OOP LAB

Assignment – 03

**Q1. Write a java program to illustrate 4 different possible NullPointerExeptions cases**

**possible.**

**Solu.**

/\*  
 \* Copyright (c) 2021.  
 \* Divyanshu Tyagi  
 \* NIT Warangal  
 \* 207919  
 \*/  
  
package Assignment3;  
  
public class sol1 {  
 public static void main(String[] args) {  
 System.*out*.println("Null pointer Exception");  
  
 try {  
 System.*out*.println("1. Invoking a method from a null object");  
 String str = null;  
 String str1 = "hello";  
 if(str.equals(str1)) {  
 System.*out*.println("Str equals to hello");  
 }else {  
 System.*out*.println("Str not eqal to hello");  
 }  
 }catch (NullPointerException exp) {  
 System.*out*.println("NullPointerException : Case 1");  
 }  
 System.*out*.println("---------------------------------------");  
 try {  
 System.*out*.println("2. Accessing or modifying a null object’s field");  
 Integer x = null;  
 x++;  
 System.*out*.println("New Value of x is : " + x);  
 }catch (NullPointerException exp) {  
 System.*out*.println("NullPointerException : Case 2");  
 }  
 System.*out*.println("---------------------------------------");  
 try {  
 System.*out*.println("3. Taking the length of null, as if it were an array");  
 int arr[] = null;  
 System.*out*.println("Length of array is : " + arr.length);  
 }catch (NullPointerException exp) {  
 System.*out*.println("NullPointerException : Case 3");  
 }  
 System.*out*.println("---------------------------------------");  
 try {  
 System.*out*.println("4. Throwing null, as if it were a Throwable value");  
 throw null;  
 }catch (NullPointerException exp) {  
 System.*out*.println("NullPointerException : Case 4");  
 }  
 }  
}

**OUTPUT:**

****

**Q2. You have been assigned to create a student database (Just information) of our**

**institute. For a class named StudentInfo, create several objects for this class, each**

**object representing one student (means array of objects). StudentInfo class is**

**different and main class is different.The number of students should be taken from the user console. Student details should be like id, name, department name, college name. College name should be common to all of them (means this should be class variable). Now**

**student id of each student should be unique, if any of the student id is repeating**

**then handle with an user-defined exception. Name and dept name should not be**

**empty, if they are empty then handle with user-defined exception. After all store**

**the student’s data inside a file using stream classes not by using file classes. And**

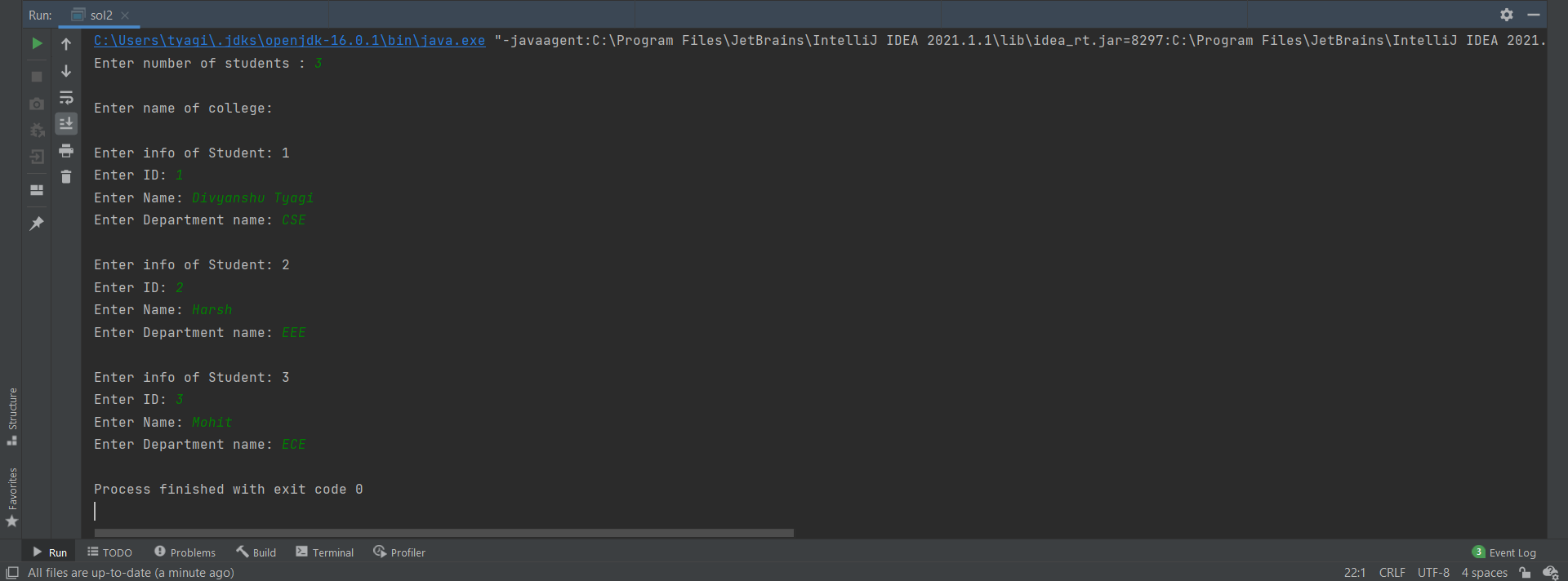
**number of students should be minimum of 10.**

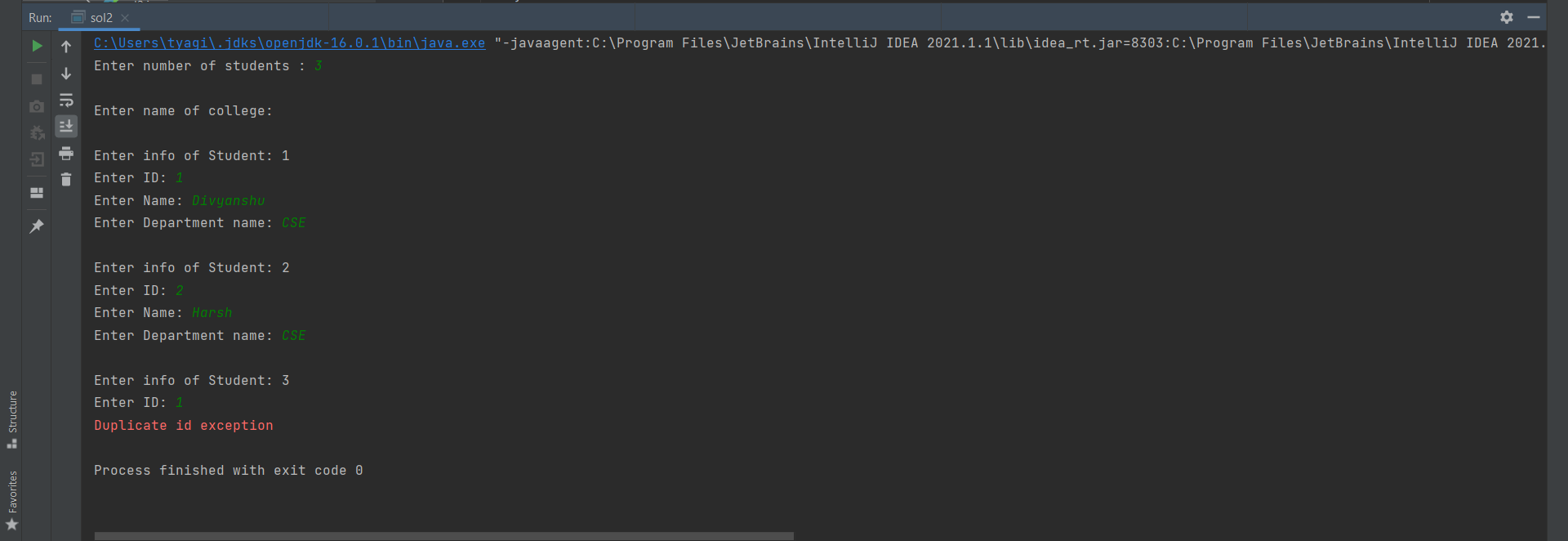
**Solu.**

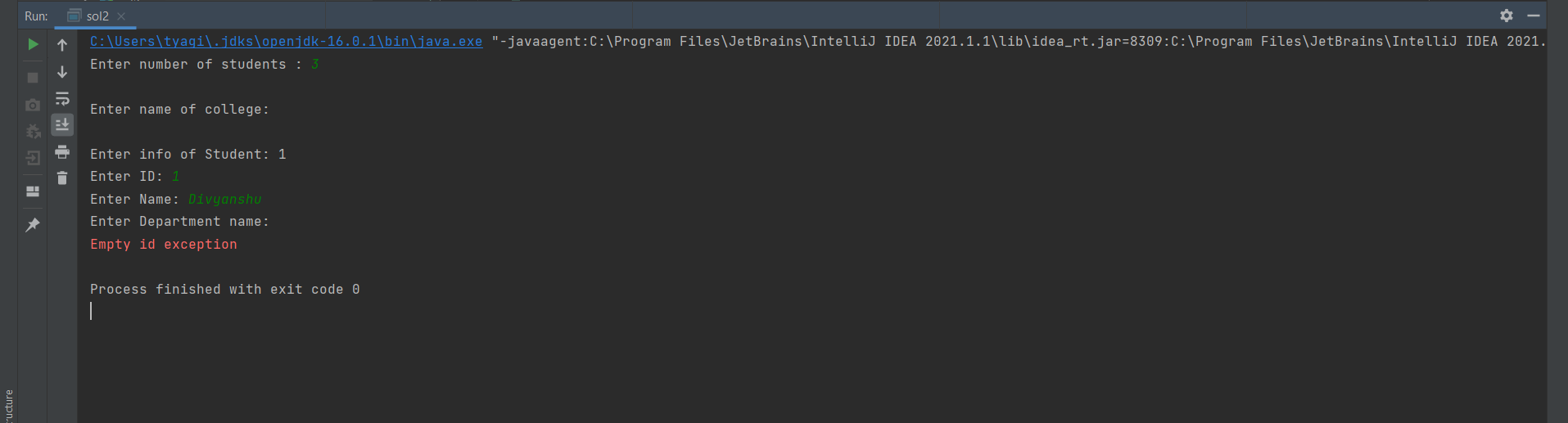
/\*  
 \* Copyright (c) 2021.  
 \* Divyanshu Tyagi  
 \* NIT Warangal  
 \* 207919  
 \*/  
  
package Assignment3;  
  
import java.io.\*;  
import java.nio.charset.StandardCharsets;  
import java.util.HashSet;  
import java.util.Scanner;  
  
class DuplicateIdException extends Exception{  
 DuplicateIdException(String prompt) {  
 super(prompt);  
 }  
}  
  
class EmptyStringException extends Exception {  
 EmptyStringException(String prompt) {  
 super(prompt);  
 }  
}  
  
class StudentInfo {  
  
 private String name, depart\_name;  
 private Integer id;  
 static private String *college\_name*;  
 public StudentInfo(Integer id,String name, String depart\_name) {  
 this.id = id;  
 this.name = name;  
 this.depart\_name = depart\_name;  
 }  
 public String getInfo() {  
 return id + "|" + name + "|" + depart\_name + "|" + *college\_name* + "|";  
 }  
  
 public static void setCollege(String col\_name) {  
 *college\_name* = col\_name;  
 }  
}  
public class sol2 {  
 static private HashSet<Integer> *idSet* = new HashSet<Integer>();  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter number of students : ");  
 int n = sc.nextInt();  
 System.*out*.println();  
 StudentInfo students[] = new StudentInfo[n];  
 try {  
 System.*out*.print("Enter name of college: ");  
 String college = sc.nextLine();  
 System.*out*.println();  
 StudentInfo.*setCollege*(college);  
 for (int i = 0; i < n; i++) {

System.out.println();  
 System.*out*.println("Enter info of Student: " + (i+1));  
 StudentInfo tempStudent = *input*();  
 students[i] = tempStudent;  
 }  
 }catch (DuplicateIdException e) {  
 System.*err*.println("Duplicate id exception");  
 System.*exit*(0);  
 }catch (EmptyStringException e) {  
 System.*err*.println("Empty id exception");  
 System.*exit*(0);  
 }  
  
 FileOutputStream fout;  
 BufferedOutputStream buff;  
  
 try {  
 fout = new FileOutputStream("students.txt");  
 buff = new BufferedOutputStream(fout);  
 for (int i = 0; i < n; i++) {  
 byte data[] = students[i].getInfo().getBytes(StandardCharsets.*UTF\_8*);  
 buff.write(data);  
 }  
 buff.flush();  
 buff.close();  
 fout.close();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static StudentInfo input() throws DuplicateIdException, EmptyStringException {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter ID: ");  
 Integer id = Integer.*parseInt*(sc.nextLine());  
 if(!*idSet*.isEmpty() && *idSet*.contains(id)) {  
 throw new DuplicateIdException("Student data already contains id : " + id);  
 }else {  
 *idSet*.add(id);  
 }  
 System.*out*.print("Enter Name: ");  
 String name = sc.nextLine();  
 System.*out*.print("Enter Department name: ");  
 String department = sc.nextLine();  
 // check if name or department name is empty or not  
 if(name.isEmpty() || department.isEmpty()) {  
 throw new EmptyStringException("Enter name or department name is empty");  
 }  
 return new StudentInfo(id,name,department);  
 }  
}

**OUTPUT:**

****

****

****

**Q3. Write a java program to throw a custom exception when you are trying to read**

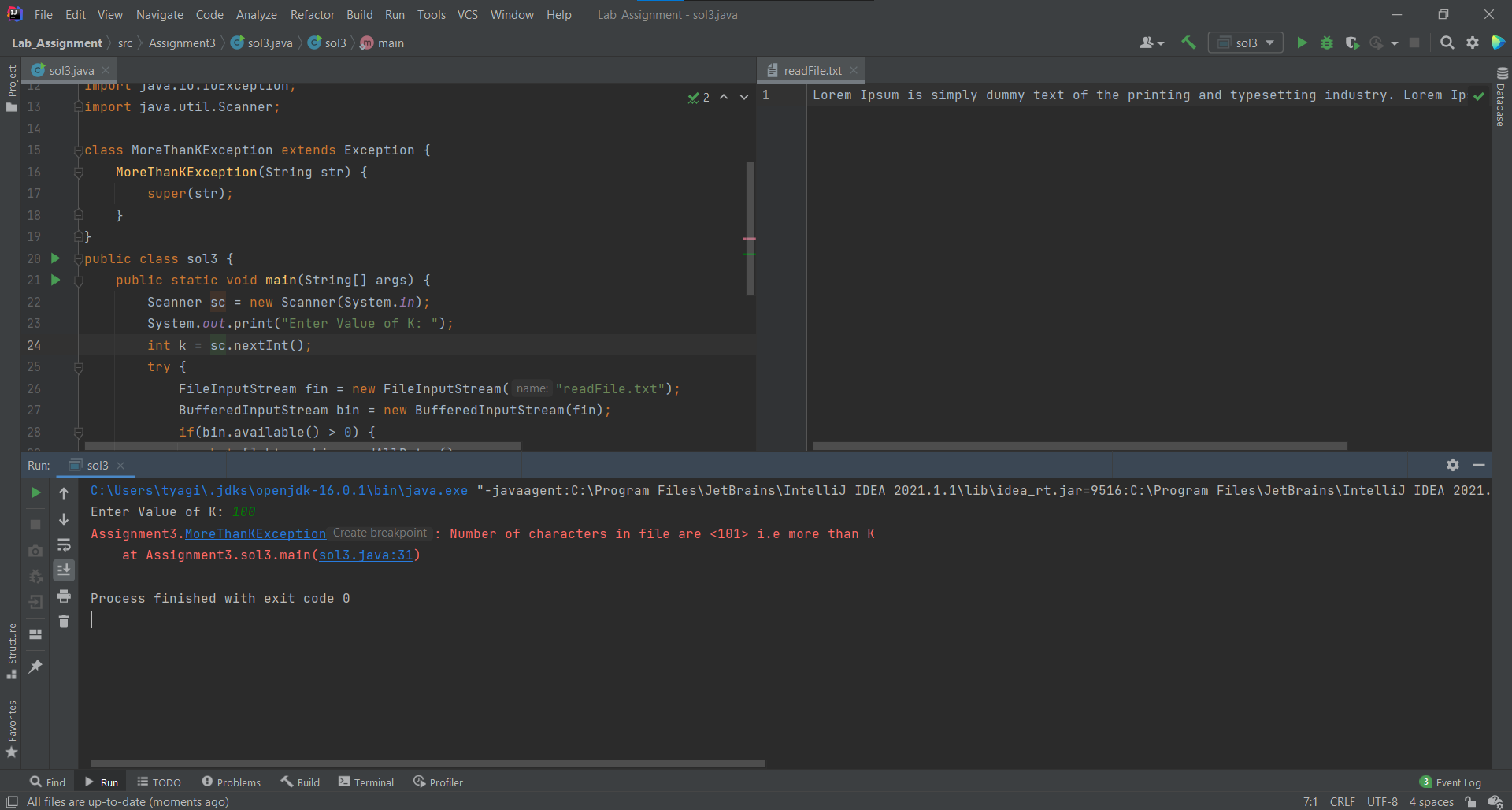
**characters from a file, if number of characters in the file are more than specified**

**number (K), Note K should be >=100.**

**Solu.**

/\*  
 \* Copyright (c) 2021.  
 \* Divyanshu Tyagi  
 \* NIT Warangal  
 \* 207919  
 \*/  
  
package Assignment3;  
  
import java.io.BufferedInputStream;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.util.Scanner;  
  
class MoreThanKException extends Exception {  
 MoreThanKException(String str) {  
 super(str);  
 }  
}  
public class sol3 {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter Value of K: ");  
 int k = sc.nextInt();  
 try {  
 FileInputStream fin = new FileInputStream("readFile.txt");  
 BufferedInputStream bin = new BufferedInputStream(fin);  
 if(bin.available() > 0) {  
 byte[] bts = bin.readAllBytes();  
 if(bts.length > k)  
 throw new MoreThanKException("Number of characters in file are <" + bts.length + "> i.e more than K");  
  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 } catch (MoreThanKException e) {  
 e.printStackTrace();  
 System.*exit*(0);  
 }  
 }  
}

**OUTPUT:**

****

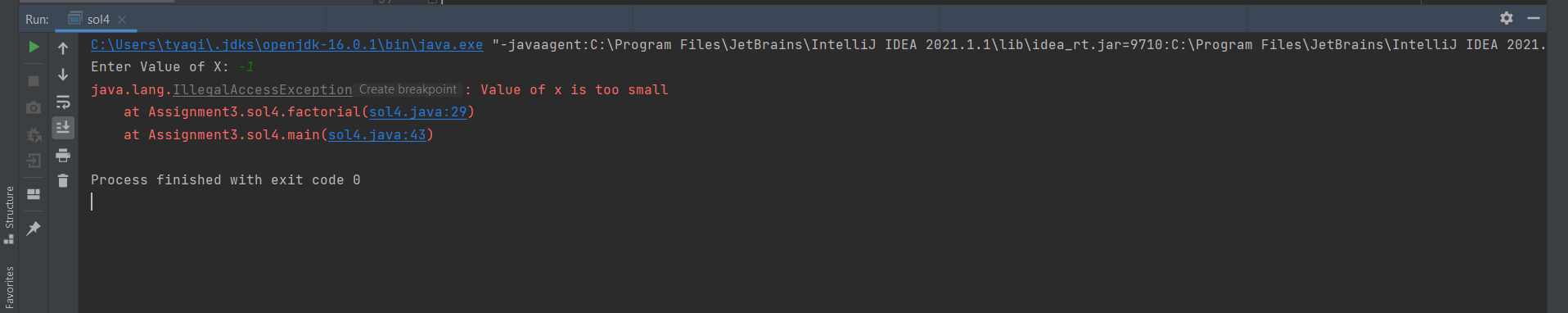
**Q4. Write a java program called factorial.java that computes the factorials and catches**

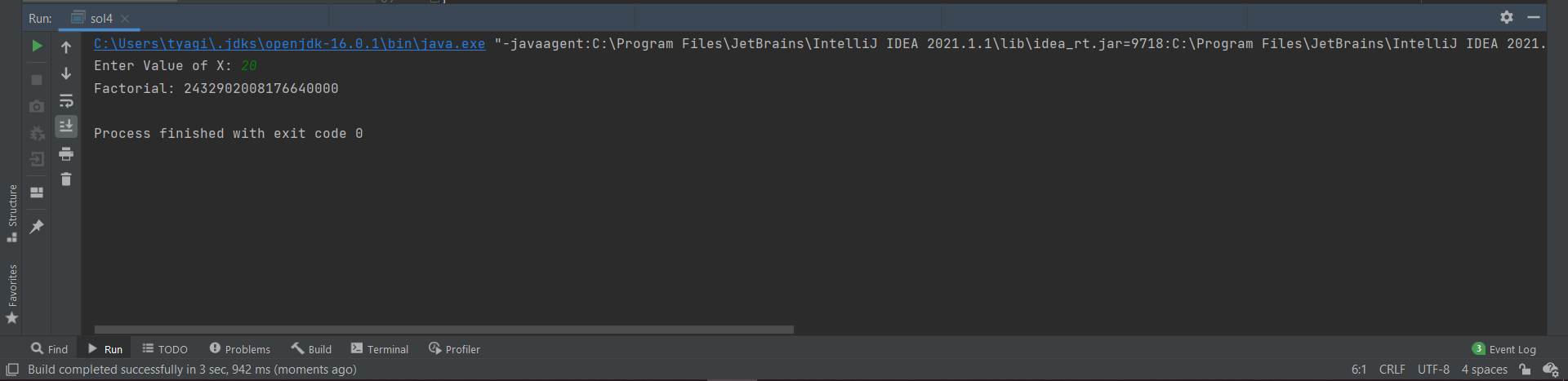
**the result in a long variable. The long type of variable has its own range. For example, 20! Is as high as the range of long type. So, check the argument passed and throw an exception, if it is too big or too small. If x is less than 0 throw an IllegalArgumentException with a message. If x is more than specific range then throw a custom exception saying “Result will overflow”. Here x is a value for which we want to find factorial (it should be user-defined).**

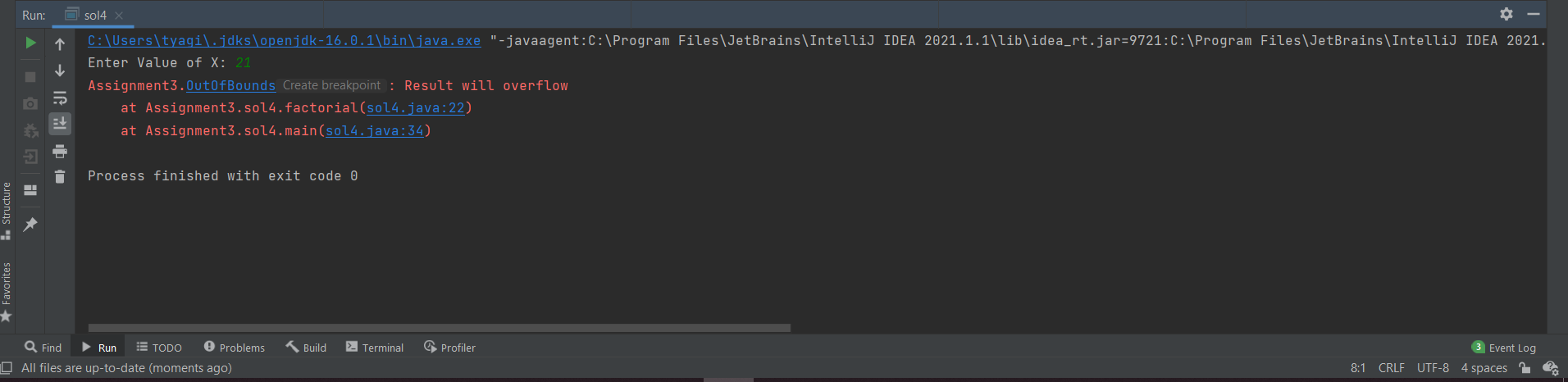
**Solu.**

/\*  
 \* Copyright (c) 2021.  
 \* Divyanshu Tyagi  
 \* NIT Warangal  
 \* 207919  
 \*/  
  
package Assignment3;  
  
import java.util.Scanner;  
  
class OutOfBounds extends Exception {  
 OutOfBounds(String str) {  
 super(str);  
 }  
}  
public class sol4 {  
 public static long factorial(long x) throws OutOfBounds, IllegalAccessException {  
 if(x < 0)  
 throw new IllegalAccessException("Value of x is too small");  
 if(x > 20)  
 throw new OutOfBounds("Result will overflow");  
 long fact = 1;  
 for(int i = 2; i <= x; i++) {  
 fact \*= i;  
 }  
 return fact;  
 }  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter Value of X: ");  
 int x = sc.nextInt();  
 try {  
 System.*out*.println("Factorial: " + *factorial*(x));  
 }catch (OutOfBounds | IllegalAccessException e) {  
 e.printStackTrace();  
 }  
 }  
}

**OUTPUT:**

****

****

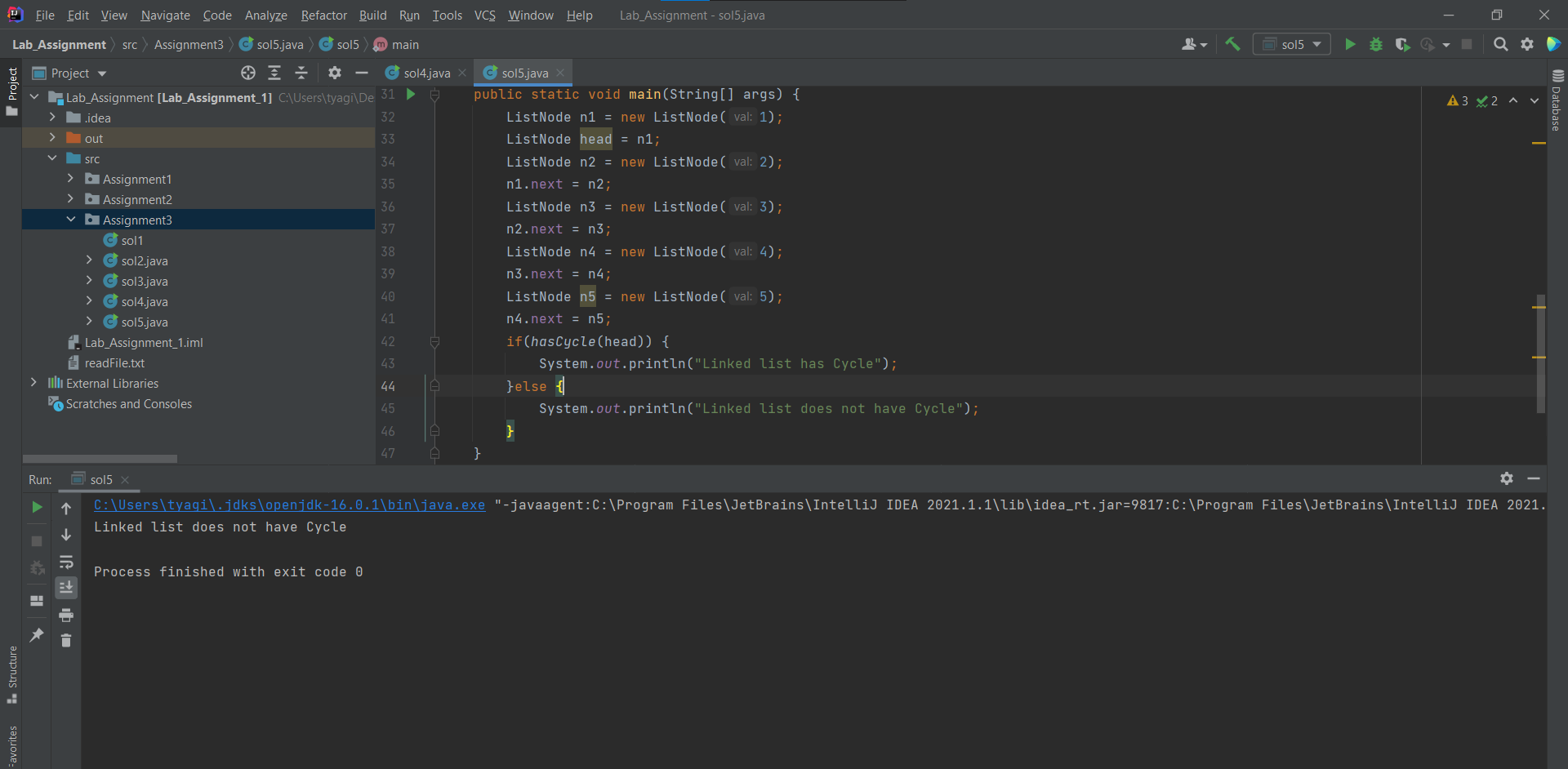
****

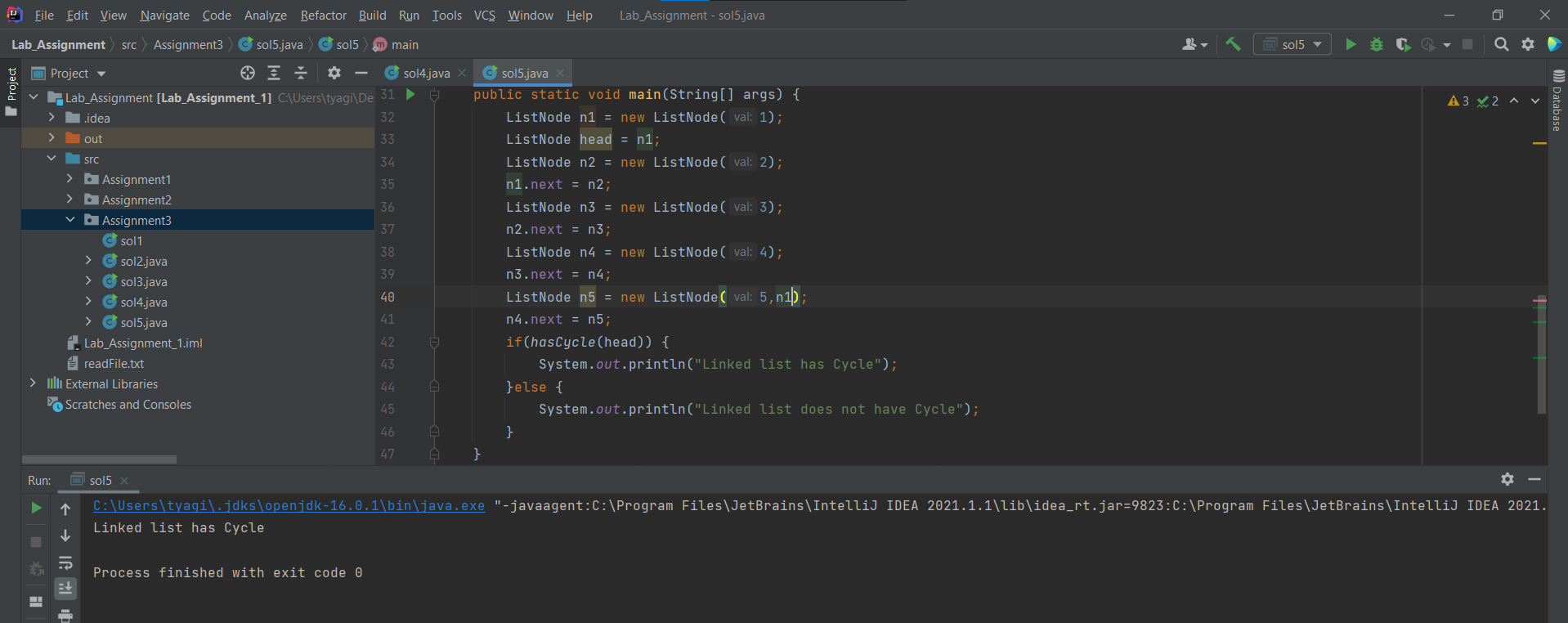
**Q5. Write a java program to detect a loop in the linked list.**

**Solu.**

/\*  
 \* Copyright (c) 2021.  
 \* Divyanshu Tyagi  
 \* NIT Warangal  
 \* 207919  
 \*/  
  
package Assignment3;  
  
import java.util.List;  
  
class ListNode {  
 int val;  
 ListNode next;  
 ListNode(int val) {this.val = val; this.next = null;}  
 ListNode(int val, ListNode next) {this.val = val; this.next = next;}  
}  
public class sol5 {  
 public static boolean hasCycle(ListNode head) {  
 if(head==null) return false;  
 ListNode slow = head;  
 ListNode fast = head;  
 while(fast.next!=null && fast.next.next!=null) {  
 slow = slow.next;  
 fast = fast.next.next;  
 if(fast == slow) return true;  
 }  
 return false;  
 }  
  
 public static void main(String[] args) {  
 ListNode n1 = new ListNode(1);  
 ListNode head = n1;  
 ListNode n2 = new ListNode(2);  
 n1.next = n2;  
 ListNode n3 = new ListNode(3);  
 n2.next = n3;  
 ListNode n4 = new ListNode(4);  
 n3.next = n4;  
 ListNode n5 = new ListNode(5,n1);  
 n4.next = n5;  
 if(*hasCycle*(head)) {  
 System.*out*.println("Linked list has Cycle");  
 }else {  
 System.*out*.println("Linked list does not have Cycle");  
 }  
 }  
}

**OUTPUT:**

****

****

**Q6. Write a java program to write name and phone numbers of 15 people inside a file**

**each line contains name and phone number. Now enter user defined variable called**

**phone number, you need to match (compare) this phone number against all the**

**phone numbers in the file, if matches then print corresponding name on to the**

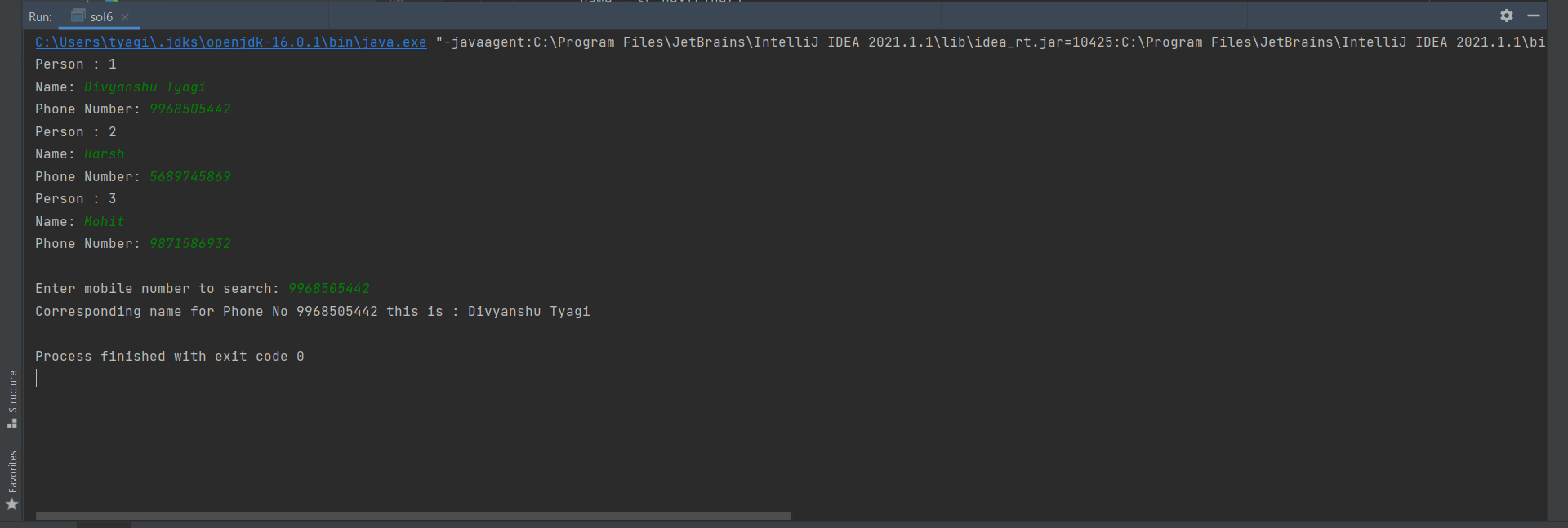
**screen, if not present then throw a custom exception.**

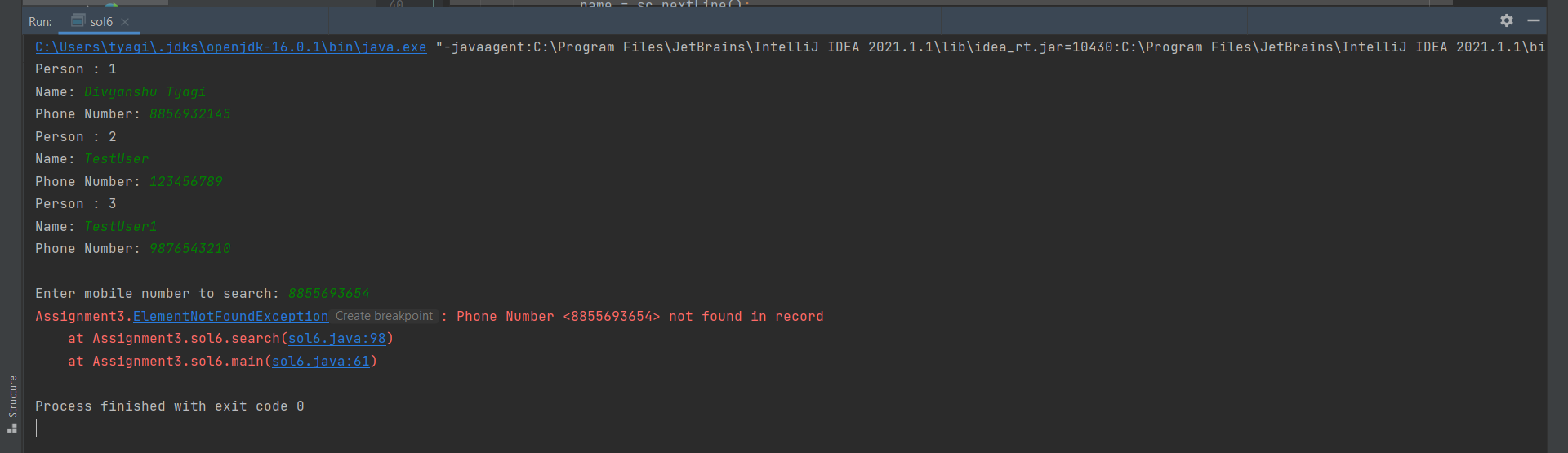
**Solu.**

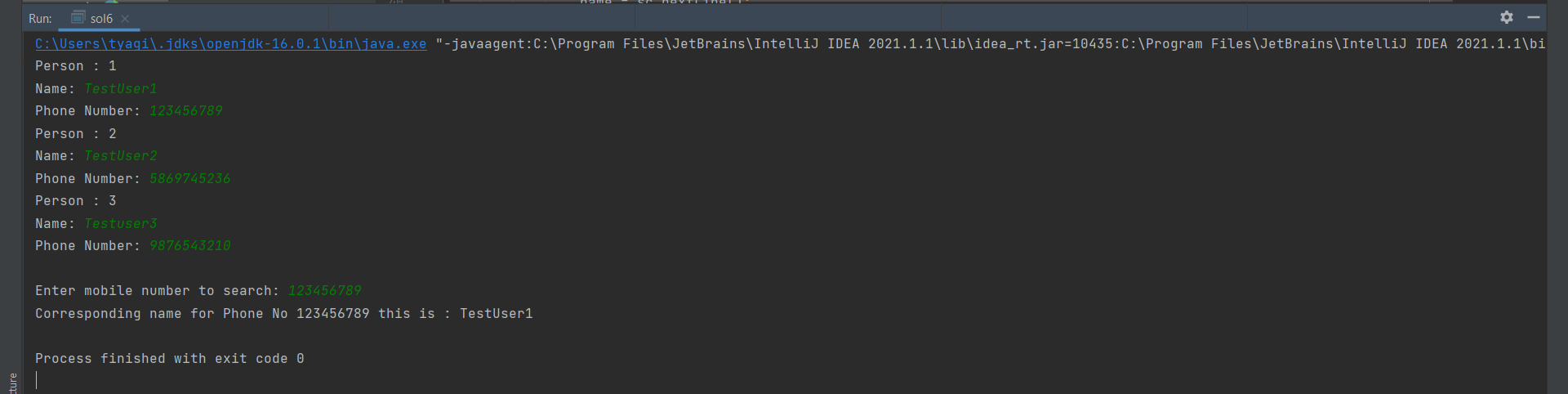
/\*  
 \* Copyright (c) 2021.  
 \* Divyanshu Tyagi  
 \* NIT Warangal  
 \* 207919  
 \*/  
  
package Assignment3;  
/\*  
Write a java program to write name and phone numbers of 15 people inside a file  
each line contains name and phone number. Now enter user defined variable called  
phone number, you need to match (compare) this phone number against all the  
phone numbers in the file, if matches then print corresponding name on to the  
screen, if not present then throw a custom exception.  
\*/  
  
import java.io.\*;  
import java.nio.charset.StandardCharsets;  
import java.util.Scanner;  
  
class ElementNotFoundException extends Exception {  
 ElementNotFoundException(String str) {  
 super(str);  
 }  
}  
  
public class sol6 {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 FileOutputStream fout = null;  
 BufferedOutputStream buff = null;  
 try {  
 fout = new FileOutputStream("phoneDir.txt");  
 buff = new BufferedOutputStream(fout);  
 for (int i = 0; i < 15; i++) {  
 System.*out*.println("Person : " + (i+1));  
 String phoneNumber;  
 String name;  
 System.*out*.print("Name: ");  
 name = sc.nextLine();  
 name += " ";  
 System.*out*.print("Phone Number: ");  
 phoneNumber = sc.nextLine();  
 byte bt[] = name.getBytes(StandardCharsets.*UTF\_8*);  
 buff.write(bt);  
 bt = phoneNumber.getBytes(StandardCharsets.*UTF\_8*);  
 buff.write(bt);  
 buff.write(System.*getProperty*("line.separator").getBytes());  
 }  
 buff.flush();  
 buff.close();  
 fout.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 System.*out*.println();  
 String ph = null;  
 System.*out*.print("Enter mobile number to search: ");  
 ph = sc.nextLine();  
 *search*(ph);  
 }  
  
 static void search(String phoneNumber) {  
 boolean found = false;  
 FileInputStream fout = null;  
 DataInputStream in = null;  
 BufferedReader br = null;  
 try {  
 fout = new FileInputStream("phoneDir.txt");  
 in = new DataInputStream(fout);  
 br = new BufferedReader(new InputStreamReader(in));  
 String strLine;  
 while ((strLine = br.readLine()) != null) {  
 String[] tokens = strLine.split(" ");  
 String phNo = null;  
 if(tokens.length == 2) {  
 phNo = tokens[1];  
 if(phoneNumber.equals(phNo)) {  
 System.*out*.println("Corresponding name for Phone No " + phoneNumber + " this is : " + tokens[0]);  
 found = true;  
 break;  
 }  
 }  
 else if(tokens.length == 3) {  
 phNo = tokens[2];  
 if(phoneNumber.equals(phNo)) {  
 System.*out*.println("Corresponding name for Phone No " + phoneNumber + " this is : " + tokens[0] + " " + tokens[1]);  
 found = true;  
 break;  
 }  
 }  
 }  
 br.close();  
 in.close();  
 fout.close();  
 if(!found) {  
 throw new ElementNotFoundException("Phone Number <" + phoneNumber + "> not found in record");  
 }  
 }catch (IOException | ElementNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
}

**OUTPUT:**

**(For simplicity’s sake searching within 3 records)**

****

****

****