# EXPERIMENT 10

**Aim :**

To prepare Class Diagram for the Application Software – **Bill Management System.**

**Theory :**

**What is Class Diagram ?**

Class Diagrams are a type of structure diagram because they describe what must be present in the system being modelled. The various components in a class diagram can represent the classes that will actually be programmed, the main objects, or the interactions between classes and objects.

The class shape itself consists of a rectangle with three rows. The top row contains the name of the class, the middle row contains the attributes of the class, and the bottom section expresses the methods or operations that the class may use. Classes and subclasses are grouped together to show the static relationship between each object.

**Benefits of Class Diagram :**

- Illustrate data models for information systems, no matter how simple or complex.
- Better understand the general overview of the schematics of an application.
- Visually express any specific needs of a system and disseminate that information throughout the business.
- Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure.
- Provide an implementation – independent description of types used in a system that are later passed between its components.

**What are the Components of Class Diagram ?**

- **Upper Section :** Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.
- **Middle Section :** Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.
- **Bottom Section :** Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

**Member Access Modifiers :**

| | |
|---|---|
| Public | **( + )** |
| Private | **( - )** |
| Protected | **( # )** |
| Package | **( ~ )** |
| Derived | **( / )** |
| Static | **( underlined )** |

**Member Scopes :**

There are two scopes for members : Classifiers and Instances. Classifiers are static members while Instances are the specific instances of the class.

**Additional Class Diagram Components :**

- **Classes :** A template for creating objects and implementing behaviour in a system. A class represents an object or a set of objects that share a common structure and behaviour. They're represented by a rectangle that includes rows of the class name, its attributes, and its operations.

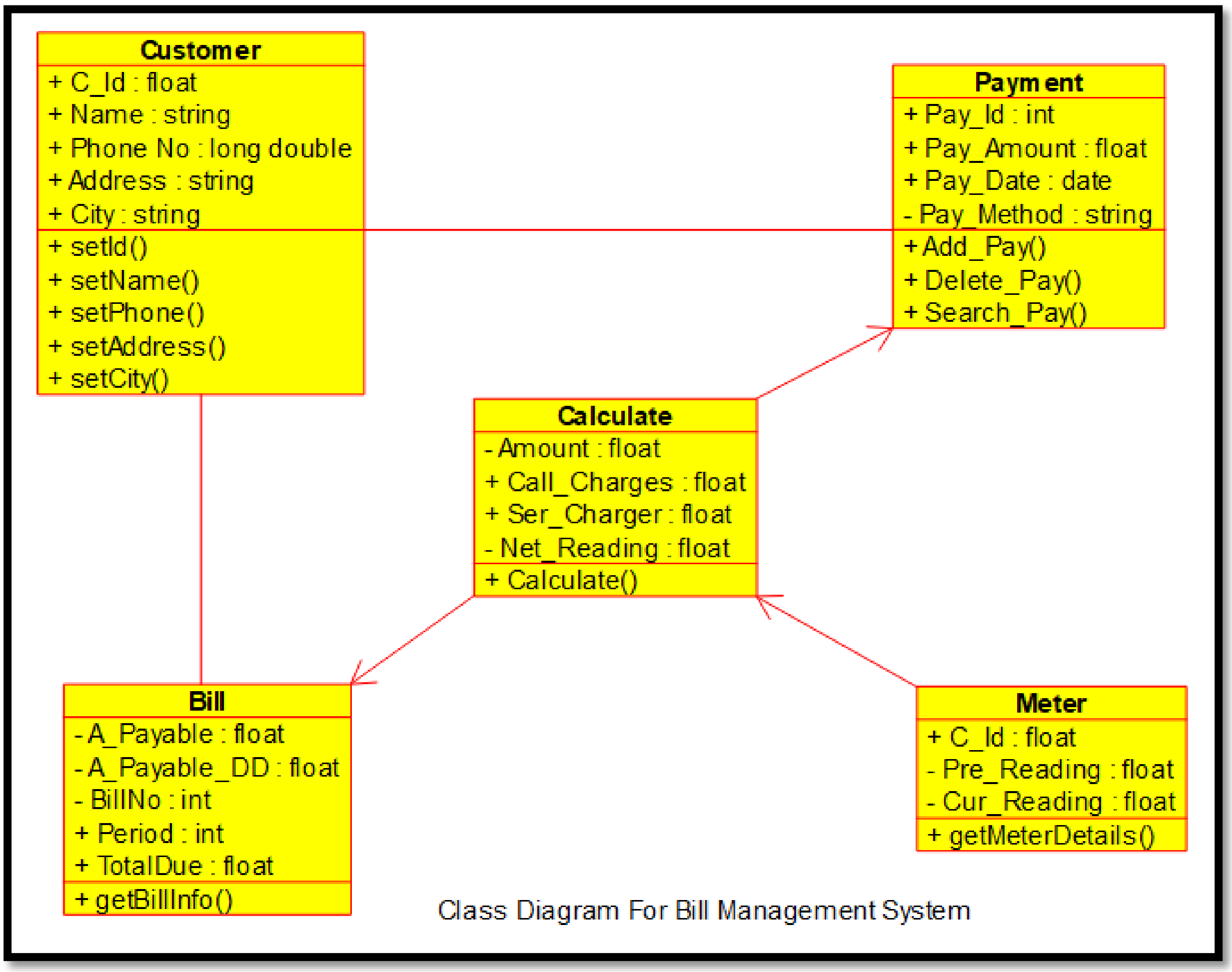
    - **Name :** The first row in a class shape.
    - **Attributes :** The second row in a class shape. Each attribute of the class is displayed on a separate line.
    - **Methods :** The third row in a class shape. Also known as operations, methods are displayed in list format with each operation on its own line.

- **Signals :** Symbols that represent one-way, asynchronous communications between active objects.
- **Data Types :** Classifiers that define data values. Data types can model both primitive types and enumerations.
- **Packages :** Shapes designed to organize related classifiers in a diagram. They are symbolized with a large tabbed rectangle shape.
- **Interfaces :** A collection of operation signatures and/or attribute definitions that define a cohesive set of behaviours.
- **Enumerations :** Representations of user – defined data types. An enumeration includes groups of identifiers that represent values of the enumeration.
- **Objects :** Instances of a class or classes. Objects can be added to a class diagram to represent either concrete or prototypical instances.
- **Artefacts :** Model elements that represent the concrete entities in a software system, such as documents, databases, executable files, software components, etc.

**Interactions :**

- **Inheritance :** The process of a child or sub – class taking on the functionality of a parent or superclass, also known as generalization. It's symbolized with a straight connected line with a closed arrowhead pointing towards the superclass.
- **Bidirectional Association :** The default relationship between two classes. Both classes are aware of each other and their relationship with the other. This association is represented by a straight line between two classes.
- **Unidirectional Association :** A slightly less common relationship between two classes. One class is aware of the other and interacts with it. Unidirectional association is modelled with a straight connecting line that points an open arrowhead from the knowing class to the known class.

Class Diagram for the Application Software – **Bill Management System** :



**Conclusion** :

The Class Diagram for the Application Software – **Bill Management System** was prepared successfully.