S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 4

Aim: Analyze the Domain classes and design class diagram for the given problem statement.

Name of Student	
Roll No	
Semester/Year	
Academic Session	
Date of Performance	
Date of Submission	

AIM: Analyze the Domain classes and design class diagram for the given problem statement.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

- Understand the concept of domain classes
- Identify a list of potential domain classes from a given problem statement.
- Graphically represent a class, and associations among different classes
- Identify the logical sequence of activities undergoing in a system, and represent them pictorially

HARDWARE AND SOFTWARE REQUIRMENTS:

Hardware Requirement

- Processor: Dual Core
- RAM: 1GB
- Hard Disk Drive: > 80 GB

Software Requirement

- Operating System – Windows
- StarUML

THEORY

Same types of objects are typically implemented by class in object-oriented programming. As the structural unit of the system can be represented through the classes, so, it is very important to identify the classes before start implementing all the logical flows of the system. In this experiment we will learn how to identify the classes from a given problem statement.

Domain Class

In Object Oriented paradigm Domain Object Model has become subject of interest for its excellent problem comprehending capabilities towards the goal of designing a good software system. Domain Model, as a conceptual model gives proper understanding of problem description through its highly effective component – the Domain Classes. Domain classes are the abstraction of key entities, concepts or ideas presented in the problem statement. As stated in, domain classes are used for representing business activities during the analysis phase.

Below we discuss some techniques that can be used to identify the domain classes.

Traditional Techniques for Identification of Classes

Grammatical Approach Using Nouns

This object identification technique was proposed by Russell J. Abbot, and Grady Booch made the technique popular. This technique involves grammatical analysis of the problem statement to identify list of potential classes. The logical steps are:

1. Obtain the user requirements (problem statement) as a simple, descriptive English text. This basically corresponds to the use-case diagram for the problem statement.
2. Identify and mark the nouns, pronouns and noun phrases from the above problem statements
3. List of potential classes is obtained based on the category of the nouns (details given later). For example, nouns that direct refer to any person, place, or entity in general, correspond to different objects. And so does singular proper nouns. On the other hand, plural nouns and common nouns are candidates that usually map into classes.

Advantages

This is one of the simplest approaches that could be easily understood and applied by a larger section of the user base. The problem statement does not necessarily be in English, but in any other language.

Disadvantages

The problem statement always may not help towards correct identification of a class. At times it could give us redundant classes. At times the problem statement may use abbreviations for large systems or concepts, and therefore, the identified class may actually point to an aggregate of classes. In other words, it may not find all the objects.

Using Generalization

In this approach, all potential objects are classified into different groups based on some common behaviour. Classes are derived from these groups.

Using Subclasses

Here, instead of identifying objects one goes for identification of classes based on some similar characteristics. These are the specialized classes. Common characteristics are taken from them to form the higher-level generalized classes.

Steps to Identify Domain Classes from Problem Statement

We now present the steps to identify domain classes from a given problem statement. This approach is mostly based on the “Grammatical approach using nouns” discussed above, with some insights from.

1. Make a list of potential objects by finding out the nouns and noun phrases from narrative problem statement
2. Apply subject matter expertise (or domain knowledge) to identify additional classes
3. Filter out the redundant or irrelevant classes
4. Classify all potential objects based on categories. We follow the category table as described by Ross.

Categories	**Explanation**
People	Humans who carry out some function
Places	Areas set aside for people or things
Things	Physical objects
Organizations	Collection of people, resources, facilities and capabilities having a defined mission
Concepts	Principles or Ideas not tangible
Events	Things that happen (usually at a given date and time), or as a steps in an ordered sequence

5. Group the objects based on similar attributes. While grouping we should remember:

- ✓ Different nouns (or noun phrases) can actually refer to the same thing (examples: house, home, abode)
- ✓ Same nouns (or noun phrases) could refer to different things or concepts (example: I go to school every day / This school of thought agrees with the theory)

6. Give related names to each group to generate the final list of top-level classes
7. Iterate over to refine the list of classes

Class

Classes are the structural units in object-oriented system design approach, so it is essential to know all the relationships that exist between the classes, in a system. All objects in a system are also interacting to each other by means of passing messages from one object to another. Sequence diagram shows these interactions with time ordering of the messages.
In this Experiment, we will learn about the representation of class diagram and sequence diagram. We also learn about different relationships that exist among the classes, in a system.
From the experiment of sequence diagram, we will learn about different types of messages passing in between the objects and time ordering of those messages, in a system.

Structural and Behavioral aspects

Developing a software system in object-oriented approach is very much dependent on understanding the problem. Some aspects and the respective models are used to describe problems and in context of those aspects the respective models give a clear idea regarding the problem to a designer. For developer, structural and behavioral aspects are two key aspects to see through a problem to design a solution for the same.

Class diagram

It is a graphical representation for describing a system in context of its static construction.

Elements in class diagram

Class diagram contains the system classes with its data members, operations and relationships between classes.

Class

A set of objects containing similar data members and member functions is described by a class. In UML syntax, class is identified by solid outline rectangle with three compartments which contain

- **Class name**

A class is uniquely identified in a system by its name. A textual string is taken as class name. It lies in the first compartment in class rectangle.

- **Attributes**

Property shared by all instances of a class. It lies in the second compartment in class rectangle.

- **Operations**

An execution of an action can be performed for any object of a class. It lies in the last compartment in class rectangle.

Example

To build a structural model for an Educational Organization, 'Course' can be treated as a class which contains attributes 'courseName' & 'courseID' with the operations 'addCourse()' & 'removeCourse()' allowed to be performed for any object to that class.

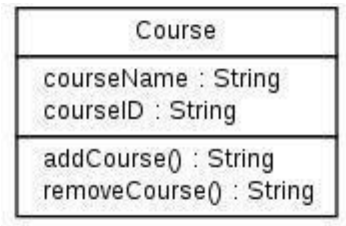


Figure-01:Class Diagram of Course

- **Generalization/Specialization**

It describes how one class is derived from another class. Derived class inherits the properties of its parent class.

Example

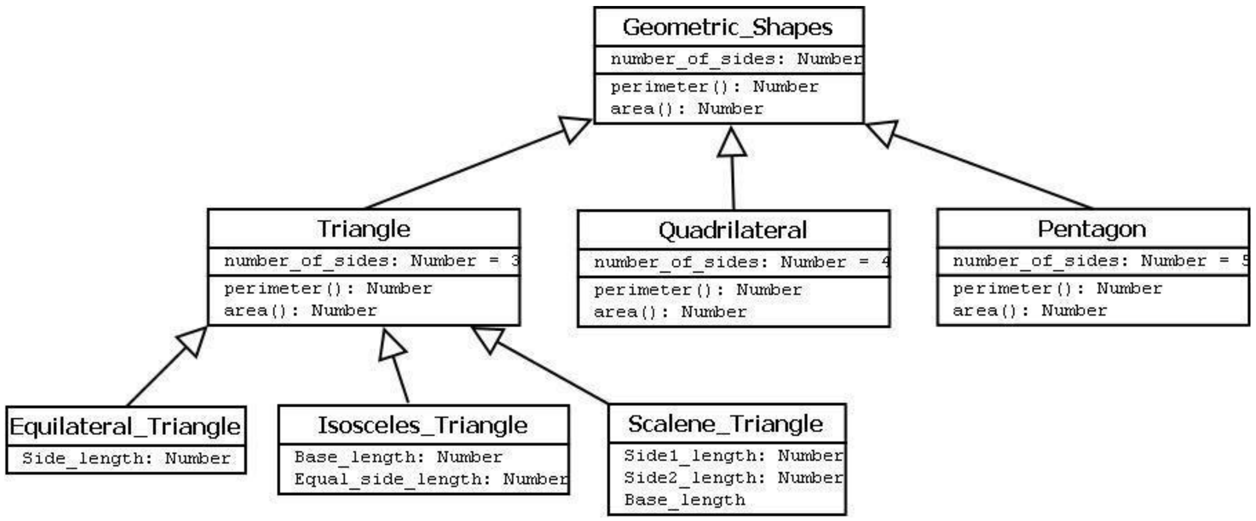


Figure-02: Relation between classes

Geometric_Shapes is the class that describes how many sides a particular shape has. Triangle, Quadrilateral and Pentagon are the classes that inherit the property of the Geometric_Shapes class. So, the relations among these classes are generalization. Now Equilateral_Triangle, Isosceles_Triangle and Scalene_Triangle, all these three classes inherit the properties of Triangle class as each one of them has three sides. So, these are specialization of Triangle class.

Relationships

Existing relationships in a system describe legitimate connections between the classes in that system.

- **Association**

It is an instance level relationship that allows exchanging messages among the objects of both ends of association. A simple straight line connecting two class boxes represent an association. We can give a name to association and also at the both end we may indicate role names and multiplicity of the adjacent classes. Association may be uni-directional.

Example
In structure model for a system of an organization an employee (instance of 'Employee' class) is always assigned to a particular department (instance of 'Department' class) and the association can be shown by a line connecting the respective classes.

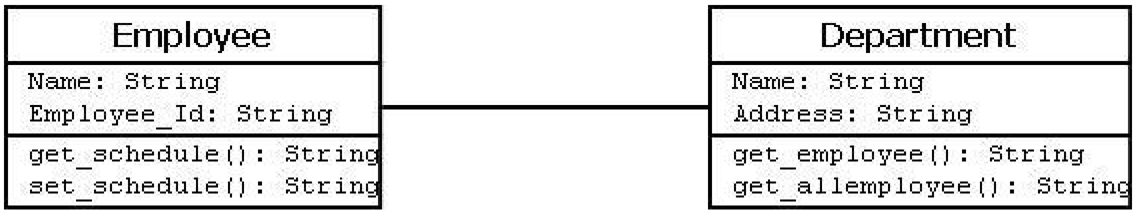


Figure-03: Association between classes

- **Aggregation**

It is a special form of association which describes a part-whole relationship between a pair of classes. It means, in a relationship, when a class holds some instances of related class, then that relationship can be designed as an aggregation.
Example
For a supermarket in a city, each branch runs some of the departments they have. So, the relation among the classes 'Branch' and 'Department' can be designed as an aggregation. In UML, it can be shown as in the fig. below

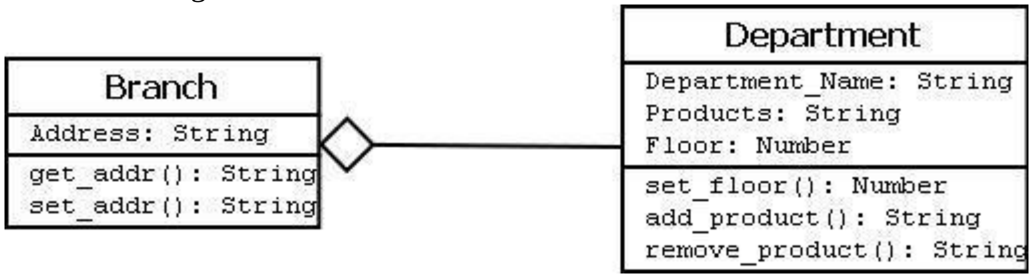


Figure-04: Aggregation between classes

- **Composition**

It is a strong from of aggregation which describes that whole is completely owns its part. Life cycle of the part depends on the whole.
Example
Let consider a shopping mall has several branches in different locations in a city. The existence of branches completely depends on the shopping mall as if it is not exist any branch of it will no

longer exists in the city. This relation can be described as composition and can be shown as below

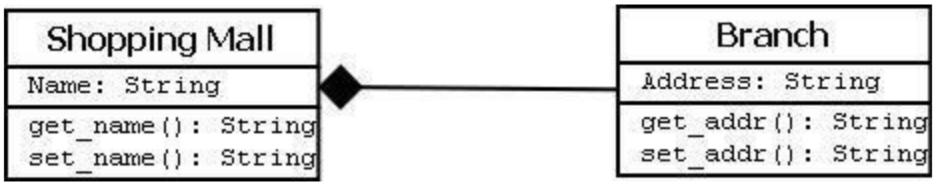
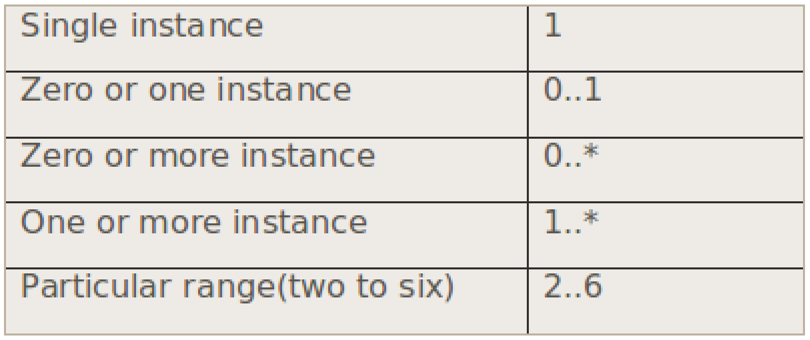


Figure-05:Composition between classes

- **Multiplicity**

It describes how many numbers of instances of one class is related to the number of instances of another class in an association.

Notation for different types of multiplicity:



Single instance	1
Zero or one instance	0..1
Zero or more instance	0..*
One or more instance	1..*
Particular range(two to six)	2..6

Figure-06: Different types of multiplicity

Example

One vehicle may have two or more wheels

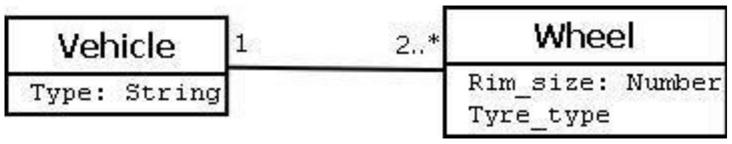


Figure-07: Multiplicity between classes

Sequence diagram

It represents the behavioral aspects of a system. Sequence diagram shows the interactions between the objects by means of passing messages from one object to another with respect to time in a system.

Elements in sequence diagram

Sequence diagram contains the objects of a system and their life-line bar and the messages passing between them

Object

Objects appear at the top portion of sequence diagram. Object is shown in a rectangle box. Name of object precedes a colon ‘:’ and the class name, from which the object is instantiated. The whole string is underlined and appears in a rectangle box. Also, we may use only class name or only instance name.

Objects which are created at the time of execution of use case and are involved in message passing, are appear in diagram, at the point of their creation.

Life-line bar

A down-ward vertical line from object-box is shown as the life-line of the object. A rectangle bar on life-line indicates that it is active at that point of time.

Messages

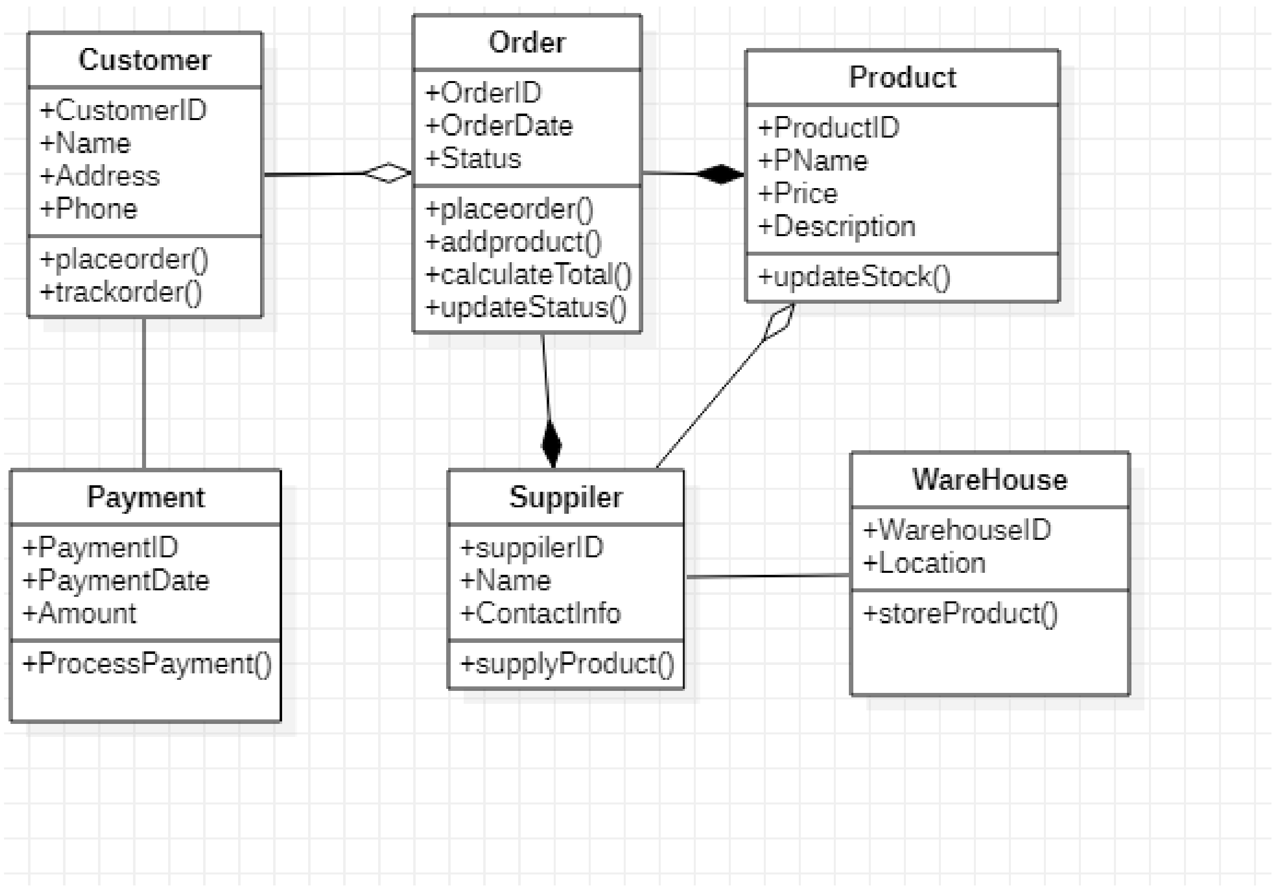
Messages are shown as an arrow from the life-line of sender object to the life-line of receiver object and labeled with the message name. Chronological order of the messages passing throughout the objects’ life-line show the sequence in which they occur. There may exist some different types of message:

- **Synchronous messages:** Receiver start processing the message after receiving it and sender needs to wait until it is made. A straight arrow with close and fill arrow-head from sender life-line bar to receiver end, represent a synchronous message.
- **Asynchronous messages:** For asynchronous message sender needs not to wait for the receiver to process the message. A function call that creates thread can be represented as an asynchronous message in sequence diagram. A straight arrow with open arrow-head from sender life-line bar to receiver end, represent an asynchronous message.
- **Return message:** For a function call when we need to return a value to the object, from which it was called, then we use return message. But it is optional, and we are using it when we are going to model our system in much detail. A dashed arrow with open arrow-head from sender life-line bar to receiver end, represent that message.
- **Response message:** One object can send a message to self. We use this message when we need to show the interaction between the same object.

Message Type	**Notation**
Synchronous message	——▶
Asynchronous message	——>
Response message	◀- - - - -

Figure-08: Notation used in Sequence diagram

OBSERVATION:



CONCLUSION:

____

____

____

____

____

____

DISCUSSION QUESTIONS?

1. How do you identify the domain class of a problem statement?

   ____

   ____

   ____

   ____

   ____

   ____

   ____

2. What is the purpose of the domain class model?

   ____

   ____

   ____

   ____

   ____

   ____

   ____

3. What are problem domain classes?

   ____

   ____

   ____

   ____

4. How is a domain class different from a design class?

5. Why do we use domain?

6. What is the purpose of sequence diagram and class diagram?

7. What are the 3 main elements of a class diagram?

8. How does the UML model use sequence diagrams?

9. Which type of UML diagrams are used for process models?

10. What are the symbols used in sequence diagram?

REFERENCES:

- http://vlabs.iitkgp.ernet.in/se/1/
- https://sites.google.com/view/ait-se/Home/practicals
- https://www.javatpoint.com/software-requirement-specifications