



EmployeeTrackingSystem

Core JAVA PROJECT

ABSTRACT

The Employee tracking application is prepared as a console-based core java project. The system tracks the performance of all registered employees in an organisation, involved in different projects. The detailed address of the employees is stored in a different entity and may be referred to when tracking of employees is necessary or their proximity to each other and involvement in different projects may need to be conveniently assigned.

The Manager entity of the system benefits the managerial staff to track the employees efficiently to know their allotment, department, progress, and scheduling data.

This system requires no web server and may be executed from a digital device within the onsite premises of an organisation which can effectively use it to understand the performance of its employees.

The system helps in

- Division of labour
- Understanding effectiveness of each employee
- Understanding employee potential
- Project constraints
- Effective scheduling
- Effective running of organisational framework
- Less cost involvement
- Tracking employee details

1. INTRODUCTION

1.1 Project Objectives

Admin Objectives

It is an admin dependent system which has no user functionality. The salient features of the Employee Tracking System are:

- It is easy to execute the java classes which is possible just with the presence of jdk .
- It is an instant system of entry and checking entries of employee details, which involve less complication and hassle.
- It does not involve a web server at present as it does not give a facility of browser execution or deployment as such.
- It does not involve a database. It therefore hardly requires storage space.
- However it facilitates updation as it is prepared in an IDE and can be easily converted into a web based project using hibernate + MySQL or any other database in future.

1.2. Constraints

A few functionalities could not be implemented.

- The system suffers from the drawback of not being able to function at times due to session creation error for the application.
- The employer or the manager constructor which is created as a inherited class from the employee class, has no existence if employees do not exist. So, Manager class bears a IS-A relationship with the Employee class. The Manager class is created as the child class of Employee. Employee is the parent class of the manager class.
- Address bears a HAS-A relation with Employee. It is an Association of Composition type, with the Employee class. Mysql was not used as JDBC was not to be used in this project.

○ 1.3. Risks

Document specific risks that have been identified or that should be considered.

- The authenticity of users is not determined.
- The Console based project is not equipped to provide detailed view of employees. The database is not used which makes calling of employees view on the basis of employee id not possible.
- The project may suffer from updation and deletion anomalies for which the development need to be converted to different classes for insertion or deletion.

1.4. Out of Scope

1. The project is developed in the local system and the code in github repository.
2. The project is not hosted in any cloud platform and is still not available in on internet.
3. Setting of parameters in database for further use is not possible.
4. Employee authentication or the authentication of user of the application is out of scope.

2. TECHNICAL SPECIFICATIONS

- It includes the hardware software and other technical requirements of the system. Any platform and machine with an installed jdk can effectively run the application. As the project is developed in java it is:
- Portable
- Simple to understand and implement
- Any database can be used for stating minimum employee details
- Minimum storage and any RAM that effectively runs a java application can be used.
- The application uses the System Library which may vary from machine to machine.
- The application is developed in Eclipse IDE. It may be opened in any IDE or it may even run in command prompt.

2.1. Hardware Requirements

- Hard disk : 256 MB required
- Main memory: 512 MB
- Processor: Intel Pentium microprocessor with RYZEN
- Keyboard: Standard
- Monitor: 600x800 Resolution or above
- Mouse: Scroll

2.2. Software Requirements

1.Tools and platforms used

- Operating System: Windows 11
- Platform: ECLIPSE IDE [2022-09]
- Language: CORE JAVA

1.Software interfaces

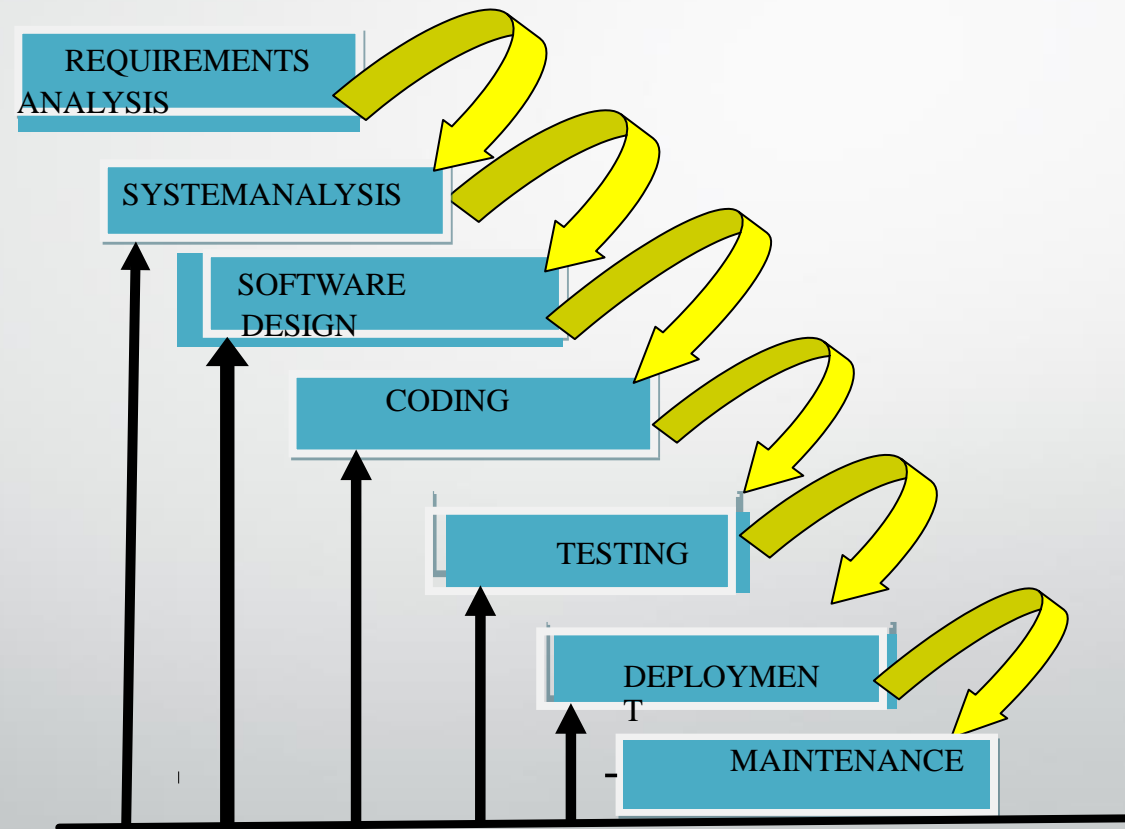
- Application: Eclipse IDE [2022-09]
- Jdk : 8
- Additional API: MS OFFICE

No **DATABASE** used and no schema needed to be developed. Technological Specification was not a matter of concern for the project.

- 3. SOFTWARE DEVELOPMENT METHODOLOGY

- 3.1. SOFTWARE LIFECYCLE MODEL

1. The software to be developed depends on the series of identifiable stages that would eventually lead to the product. The diagrammatic representation would follow in building the logical framework. It would be based on the requirements analysis and the design phase. The proposed software is planned after the requirements analysis and therefore may be developed by means of the iterative waterfall model. In order to give it a stage to analyse the effects of the prior stage, the iterative approach has been followed.



3.2. REQUIREMENTS ANALYSIS

3.2.1. PROBLEM SPECIFICATION

The system to be developed is based on core java as a console-based project. The system is developed to track which employees are involved in which project and their details.


- ✚ Employees and their details are inserted in the system.
- ✚ Employee details, as Address may be updated in the system.
- ✚ Employees may be removed from a project.
- ✚ A Manager view is created in the manager class.
- ✚ No database and no connector are involved. Employees are stored in employee array.
- ✚ Linked hash map is used for a quick view of empid and employee_ name.
- ✚ The code is portable and easily maintained. It may well be enhanced and modified.

3.2.2. FEASIBILITY STUDY: The development of any Software depends on the fact that whether it is feasible for development or not. This study is done for the various factors which might affect the software development, deployment and maintenance. It is also targeted for Customer Relationship Management in future.

3.2.2.1. TECHNICAL FEASIBILITY:

The software is a simple console-based application. It is purely coded in java.

The Software is to be managed and maintained after deployment on an iterative basis,



3.2.2.2. SOCIAL FEASIBILITY: This software is socially feasible as a fast checking of projects and employees involved in the projects. Visibility to employee details especially with their address and contact details becomes a necessity.

Creation of a manager view is of importance to checking employee records. The present system is a handy method of instant updating and checking.

3.2.2.3. ECONOMIC FEASIBILITY: The system does not need any installations. It does not involve a database or a web server. As it is coded in java it can be run on cmd if an IDE is not available. It implements the java features and the code being portable is a handy method.

3.2.2.4. LEGAL FEASIBILITY:

The Legal feasibility of the system development is based on whether the employee details checked are authentic or not. As employees are generally distributed and assigned projects according to their domain and specific study fields, they might need to share details with one another. This problem makes it necessary to understand the Residential Address authenticity of the employees. This problem may be solved in future by employing database storage facilities with accepting files for address proof.

The feasibility study conducted on the system has helped its development and further maintenance

3.3. SOFTWARE REQUIREMENTS SPECIFICATION

3.3.1. This part of the document provides a comprehensive description of the Software to be developed by the system. The different subsections provide the information of the Software and hardware to be used by the system.

4.3.1. PURPOSE: The SRS aims at the development of the system requirements. The employee tracking system is a must have for all concerned organizations.

The vision of the system is a unified platform of Employee details to track their whereabouts and their involvement in projects.

The mission is to strive towards the goal with a reduction of insertion and deletion anomalies to have a better track of every employed person involved in a company.

1.SCOPE: The system may be used by an individual data entry operator of an organisation with very few instructions to follow.

- The system runs offline and requires no web server installation.
- It gives instant entry and check facilities with less typing and entry hassles.
- The portable system created in pure core java, runs at present on the simple build level without any database and has a lot of maintenance scope and can be easily connected in time with any database just by integrating with Hibernate or by usage of JDBC.
- . ABBREVIATIONS:

Table : 5.1.

Specification: IEEE STD 830-1993

SRS	Software Requirements Specification
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
ID	Identification Definition
jdk	Java development Kit
jre	Java runtime environment
IDE	Integrated Development Environment

3.3.2. FUNCTIONAL REQUIREMENTS:

This gives specific details about how the system is supposed to behave after execution in the virtual machine. It also gives what inputs are provided to which process and what is the expected output of each. It also denotes how the system might behave and what are the specific data manipulations and calculations.

3.3.2.1. Employee insertion [address class gets executed first]:

Employee address is given as an input

Input: Address address (), zip_code, street, city, state, country

Output: address input success

3.3.2.2. Employee class insertion:

Input: empid, employee _ name, employee _ email, employee _ phone, project _ name

Output: employee class inserted successfully

3.3.2.3. Manager class insertion:

Input: department _ name, no _ individuals _ reporting, timesheet _ begin , timesheet _ end,

Employee object

Output: status, save()

3.3.2.4. Employee selection:

Input: empid

Output: employee _ name, employee _ email, employee _ contact, project _ name, address, department _ name,

3.3.2.5. Employee updation(address):

Input: empid, Address new _address

Output: save(), status

3.3.2.6. Employee deletion :

Input empid

Output: status , save()



1.NON-FUNCTIONAL REQUIREMENTS: These are directly related to the functioning of the system. The main constraints of the system are

1.Hardware interface:

- Screen Resolution of 600x800.
- Mouse for scroll

Maintainability: This is achieved by updating the system on the basis of present requirements and implementation of Client demand techniques, as gathered from feedback of users.

Portability: This feature is achieved by using JAVA as a programming language. The OOPS feature helps the system to have a portable feature. It is therefore made to run on any

- Operating System on any machine.
- PC or Laptop

1.Software Interfaces:

☪ Eclipse IDE for developing java Code

☪ Windows 11 OS

jre used: 1.8

1.Communications Interfaces: None

4.1. 3..6. Performance requirements: java classes

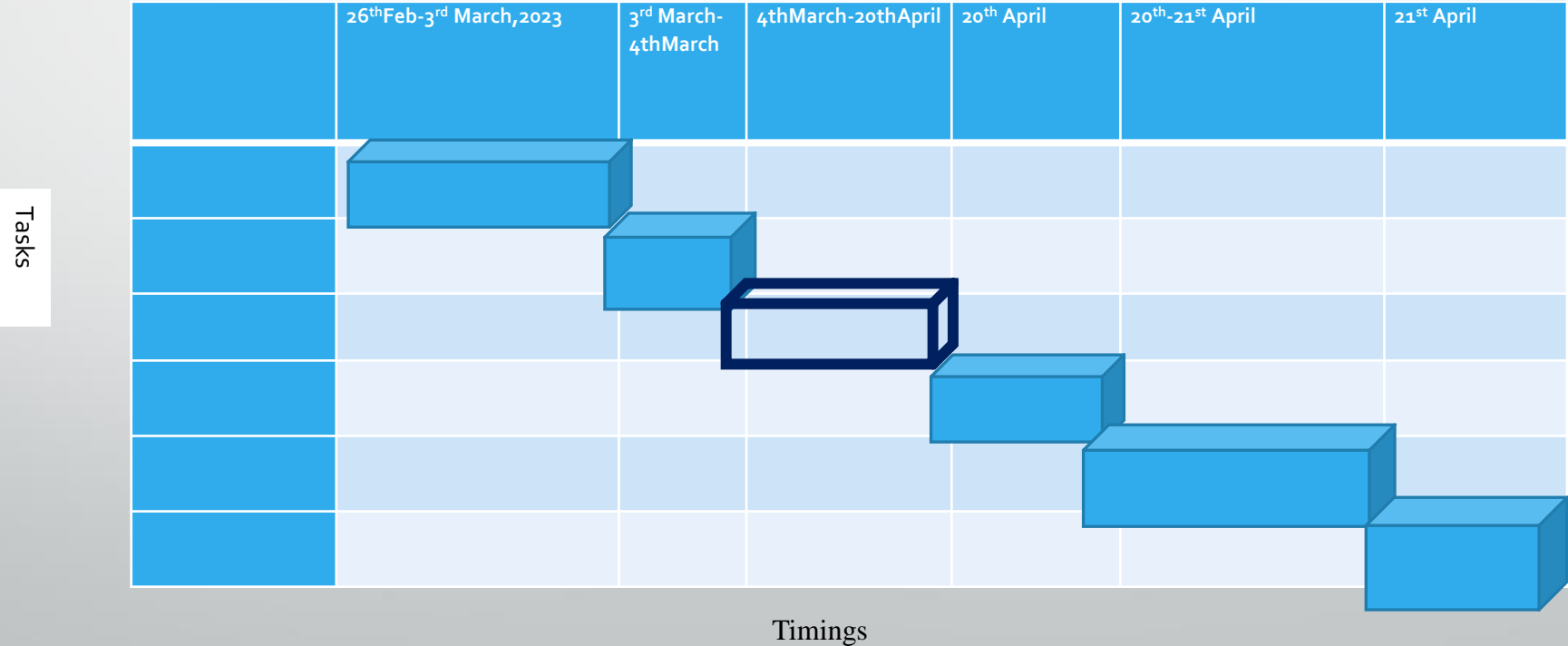
1.Software System attributes:

- **PROJECT PLANNING AND SCHEDULING**

NAME OF PHASES	SUB PHASES OR DESCRIPTION	
Requirements analysis	Problem definition	
	Feasibility study	
	Software requirementsanalysis	
Milestone : Successful SRS and Feasible System. Proceeding to Designing the System		
System analysis	Project planning and Scheduling	
	System DFD	
	System Designing	
	Structure designing	
Milestone :Completion of Design. Proceed to code the System		
Coding	Coding with Comments	
	Code Efficiency	
	Error handling	
Milestone : Error free Code		
	System Testing	
	Debugging	
Milestone : Successful Testing		
Implementation andMaintenance	improvement	
Milestone : Successful Implementation		

1.GANTT CHART: A horizontal bar chart which visually represents a project plan over time. The chart shows status of each task in the project.

TASK	START DATE	DAYS TO COMPLETE
Requirements analysis	26 th February, 2023	5 days
System Analysis	3 rd march	1 day
Procrastination + trial with Hibernate Integration		
System Design	20 th April	1 day
Coding	20 th April-21 st April	2 days
Testing	21 st April	1 day
Build	21 st April, 2023	-

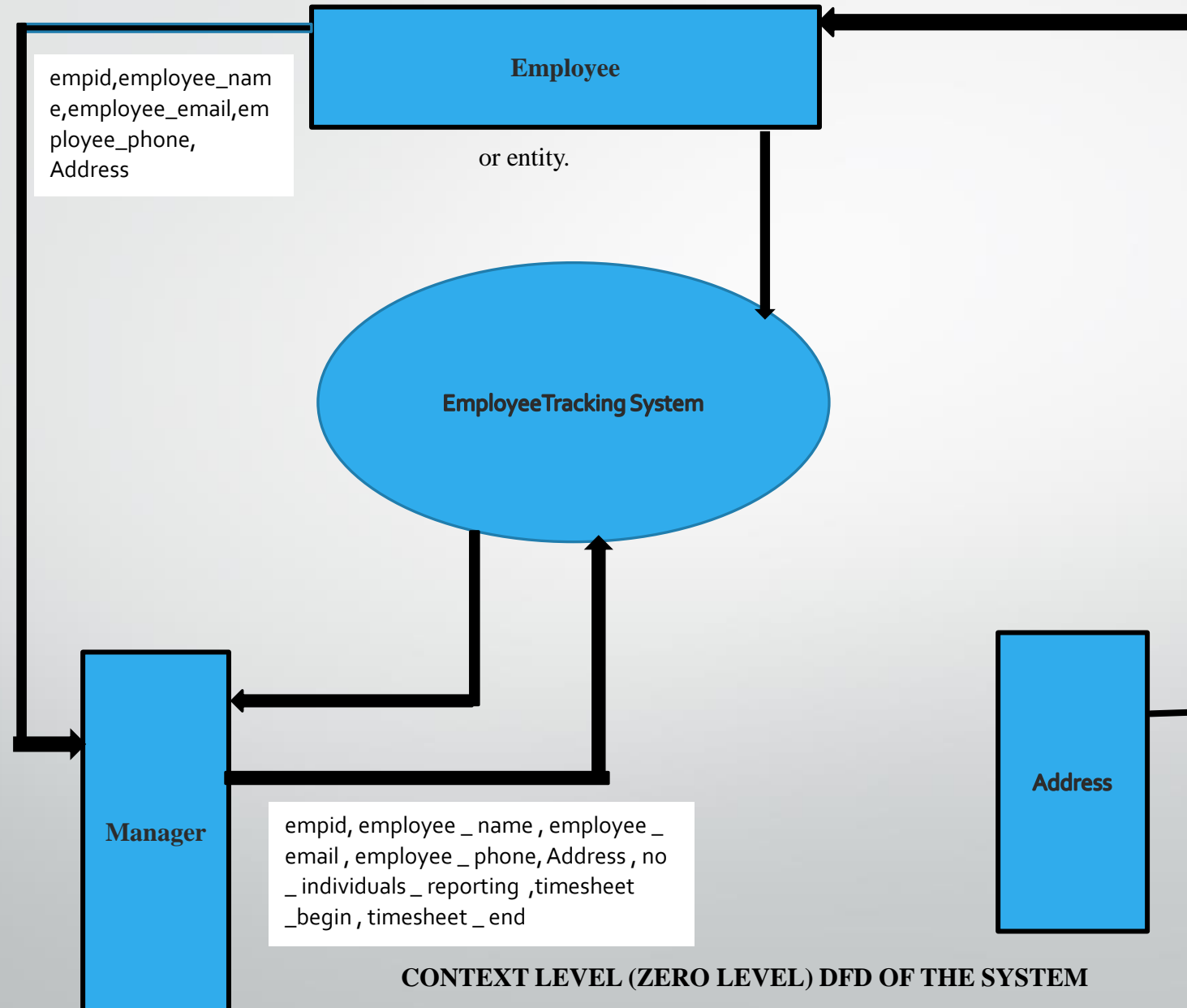


4. SYSTEM ANALYSIS

zip_code,street,city,state,country

4.1. DATA FLOW DIAGRAMS

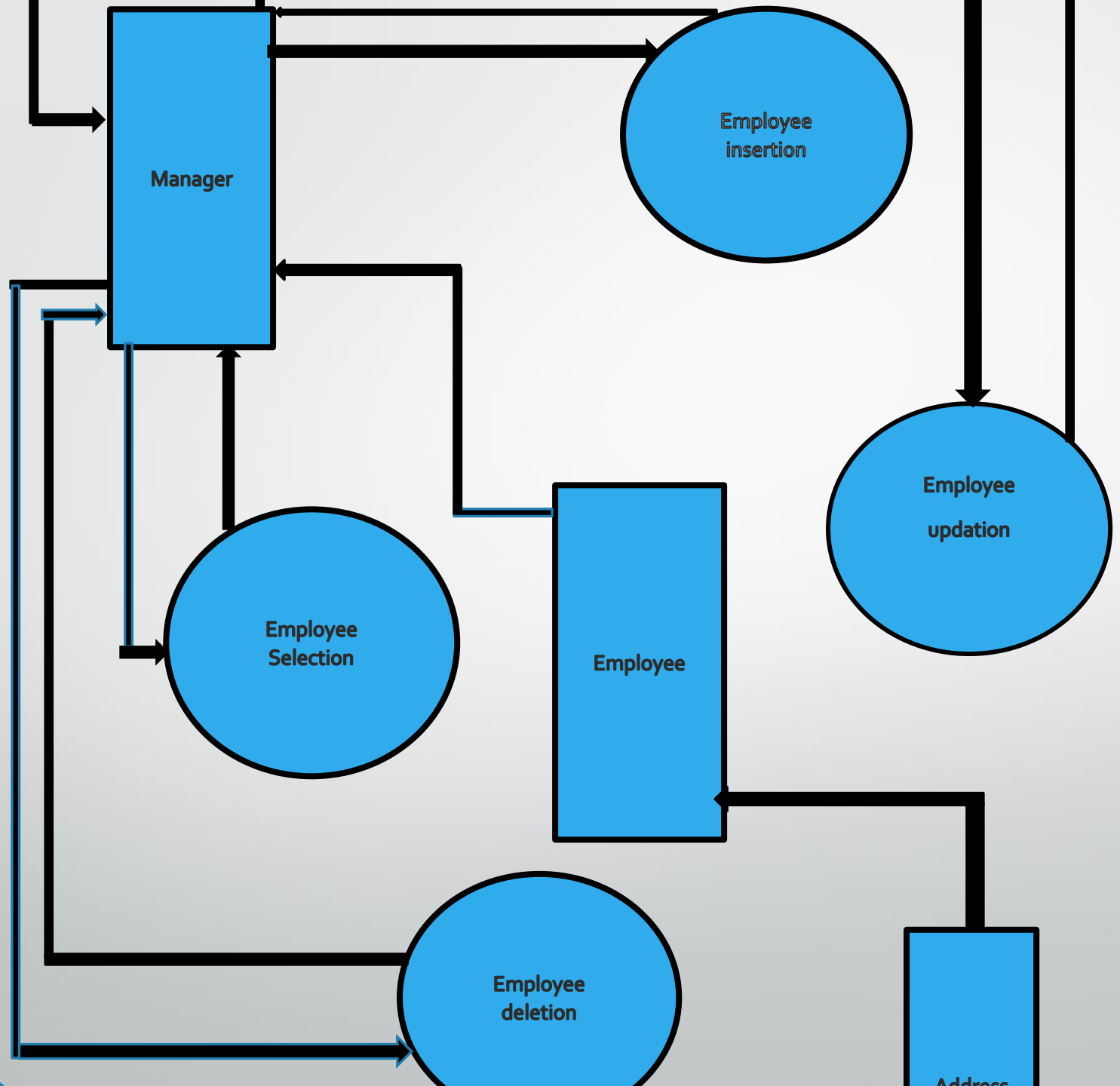
(DFD): Drawn to show the flow of data between processes and entities of a System. It has no control flow. It is a mapping to demonstrate the flow of input and output from a process

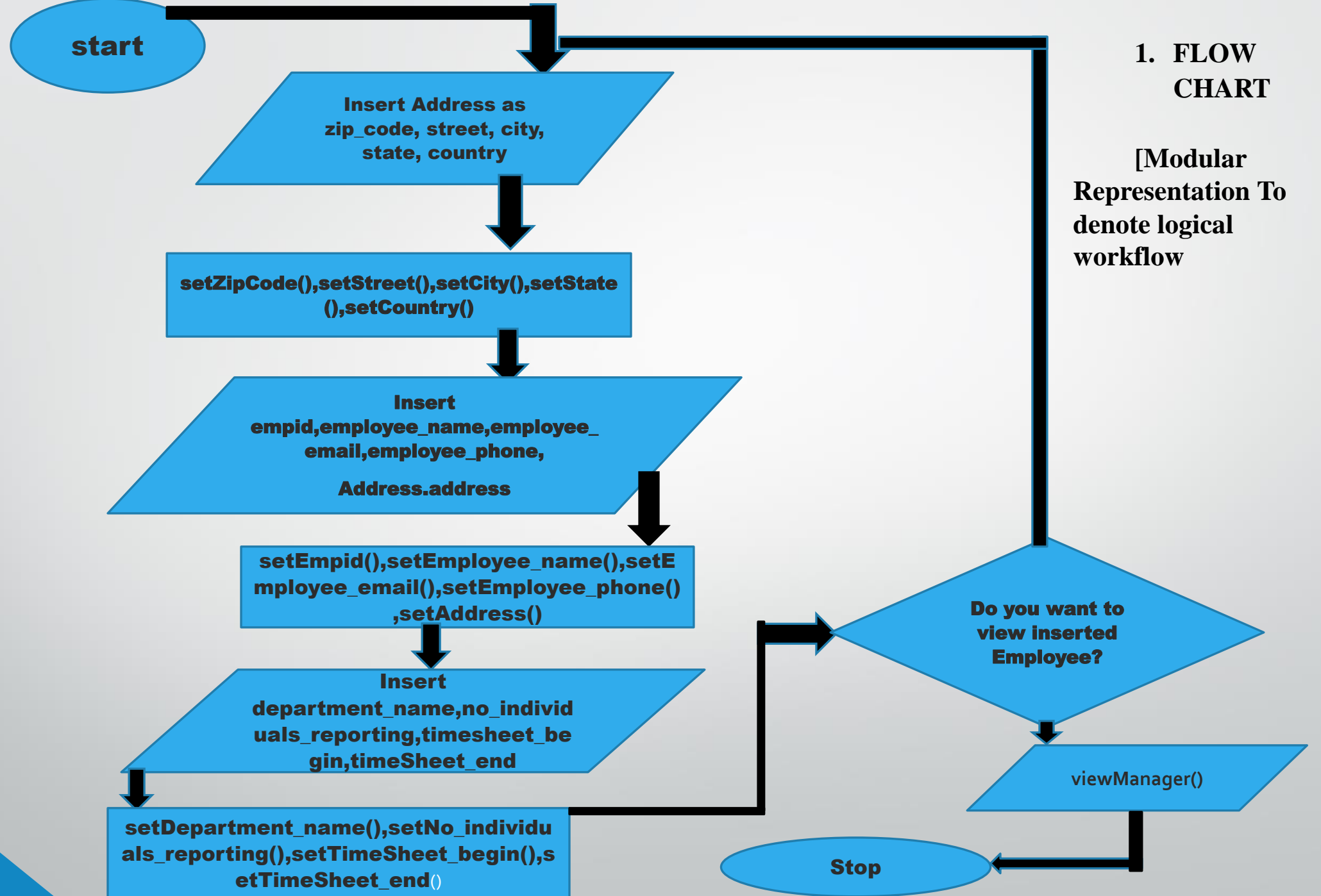


CONTEXT LEVEL (ZERO LEVEL) DFD OF THE SYSTEM

6.1.2.

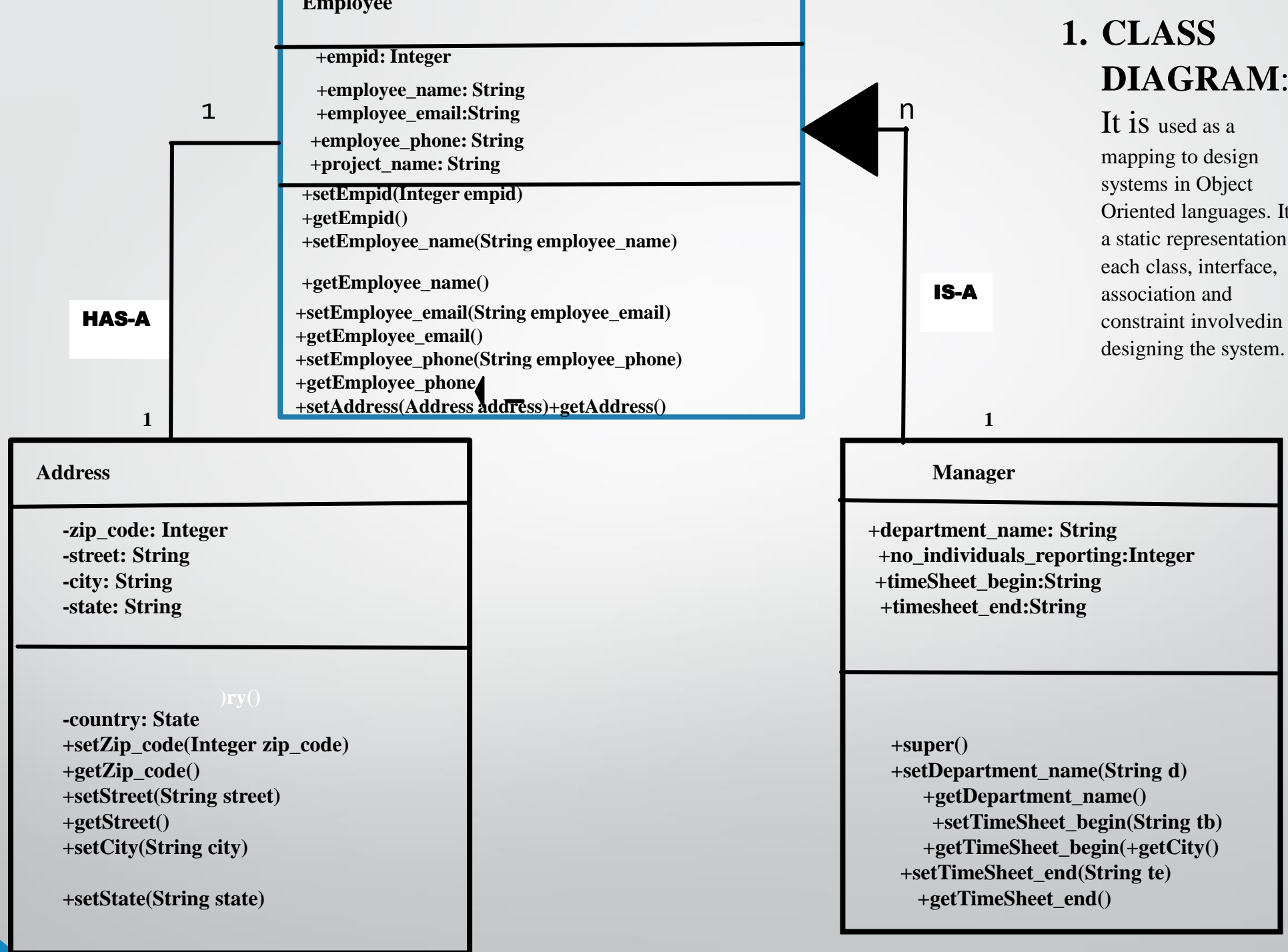
Level-1 DFD



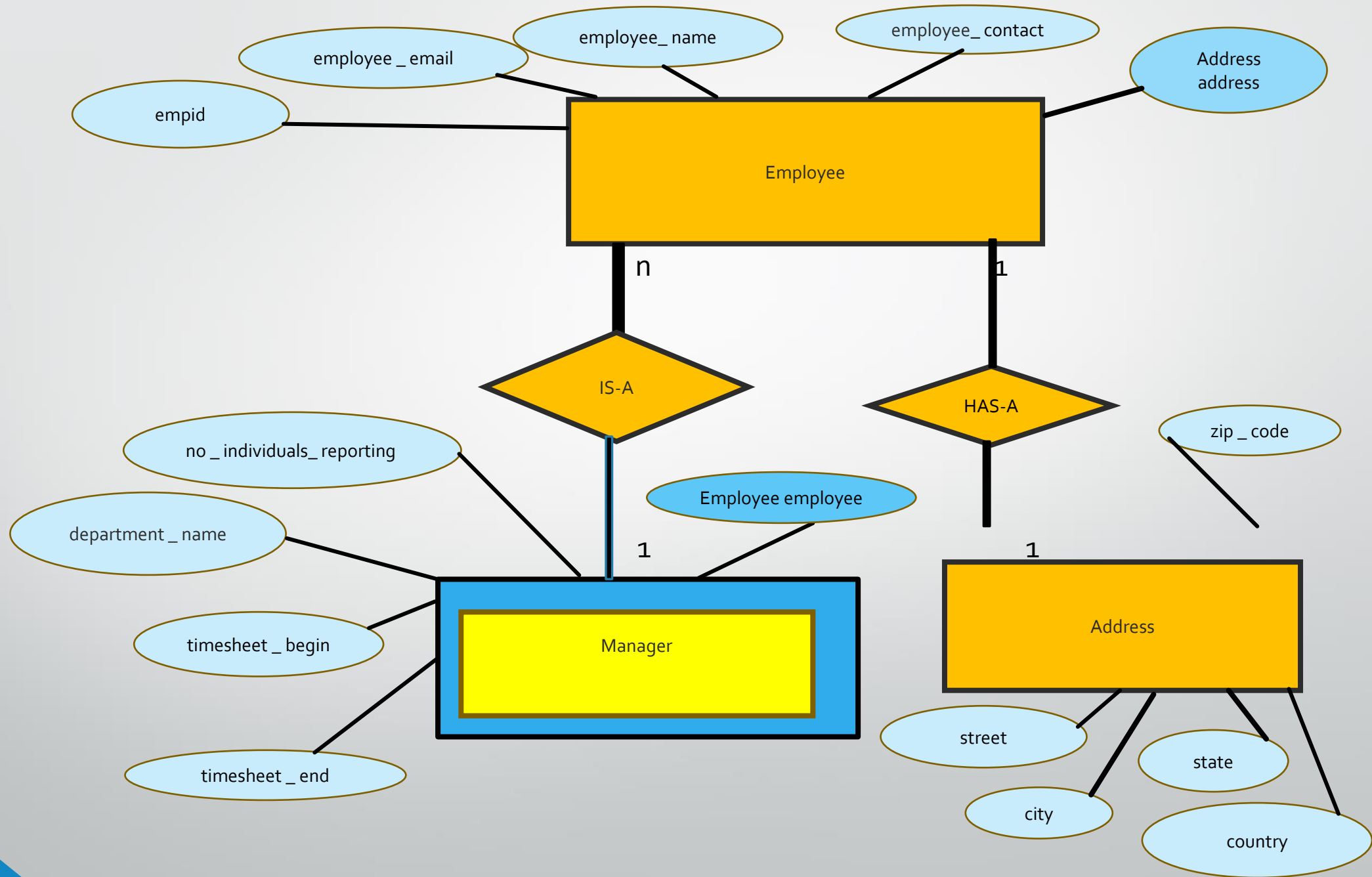


1. CLASS DIAGRAM:

It is used as a mapping to design systems in Object Oriented languages. It is a static representation of each class, interface, association and constraint involved in designing the system.



ENTITY RELATIOSHIP DIACRAM



- **SYSTEM TESTING**

Software testing is a crucial element and it represents the ultimate review of specification design & coding. There are two types of test approaches. They are-

- Black Box Testig
- White box testing

When computer software is considered, *black-box testing* alludes to tests that are conducted at the software interface. Black-box tests are used to demonstrate that software functions are operational, that input is properly accepted and output is correctly produced, and that the integrity of external information is maintained.

Other Testing methods are:

- **Unit testing:**

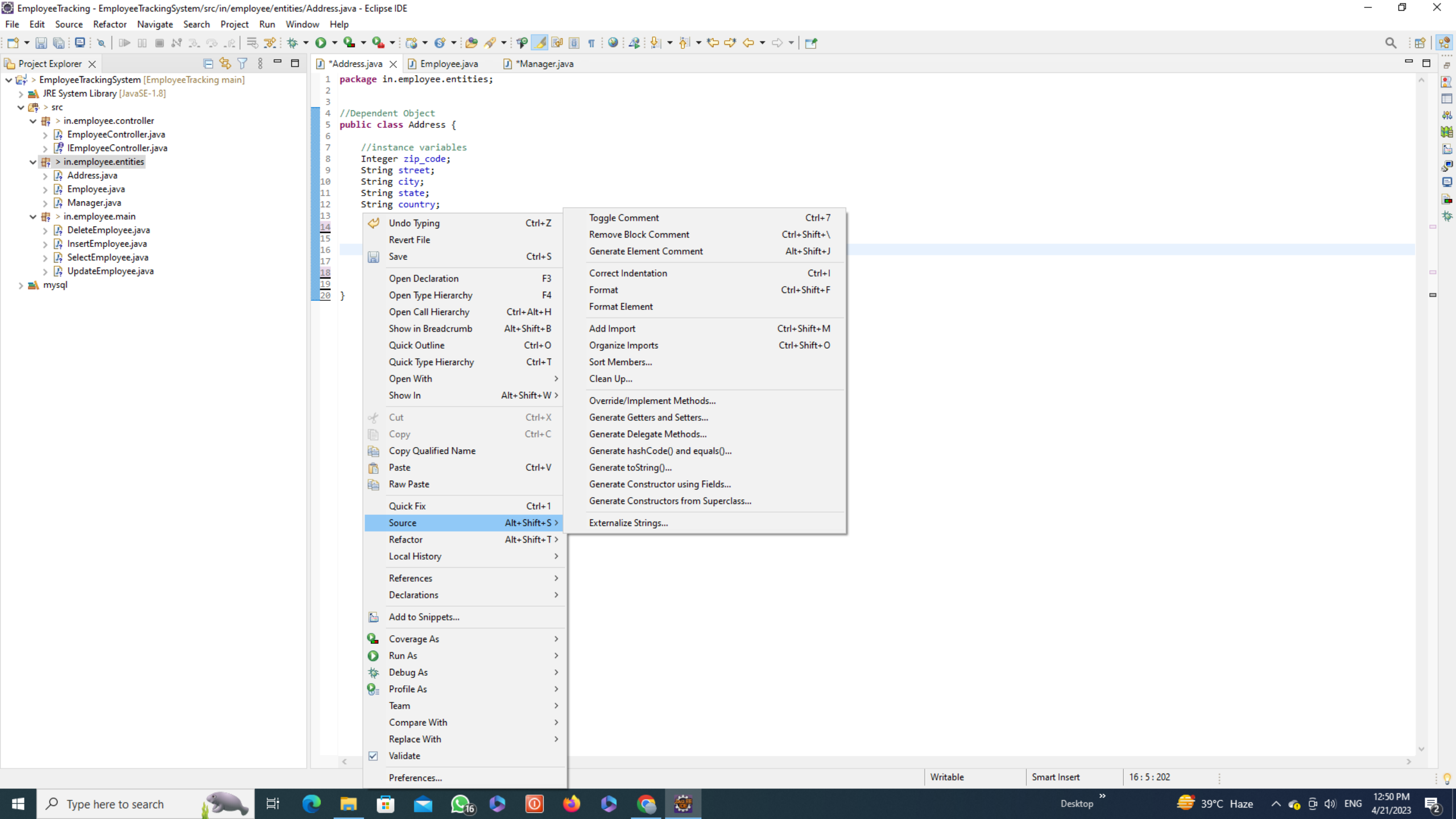
Unit testing focuses verification effort on the smallest unit of software design—the software component or module. Here each module is tested individually and it was ensured that all the modules function as required.

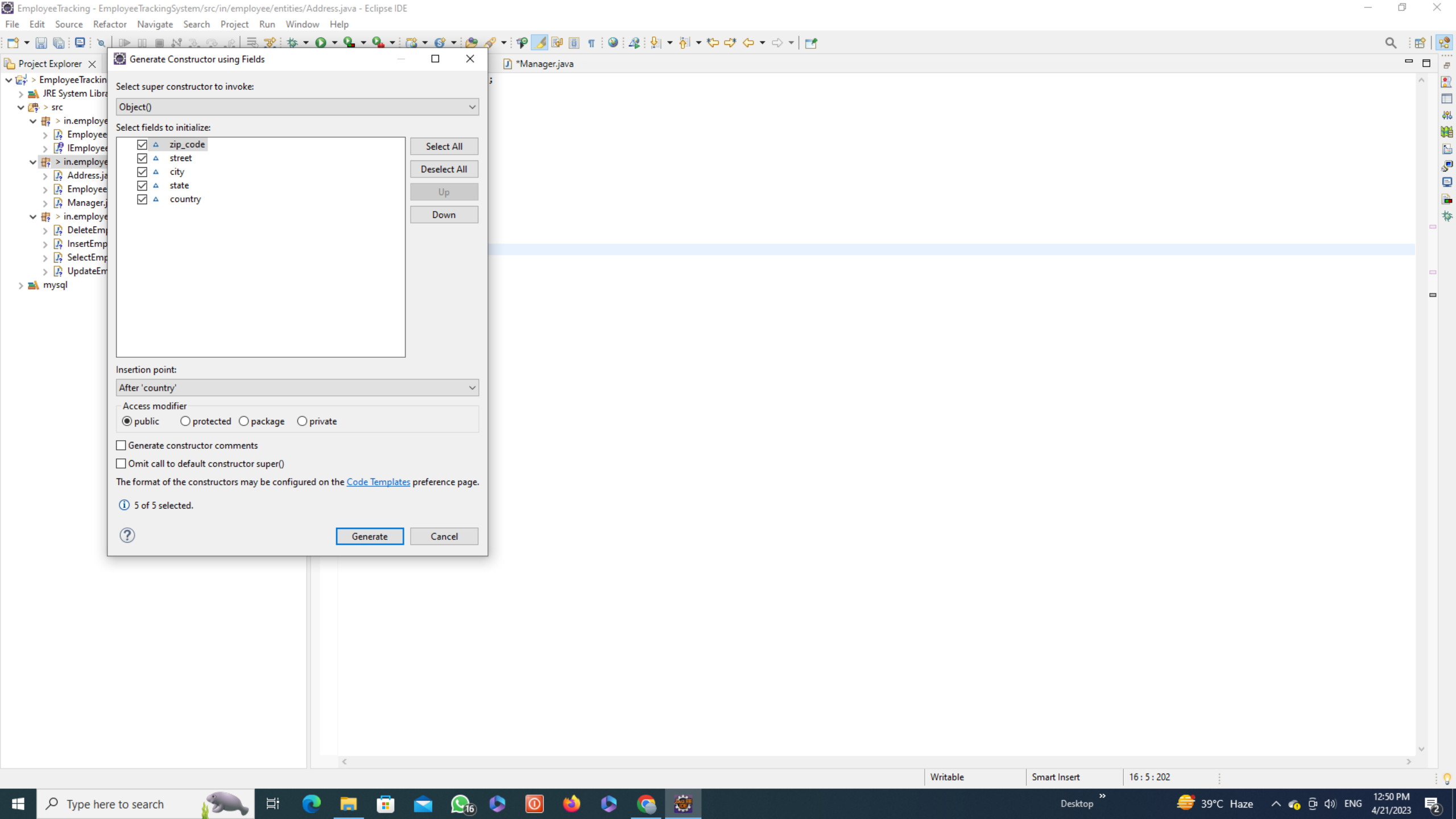
- **Integrated testing:**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing.

The objective is to take unit tested components and build a program structure that has been dictated by design. Here the tested modules is combined together and were tested to work properly as a group of interactive modules. The main purpose of this testing is to check the interface between the modules.

Some test cases are supplied:





Project Explorer

- EmployeeTrackingSystem [EmployeeTrackingSystem]
- JRE System Library [JavaSE-1.8]
- src
 - in.employee.controller
 - EmployeeController.java
 - IEmployeeController.java
 - in.employee.entities
 - Address.java
 - Employee.java
 - Manager.java
 - in.employee.main
 - DeleteEmployee.java
 - InsertEmployee.java
 - SelectEmployee.java
 - UpdateEmployee.java
 - mysql

Generate Getters and Setters

Select getters and setters to create:

- ☒ city
- ☒ country
- ☒ state
- ☒ street
- ☒ zip_code

Select All
Deselect All
Select Getters
Select Settings

☐ Allow setters for final fields (remove 'final' modifier from fields if necessary)

Insertion point:
After 'getCountry()'

Sort by:
Fields in getter/setter pairs

Access modifier
☒ public ☐ protected ☐ package ☐ private
☐ final ☐ synchronized

☐ Generate method comments

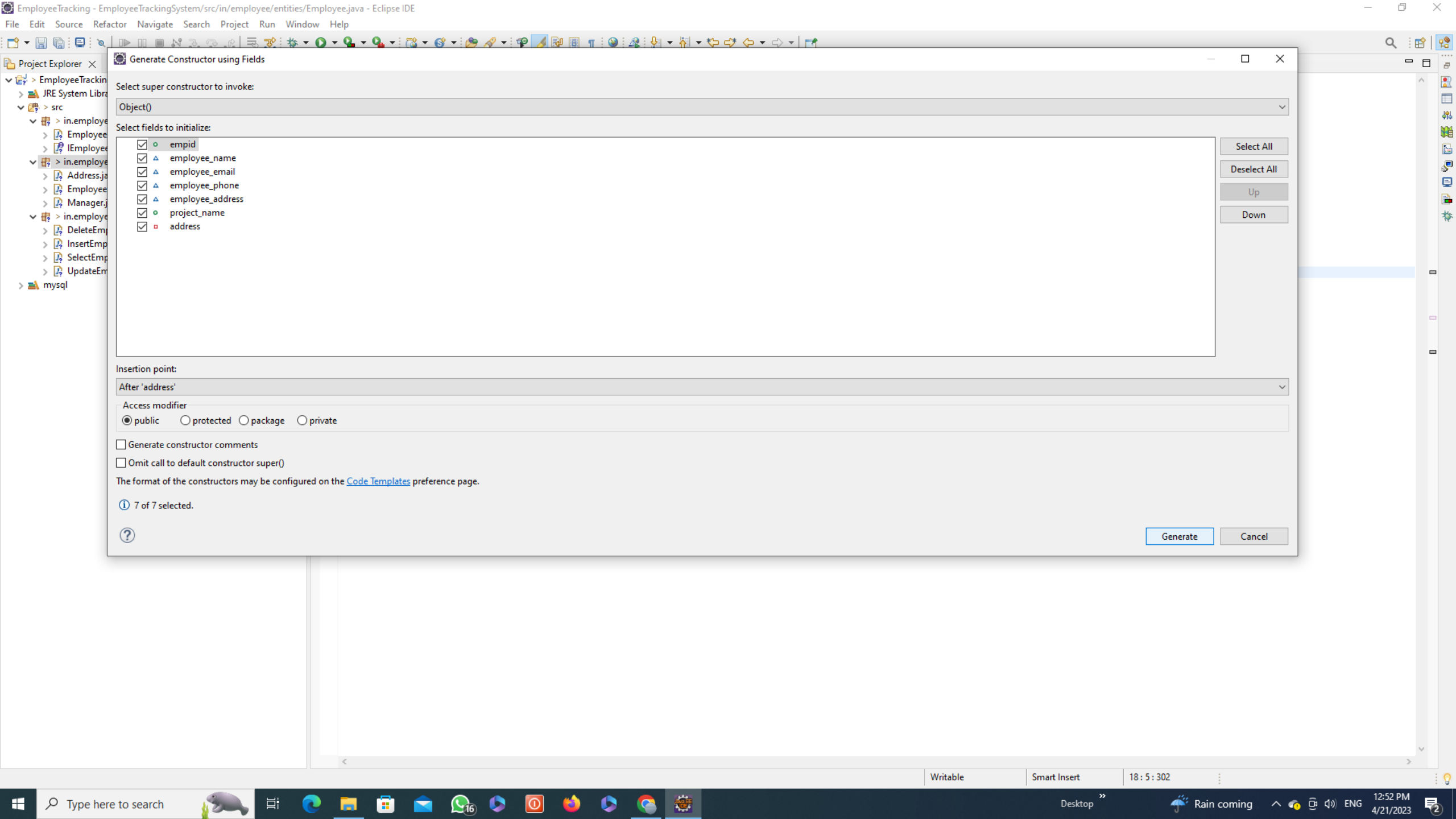
The format of the getters/setters may be configured on the [Code Templates](#) preference page.

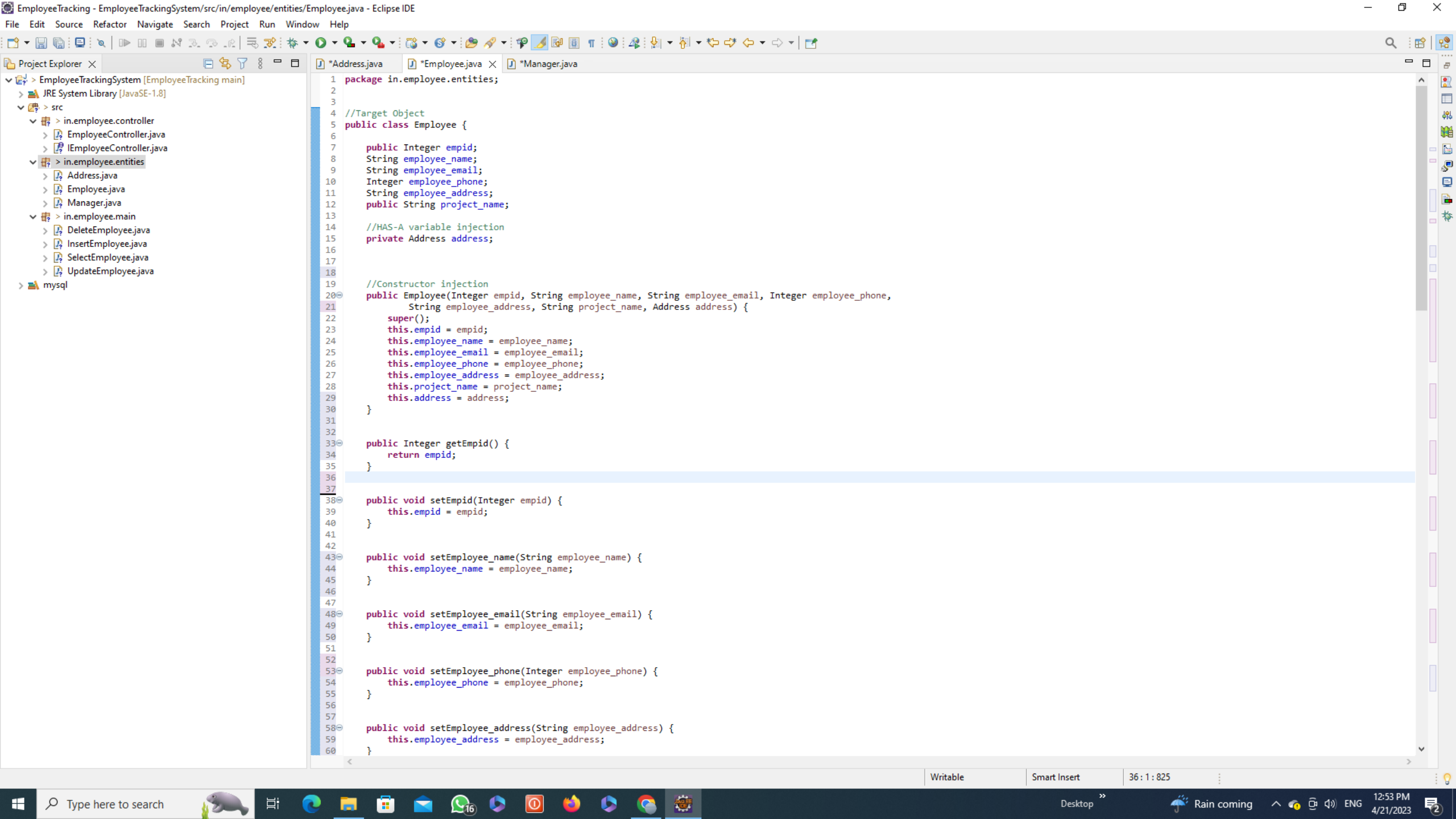
5 of 5 selected.

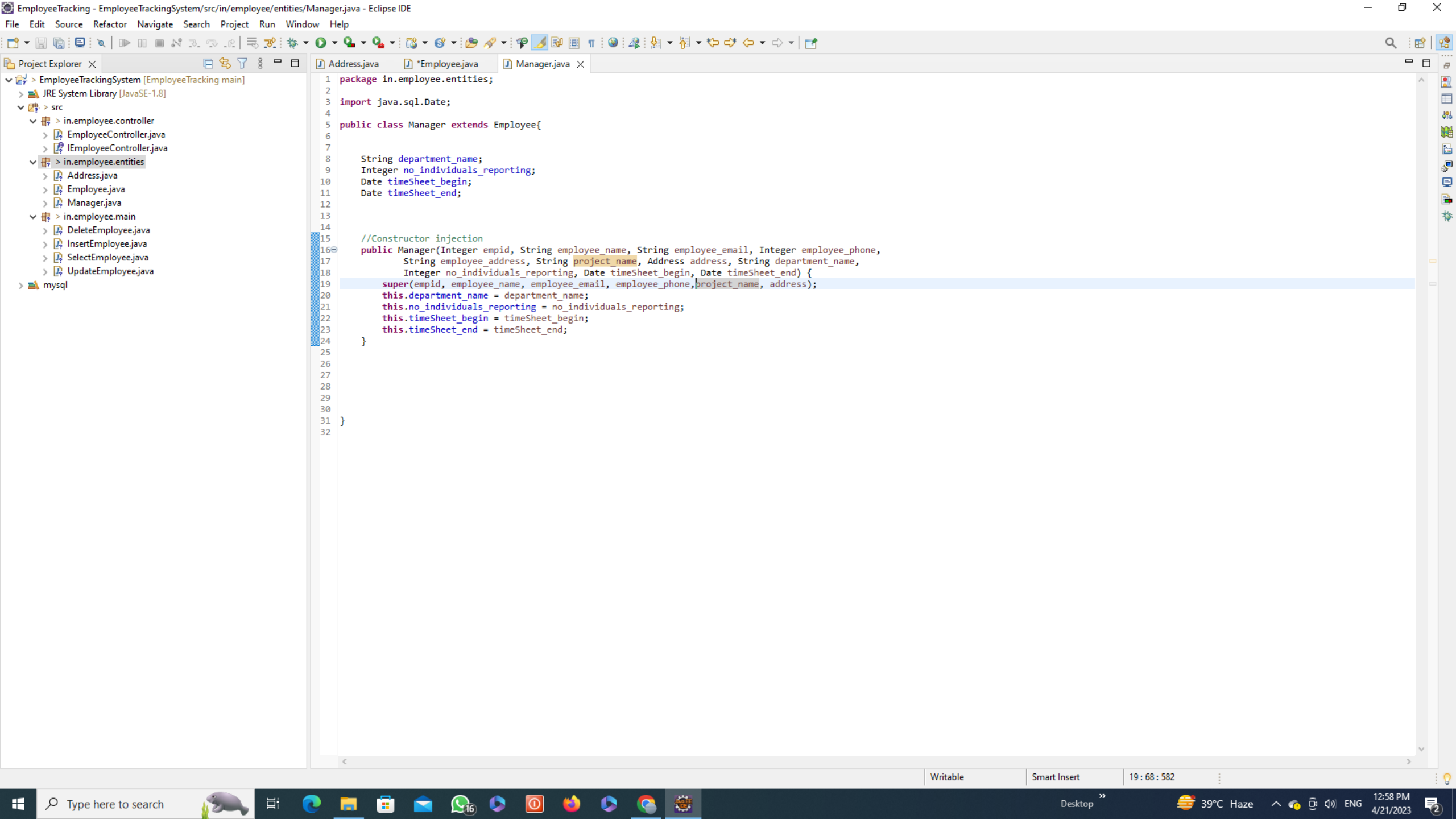
Generate Cancel

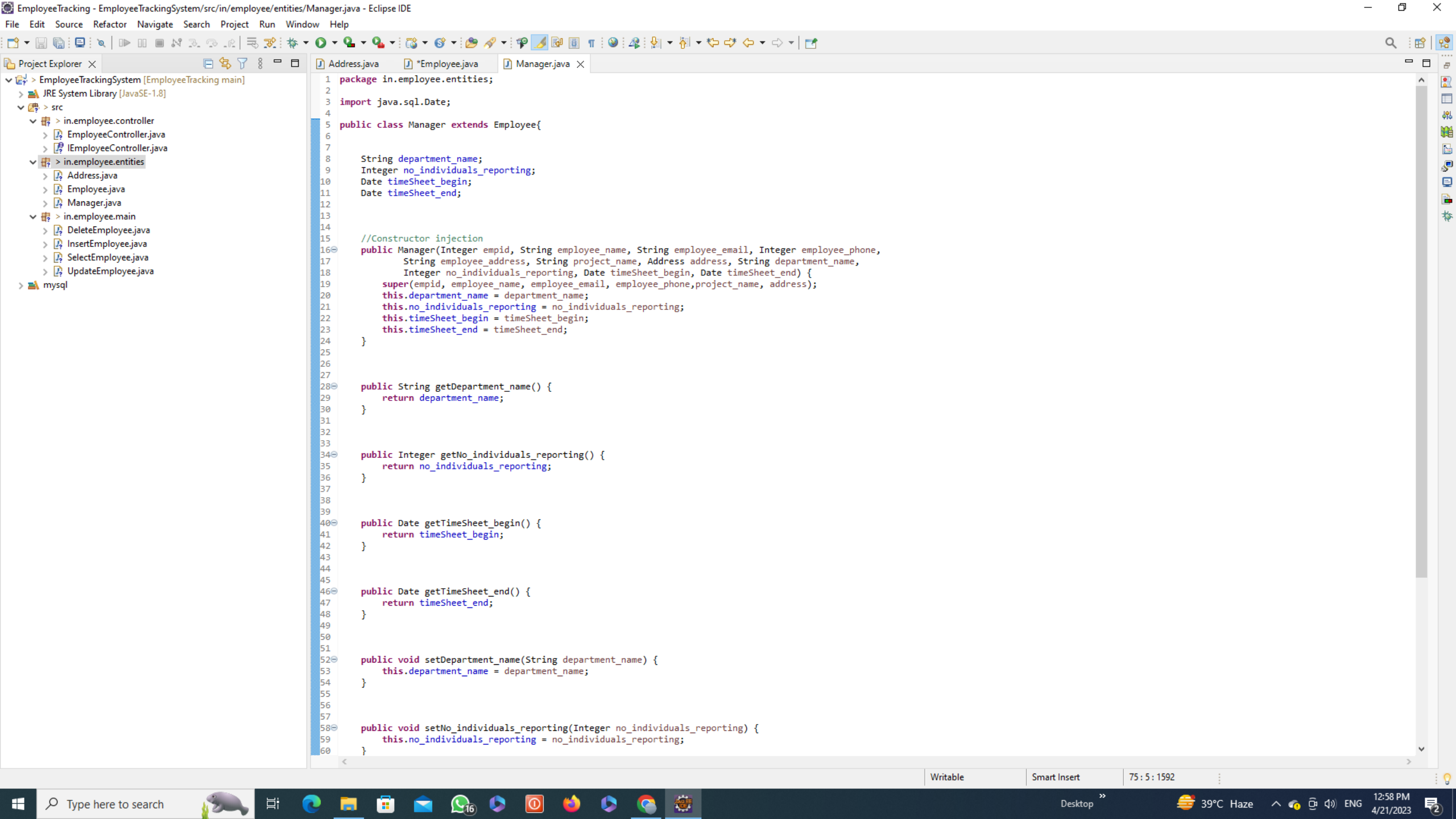
```
street, String city, String state, String country) {
```

```
40  
41 public String getState() {  
42     return state;  
43 }  
44  
45  
46 public String getCountry() {  
47     return country;  
48 }  
49  
50  
51  
52  
53  
54  
55  
56  
57 }
```



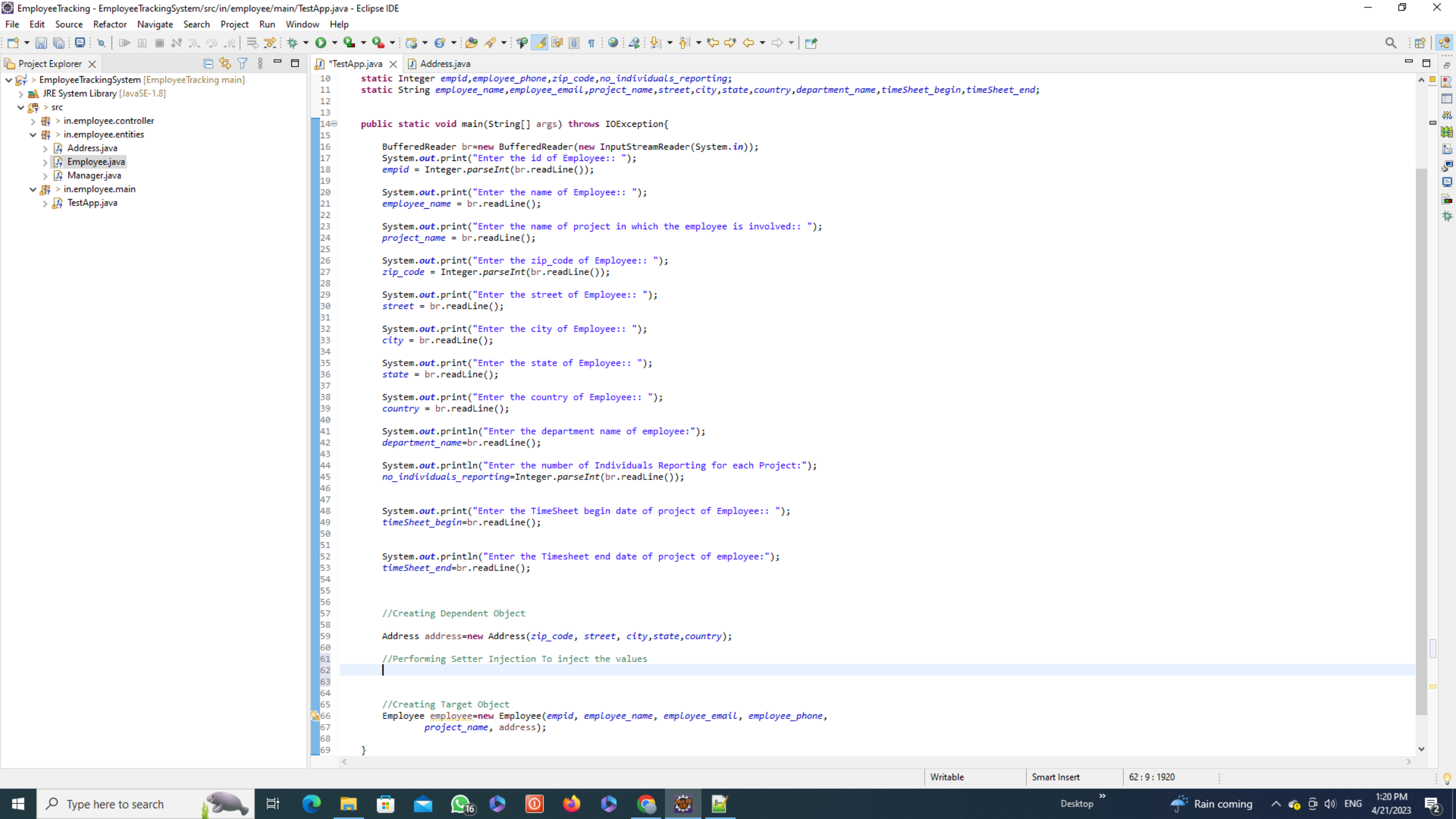


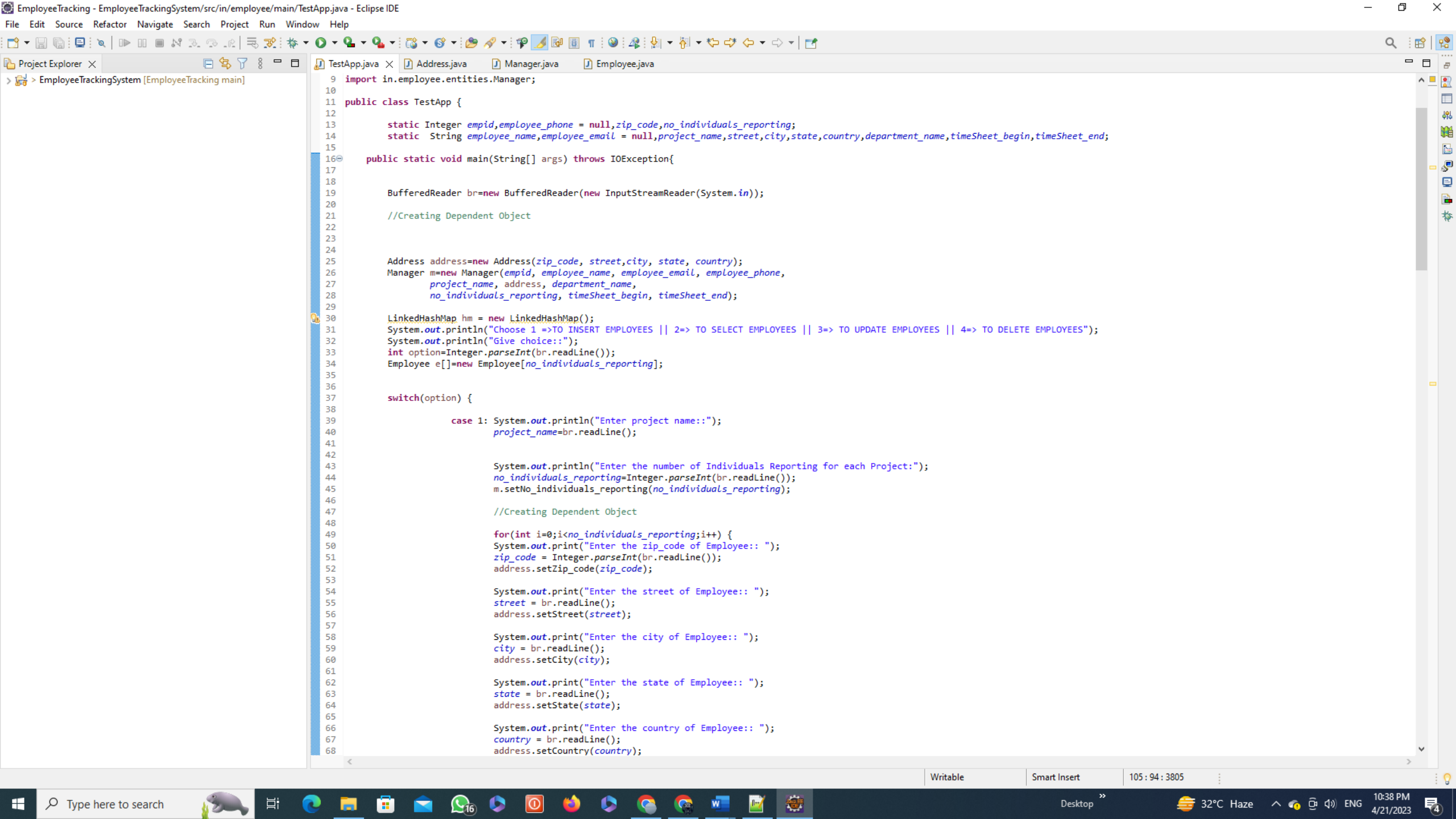




```
EmployeeTrackingSystem - EmployeeTrackingSystem/src/in/employee/entities/Manager.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
EmployeeTrackingSystem [EmployeeTracking main]
  JRE System Library [JavaSE-1.8]
  src
    in.employee.controller
      EmployeeController.java
      IEmployeeController.java
    in.employee.entities
      Address.java
      Employee.java
      Manager.java
    in.employee.main
      DeleteEmployee.java
      InsertEmployee.java
      SelectEmployee.java
      UpdateEmployee.java
  mysql

Address.java
Employee.java
Manager.java
1 package in.employee.entities;
2
3 import java.sql.Date;
4
5 public class Manager extends Employee{
6
7     String department_name;
8     Integer no_individuals_reporting;
9     Date timeSheet_begin;
10    Date timeSheet_end;
11
12
13
14
15    //Constructor injection
16    public Manager(Integer empid, String employee_name, String employee_email, Integer employee_phone,
17        String employee_address, String project_name, Address address, String department_name,
18        Integer no_individuals_reporting, Date timeSheet_begin, Date timeSheet_end) {
19        super(empid, employee_name, employee_email, employee_phone,project_name, address);
20        this.department_name = department_name;
21        this.no_individuals_reporting = no_individuals_reporting;
22        this.timeSheet_begin = timeSheet_begin;
23        this.timeSheet_end = timeSheet_end;
24    }
25
26
27
28    public String getDepartment_name() {
29        return department_name;
30    }
31
32
33
34    public Integer getNo_individuals_reporting() {
35        return no_individuals_reporting;
36    }
37
38
39
40    public Date getTimeSheet_begin() {
41        return timeSheet_begin;
42    }
43
44
45
46    public Date getTimeSheet_end() {
47        return timeSheet_end;
48    }
49
50
51
52    public void setDepartment_name(String department_name) {
53        this.department_name = department_name;
54    }
55
56
57
58    public void setNo_individuals_reporting(Integer no_individuals_reporting) {
59        this.no_individuals_reporting = no_individuals_reporting;
60    }
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
Writable Smart Insert 75: 5: 1592
Desktop 39°C Haze 12:58 PM 4/21/2023
```





EmployeeTracking - EmployeeTrackingSystem/src/in/employee/entities/Manager.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

EmployeeTrackingSystem [EmployeeTracking main]
JRE System Library [JavaSE-1.8]
src
in.employee.entities
Address.java
Employee.java
Manager.java
in.employee.main
TestApp.java

TestApp.java Address.java Manager.java Employee.java

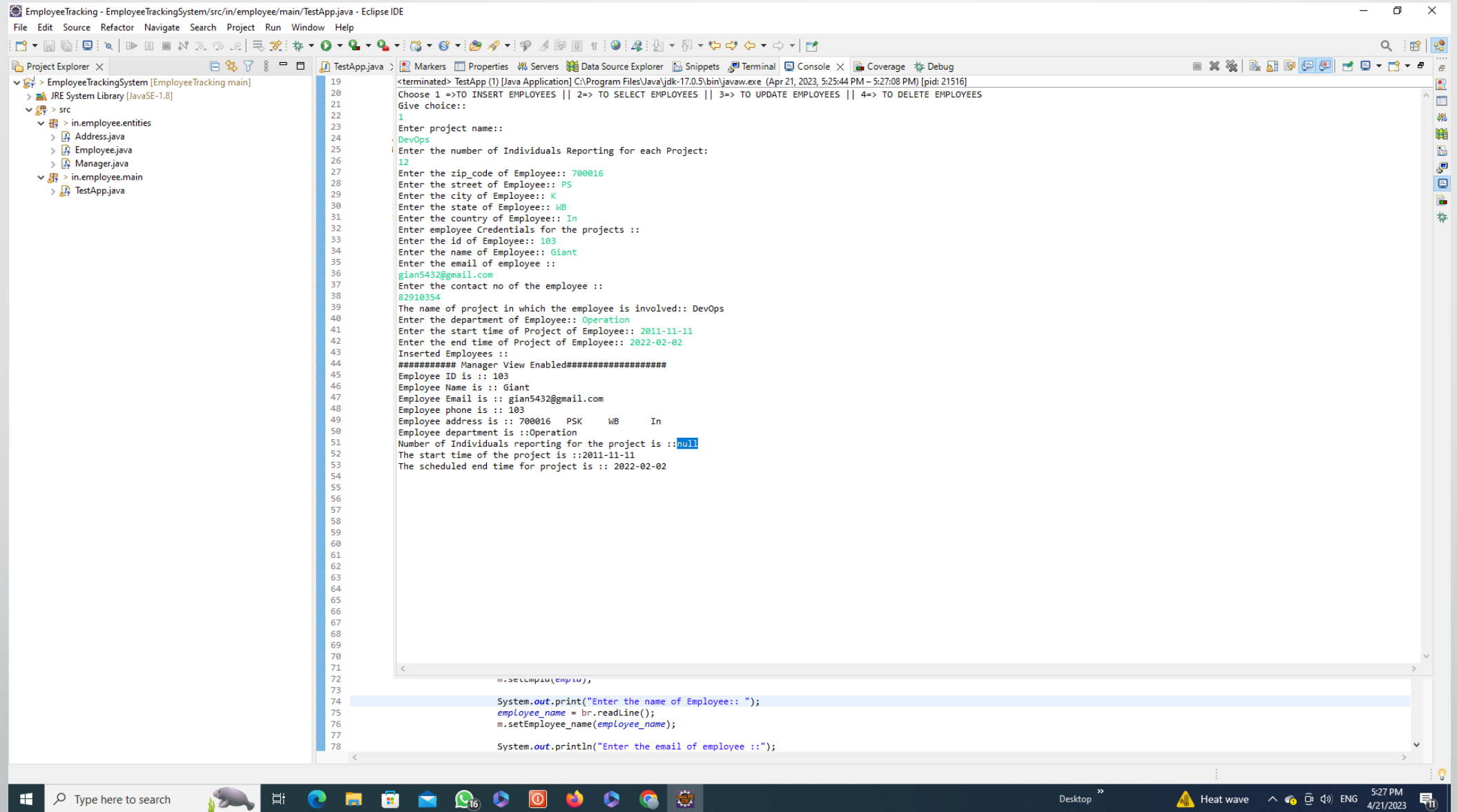
```
37
38 public Integer getNo_individuals_reporting() {
39     return no_individuals_reporting;
40 }
41
42
43
44 public String getTimeSheet_begin() {
45     return timeSheet_begin;
46 }
47
48
49
50 public String getTimeSheet_end() {
51     return timeSheet_end;
52 }
53
54
55
56 public void setDepartment_name(String department_name) {
57     this.department_name = department_name;
58 }
59
60
61
62 public void setNo_individuals_reporting(Integer no_individuals_reporting) {
63     this.no_individuals_reporting = no_individuals_reporting;
64 }
65
66
67
68 public void setTimeSheet_begin(String timeSheet_begin) {
69     this.timeSheet_begin = timeSheet_begin;
70 }
71
72
73
74 public void setTimeSheet_end(String timeSheet_end) {
75     this.timeSheet_end = timeSheet_end;
76 }
77
78
79
80 public void viewManager() {
81     System.out.println("##### Manager View Enabled#####");
82     System.out.println("Employee ID is :: "+super.getEmpid());
83     System.out.println("Employee Name is :: "+super.getEmployee_name());
84     System.out.println("Employee Email is :: "+super.getEmployee_email());
85     System.out.println("Employee phone is :: "+super.getEmpid());
86     System.out.println("Employee address is :: "+super.getAddress().getZip_code()+"\t"+super.getAddress().getStreet()+super.getAddress().getCity()+"\t"+super.getAddress().getState()+"\t"+super.getAd
87     System.out.println("Employee department is :: "+this.department_name);
88     System.out.println("Number of Individuals reporting for the project is :: "+this.no_individuals_reporting);
89     System.out.println("The start time of the project is :: "+this.timeSheet_begin);
90     System.out.println("The scheduled end time for project is :: "+this.timeSheet_end);
91     System.out.println();
92 }
93
94
95 }
96
```

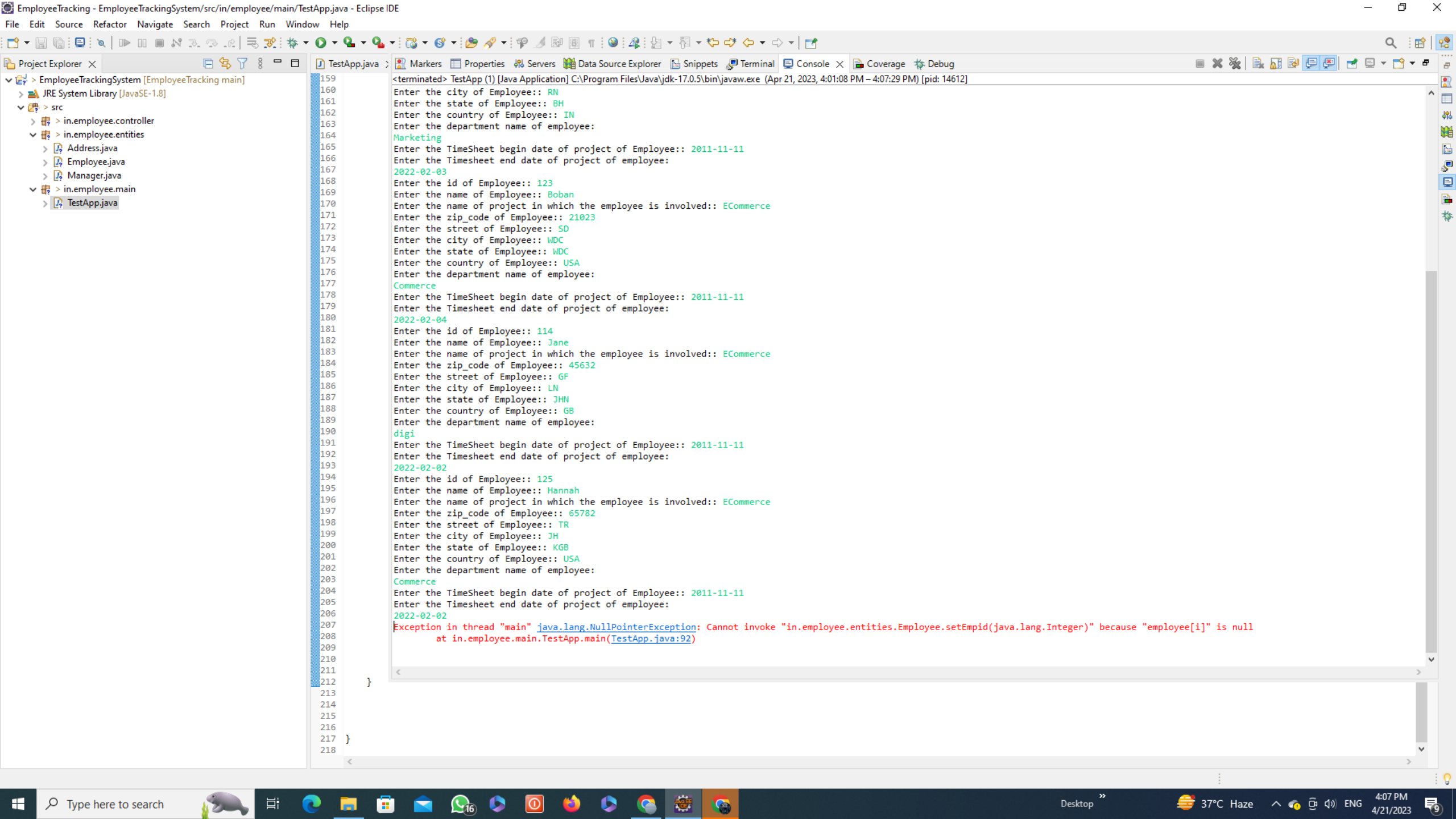
Address in.employee.entities.Employee.getAddress()

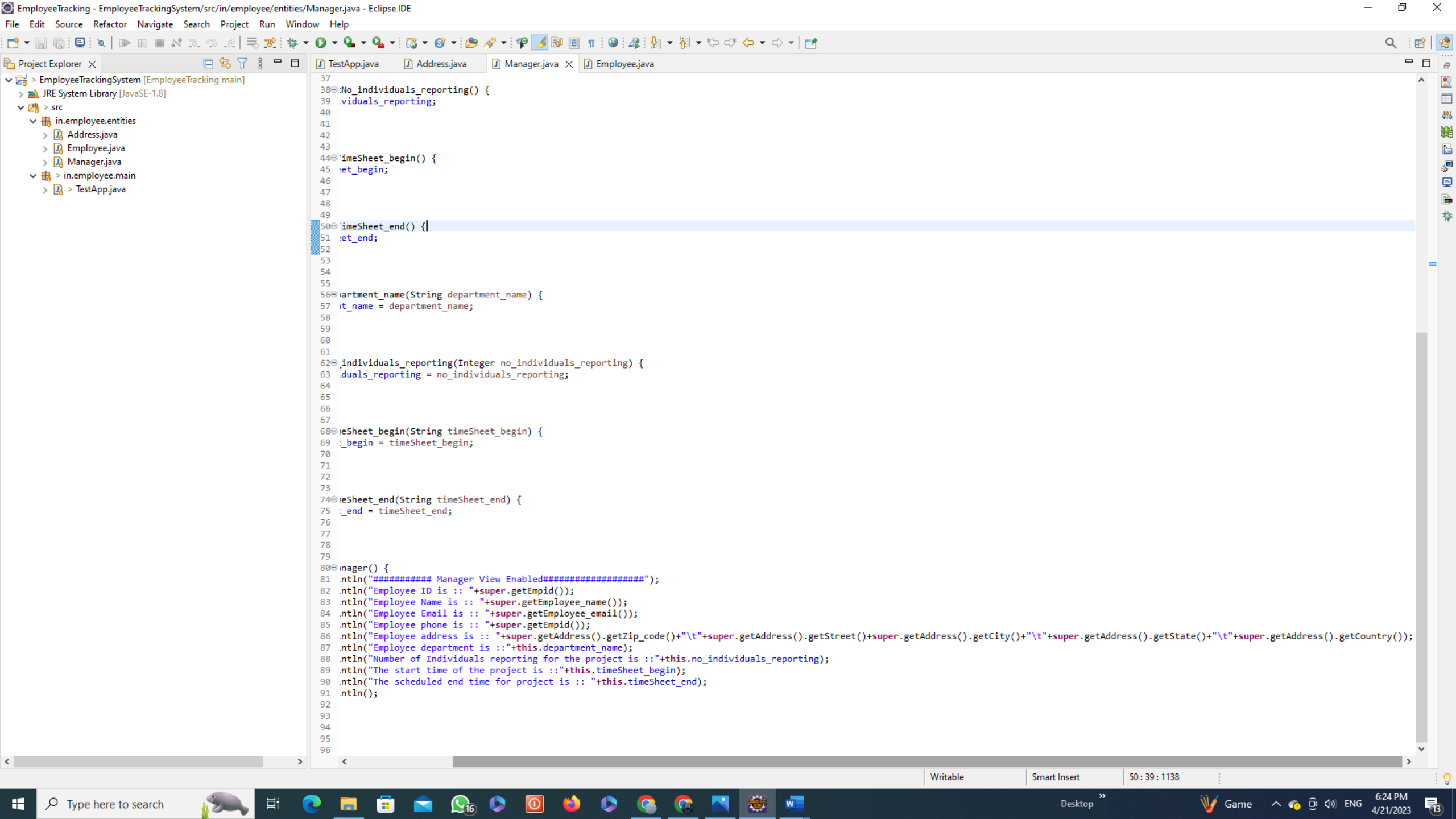
Press 'F2' for focus

Writable Smart Insert 50:39:1138

Desktop 34°C Haze 6:04 PM 4/21/2023







EmployeeTracking - EmployeeTrackingSystem/src/in/employee/entities/Manager.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

EmployeeTrackingSystem [EmployeeTracking main]
JRE System Library [JavaSE-1.8]
src
in.employee.entities
Address.java
Employee.java
Manager.java
in.employee.main
TestApp.java

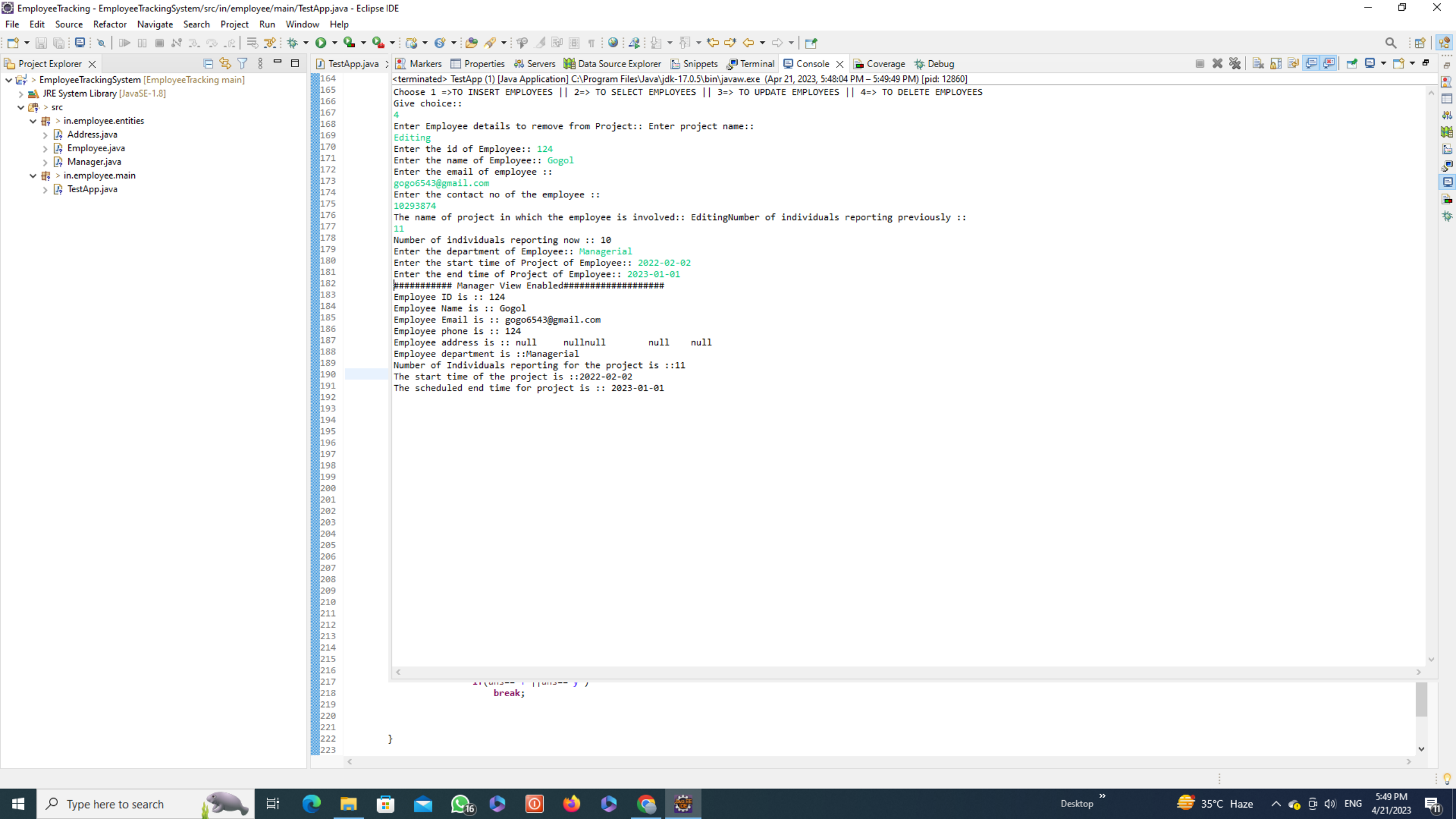
TestApp.java Address.java Manager.java Employee.java

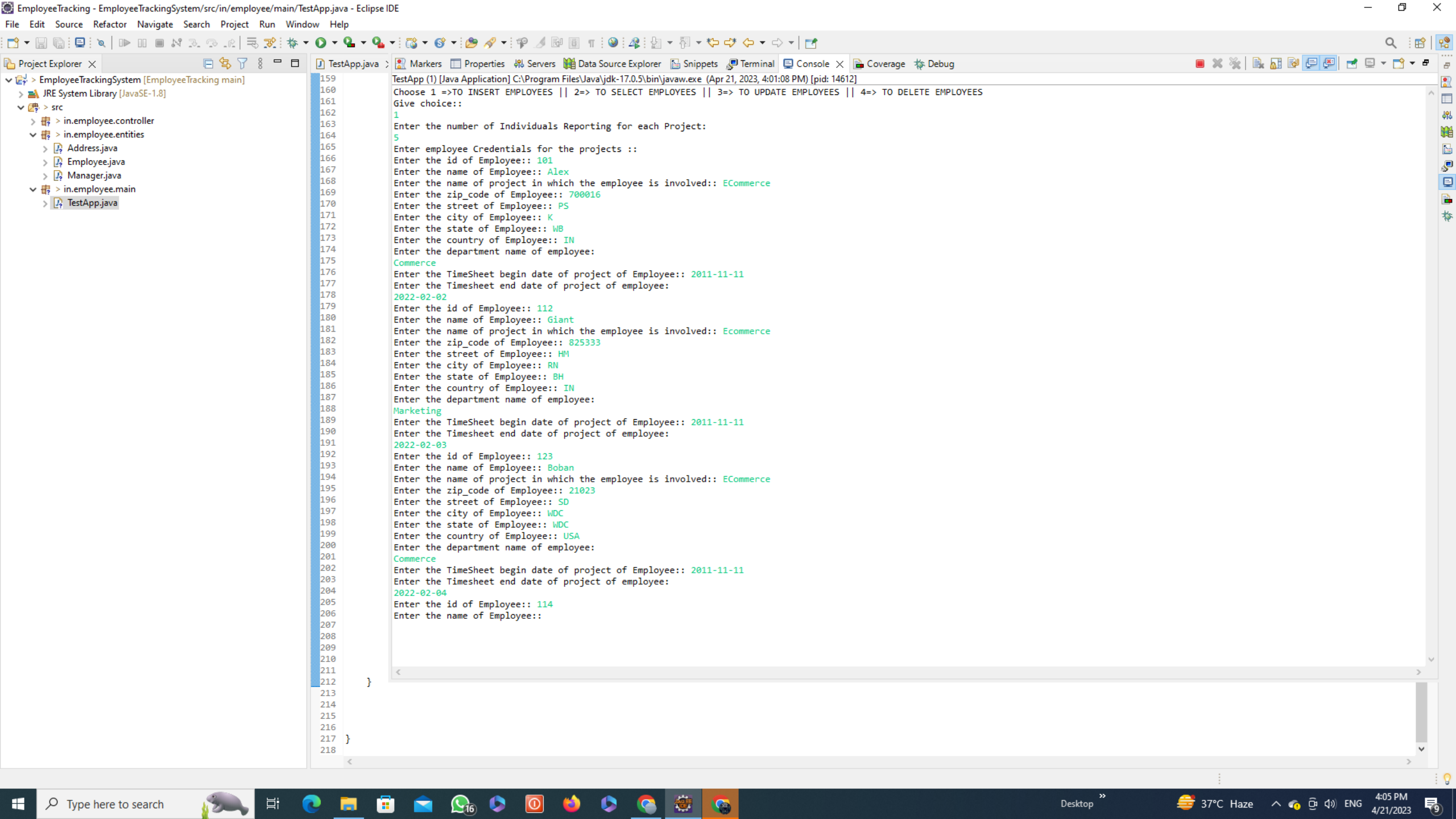
37
38 public Integer getNo_individuals_reporting() {
39 return no_individuals_reporting;
40 }
41
42
43
44 public String getTimeSheet_begin() {
45 return timeSheet_begin;
46 }
47
48
49
50 public String getTimeSheet_end() {
51 return timeSheet_end;
52 }
53
54
55
56 public void setDepartment_name(String department_name) {
57 this.department_name = department_name;
58 }
59
60
61
62 public void setNo_individuals_reporting(Integer no_individuals_reporting) {
63 this.no_individuals_reporting = no_individuals_reporting;
64 }
65
66
67
68 public void setTimeSheet_begin(String timeSheet_begin) {
69 this.timeSheet_begin = timeSheet_begin;
70 }
71
72
73
74 public void setTimeSheet_end(String timeSheet_end) {
75 this.timeSheet_end = timeSheet_end;
76 }
77
78
79
80 public void viewManager() {
81 System.out.println("##### Manager View Enabled#####");
82 System.out.println("Employee ID is :: "+super.getEmpid());
83 System.out.println("Employee Name is :: "+super.getEmployee_name());
84 System.out.println("Employee Email is :: "+super.getEmployee_email());
85 System.out.println("Employee phone is :: "+super.getEmpid());
86 System.out.println("Employee address is :: "+super.getAddress().getZip_code()+"\t"+super.getAddress().getStreet()+super.getAddress().getCity()+"\t"+super.getAddress().getState()+"\t"+super.getAd
87 System.out.println("Employee department is :: "+this.department_name);
88 System.out.println("Number of Individuals reporting for the project is :: "+this.no_individuals_reporting);
89 System.out.println("The start time of the project is :: "+this.timeSheet_begin);
90 System.out.println("The scheduled end time for project is :: "+this.timeSheet_end);
91 System.out.println();
92 }
93
94
95 }
96

Address in.employee.entities.Employee.getAddress()
Press 'F2' for focus

Writable Smart Insert 50 : 39 : 1138

Desktop 34°C Haze 6:04 PM 4/21/2023





EmployeeTracking - EmployeeTrackingSystem/src/in/employee/main/TestApp.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

EmployeeTrackingSystem [EmployeeTracking main]
JRE System Library [JavaSE-1.8]
src
in.employee.entities
Address.java
Employee.java
Manager.java
in.employee.main
TestApp.java

TestApp.java

31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86

<terminated> TestApp (1) [Java Application] C:\Program Files\Java\jdk-17.0.5\bin\javaw.exe (Apr 21, 2023, 5:28:33 PM – 5:31:04 PM) [pid: 14180]
Choose 1 =>TO INSERT EMPLOYEES || 2=> TO SELECT EMPLOYEES || 3=> TO UPDATE EMPLOYEES || 4=> TO DELETE EMPLOYEES
Give choice::
1
Enter project name::
ECommerce
Enter the number of Individuals Reporting for each Project:
8
Enter the zip_code of Employee:: 825432
Enter the street of Employee:: HM
Enter the city of Employee:: DN
Enter the state of Employee:: JH
Enter the country of Employee:: IN
Enter employee Credentials for the projects ::
Enter the id of Employee:: 212
Enter the name of Employee:: Electro
Enter the email of employee ::
elet9078@gmail.com
Enter the contact no of the employee ::
90076543
The name of project in which the employee is involved:: ECommerce
Enter the department of Employee:: Commerce
Enter the start time of Project of Employee:: 2021-05-05
Enter the end time of Project of Employee:: 2022-10-10
Inserted Employees ::
Manager View Enabled#####
Employee ID is :: 212
Employee Name is :: Electro
Employee Email is :: elet9078@gmail.com
Employee phone is :: 212
Employee address is :: 825432 HMDN JH IN
Employee department is ::Commerce
Number of Individuals reporting for the project is ::8
The start time of the project is ::2021-05-05
The scheduled end time for project is :: 2022-10-10

Markers Properties Servers Data Source Explorer Snippets Terminal Console Coverage Debug

void java.io.PrintStream.println()

Terminates the current line by writing the line separator string. The line separator string is defined by the system property line.separator, and is not necessarily a single newline character ('\n').

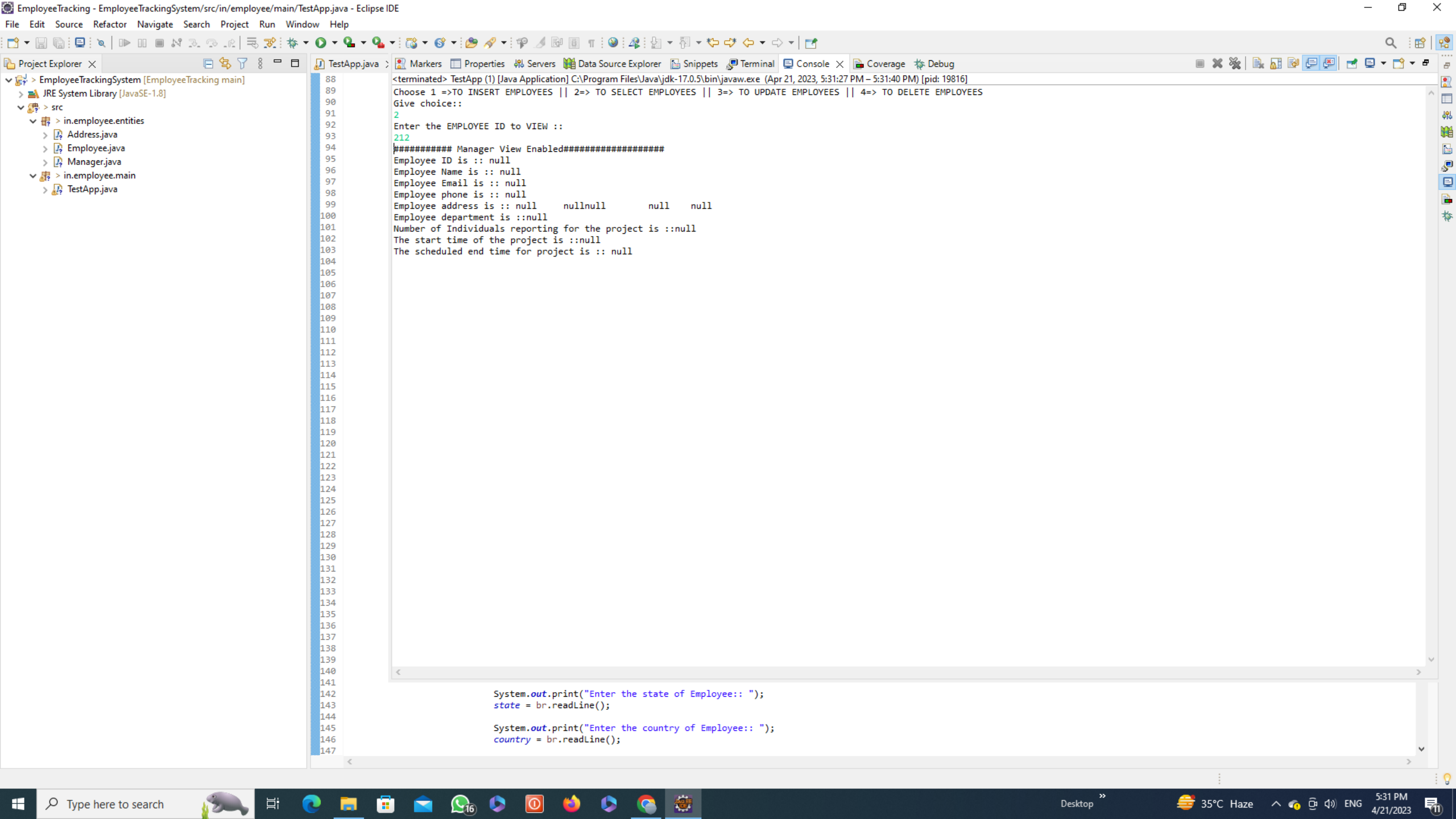
employee_phone=Integer.parseInt(ui.getTextLine());
m.setEmployee_phone(employee_phone);

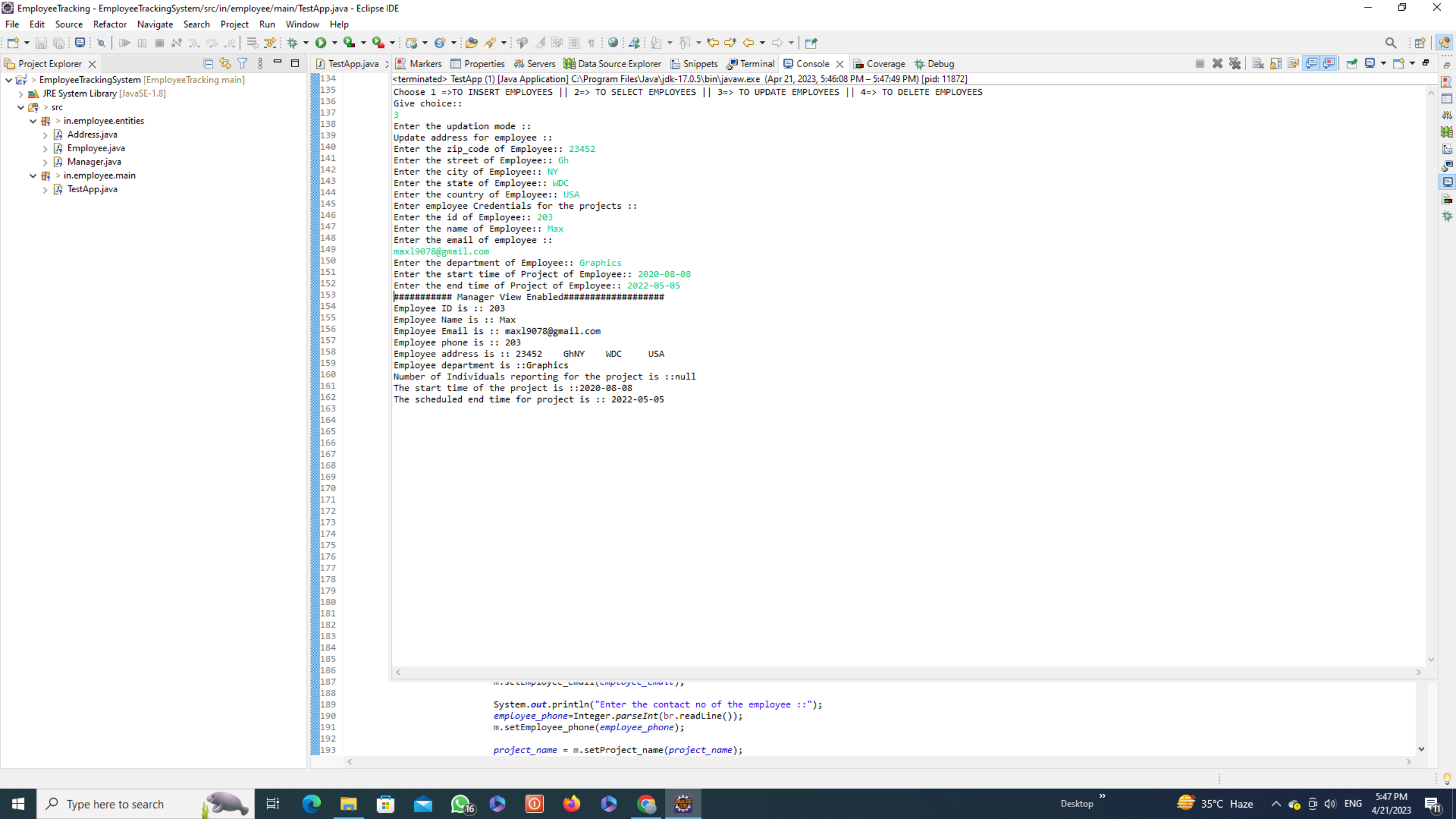
me = m.setProject_name(project_name);
print("The name of project in which the employee is involved: "+m.getProject_name());
println();

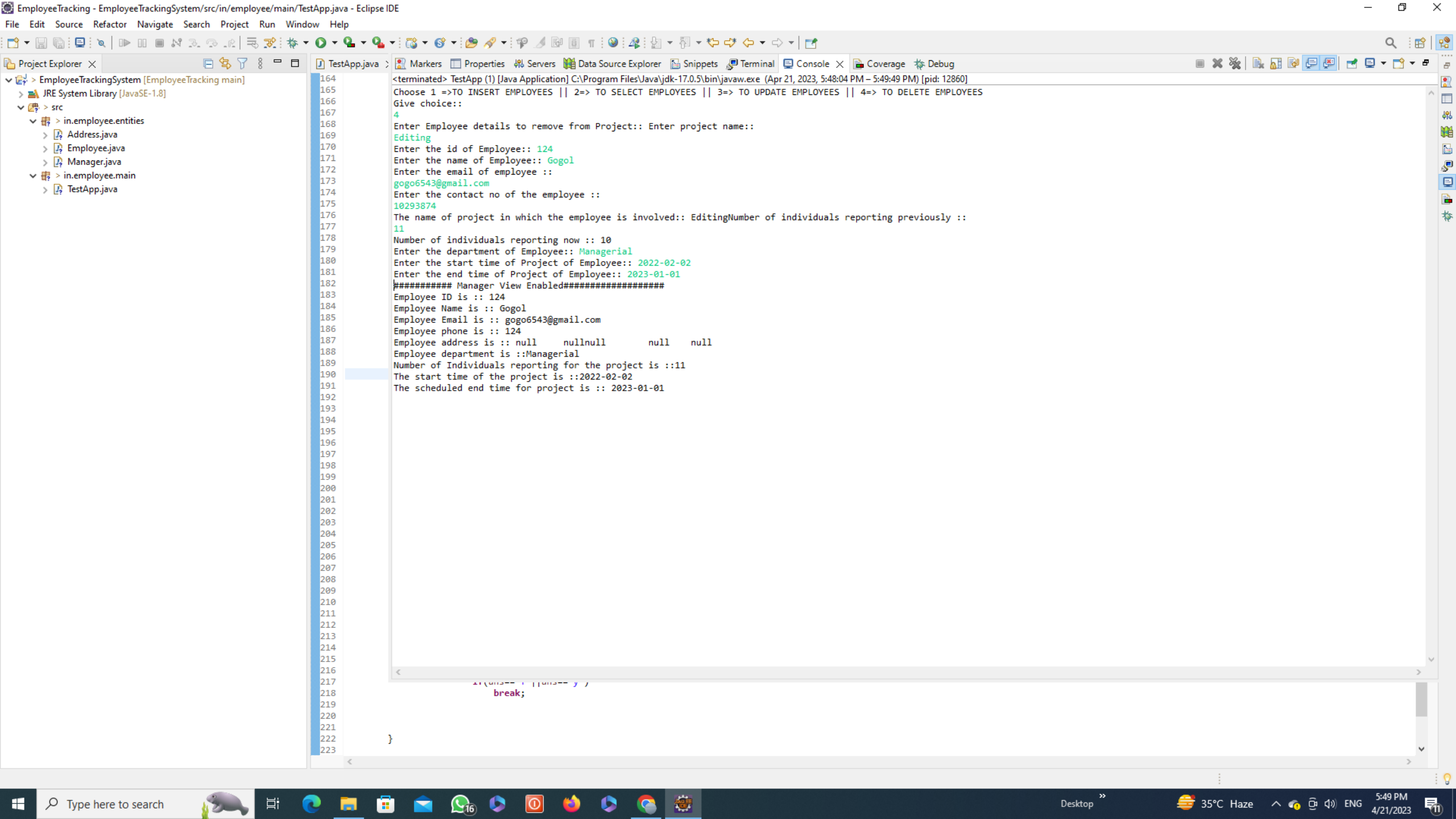
Desktop

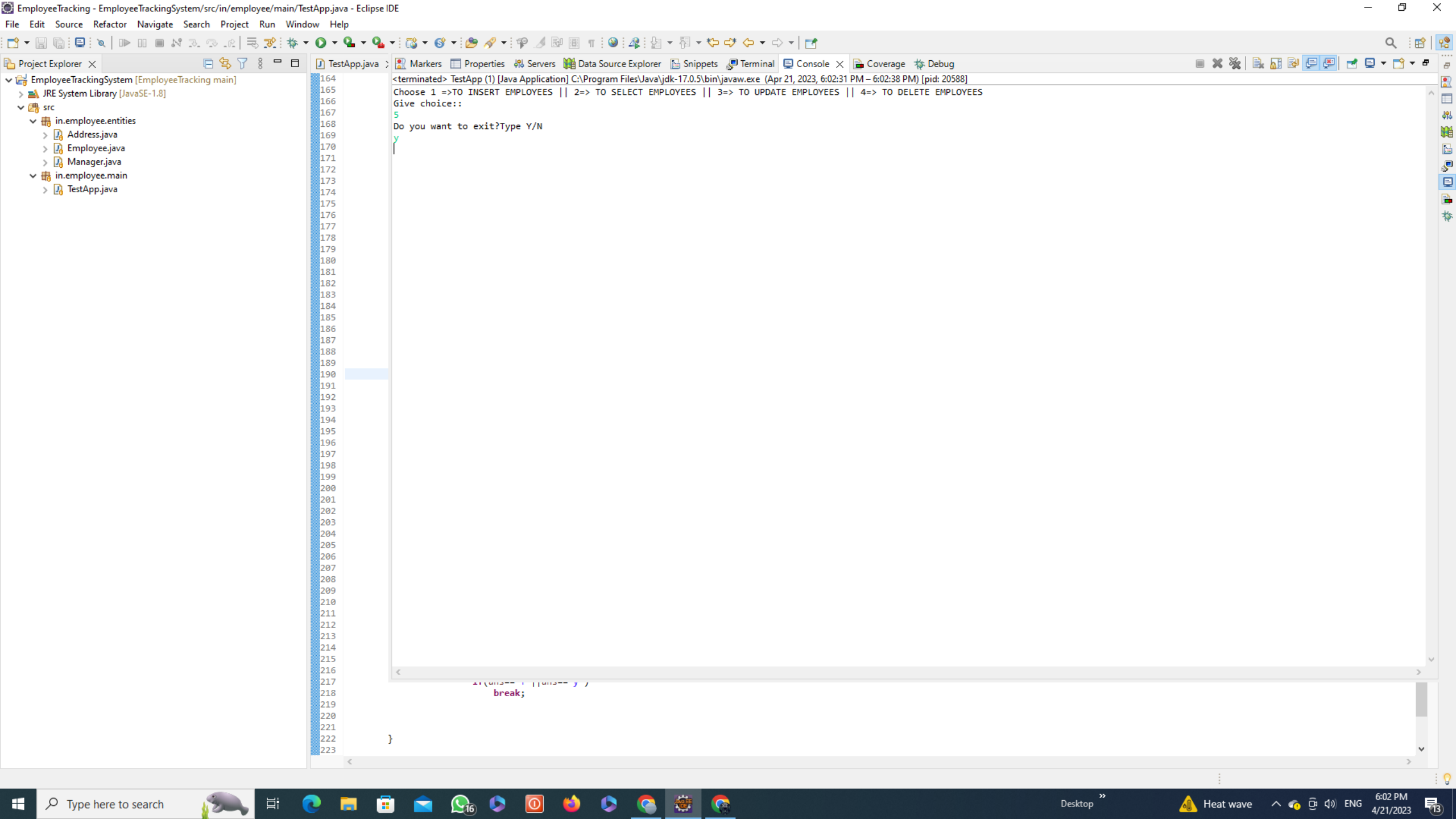
35°C Haze

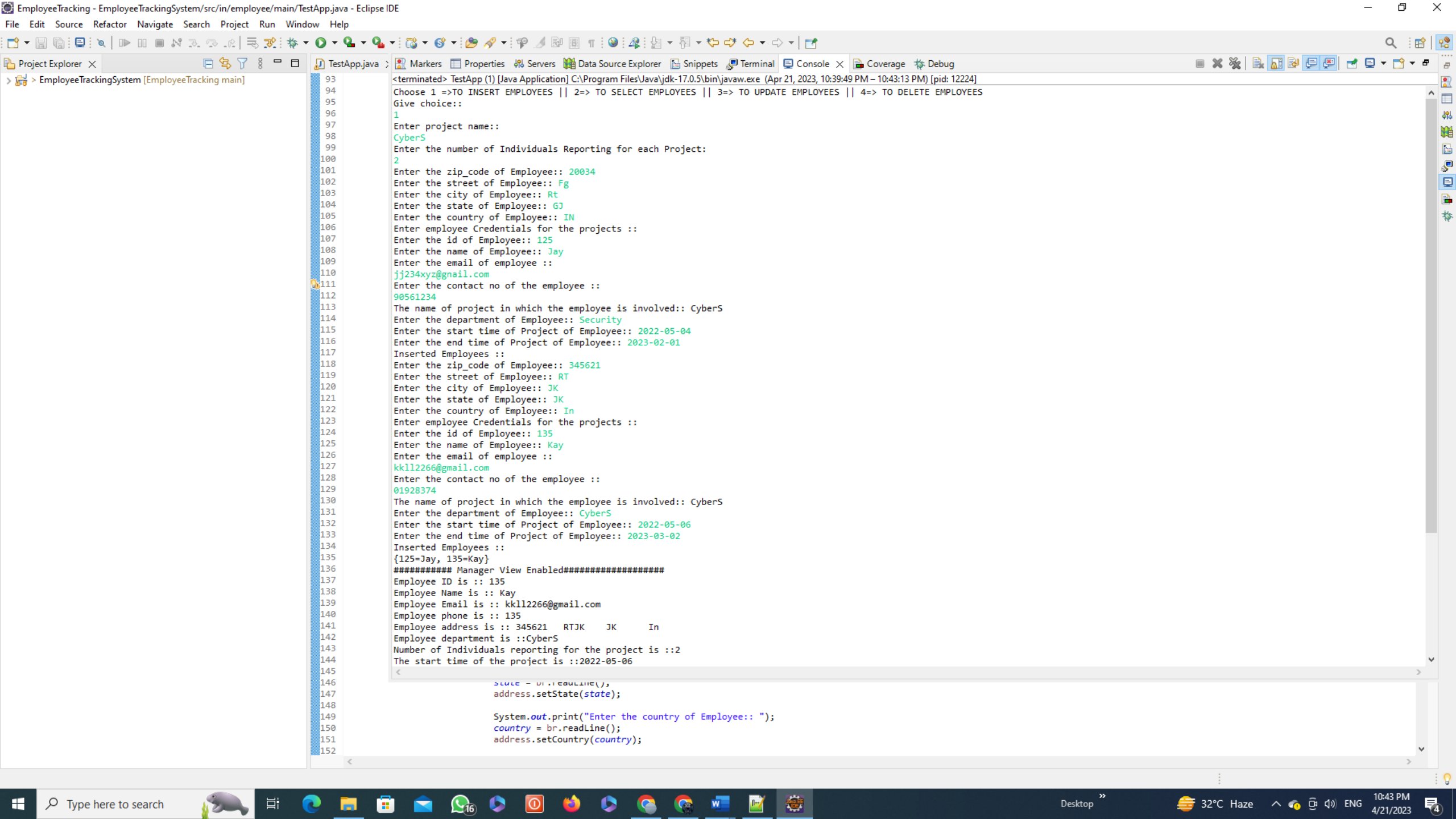
5:31 PM 4/21/2023

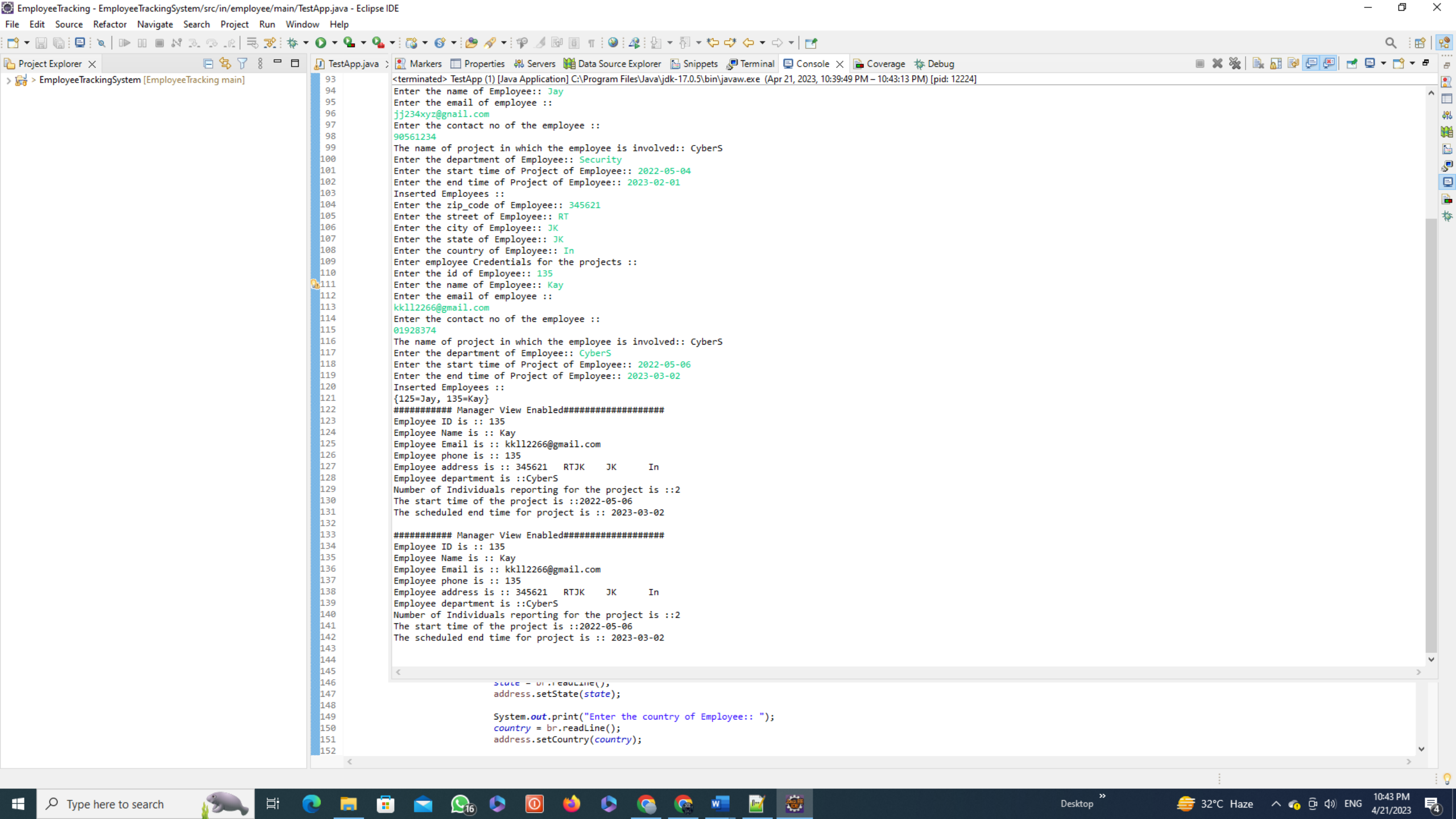












FUTURE SCOPE:

- ✚ **Hibernate integration**
- ✚ **Database connectivity**
- ✚ **Connecting jars in build path of project**

PROJECT AVAILABILITY:

Project-code url in github repository:

<https://github.com/paramita22/EmployeeTrackingSystem/tree/main/EmployeeTrackingSystem>

Project post in Linked in:

<https://www.linkedin.com/feed/update/urn:li:activity:7055241634241785856/>

Project demo video posted in linkedin:

<https://www.linkedin.com/feed/update/urn:li:activity:7055237843085885440/>