Assignment NO·8

Name- Kudharan Sumit Dattatraya
Roll NO- 20U401
class- TE4
Subject -

Problem statement -

Consider online shopping system. Perform the following tasks & draw the class diagram using UML tool. Represent the individual classes & objects Add methods Represent relationships & other classifiers like interfaces.

Theory -

* Introduction of Class Diagram -

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing & documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes & operations of a class and also the constraints imposed on the system. The class diagrams are coidely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations & constraints. It is also known a structural diagram. The class diagram depicts a static view of an application It represents the types of objects residing in the system and the relationships between them.

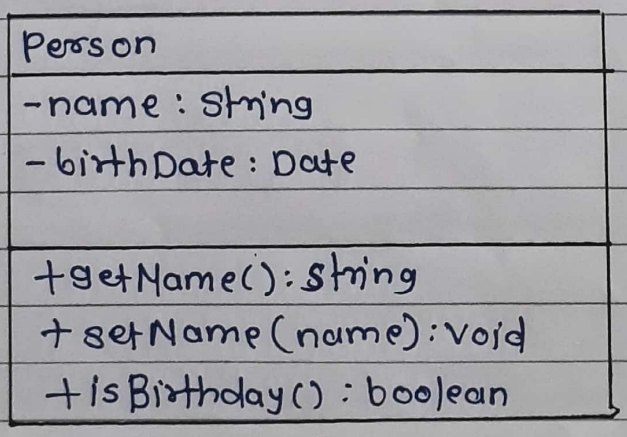
A class consists of its objects, & also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system & also construct ~~prompt~~ executable software code. It shows the

attributes, classes, functions & relationships to give an overview of the software system. It constitutes class names, attributes & functions in a sperate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations & constraints it is termed as a structural diagram.

The class diagram is the main building block of object oriented modelling. It is used for general conceptual modelling of the structure of the application, and for detailed modelling, translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application & the classes to be programmed.

In the diagram classes are represented with boxes that contain three components:

- The top compartment contains the name of the class. It is printed in bold & centered, and the first letter is capitalized.
- The middle compartment contains the attributes of the class. They are left-aligned & the first letter is lowercase.
- The bottom compartment contains the operations the class can execute. They are also left-aligned & the first letter is lowercase.



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.

All classes have different access levels depending on the access modifier (visibility). Here are the access levels with their corresponding symbols:

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

The following topics describe model elements in class diagram -

Classes - In UML, a class represents an object or a set of objects that share a common structure & behaviour. Classes, or instances of classes are common model elements in UML diagrams.

Objects - In UML models objects are model elements that represent instances of a class or of classes. You can add objects to your model to represent concrete & prototypical instances. A concrete instance represents an actual person or thing in the real world. For example a concrete instance of a customer class represents an actual customer. A prototypical instance of a customer class contains data that represents a typical customer.

Packages - Packages group related model elements that are independent of the classifiers of all types including other packages.

Signals - In UML models, signals are model elements that are independent of the classifiers that handle them. Signals specify one-way, asynchronous communications between active objects.

Enumerations - In UML models, enumerations are model elements in class diagrams that represent user-defined data types. Enumerations contains set of named identifiers that represent the values of the enumeration. These values are called enumeration literals.

Data Types - In UML diagrams data types are model elements that define data values. you typically use data types to represent primitive types, such as integer or string types & enumerations such as user defined data types.

Artifacts - In UML Models artifacts are model elements that represent the physical entities in a software system. Artifacts represent physical implementation units, such as executable files, libraries, software components, documents & databases.

Relationship in class diagram - In UML a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure & behaviour between model elements.

Qualifiers on association ends - In UML, qualifiers are properties of binary associations and are an optional part of association ends. A qualifier holds a list of association attributes each with a name & a type. Association attributes should model the keys that are used to index a subset of relationship instances.

* Relationships in class diagrams -

In UML a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure & behaviour between model elements.

UML relationship are grouped into the following categories:

Category	Function
Activity edges	Represents the flow between activities
Associations	Indicate the instances of one model element are connected to instances of another model element
Dependencies	Indicate that a change to one model element can affect another model element.
Generalizations	Indicate that one model element is a specialization of another model element.
Realizations	Indicate that one model element provides a specification that another model element implements.
Transitions	Represent changes in state.

Relationships in class diagram show the interaction between classes & classifiers. Such relationship indicate the classifiers that are associated with each other, those that are generalizations & realizations, & those that have dependencies on other classes & classifiers.

The following topics describe the relationships that you can use in class diagrams:

Abstraction relationships - An abstraction relationship is a dependency between model elements that represents the same concept at different levels of abstraction or from different viewpoints. You can add abstraction relationships to a model in several diagrams, including use-case, class & component diagrams.

Aggregation relationships - In UML models, an aggregation relationship shows a classified as a part of or subordinate to another classifier.

Association relationships – In UML models an association is a relationship between two classifiers such as classes or use cases that describes the reasons for the relationship & the rules that govern the relationship.

Association classes - In UML diagrams an association class is a class that is part of an association relationship between two other classes.

Binding relationships - In UML models, a binding relationship is a relationship that values to template parameters & generates a new model element from the template.

Composition association relationships - A composition association relationship represents a whole part relationship & is a form of aggregation. A composition association relationship specifies that the lifetime of the part classifier is dependent on the lifetime of the whole classifier.

Dependancy relationships - In UML, a dependancy relationship is a relationship in which one element, the client, uses or depends on another element, the supplier. You can use dependancy relationships in class diagrams, component diagrams, deployment diagrams & use-case diagrams to indicate that a change to the supplier might require a change to the client.

Directed association relationships - In UML models directed association relationships are associations that are navigable in only one direction.

Element import relationship - In UML diagrams an element import relationship identifies a model element in another package & allows the element in the other package to be referenced by using its name without a qualifier.

Generalization relationship - In UML modelling a generlization relationship is a relationship, in which one model element (the child) is based on another model element (the parent). Generalization relationships are used in class, component, deployment,

and use-case diagrams to indicate that the child receives all of the attributes, operations & relationships that are defined in the parent.

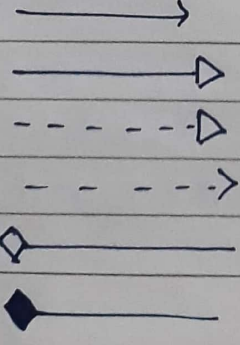
Interface realization relationships - In UML diagrams an interface realization relationship is a specialized type of implementation relationship between a classifier & a provided interface. The interface realization relationship specifies that the realizing classifier must conform to the contract that the provided interface specifies.

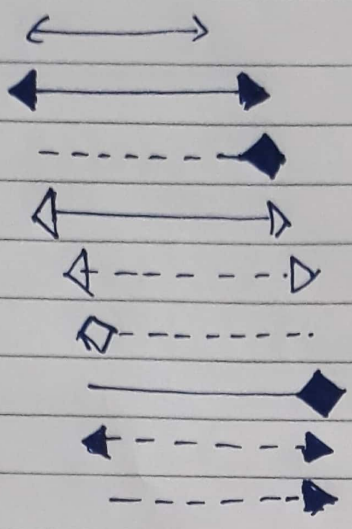
Instantiation relationships - In UML diagrams, an instantiation relationship is a type of usage dependency between classifiers that indicates that the operations in one classifier create instances of other classifier.

Package import relationship - In UML diagrams, a package import relationship allows other namespaces to use unqualified names to refer to package members.

Realization relationships - In UML modelling, a realization relationship is a relationship, between two model elements, in which one model element (the client) realizes the behaviour that the other model element (the supplier) specifies. Several clients can realize the behaviour of a single supplier. You can use realization relationships in class diagrams & Component diagrams.

Usage Relationships - In UML modelling a usage relationship is a type of dependancy relationship in which one model element (the client) requires another model element (the supplier) for full implementation or operation.





Redundant
Temporally Coupled
Replaces
Unrelated
Inexplicably Related
Estate Inheritance
Attorney-client Relationship
Bound by Blood
Father's Brother's Nephew's
Cousni's Former Roomate

* Purpose of class Diagrams-

The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object oriented languages. It is one of the most popular UML diagrams. Following are the purpose of class diagrams given below:

- It analyses & designs a static view of an application.
- It describes the major responsibilities of a system.
- It is a base of component & deployment diagrams.
- It incorporates forward & reverse engineering.

* purpose of class Diagrams-

The main purpose of class diagram, is to build a static view of an application. It is the only diagram that is widely used for construction & it can be mapped with object oriented languages. It is one of the most popular UML diagrams. ~~Following are~~

* Class Diagram in software Development Lifecycle-

class diagrams can be used in various software development phases. It helps in modelling class diagrams in three different perspectives.

① Conceptual Perspective -

Conceptual diagrams are describing things in the real world. you should draw a diagram that represents the concepts in the domain under study. These concepts related to class & it is always language - independent.

② Specification perspective -

Specification perspective describes software abstractions or components with specifications & interfaces. However, it does not give any commitment to specific implementation.

③ Implementation perspective -

This type of class diagram is used for implementations in a specific language or application. Implementation perspective, use for software implementation.

Here, are some points which should be kept in mind while drawing a class diagram:

- The name given to the class diagram must be meaningful. Moreover it should describe the real aspect of the system.
- The relationship between each element needs to be identified in advance.
- The responsibility for every class needs to be identified.
- For every class, minimum number of properties should be specified. Therefore, unwanted properties can easily make the diagram complicated.

* Advantages of class diagram -

- It can represent the object model for complex systems.
- It reduces the maintaince time by providing an overview of how an application is structured before coding.
- It provides a general schematic of an application for better understanding.
- It is helpful for the stakeholders & the developers.

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

* Disadvantages of class Diagram -

- The class diagrams might often take a longer time to manage & maintain which is sometimes annoying for a developer.
- It requires time for the synchronization with the software code to set it up & maintain.
- A lack of clarity in understanding the beneficiary of the diagram is also a disadvantage
- An overcomplicated or overwhelming diagram doesn't help software developers in their work.
- Mapping out every single scenario could make the diagram messy & hard to work with.

Conclusion -

To design & visualize the software system artifacts the standard language used is the UML. The relationship between the different objects is described by the class diagram which ensures the design & analysis of an application & views it in its static form. Being the most important UML diagram the class diagram consists of class, attributes & relationships which are its essential elements. To get an idea of the application structure, the class diagram is used, which helps in reducing the maintainee time.