Low Level Design (LLD)

Employee Tracking System

Last date of revision: 21/04/2023

Paramita Pal

Document Version Control

Date Issued	Version	Description	Author
26th February,2023	**1.1**	**First Draft**	Paramita
27th February	**1.2**	**Workflow process added**	-Do-
20th April	**1.3**	**Constraints and Exceptions added**	-Do-
21st April	**1.4**	**Class Diagram added**	-Do-
21st April	**1.5**	**Flowcharts added denoting input and output**	-Do-
21st April	**1.6**	**Key performance indexes added**	-Do-
21st April	**1.7**	**Test cases discussed**	-Do-
21st April	**1.8**	**LLD ready for Submission**	

Contents

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 Need of a Low-Level Design Document:

The low-level Design Document contains a detailed description of the application to be developed. Low level designing contains bulky data and document to delve into details the development phase. The LLD is a reference document for developers, reference material with all the technicalities involved.

Low level designing aims at the technical detailing of the project. It consists of the algorithm and details about classes/ methods to achieve the required functionality in terms of business requirements. The Low Levell design aims to achieve the functional and non-functional requirements by giving a technical roadmap for it. The document divided into various sections to make the code reusable and scalable.

The main objective of the project is to make create a console-based application to be run in onsite premises to get a hands-on information about the ongoing projects and details of employees involved.

This project shall be delivered in a manner that suggestive changes may be easily implemented without disturbing the already existing data.:

1.2 Scope

This software system will be a console-based application with Core Java. This system will be designed to gather a first-hand information on which employee is involved in which project, their department name and location details.

In the application, employees may be tracked by the employer for the benefit of creating a clear understanding of their employees.

The project deals with a employee view as well as a manager view, to get a ready information on employees and projects.

1.3 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 with the Employee class.

Mysql was not used as JDBC was not to be used in this project.

1.4 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.

1.5 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

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

2.2. Software Requirements

2.2.1. Tools and platforms used

- Operating System: Windows11
- Platform: ECLIPSE IDE [2022-09]
- Language: CORE JAVA

2.2.2. Software interfaces

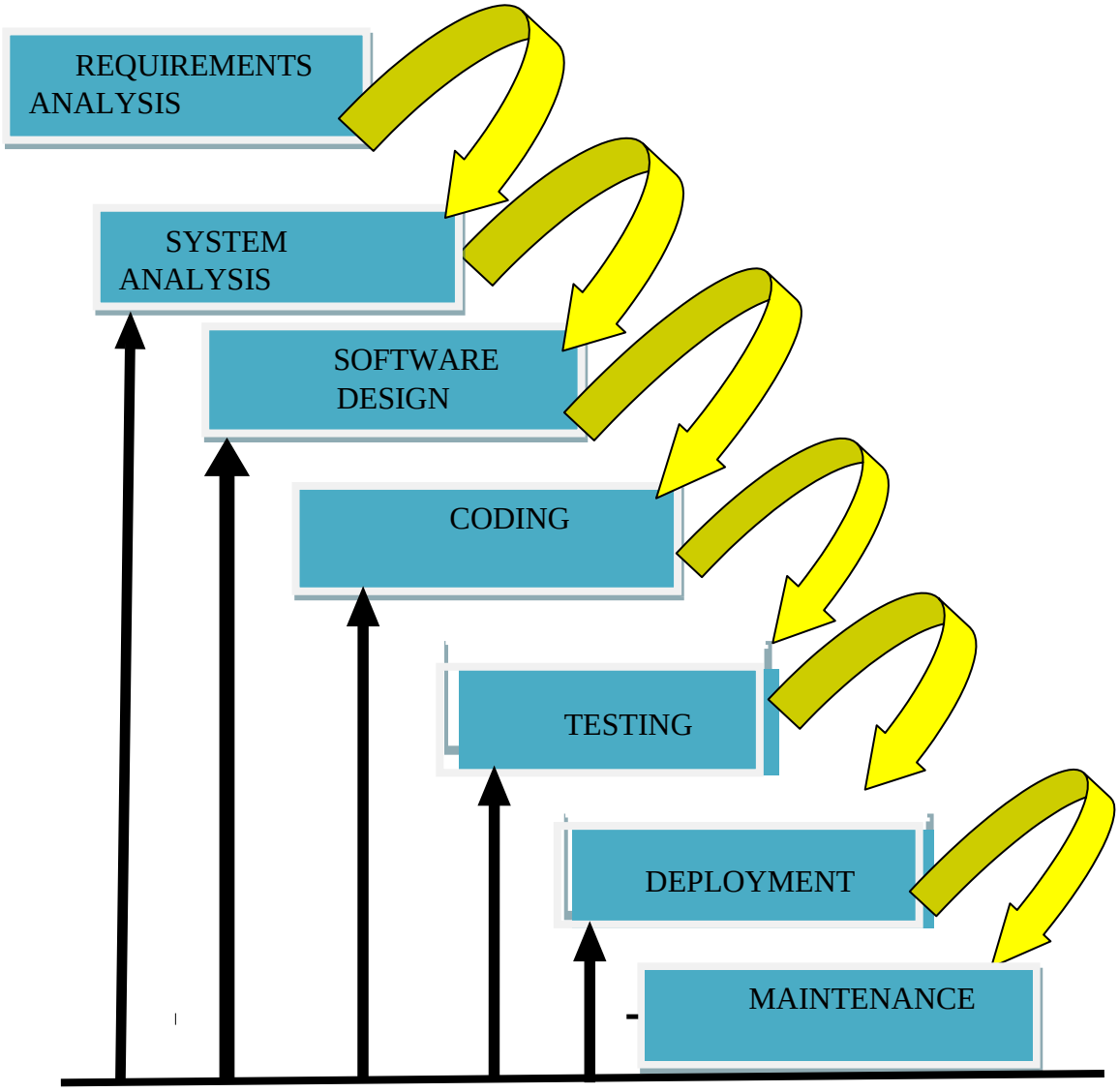
- Application: Eclipse IDE [2022-09]
- Jdk : 8
- Additional API : MS OFFICE

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

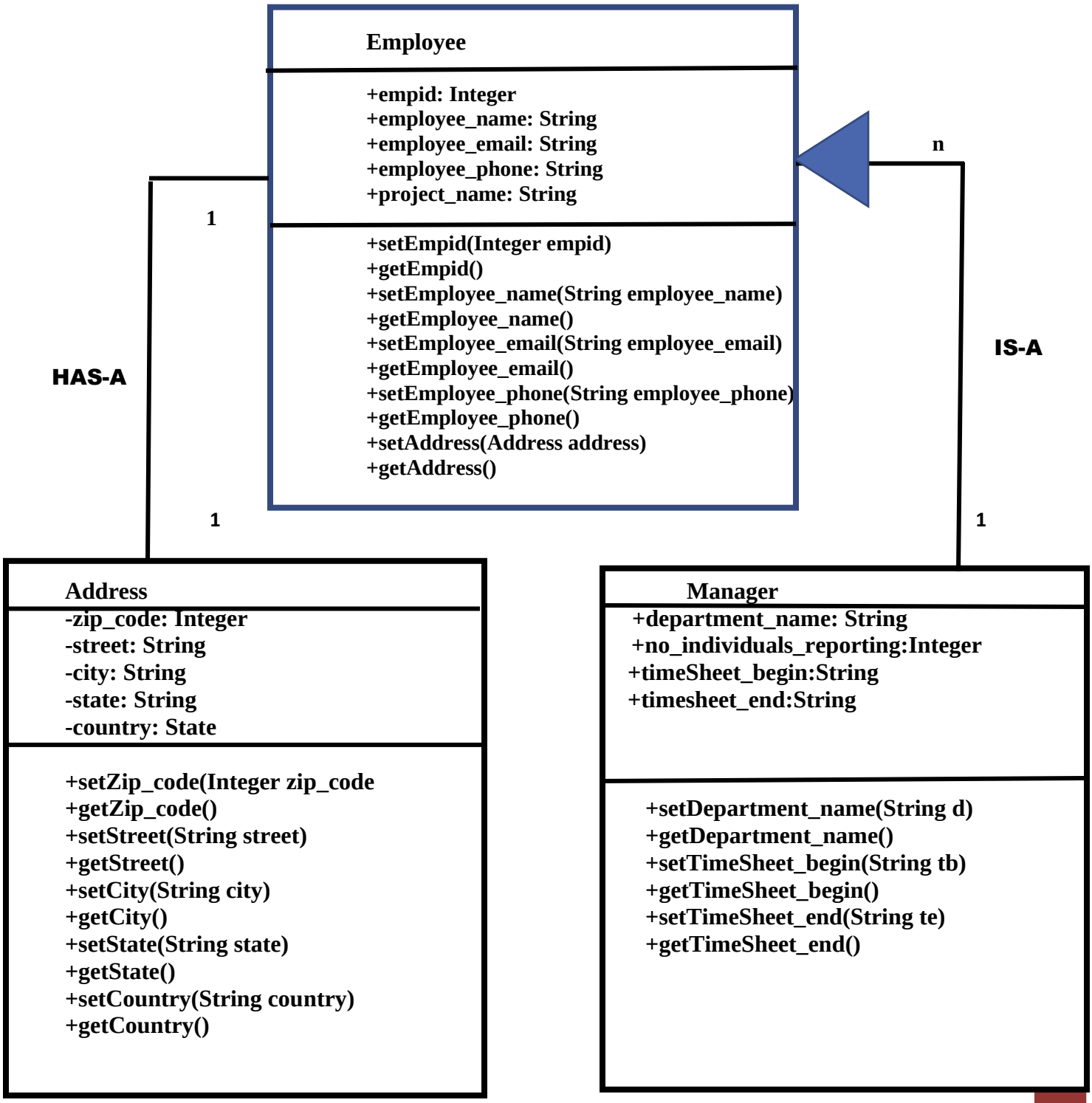
3. SOFTWARE DEVELOPMENT METHODOLOGY

3.1. SOFTWARE LIFECYCLE MODEL

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.



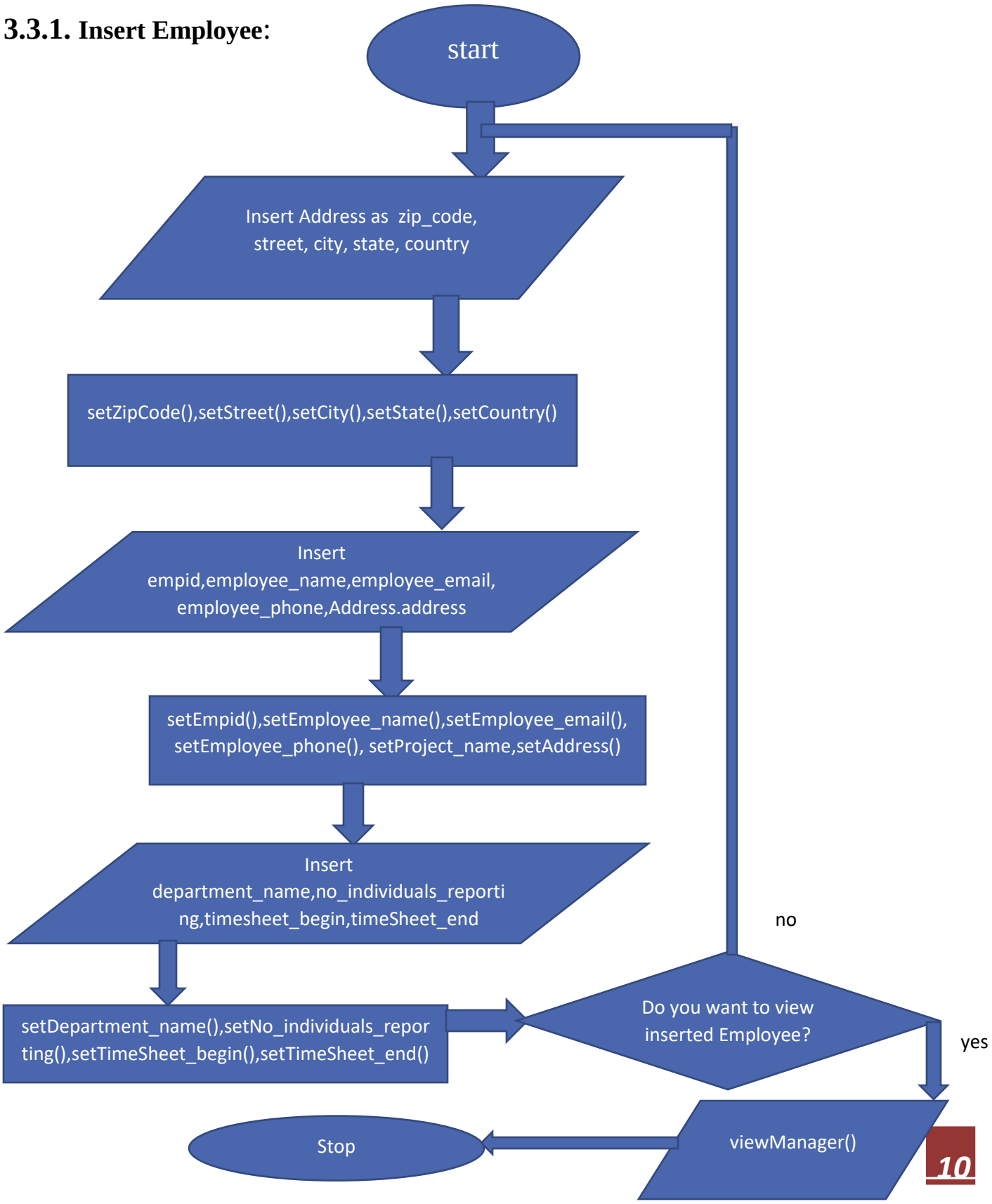
3.2. CLASS DIAGRAM: It is used as a mapping to design systems in Object Oriented languages. It is a static representation of each class, interface, association and constraint involvedin designing the system.



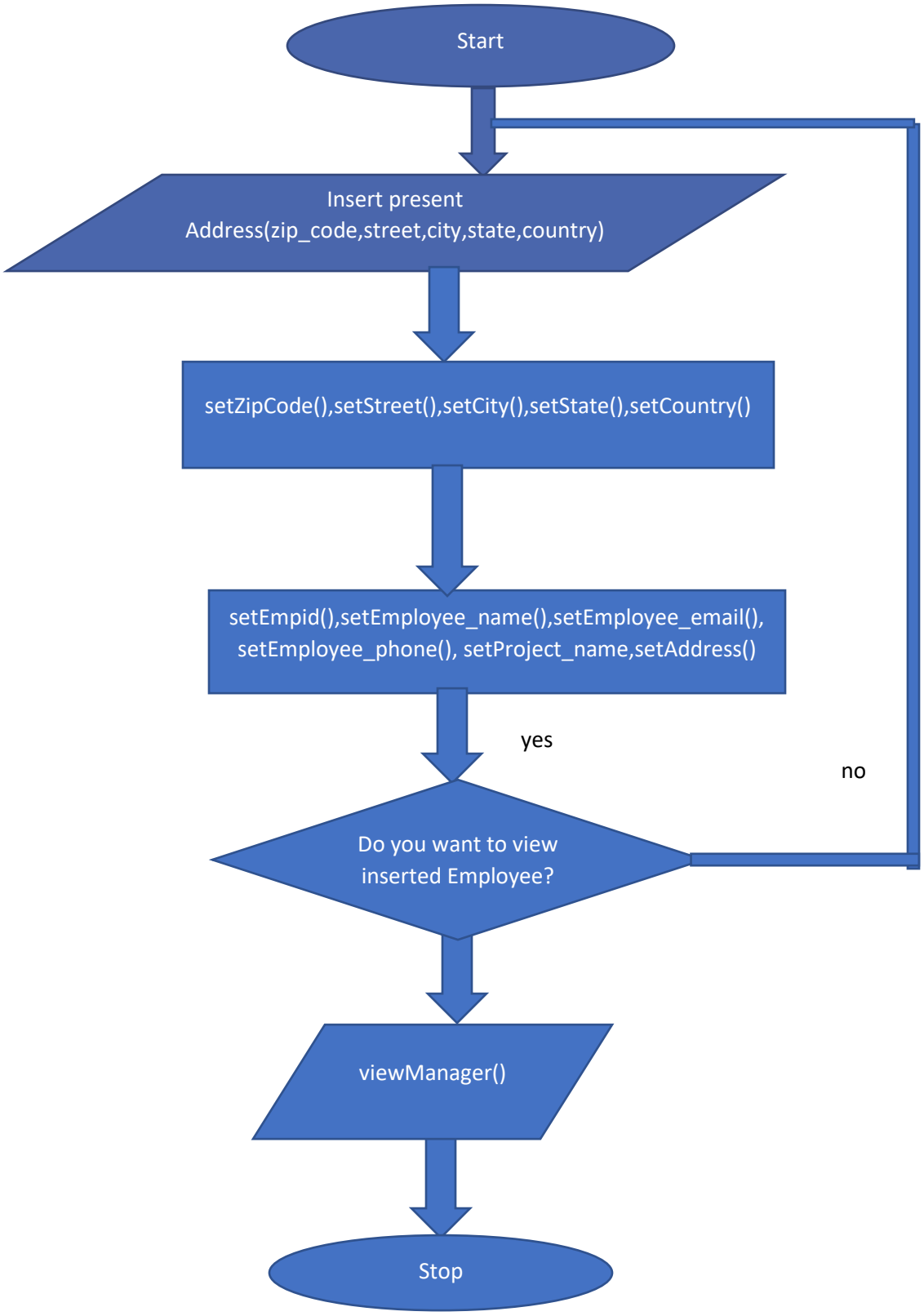
3.3. FLOW CHART

[Modular Representation To denote logical workflow]

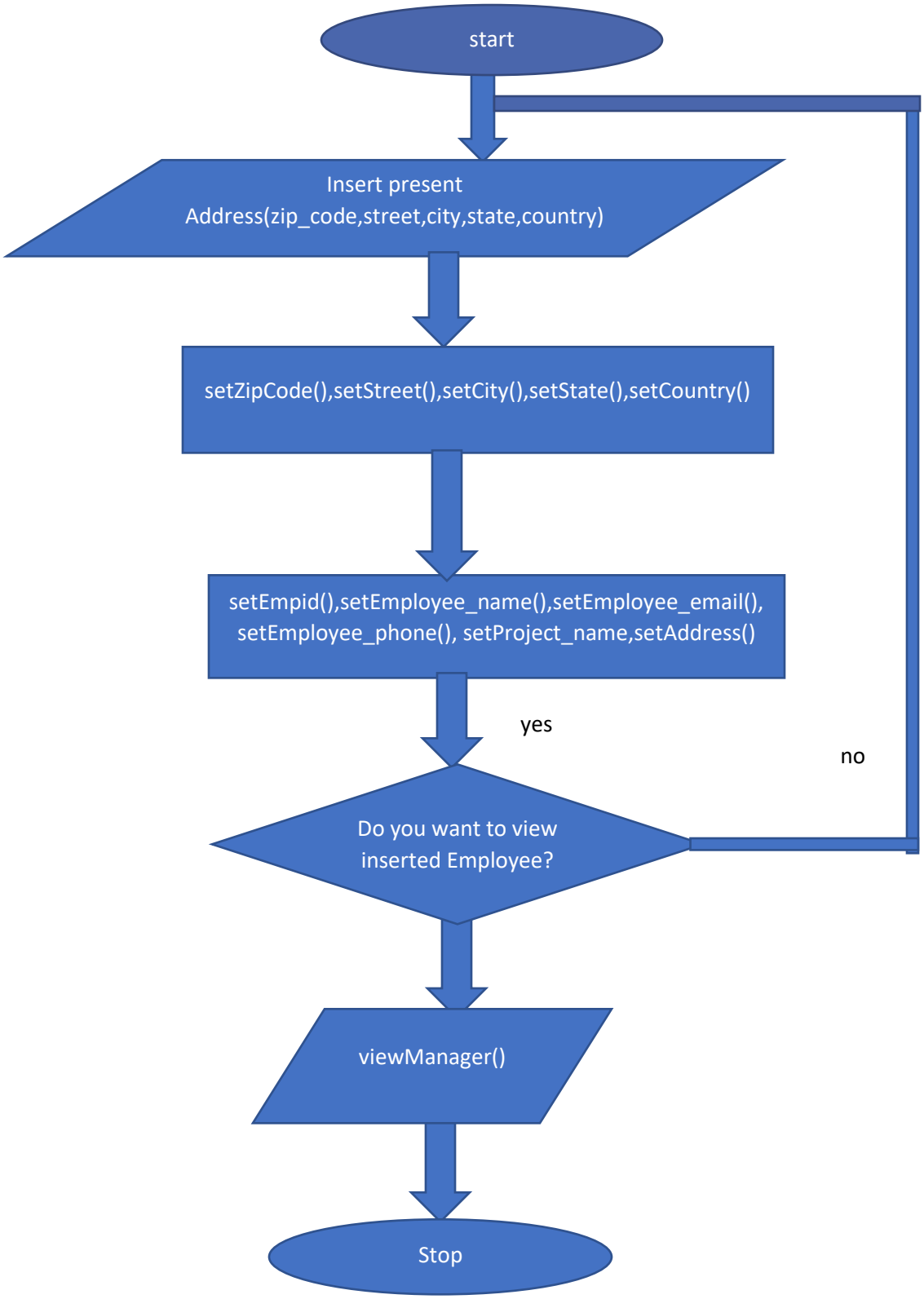
3.3.1. Insert Employee:



3.3.2. Update Employee [Address]



3.3.3. Delete Employee [Remove Employee from project]



4. KEY PERFORMANCE INDEX

The key performances expected from the system are:

- The system is primarily designed for entry of employee details involved in each project.
- The java programming characteristics of Inheritance and Association are implemented in the code.
- It is a console-based project, so no database or storage of any kind is involved.
- It is coded in java so is portable and can work on any platform.
- It gives an instant check on insertions and updates.
- It is maintainable and can be easily connected with a database either with JDBC or using hibernate.
- Linked hash map helps to track employees on the basis of empid and employee_ name for a quick view.
- An employee array helps to insert and view several employees involved in the project.
- All fields of the employee class and the address class are accessed by the Manager class object which inherits the Employee class.
-

5. TEST CASES:

Test case	Steps to perform test case	Module	Pass/Fail
1	Enter zip_ code, street, city, state, country	Address class	Success [80%]
2.	Enter empid, employee_ name, employee_ email, employee_ phone	Employee class	Success [80%]
3.	Address object received by constructor injection. Address with Composition [Association with Employee class]	Address object sets all address fields	Success [90%]
4.	Enter department_ name, no_ individuals reporting	Manager class	Success [90%]
5.	Enter timesheet_ begin, timesheet_ end in Manager class	Manager class	Success [90%]
6.	Accessing all fields of Employee class in Manager class object as Manager class inherits Employee class	Manager class	Success [90%]
7.	Entering several employees as an array in Employee class	Employee class object	Success [70%]

6. EXCEPTIONAL SCENARIOS

MODULES	EXCEPTIONS	OCCURANCE	SOLUTION
Option to insert in TestApp	• Number format exception • NullPointerException	⇨ For not parsing to Integer those variables declared as Integer ⇨ Not being able to access fields not declared public or static in other classes and packages	Integer. parseInt(var) Changing fields to public or static
Accessing the employee object before Address object	NullPointerException	➔ Fields accepted showing null in console	Accessing the dependent object of Address in the beginning
Employee array declared before accepting number of individuals in the project	NullPointerException	=>e[i] array showing null as empid is null	Declaring e[] after accepting the number of individuals reporting[as it is the size of the given array]