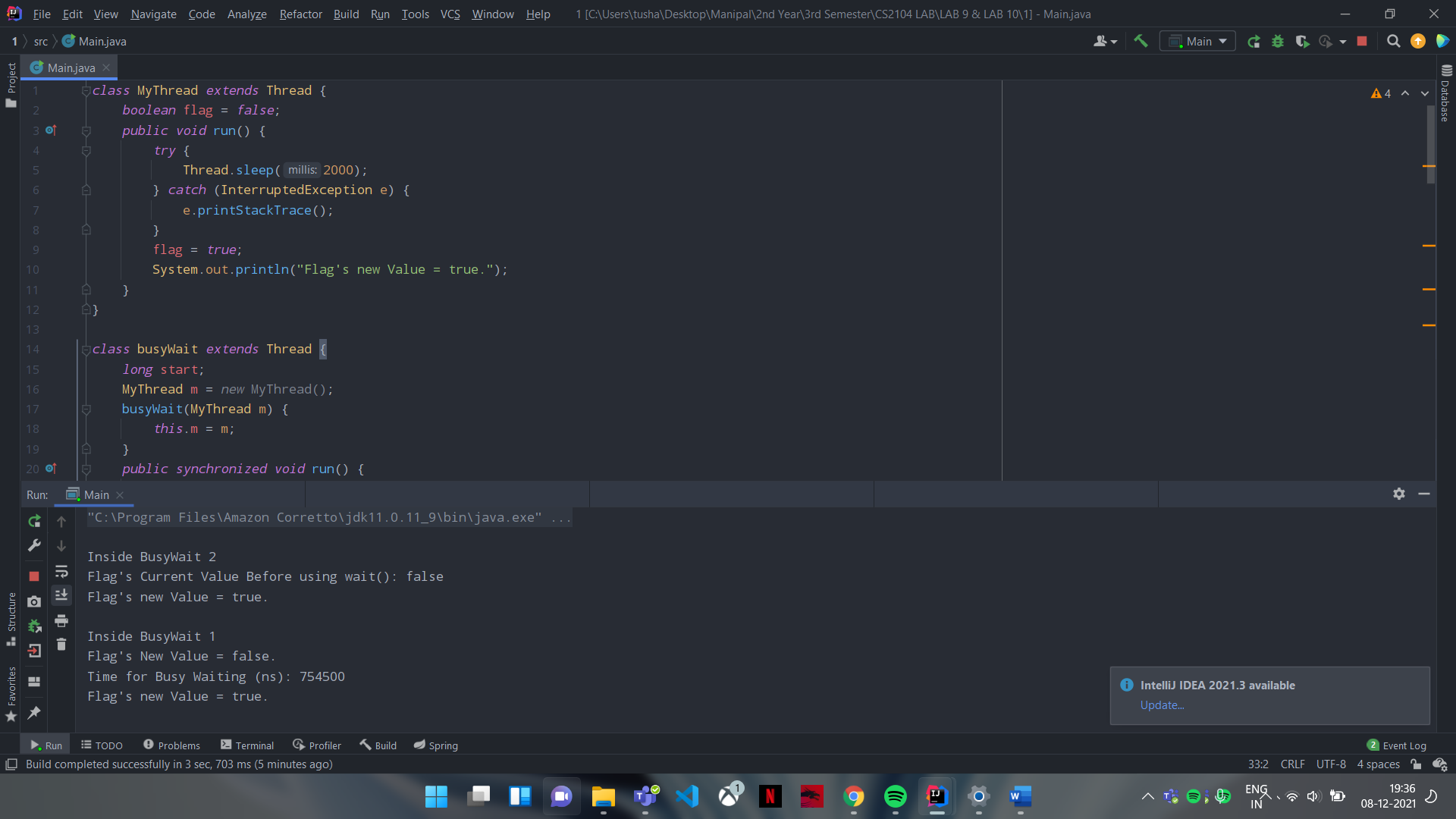
Name: Tushar(209301387) Sec: B

LAB MODULE 9&10

Exercise LAB Questions:

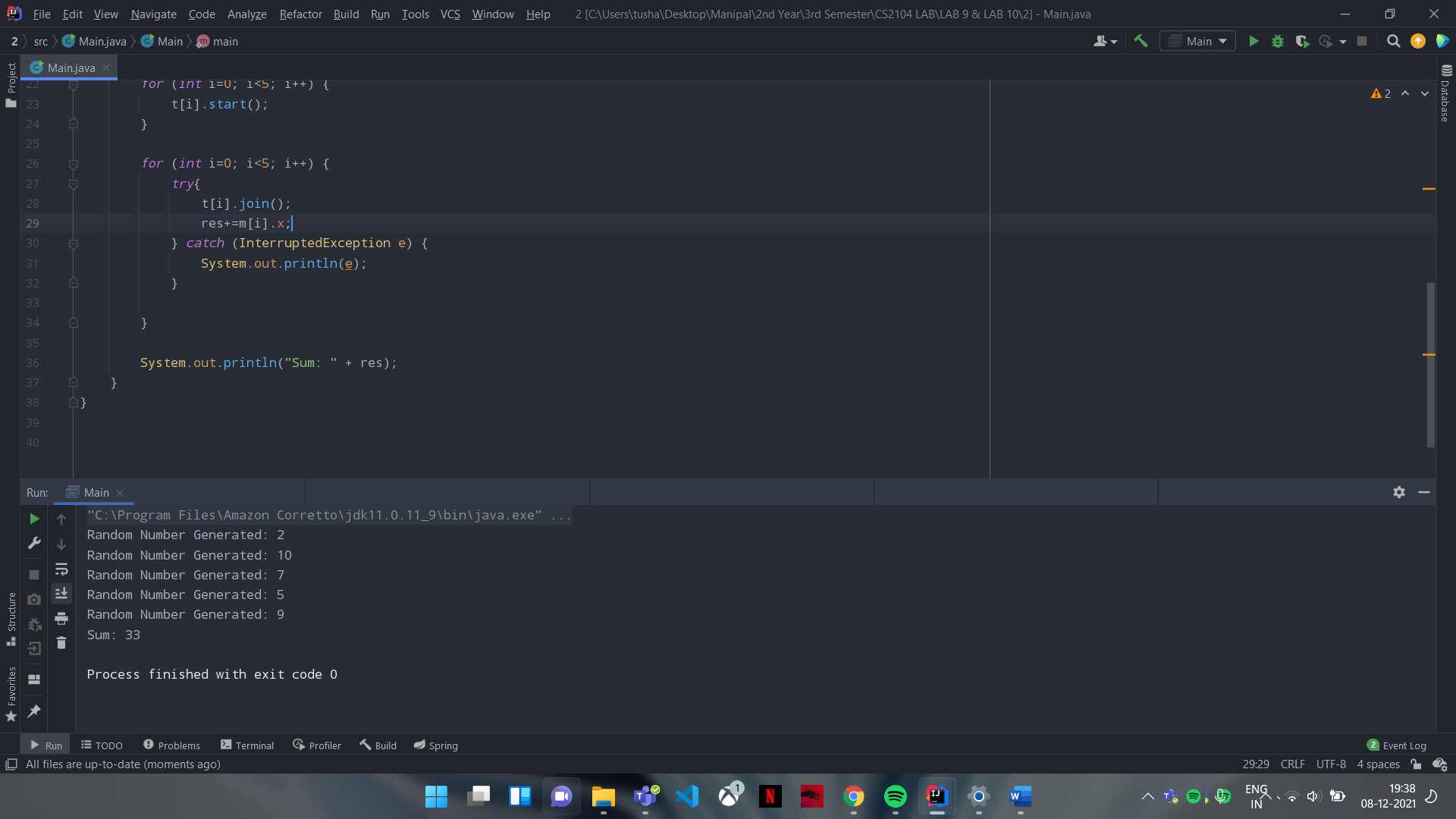
Q1.

**class** MyThread **extends** Thread {  
 **boolean flag** = **false**;  
 **public void** run() {  
 **try** {  
 Thread.*sleep*(2000);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **flag** = **true**;  
 System.***out***.println(**"Flag's new Value = true."**);  
 }  
}  
  
**class** busyWait **extends** Thread {  
 **long start**;  
 MyThread **m** = **new** MyThread();  
 busyWait(MyThread m) {  
 **this**.**m** = m;  
 }  
 **public synchronized void** run() {  
  
 **while**(!Thread.*interrupted*()) {  
 **start** = System.*nanoTime*();  
 **if**(**m**.**flag**) {  
 **m**.**flag** = **false**;  
 System.***out***.println();  
 System.***out***.println(**"Inside BusyWait 1"**);  
 System.***out***.println(**"Flag's New Value = false."**);  
 System.***out***.println(**"Time for Busy Waiting (ns): "** + (System.*nanoTime*()-**start**));  
 }  
 }  
 }  
}  
  
**class** busyWait2 **extends** Thread {  
 MyThread **m** = **new** MyThread();  
 busyWait2(MyThread m) {  
 **this**.**m** = m;  
 }  
 **public synchronized void** run() {  
 System.***out***.println(**"Inside BusyWait 2"**);  
 System.***out***.println(**"Flag's Current Value Before using wait(): "** + **m**.**flag**);  
 **while**(!**m**.**flag**) {  
 **try** {  
 wait();  
 } **catch** (InterruptedException e) {  
 System.***out***.println(e);  
 }  
 **m**.**flag**=**false**;  
 System.***out***.println(**"Flag value has been set to false."**);  
 }  
 }  
}  
  
**public class** Main {  
 **public static void** main(String[] Args) **throws** InterruptedException {  
 MyThread m1 = **new** MyThread();  
 MyThread m2 = **new** MyThread();  
  
 busyWait b1 = **new** busyWait(m1);  
 Thread t1 = **new** Thread(b1);  
 Thread t2 = **new** Thread(m1);  
 t2.start();  
 t1.start();  
  
 **try** {  
 Thread.*sleep*(300);  
 } **catch**(InterruptedException e) {  
 System.***out***.println(**"sleep interrupted in main()"**);  
 }  
  
 System.***out***.println();  
  
 busyWait2 b2 = **new** busyWait2(m2);  
 t1 = **new** Thread(m2);  
 t2 = **new** Thread(b2);  
 t2.start();  
 t1.start();  
  
 **try** {  
 Thread.*sleep*(2000);  
 } **catch**(InterruptedException e) {  
 System.***out***.println(**"sleep interrupted in main()"**);  
 }  
}}



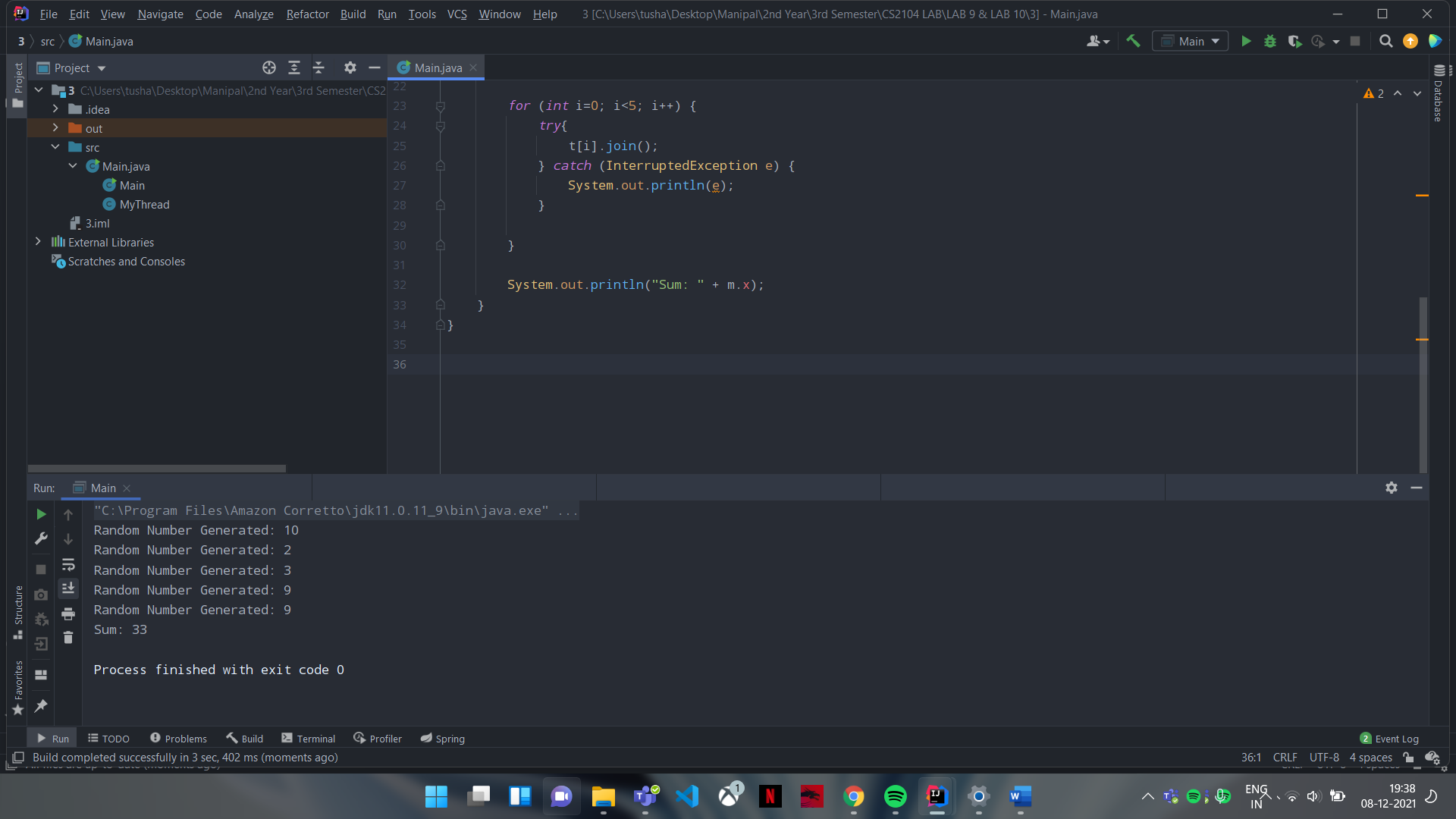
Q2.

**class** MyThread **implements** Runnable {  
 **int x**;  
 **public void** run() {  
 **int** n = (**int**) (Math.*random*()\*10 + 1);  
 System.***out***.println(**"Random Number Generated: "** + n);  
 **this**.**x**+=n;  
 }  
}  
  
**public class** Main {  
 **public static void** main(String[] Args) **throws** InterruptedException {  
 MyThread[] m = **new** MyThread[5];  
 Thread[] t = **new** Thread[5];  
  
 **int** res=0;  
  
 **for** (**int** i=0; i<5; i++) {  
 m[i] = **new** MyThread();  
 t[i] = **new** Thread(m[i]);  
 }  
  
 **for** (**int** i=0; i<5; i++) {  
 t[i].start();  
 }  
  
 **for** (**int** i=0; i<5; i++) {  
 **try**{  
 t[i].join();  
 res+=m[i].**x**;  
 } **catch** (InterruptedException e) {  
 System.***out***.println(e);  
 }  
  
 }  
  
 System.***out***.println(**"Sum: "** + res);  
 }  
}



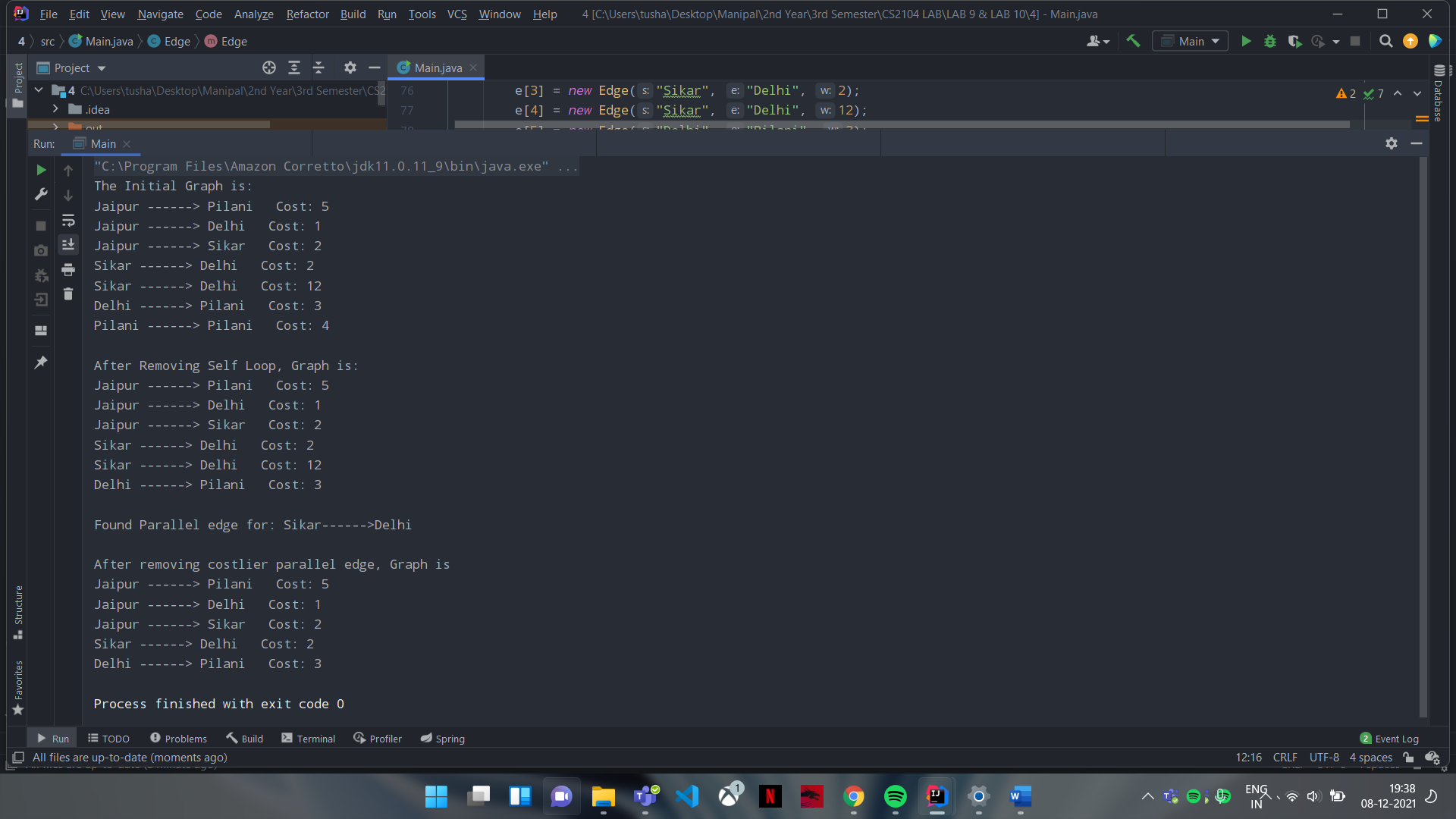
Q3.

**class** MyThread **implements** Runnable {  
 **int x**;  
 **public void** run() {  
 **int** n = (**int**) (Math.*random*()\*10 + 1);  
 System.***out***.println(**"Random Number Generated: "** + n);  
 **this**.**x**+=n;  
 }  
}  
  
**public class** Main {  
 **public static void** main(String[] Args) **throws** InterruptedException {  
 MyThread m = **new** MyThread();  
 Thread[] t = **new** Thread[5];  
  
 **for** (**int** i=0; i<5; i++) {  
 t[i] = **new** Thread(m);  
 }  
  
 **for** (**int** i=0; i<5; i++) {  
 t[i].start();  
 }  
  
 **for** (**int** i=0; i<5; i++) {  
 **try**{  
 t[i].join();  
 } **catch** (InterruptedException e) {  
 System.***out***.println(e);  
 }  
  
 }  
  
 System.***out***.println(**"Sum: "** + m.**x**);  
 }  
}



Q4.

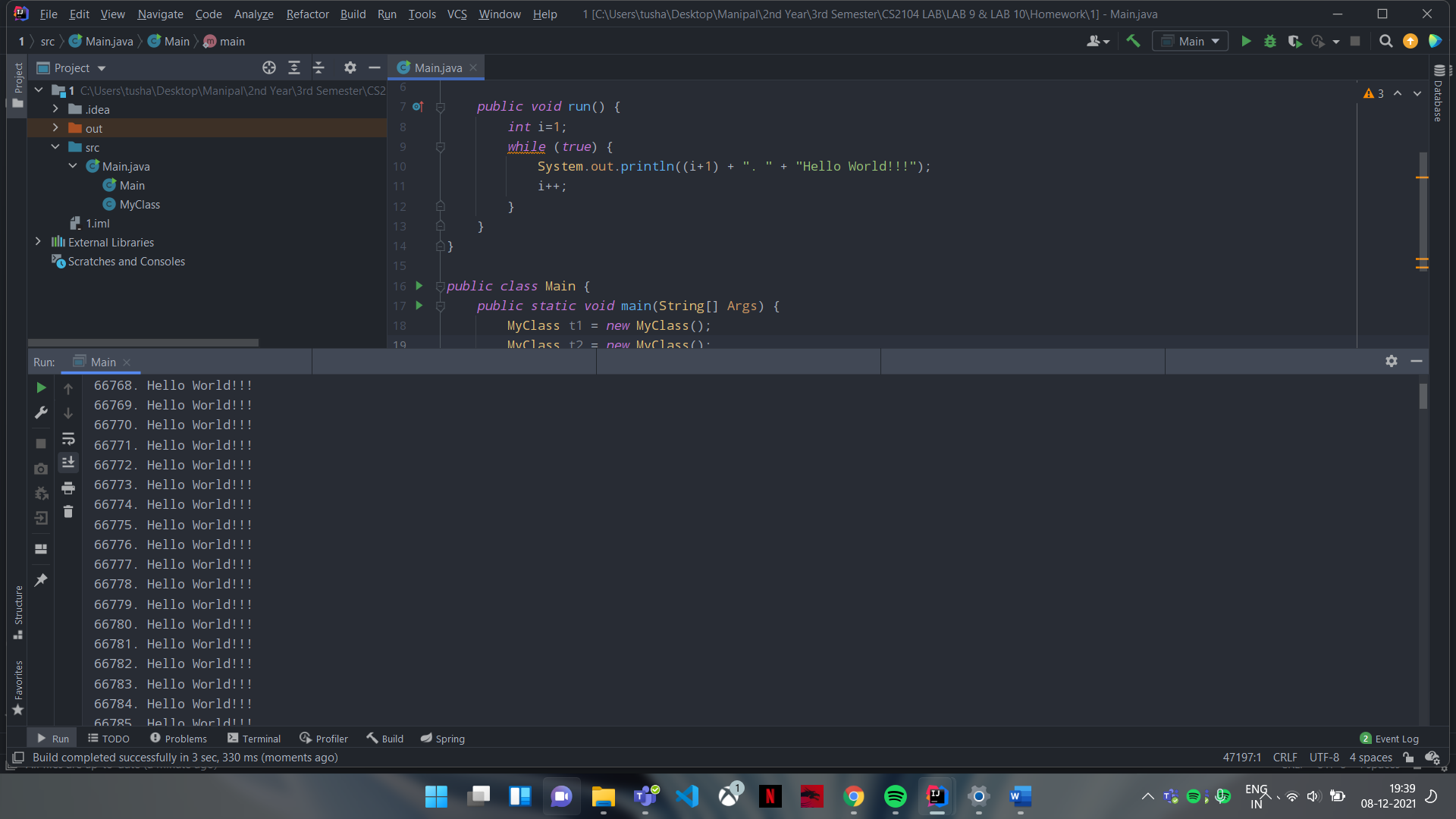
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
**import** java.util.Scanner;  
  
**class** Edge {  
 **public** String **start**; *// start node of an edge* **public** String **end**; *// End node of an edge* **public int cost**; *// cost of wire* Edge(String s, String e, **int** w) {  
 **start** =s;  
 **end** =e;  
 **cost** =w;  
 }  
} *// end of edge class***class** Graph {  
 **public static** ArrayList<Edge> *list\_edges*; *// list of edges of a graph* Graph(ArrayList<Edge> n) {  
 *list\_edges* = n;  
 }  
  
 **public void** removeLoop() {  
 *//Q.1.1 write the java code to remove loops from the list\_edges:ArrayList. \*/  
  
 // int temp = list\_edges.indexOf(x);  
 list\_edges*.removeIf(x -> x.**start**.equals(x.**end**));  
 }  
  
 **public** Edge findParallelEdge(Edge n) {  
 *//Q.1.2 This function finds a parallel edge of n:Edge inlist\_edges:ArrayList.  
 // If parallel edge exists it returns parallel edge otherwise null. Write the java code\*/* **for** (Edge x: *list\_edges*) {  
 **if**(x.**start**.equals(n.**start**) && x.**end**.equals(n.**end**)) {  
 **return** x;  
 }  
 }  
  
 **return null**;  
 }  
  
 **public void** removeParallel() {  
 *//Q.1.3 This method traverses list\_edges:ArrayList and finds a parallel edge.  
 // if an Edge’s parallel edge is found and Edge’s cost is higher than parallel edge it removes Edge.  
 // Write the java code \*/* **for** (**int** i=0; i<*list\_edges*.size(); i++) {  
 **for** (**int** j=i+1; j<*list\_edges*.size(); j++) {  
 **if**(*list\_edges*.get(i).**start**.equals(*list\_edges*.get(j).**start**) && *list\_edges*.get(i).**end**.equals(*list\_edges*.get(j).**end**)) {  
 **if**(*list\_edges*.get(i).**cost**<=*list\_edges*.get(j).**cost**) {  
 *list\_edges*.remove(j);  
 } **else** {  
 *list\_edges*.remove(i);  
 }  
 }  
 }  
 }  
 }  
  
 **public void** print() {  
 *// Q.1.4 write the java code to print edges of list\_edges:ArrayList* **for** (Edge x: *list\_edges*) {  
 System.***out***.println(x.**start** + **" ------> "** + x.**end** + **" Cost: "** + x.**cost**);  
 }  
 }  
}  
  
**public class** Main {  
 **public static void** main(String[] Args) {  
 Scanner sc = **new** Scanner(System.***in***);  
 Edge[] e = **new** Edge[7];  
  
 e[0] = **new** Edge(**"Jaipur"**, **"Pilani"**, 5);  
 e[1] = **new** Edge(**"Jaipur"**, **"Delhi"**, 1);  
 e[2] = **new** Edge(**"Jaipur"**, **"Sikar"**, 2);  
 e[3] = **new** Edge(**"Sikar"**, **"Delhi"**, 2);  
 e[4] = **new** Edge(**"Sikar"**, **"Delhi"**, 12);  
 e[5] = **new** Edge(**"Delhi"**, **"Pilani"**, 3);  
 e[6] = **new** Edge(**"Pilani"**, **"Pilani"**, 4);  
  
 ArrayList<Edge> graph = **new** ArrayList<>(Arrays.*asList*(e).subList(0, 7));  
  
  
 Graph g = **new** Graph(graph);  
 System.***out***.println(**"The Initial Graph is: "**);  
 g.print();  
 System.***out***.println();  
  
 g.removeLoop();  
 System.***out***.println(**"After Removing Self Loop, Graph is: "**);  
 g.print();  
 System.***out***.println();  
  
 Edge temp = g.findParallelEdge(e[3]);  
 **if**(temp!=**null**) {  
 System.***out***.println(**"Found Parallel edge for: "** + temp.**start** + **"------>"** + temp.**end**);  
 } **else** {  
 System.***out***.println(**"Not having parallel Edge"**);  
 }  
 System.***out***.println();  
  
 g.removeParallel();  
 System.***out***.println(**"After removing costlier parallel edge, Graph is"**);  
 g.print();  
 }  
  
}



Homework LAB Questions:

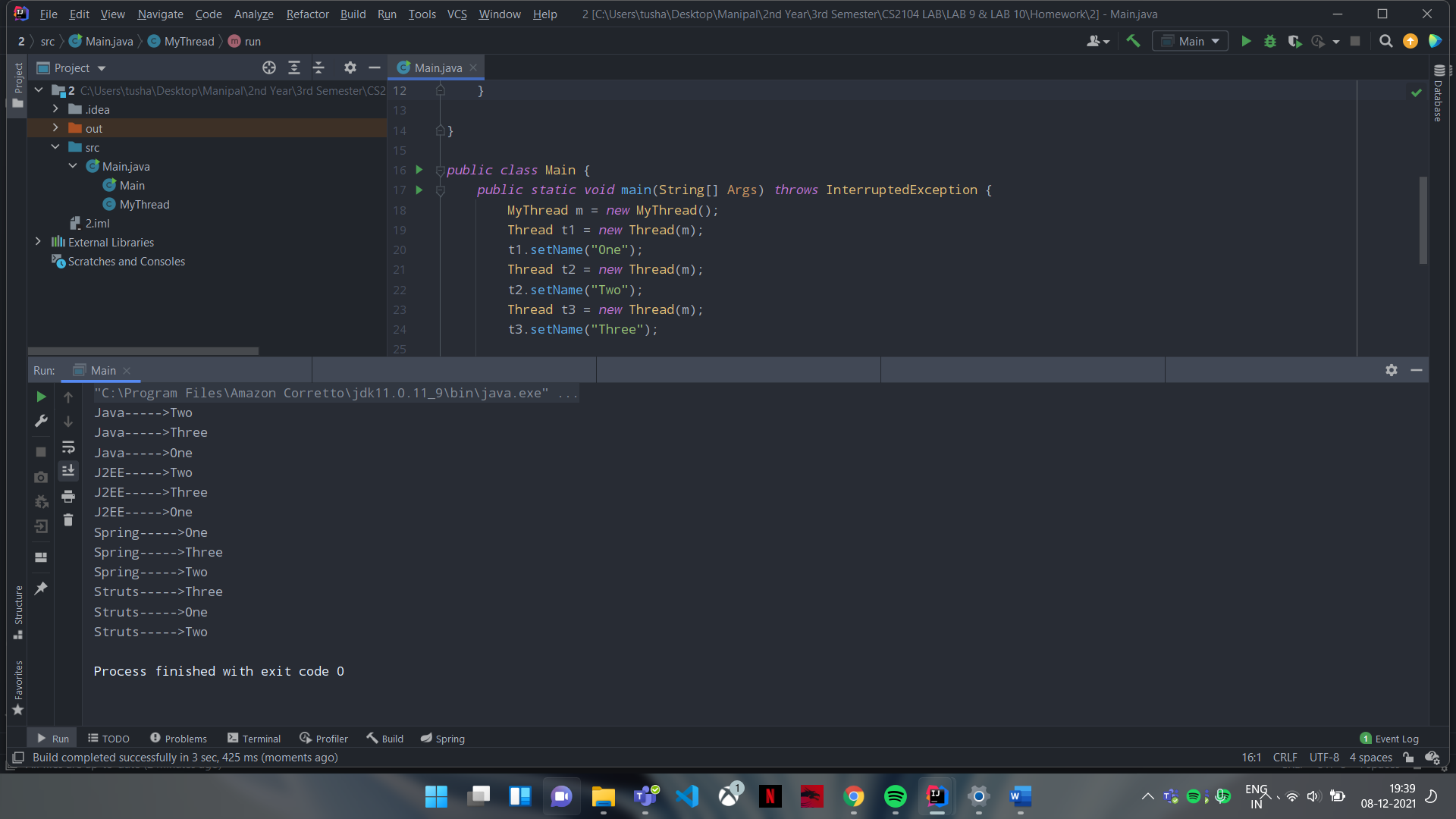
Q1.

**class** MyClass **extends** Thread {  
  
 MyClass() {  
 **this**.start();  
 }  
  
 **public void** run() {  
 **int** i=1;  
 **while** (**true**) {  
 System.***out***.println((i+1) + **". "** + **"Hello World!!!"**);  
 i++;  
 }  
 }  
}  
  
**public class** Main {  
 **public static void** main(String[] Args) {  
 MyClass t1 = **new** MyClass();  
 MyClass t2 = **new** MyClass();  
 }  
}



Q2.

**class** MyThread **implements** Runnable {  
 String[] **course** = {**"Java"**, **"J2EE"**, **"Spring"**, **"Struts"**};  
 **public void** run() {  
 **for** (String x: **course**) {  
 System.***out***.println(x + **"----->"** + Thread.*currentThread*().getName());  
 **try** {  
 Thread.*sleep*(1000);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
}  
  
**public class** Main {  
 **public static void** main(String[] Args) **throws** InterruptedException {  
 MyThread m = **new** MyThread();  
 Thread t1 = **new** Thread(m);  
 t1.setName(**"One"**);  
 Thread t2 = **new** Thread(m);  
 t2.setName(**"Two"**);  
 Thread t3 = **new** Thread(m);  
 t3.setName(**"Three"**);  
  
 t1.start();  
 t2.start();  
 t3.start();  
  
 Thread.*sleep*(1000);  
 }  
}



Q3.

**class** Inventory {  
 **static int** *quantityOnHand* = 500;  
 **static int** *req* = 0;  
  
 **public static void** request(**int** order) {  
 **synchronized** (Inventory.**class**) {  
 **if** (order <= *quantityOnHand*) {  
 System.***out***.println(**"Quantity ordered :"**+order);  
 *quantityOnHand* -= order;  
 *req* += order;  
 System.***out***.println(**"Quantity on hand :"**+*quantityOnHand*);  
 System.***out***.println(**"Total quantify taken away by way of order :"**+*req*);  
 } **else** {  
 System.***out***.println(**"Ordered quantity more than quantity on hand"**);  
 }  
 System.***out***.println();  
 }  
 }  
}  
  
**class** OurThread **extends** Thread {  
 OurThread() {  
 **super**(**"test thread"**);  
 System.***out***.println(**this**);  
 start();  
 }  
  
 **public void** run() {  
 **for** (**int** i = 5; i > 0; i--) {  
 **try** {  
 *sleep*(100);  
 } **catch** (InterruptedException e) {  
 }  
 Inventory.*request*((**int**) (Math.*random*() \* 100));  
 }  
 }  
}  
  
**public class** Main {  
 **public static void** main(String args[]) {  
 **new** OurThread();  
 **new** OurThread();  
  
 **try** {  
 **for** (**int** p = 3; p > 0; p--) {  
 Thread.*sleep*(1000);  
 }  
 } **catch** (InterruptedException e) {  
 System.***out***.println(e);  
 }  
 }  
}

