Exercise-1

- Create a thread class and pass the name of the thread to its constructor
- The thread will print “Hello from thread_name”
- In main method, Create 2 objects of the thread class and after the threads have terminated print “main is terminating”

```java
class myThread extends Thread{
    myThread(String name){
        super(name);
    }
    public void run(){
        System.out.println("Hello from "+getName());
    }
}
```

Exercise-1(cntd)

You must handle this exception for join method, because the thread(main) calling the join method may get interrupted while waiting for other threads to finish

```
class ThreadTest{
    public static void main(String[] args) throws InterruptedException{
        myThread thread1=new myThread("thread 1");
        myThread thread2=new myThread("thread 2");
        //start the threads
        thread1.start();
        thread2.start();
        //wait for the threads to terminate
        thread1.join();
        thread2.join();
        System.out.println("main is terminating");
    }
}
```

Exercise-2

- Create a thread class “SumThread” implementing runnable interface. It takes an array as an argument in the constructor. In its run method, it will compute the sum of the array.
- In main method, create an object of the thread and start the thread. Finally, print the sum of array.

```
class sumThread implements Runnable{
    int arr[];
    int sum=0;
    sumThread(int a[]){
        arr=a;
    }
    public void run(){
        for(int x:arr) sum+=x;
    }
}
```

Exercise-2(cntd)

- Create a thread class “SumThread” implementing runnable interface. It takes an array as an argument in the constructor. In its run method, it will compute the sum of the array.
- In main method, create an object of the thread and start the thread. Finally, print the sum of array.

```
class ArraySum{
    public static void main(String[] args) throws InterruptedException {
        int a[]={1,3,4,5,7};
        sumThread s=new sumThread(a);
        Thread thread=new Thread(s);
        thread.start();
        thread.join();
        System.out.println("sum:"+s.sum);
    }
}
```

Exercise-3

- Create a thread class “ComputeArraySum” extending Thread class.
- In main method, create 2 objects of the thread class. Split the array equally and let the 2 threads compute the sum of half array separately and print the sum in main method.

```
class ComputeArraySum extends Thread {
    int arr[];
    public int start,end;
    static int sum=0;
    ComputeArraySum(int a[],int s,int e){
        arr=a;start=s;end=e;
    }
    public void run(){
        for(int i=start;i<=end;i++) sum+=arr[i];
    }
}
```

Passing the start and end index to the constructor

```
class ComputeSum{
    public static void main(String[] args) throws InterruptedException {
        int a[]={1,3,4,4,2,5,7};
        ComputeArraySum c1=new ComputeArraySum(a, 0, a.length/2);
        ComputeArraySum c2=new ComputeArraySum(a, (a.length/2) + 1 , a.length-1);
        c1.start();
        c2.start();
        c1.join();
        c2.join();
        System.out.println("sum: "+c1.sum);
    }
}
```

Splitting the array into 2 equal portion

Waiting for the threads to complete the computation

Exercise-4

- Create a thread class “countNumbers” implementing runnable interface.
- In main method, create 2 objects of the thread to implement a shared counter that will count from 1 to n.

```java
class countNumber implements Runnable{
    public static int counter=1;
    int n;
    Thread t;
    countNumber(int n){
        this.n=n;
        t=new Thread(this);
        t.start();
    }
    public void run(){
        while(true){
            if(counter>n) break;
            System.out.println(counter++);
        }
    }
}
```

Exercise-4(cntd)

- Create a thread class “countNumbers” implementing runnable interface.
- In main method, create 2 objects of the thread to implement a shared counter that will count from 1 to n.

```
class Count{
    public static void main(String[] args) throws InterruptedException {
        countNumber t1=new countNumber(10);
        countNumber t2=new countNumber(5);
    }
}
```

Exercise-4(cntd)

- If main waits for the threads, then we have to invoke join method.

```
class Count{
    public static void main(String[] args) throws InterruptedException
{
        countNumber t1=new countNumber(10);
        countNumber t2=new countNumber(5);
        t1.t.join();
        t2.t.join();
        System.out.println("Count completed");
    }
}
```