UML CLASS DIAGRAM

- Class diagrams are created to provide a picture or view of some or all of the classes in the model.
- The main class diagram in the logical view of the model is typically a picture of the packages in the system. Each package also has its own main class diagram, which typically displays the “public” classes of the package.

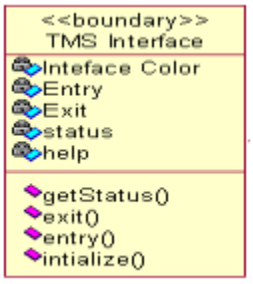
A class diagram is a picture for describing generic descriptions of possible systems. Class diagrams and collaboration diagrams are alternate representations of object models. Class diagrams contain icons representing classes, packages, interfaces, and their relationships. You can create one or more class diagrams to depict the classes at the top level of the current model; such class diagrams are themselves contained by the top level of the current model.

Class:-

A class is a description of a group of objects with common properties (attributes), common behavior (operations), common relationships to other objects, and common semantics. Thus, a class is a template to create objects. Each object is an instance of some class and objects cannot be instances of more than one class. Classes should be named using the vocabulary of the domain. For example, the Bus class may be defined with the following characteristics: Attributes - location, time offered Operations - retrieve location, retrieve time of day, add a student to the offering. Each object would have a value for the attributes and access to the operations specified by the Airline database class.

UML Representation:

- In the UML, classes are represented as compartmentalized rectangles.
- The top compartment contains the name of the class.
- The middle compartment contains the structure of the class (attributes).
- The bottom compartment contains the behavior of the class as shown below.



Analysis Class Stereotypes:

Analysis class stereotypes represent three particular kinds of class that will be encountered again and again when carrying out requirements modeling. UML DEFINITION:

Stereotype:

- A new type of modeling element that extends the semantics of the met model.
- Stereotypes must be based on certain existing types or classes in the met model.
- Stereotypes may extend the semantics but not the structure of preexisting classes.
- Certain stereotypes are defined in the UML, others may be user-defined.

- UML is designed to be capable of extension; developers can add new stereotypes depending on need. But this is only done when it is absolutely necessary. Three analysis class stereotypes of the UML are:
  - Boundary classes
  - Control classes
  - Entity classes.

1)Boundary Classes:-

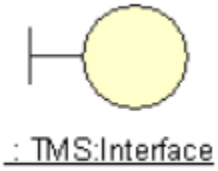
Boundary classes, it is a 'model interaction between the system and its actors'. Since they are part of the requirements model, boundary classes are relatively abstract. They do not directly represent all the different sorts of interfaces that will be used in the implementation language. The design model may well do this later, but from an analysis perspective, we are interested only in identifying the main logical interfaces with users and other systems.

This may include interfaces with other software and also with physical devices such as printers, motors, and sensors. Stereotyping these as boundary classes emphasizes that their main task is to manage the transfer of information across system boundaries. It also helps to partition the system, so that any changes to the interface or communication aspects of the system can be isolated from those parts of the system that provide the information storage.

The class TMS Interface is a typical boundary class. This style of writing the name shows that the class is TMS Interface and it belongs to the User Interface package when we write the package name in this way before the class name, it means that this class is imported from a different package from the one with which we are currently working. In this case, the current package is the Agate application package, which contains the application requirements model, and thus consists only of domain objects and classes. Alternative notations for Boundary class stereotypes can be represented as shown below

A) With the stereotype

B) Symbol



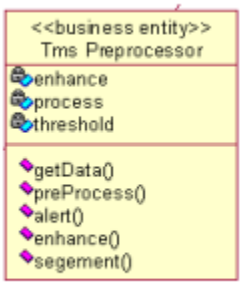
2)Entity classes:-

Entity classes represent something within the application domain, but external to the software system, about which the system must store some information. Instances of an entity class will often require persistent storage of information about the things that they represent. This can sometimes help to decide whether an entity class is the appropriate modeling construct.

For example, an actor is often not represented as an entity class. This is in spite of the fact that all actors are within the application domain, external to the software system, and important to its operation. But most systems have no need to store information about their users or to model their behavior. While there are some obvious exceptions to this (consider a system that monitors user access for security purposes), these are typically separate, specialist applications in their own right. In such a context, an actor would be modeled appropriately as an entity class, since the essential requirements for such a system would include storing information about users, monitoring their access to computer systems, and tracking their actions while logged into a system. But it is more commonly the case that the software we develop does not need to know anything about the people that use it, and so actors are not normally modeled as classes.

The following are representations for Entity classes

a) With stereotype



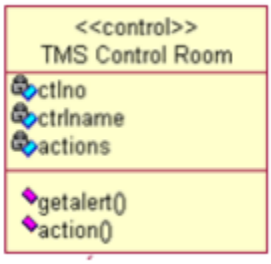
b) Symbol

3. Control classes

The third of the analysis class stereotypes is the control class, given by y the class Searching system in Search flight. Control classes 'represent coordination, sequencing, transactions, and control of other objects'. In the USDP, as in the earlier methodology Objector. It is generally recommended that there should be a control class for each use case.

In a sense, then, the control class represents the calculation and scheduling aspects of the logic of the use case, at any rate, those parts that are not specific to the behavior of a particular entity class, and that are specific to the use case. Meanwhile, the boundary class represents interaction with the user and the entity classes represent the behavior of things in the application domain and the storage of information that is directly associated with those things. The following are the notations that can be used to represent the Control class.

A) With Stereotype



B) Symbol

CLASS DIAGRAM FOR ONLINE VOTING SYSTEM