CSE310 ASSIGNMENT



**Name** : ***Kashish Gilhotra***

**Reg.No**:***11803469***

**Section** : ***K18JF***

**Roll No**. : ***20***

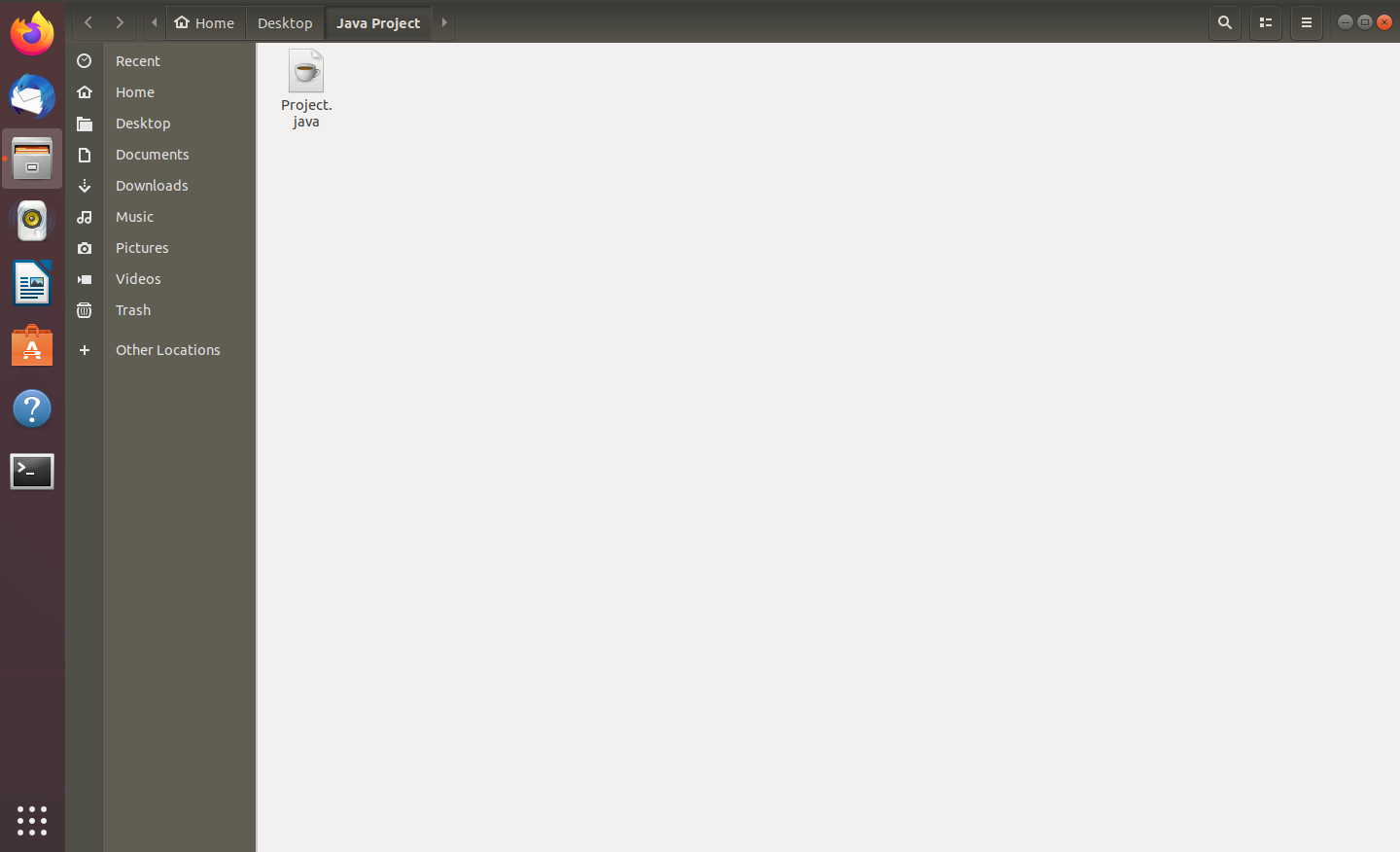
**Question :** Write a program to create following scenario:  
A **class Person** having :  
**private field : name** to store name of person  
a**bstract method : void display()**  which will be used to display details of person Use **parameterized constructor** to initialize these attributes. Make **setter** & **getter** methods for the attribute.  
Make a **class Employee** which **inherits Person** class and having : **private field: id** to store id of employee  
**date\_of\_join** to store joining date of employee  
Use a parameterized constructor to initialize these attributes. Make setter & getter methods  
for the attributes also.  
Now make a **class Test having main() method** in which you have to create at least 6 Employee objects and store them in ArrayList. Make another method named **serializeObjects()** in the same class which will read all the employee-objects from the ArrayList and serialize all those employee-records (in file “**records.txt**”) whose date of joining is before the year 2015.  
Now make another method named **deserializeObjects()** in the same class which will read and display all the records from the file.  
**Note:** Don’t make any separate instance variable other than the ones specified in above scenario. Also date\_of\_join variable is to be taken as LocalDate object rather than String object. Also, make use proper exception handling by using inbuilt exception-classes and also make your own custom exception class to deal with the exceptions such as users entering a negative no.

**Solution (CODE):**

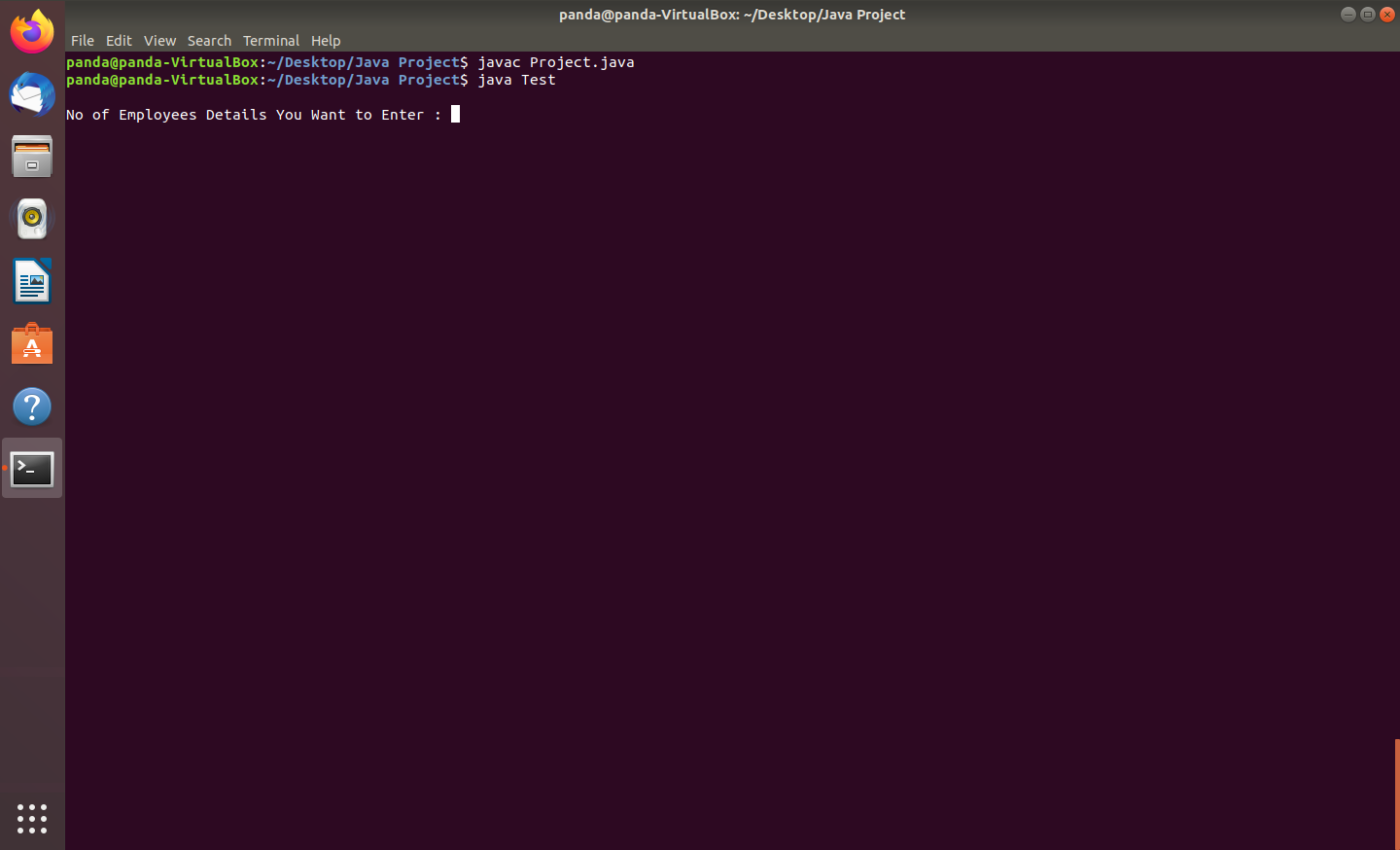
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | import java.util.\*; |  |
|  | import java.time.LocalDate; |  |
|  | import java.time.format.\*; |  |
|  | import java.io.File; |  |
|  | import java.io.FileInputStream; |  |
|  | import java.io.IOException; |  |
|  | import java.io.ObjectInputStream; |  |
|  | import java.io.FileOutputStream; |  |
|  | import java.io.ObjectOutputStream; |  |
|  | import java.io.Serializable; |  |
|  | //-------------------------------------------------------- |  |
|  | // Main Class Test |  |
|  | //------------------------------------------------------- |  |
|  | class Test { |  |
|  | private static String fileGiven = "records.txt"; |  |
|  | static ArrayList<Employee> data = new ArrayList<Employee>(); |  |
|  | static LocalDate comparer; |  |
|  | static int emps2015 = 0; |  |
|  | public static void main(String args[]) { |  |
|  | Scanner scan = new Scanner(System.in); |  |
|  | System.out.println(); |  |
|  | int noOfEmp = 0; |  |
|  | //------------------------------------------------------------- |  |
|  | // No of Employees Input |  |
|  | //------------------------------------------------------------ |  |
|  | Emps : |  |
|  | while(true) { |  |
|  | try { |  |
|  | System.out.print("No of Employees Details You Want to Enter : "); |  |
|  | noOfEmp = scan.nextInt(); |  |
|  | scan.nextLine(); |  |
|  | if(noOfEmp < 6) { // Employees Should Be at least 6 |  |
|  | throw new InvalidNoOfEmployees("No of Employees Should be at least 6"); |  |
|  | } else { |  |
|  | System.out.println("-------------------------------------------------"); |  |
|  | System.out.println(); |  |
|  | break Emps; |  |
|  | } |  |
|  | } catch(InputMismatchException s) { |  |
|  | System.out.println(s); |  |
|  | System.out.println(); |  |
|  | return; |  |
|  | } catch(Exception s) { |  |
|  | System.out.println("Exception Occured : " + s); |  |
|  | System.out.println(); |  |
|  | } |  |
|  | } |  |
|  | //----------------------------------------------------------------- |  |
|  | //Local Variables |  |
|  | //-------------------------------------------------------------------- |  |
|  | int id;String name, dte; boolean check = false; LocalDate date = LocalDate.now(); |  |
|  |  |  |
|  | //----------------------------------------------------------------------- |  |
|  | //Taking Inputs |  |
|  | //----------------------------------------------------------- |  |
|  | for(int i = 0 ; i < noOfEmp; i++) { |  |
|  | check = false; |  |
|  | System.out.print("Enter Name of " + (i + 1) + " Employee : "); |  |
|  | name = scan.nextLine(); |  |
|  |  |  |
|  | // ------------------------------------------------------- |  |
|  | id : |  |
|  | while(true) { |  |
|  | try { |  |
|  | System.out.print("Enter id of " + (i + 1) + " Employee : ") ; |  |
|  | id = scan.nextInt(); |  |
|  | scan.nextLine(); |  |
|  | if(id < 0) throw new InvalidId("Id can not be Negative"); |  |
|  | else { |  |
|  | break id; |  |
|  | } |  |
|  | } catch (InputMismatchException s) { |  |
|  | System.out.println("Exception Occur : " + s); |  |
|  | return; |  |
|  | } catch(Exception s) { |  |
|  | System.out.println(s); |  |
|  | System.out.println(); |  |
|  | } |  |
|  | } |  |
|  |  |  |
|  | // ------------------------------------------------------- |  |
|  | Date : |  |
|  | while(true) { |  |
|  | if(check) break Date; |  |
|  | try { |  |
|  | System.out.print("Enter date of Join of " + (i + 1) + " Employee (Date Format DD/MM/YYYY): "); |  |
|  | dte = scan.nextLine(); |  |
|  | DateTimeFormatter format = DateTimeFormatter.ofPattern("dd/MM/yyyy"); |  |
|  | date = LocalDate.parse(dte, format); |  |
|  | comparer = LocalDate.parse("01/01/2015", format); |  |
|  | check = true; |  |
|  | } catch(Exception s) { |  |
|  | System.out.println(s); |  |
|  | System.out.println(); |  |
|  | check = false; |  |
|  | } |  |
|  | } |  |
|  | // -------------------------------------------------------------- |  |
|  | Employee just = new Employee(name); // Initializing By Parameterized Constructor |  |
|  | // just.setName(name); // Initialize by Setter |  |
|  | just.setid(id); // Initialize by Setter |  |
|  | just.setDate\_of\_join(date); // Initialize by Setter |  |
|  | data.add(just); |  |
|  | System.out.println("-------------------------------------------------"); |  |
|  | System.out.println(); |  |
|  | } |  |
|  |  |  |
|  | // ------------------------------------------------------------- |  |
|  | //Creating a File |  |
|  | // ------------------------------------------------------------- |  |
|  |  |  |
|  | try { |  |
|  | File file = new File(fileGiven); |  |
|  | if(file.createNewFile()) System.out.println("File " + file.getName() + " is Created Successfully."); |  |
|  | System.out.println("-------------------------------------------------"); |  |
|  | System.out.println(); |  |
|  | } catch(IOException s) { |  |
|  | System.out.println("File Not Created. Some Error Occured"); |  |
|  | s.printStackTrace(); |  |
|  | } |  |
|  |  |  |
|  | //--------------------------------------------------------------- |  |
|  | //Calling Methods |  |
|  | // -------------------------------------------------------------- |  |
|  |  |  |
|  | serializeObjects(); |  |
|  | deserializeObjects(); |  |
|  | // -------------------------------------------------------------- |  |
|  | scan.close(); |  |
|  | } |  |
|  |  |  |
|  | //------------------------------------------------------------------------- |  |
|  | //Writing Objects in File |  |
|  | //------------------------------------------------------------------------- |  |
|  | public static void serializeObjects() { |  |
|  | try { |  |
|  | FileOutputStream fileOS = new FileOutputStream(fileGiven); |  |
|  | ObjectOutputStream objectOS = new ObjectOutputStream(fileOS); |  |
|  | for(Employee obj : data) { |  |
|  | LocalDate compare = obj.getdate(); |  |
|  | if(compare.isBefore(comparer)) { |  |
|  | objectOS.writeObject(obj); |  |
|  | emps2015++; |  |
|  | } |  |
|  | } |  |
|  | System.out.println("Data Written Successfully"); |  |
|  | System.out.println("-------------------------------------------------"); |  |
|  | System.out.println(); |  |
|  | objectOS.close(); |  |
|  | } catch(IOException s) { |  |
|  | s.printStackTrace(); |  |
|  | } |  |
|  | return; |  |
|  | } |  |
|  | //-------------------------------------------------------------------------- |  |
|  | //Reading Objects from File |  |
|  | //-------------------------------------------------------------------------- |  |
|  | public static void deserializeObjects() { |  |
|  | try { |  |
|  | FileInputStream fileIS = new FileInputStream(fileGiven); |  |
|  | ObjectInputStream objectIS = new ObjectInputStream(fileIS); |  |
|  | if(emps2015 == 0) { |  |
|  | System.out.println("There is No Employee Whose Joining Date is Before the Year 2015."); |  |
|  | System.out.println("-------------------------------------------------");System.out.println(); |  |
|  | } else if(emps2015 > 1){ |  |
|  | System.out.println("There are " + emps2015 + " Employees Whose Joining Date is Before the Year 2015 : "); |  |
|  | System.out.println(); |  |
|  | } else { |  |
|  | System.out.println("There is Only " + emps2015 + " Employee Whose Joining Date is Before the Year 2015 : "); |  |
|  | System.out.println(); |  |
|  | } |  |
|  | while(emps2015 != 0) { |  |
|  | Employee emp = (Employee)objectIS.readObject(); |  |
|  | emp.display(); |  |
|  | emps2015 -= 1; |  |
|  | } |  |
|  | objectIS.close(); |  |
|  | } catch(IOException s) { |  |
|  | s.printStackTrace(); |  |
|  | } catch (Exception s) { |  |
|  | s.printStackTrace(); |  |
|  | } |  |
|  | return; |  |
|  | } |  |
|  | } |  |
|  |  |  |
|  | //------------------------------------------------------------------------------ |  |
|  | //Person Class |  |
|  | //------------------------------------------------------------------------------- |  |
|  |  |  |
|  | abstract class Person implements Serializable{ |  |
|  |  |  |
|  | private String name; |  |
|  | abstract void display(); |  |
|  | //Constructor |  |
|  | Person(String name) { |  |
|  | this.name = name; |  |
|  | } |  |
|  | //Getter for Name |  |
|  | public String getName() { |  |
|  | return this.name; |  |
|  | } |  |
|  | //Setter for Name |  |
|  | public void setName(String name) { |  |
|  | this.name = name; |  |
|  | } |  |
|  | } |  |
|  |  |  |
|  | // ---------------------------------------------------------------------------- |  |
|  | // Employee Class |  |
|  | //----------------------------------------------------------------------------- |  |
|  |  |  |
|  | class Employee extends Person implements Serializable{ |  |
|  | private int id; |  |
|  | private LocalDate date\_of\_join; |  |
|  | // Constructor |  |
|  | Employee(String name) { |  |
|  | super(name); |  |
|  | } |  |
|  | //Getter For ID |  |
|  | public int getid() { |  |
|  | return this.id; |  |
|  | } |  |
|  | //Setter for ID |  |
|  | public void setid(int id) { |  |
|  | this.id = id; |  |
|  | } |  |
|  | //Getter For Date of Joining |  |
|  | public LocalDate getdate() { |  |
|  | return this.date\_of\_join; |  |
|  | } |  |
|  | //Setter for Date of Joining |  |
|  | public void setDate\_of\_join(LocalDate date\_of\_join) { |  |
|  | this.date\_of\_join = date\_of\_join; |  |
|  | } |  |
|  |  |  |
|  | public void display() { |  |
|  | System.out.println("Name of the Employee is : " + getName()); // Name by Getter |  |
|  | System.out.println("Id of the Employee is : " + getid()); //Id by Getter |  |
|  | System.out.println("Date of Joining Was : " + getdate()); // Date of Join by Getter |  |
|  | System.out.println("-------------------------------------------------"); |  |
|  | System.out.println(); |  |
|  | } |  |
|  | } |  |
|  |  |  |
|  | //-------------------------------------------------------------------------- |  |
|  | // Exception Related No of Employees |  |
|  | //-------------------------------------------------------------------------- |  |
|  |  |  |
|  | class InvalidNoOfEmployees extends Exception { |  |
|  | public InvalidNoOfEmployees(String s) { |  |
|  | super(s); |  |
|  | } |  |
|  | } |  |
|  |  |  |
|  | //----------------------------------------------- |  |
|  | // Exception Related Id (if id is Negative then) |  |
|  | //----------------------------------------------- |  |
|  |  |  |
|  | class InvalidId extends Exception { |  |
|  | public InvalidId(String s) { |  |
|  | super(s); |  |
|  | } |  |
|  | } |  |
|  |  |  |

**SCREENSHOTS OF EXCEPTIONS AND OUTPUTS :**

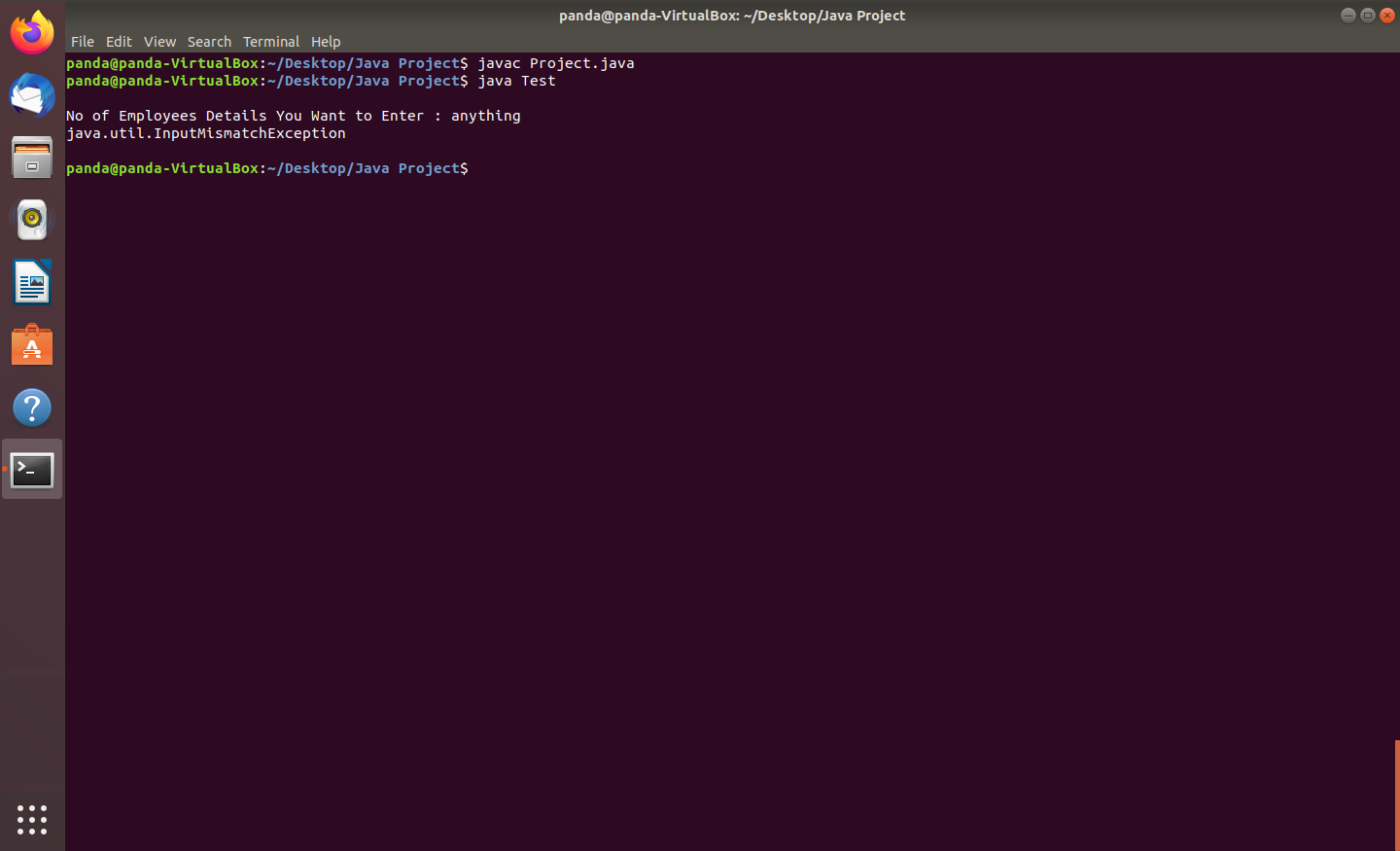
1. There is Only One File Named Project.Java (Code File)



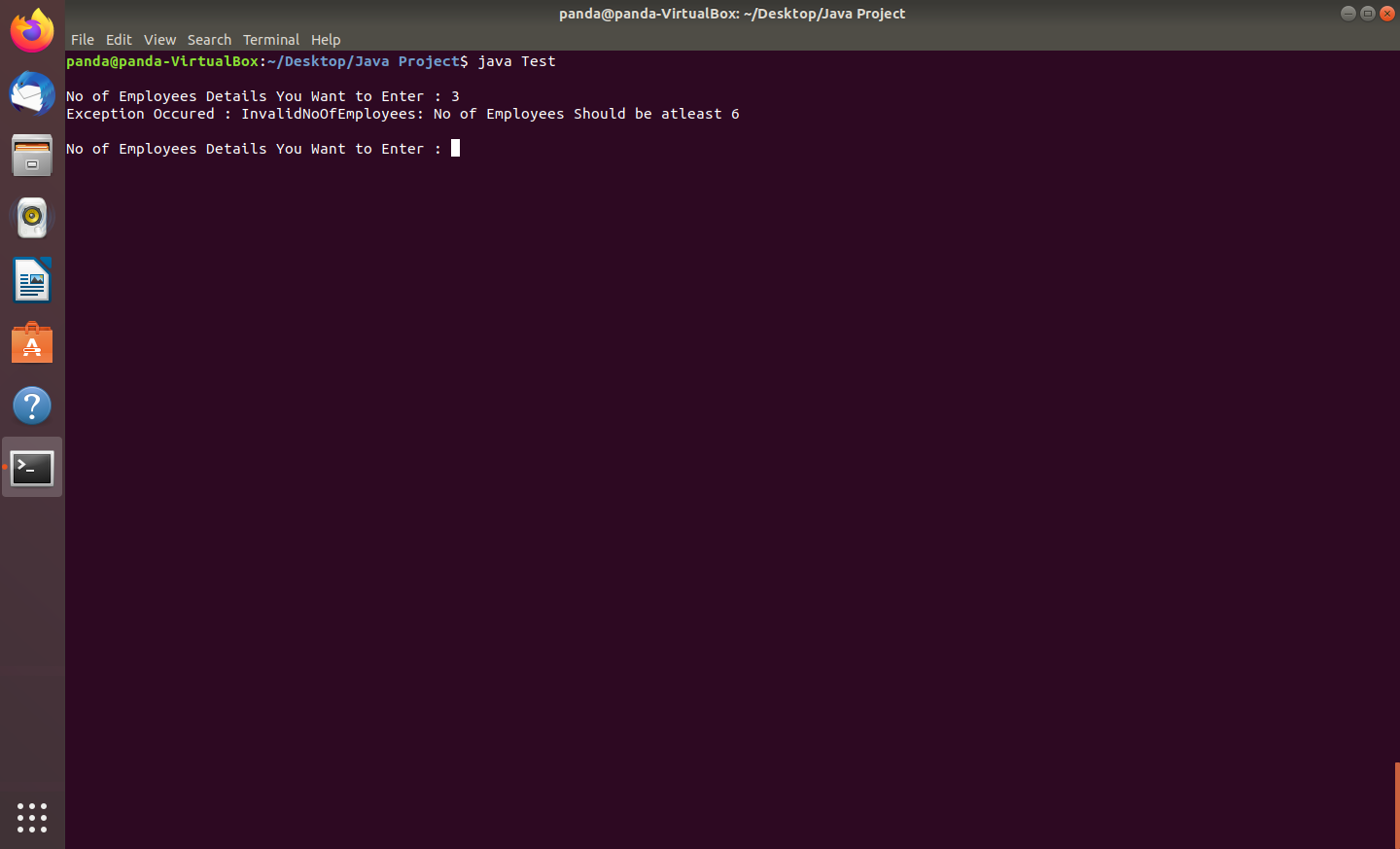
1. Compiling / Running Java File Without Complication Errors / Runtime Errors



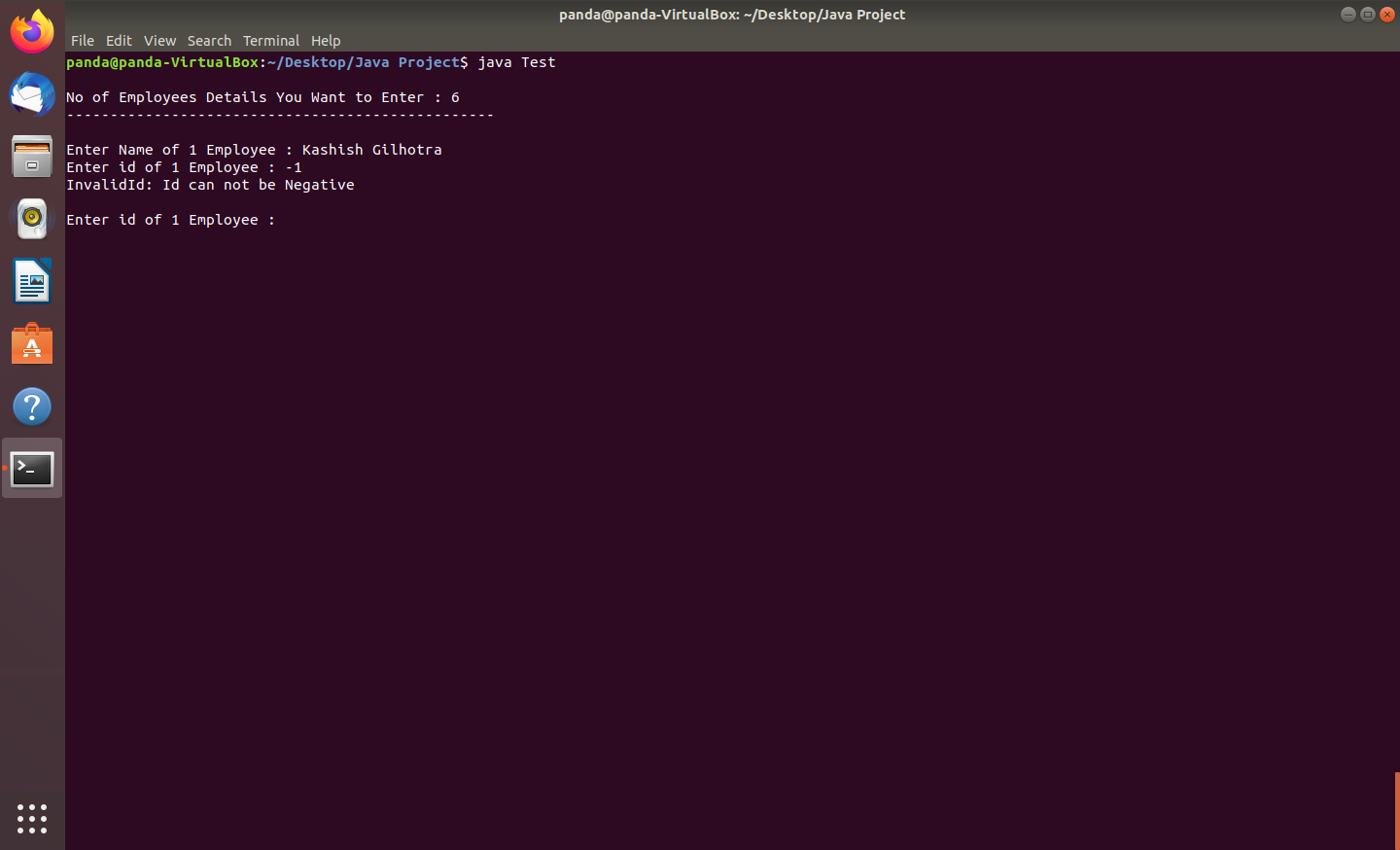
1. InputMismatch Exception (Occurs When Character is Entered in Place of INT)



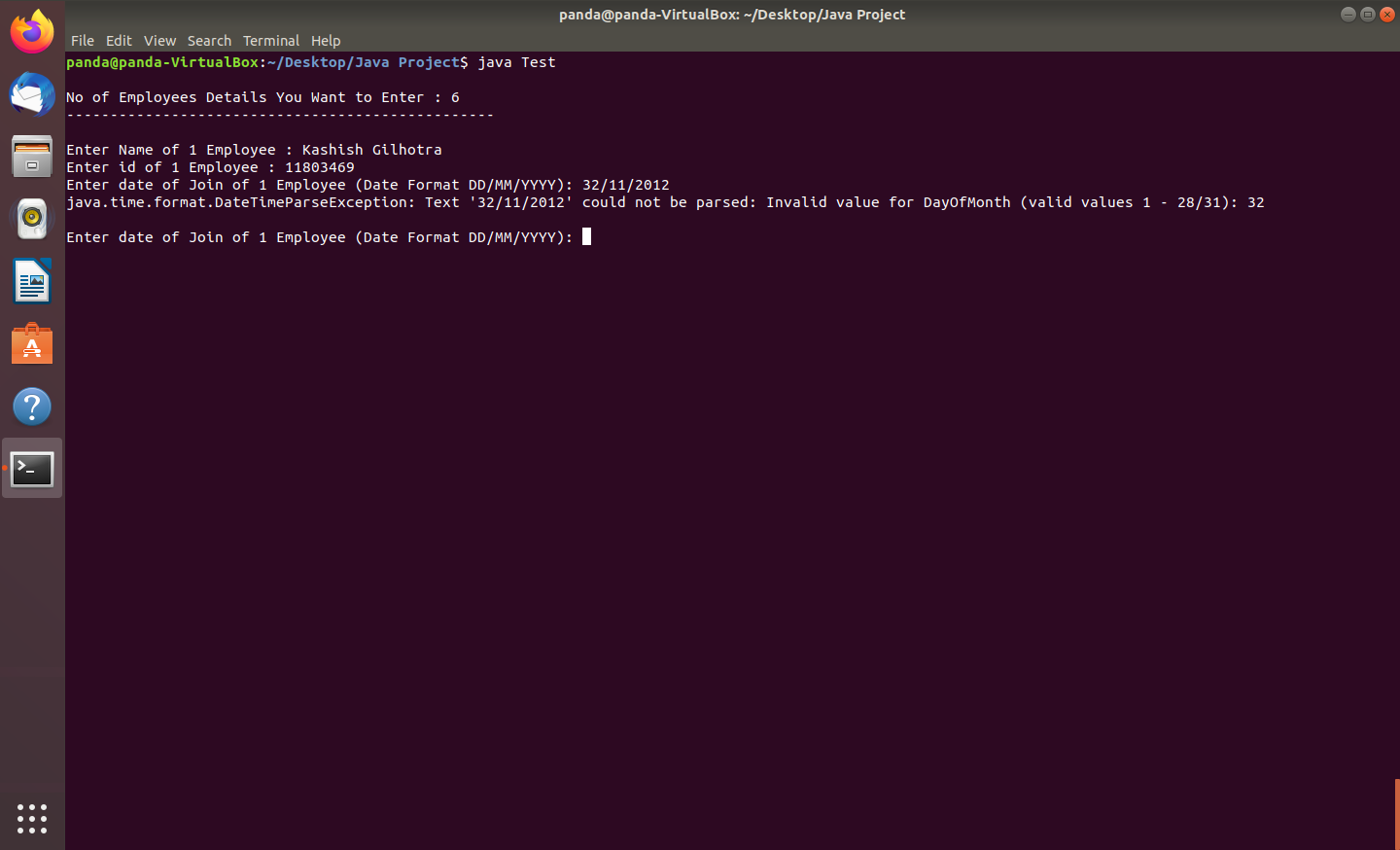
1. InvalidNoOfEmployees Exception (Occurs When No of Employees Entered are Less than 6)

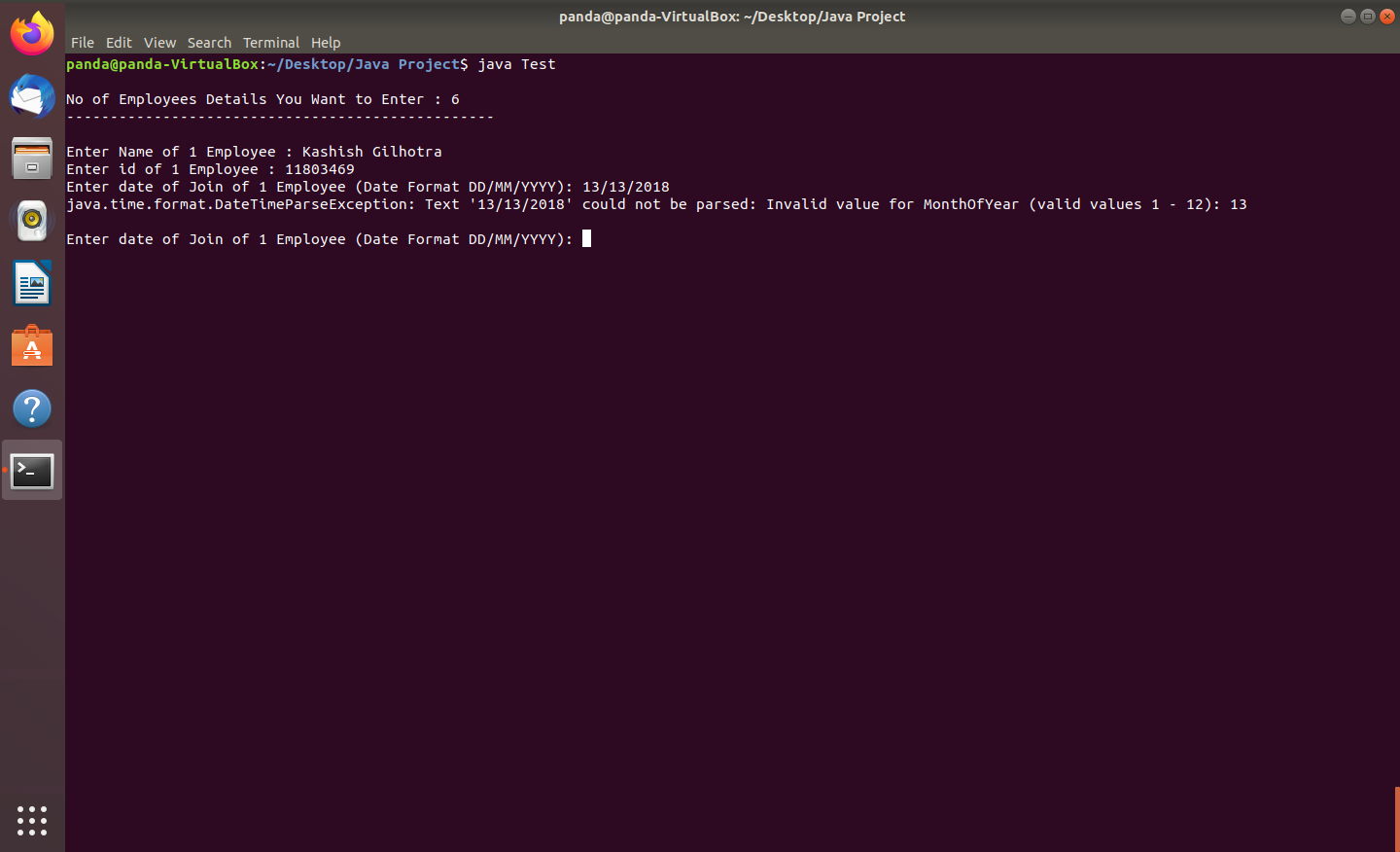


1. InvalidId Exception (Occurs When Employee Id Entered is Negative)

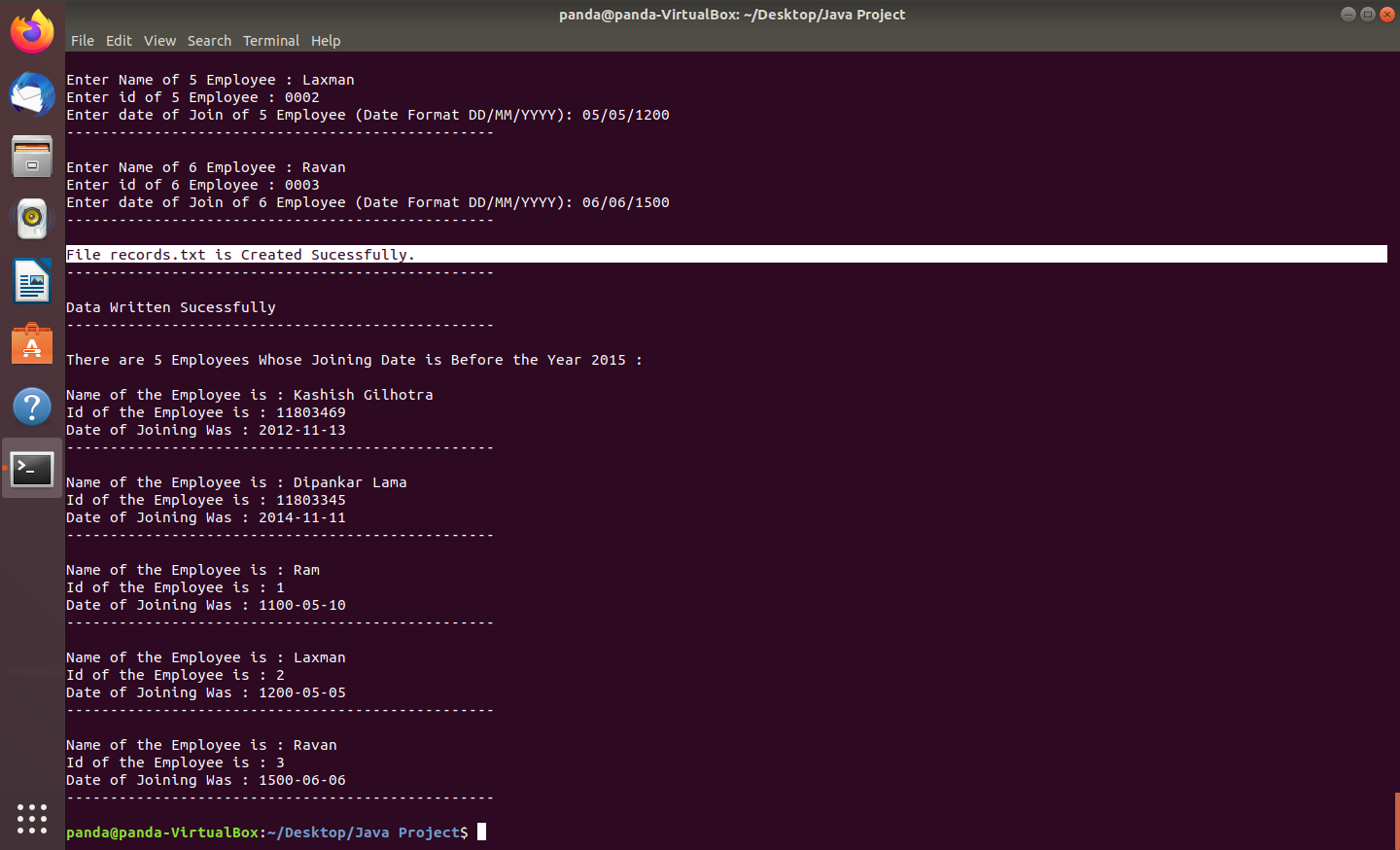


1. Java.time.format.DateTimeParse Exception (Occurs When Date / Month is Wrongly Entered)

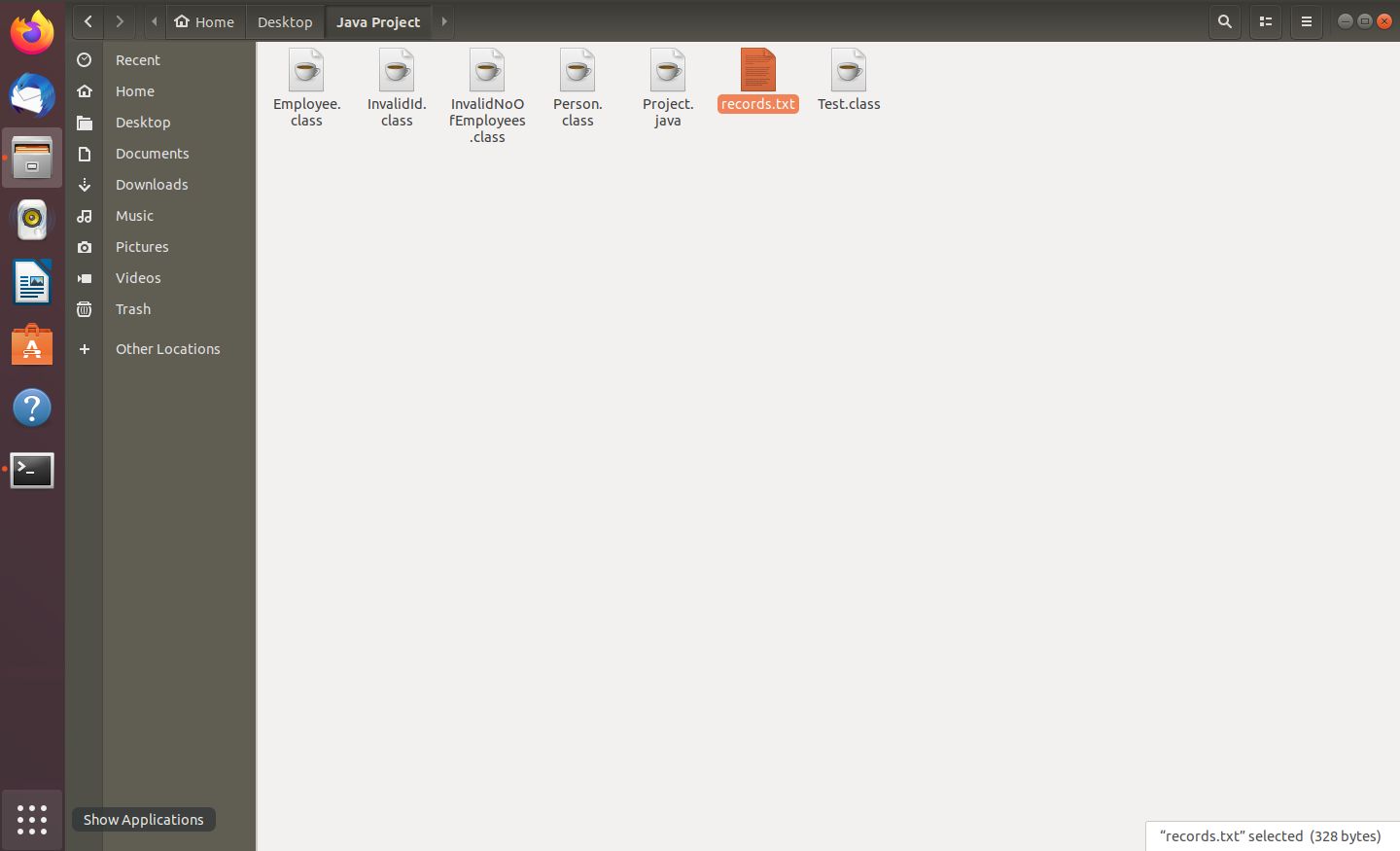
**=>** {As their No Date Like 32 in Calendar}

**=>** {As their No Month Like 13 in Whole Calendar}

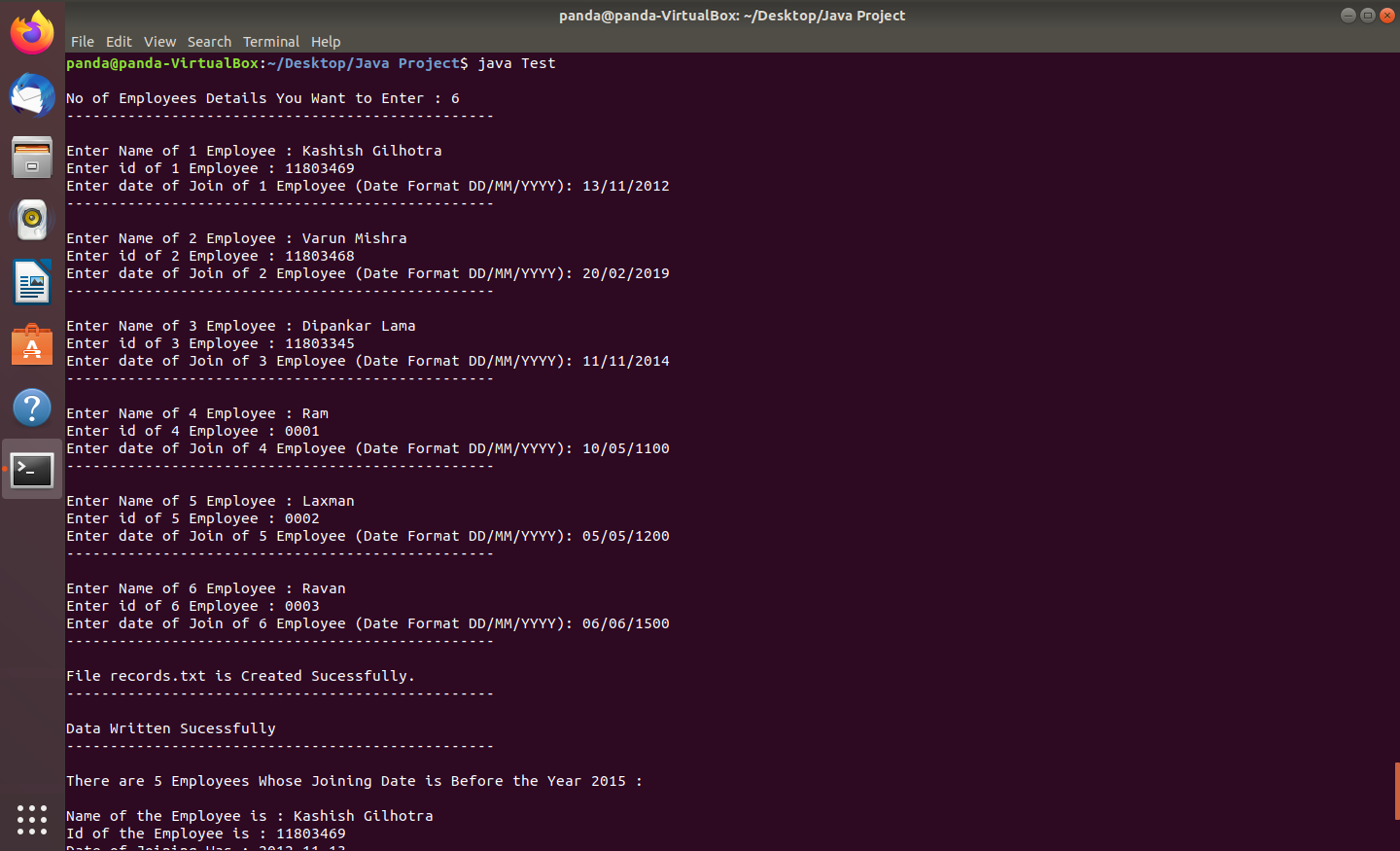
1. Creation of File

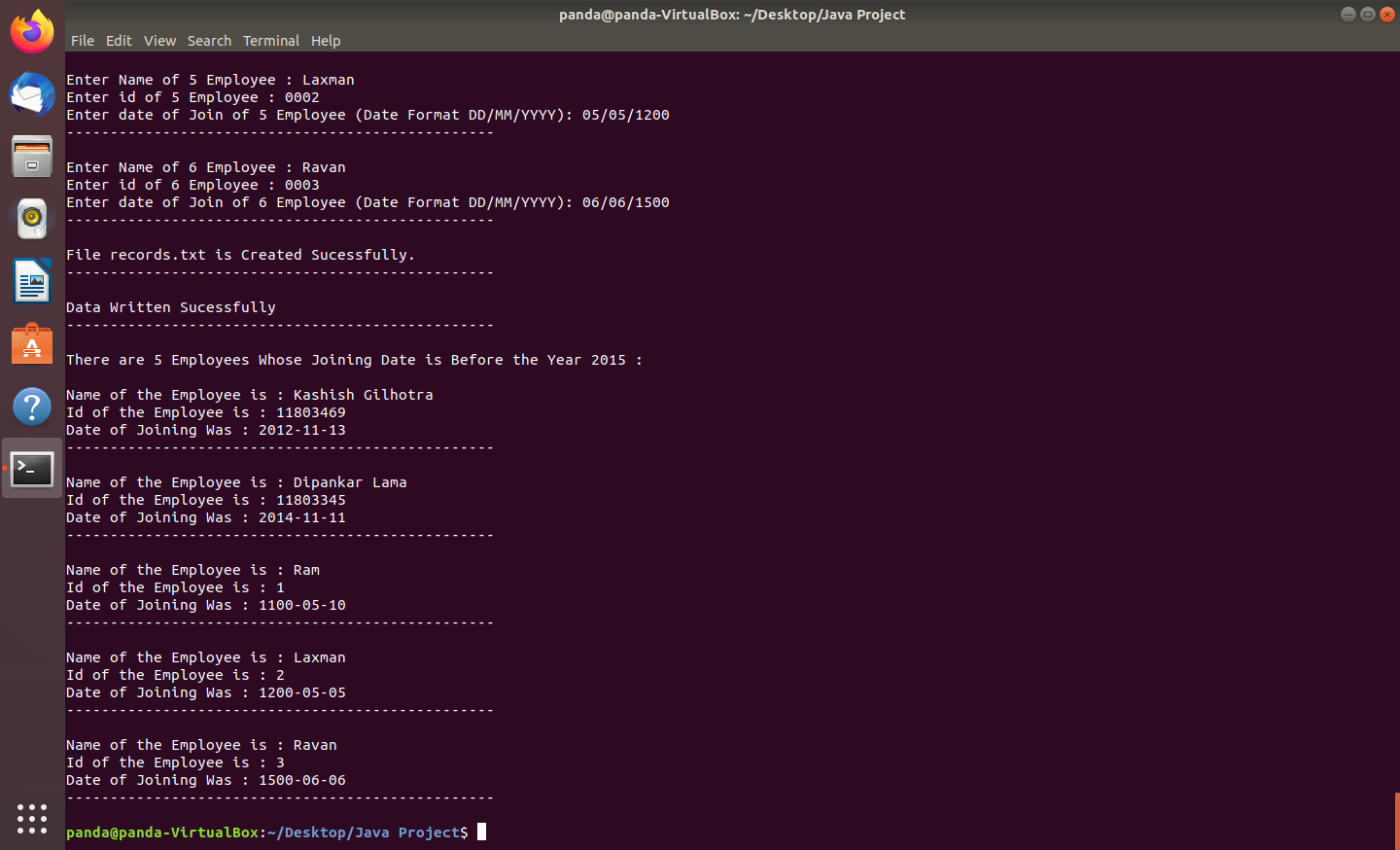


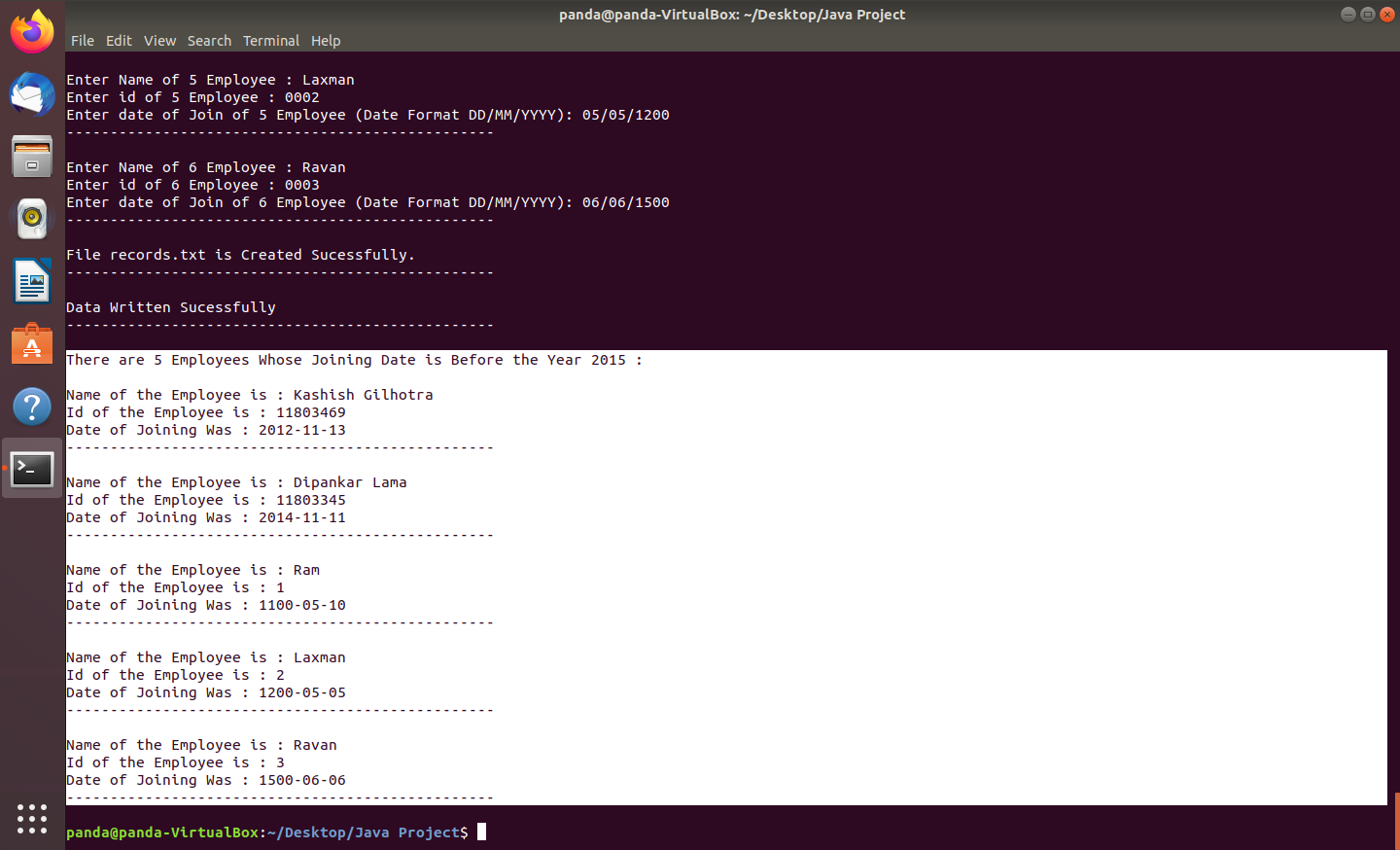
**=>** { File Created }



**FINAL OUTPUTS :**





**#** Reading Data From File only Of those Employees Whom Joining Date is Before 2015

**GitHub - Link :** [**https://github.com/Kashish001/Java-Assignment**](https://github.com/Kashish001/Java-Assignment)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*