new SumThread(numbers, 2 * chunkSize, numbers.length, "Thread
3"));


    thread1.start();
    thread2.start();
    thread3.start();


    thread1.join();
    thread2.join();
    thread3.join();


    int sum = thread1.getSum() + thread2.getSum() + thread3.getSum();
    System.out.println("Sum of all numbers: " + sum);
  }
}
```

```
class SumThread implements Runnable {
  private int[] numbers;
  private int startIndex;
  private int endIndex;
  private String name;
  private int sum;

  public SumThread(int[] numbers, int startIndex, int endIndex, String name) {
    this.numbers = numbers;
    this.startIndex = startIndex;
    this.endIndex = endIndex;
    this.name = name;
  }

  public int getSum() {
    return sum;
  }

  @Override
  public void run() {
    for (int i = startIndex; i < endIndex; i++) {
      sum += numbers[i];
    }
    System.out.println(name + ": " + sum);
  }
}
```

using the Runnable interface:

```
import java.util.Arrays;
```

```
public class Main {
  public static void main(String[] args) {
    // create an array of 9 numbers
    int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9};

    // create three threads to split the task evenly among the three threads
    Thread thread1 = new Thread(new SumTask(numbers, 0, 3));
    thread1.setName("Thread 1");
    Thread thread2 = new Thread(new SumTask(numbers, 3, 6));
    thread2.setName("Thread 2");
    Thread thread3 = new Thread(new SumTask(numbers, 6, 9));
    thread3.setName("Thread 3");

    // start the threads
    thread1.start();
    thread2.start();
    thread3.start();

    // wait for the threads to finish
    try {
      thread1.join();
      thread2.join();
      thread3.join();
    } catch (InterruptedException e) {
      e.printStackTrace();
    }

    // compute the summation of all the three threads
    int sum = thread1.getSum() + thread2.getSum() + thread3.getSum();
    System.out.println("Sum: " + sum);
  }
```

```
}

class SumTask implements Runnable {
  private int[] numbers;
  private int startIndex;
  private int endIndex;
  private int sum;

  public SumTask(int[] numbers, int startIndex, int endIndex) {
    this.numbers = numbers;
    this.startIndex = startIndex;
    this.endIndex = endIndex;
  }

  @Override
  public void run() {
    for (int i = startIndex; i < endIndex; i++) {
      sum += numbers[i];
    }
  }

  public int getSum() {
    return sum;
  }
}
```