Q1. WAP to Create a class MyThread derived from Thread class and override the run method. Create a class ThreadDemo having a main method. Create 2 objects of MyThread class and observe the behavior of threads.

```
import java.util.*;
class MyThread extends Thread{
    public void run(){
        System.out.println("Running");

    }

}
class ThreadDemo{
    public static void main(String[] args){
        MyThread t1=new MyThread();
        MyThread t2=new MyThread();
        t1.start();
        try{
        t1.sleep(2000);}
        catch(InterruptedException e){
            System.out.println("exception");
        }
        t2.start();
    }
}
```

Output:

```
Demo.java } ; if ($?) { java ThreadDemo }
Running
Running
```

Q2. WAP to Modify the above to create MyThread class by implementing Runnable interface and observe the behavior of threads.

```
class MyThread  implements Runnable{
public void run(){
```

```
    for(int i=0;i<5;i++){
        System.out.println("Running by using
Thread A");

    }

}}
class MyThread3 implements Runnable{
public void run(){
    for(int i=0;i<5;i++){
        System.out.println("Running by using
Thread B");

    }
```

```
}}
public
class Mythread1{
    public static void main(String[] args){
        MyThread t1=new MyThread();
        MyThread3 t2=new MyThread3();
        Thread r1=new Thread(t1);
        Thread r2=new Thread(t2);
        r1.start();
        try{
        r2.sleep(2000);}
        catch(InterruptedException e){
            System.out.println("exception");
        }
        r2.start();
    }
}
```

OUTPUT

```
PS C:\Users\Admin\Desktop\java> cd "c:\Users\
Admin\Desktop\java\" ; if ($?) { javac Mythre
ad1.java } ; if ($?) { java Mythread1 }
Running by using Thread A
Running by using Thread A
Running by using Thread A
Running by using Thread A
Running by using Thread A
Running by using Thread B
Running by using Thread B
Running by using Thread B
Running by using Thread B
Running by using Thread B
PS C:\Users\Admin\Desktop\java>
```

Q3. An online food delivery platform receives multiple orders at the same time.
To improve performance, each order should be processed independently and concurrently, instead of one after another.

```
class OrderProcessor extends Thread {
    private int orderId;

    public OrderProcessor(int orderId) {
        this.orderId = orderId;
    }
```

```

    public void run() {
```

```
        System.out.println("Processing order :"
+ orderId + " by " +
Thread.currentThread().getName());
        try {

            Thread.sleep(2000);
        } catch (InterruptedException e) {
            System.out.println("Exception");
        }
        System.out.println("Completed order :"
+ orderId);
    }
}
```

```
public class Food {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            OrderProcessor orderThread = new
OrderProcessor(i);
            orderThread.start();
        }
```

```
        System.out.println("Main system is free
to take new orders...");
    }
}
```

OUTPUT


PS C:\Users\Admin\Desktop\java> cd "c:\Users
Admin\Desktop\java\" ; if ($?) { javac Food.
ava } ; if ($?) { java Food }
Main system is free to take new orders...
Processing order :4 by Thread-3
Processing order :2 by Thread-1
Processing order :3 by Thread-2
Processing order :1 by Thread-0
Processing order :5 by Thread-4
Completed order :4
Completed order :3
Completed order :1
Completed order :2
Completed order :5
PS C:\Users\Admin\Desktop\java>