Assignment No. 4

Title:- Design and Implementation of Analysis model – class model

PROBLEM STATEMENT/DEFINITION :-

- Prepare Analysis Model-Class Model
- Identify Analysis Classes and assign responsibilities.
- Prepare Data Dictionary.
- Draw Analysis class Model using UML2.0 Notations.
- Implement Analysis class Model-class diagram witha suitable object oriented language.

OBJECTIVE :-

- To Identify Analysis Classes and assign responsibilities.
- To Draw Analysis class Model
- To Implement Analysis class Model-class diagram

RELEVANT THEORY :-

Class Diagram

The Class diagram shows the building blocks of any object- orientated system. Class diagrams depict the static view of the model or part of the model, describing what attributes and behaviors it has rather that detailing the methods for achieving operations.

Class diagrams are most useful to illustrate relationships between classes and interfaces. Generalizations, aggregations, and associations are all valuable in reflecting inheritance, composition or usage, and connections, respectively.

Classes

A class is an element that defines the attributes and behaviors that an object is able to generate. The behavior is the described by the possible messages the class is able to understand along with operations that are appropriate for each message. Classes may also contain definitions of constraints tagged values and stereotypes.

Class Notation

Classes are represented by rectangles which show the name of the class and optionally the name of the operations and attributes. Compartments are used to divide the class name, attributes and operations. Additionally constraints, initial values and parameters may be assigned to classes. Classes are composed of three things: a name, attributes, and operations.

Interfaces

An interface is a specification of behavior that implementers agree to meet. It is a contract. By realizing an interface, classes are guaranteed to support a required behavior, which allows the system to treat non-related elements in the same way –i.e. through the common interface. Interfaces may be drawn in a similar style to a class, with operations specified, as shown below. They may also be drawn as a circle with no explicit operations detailed. When drawn as a circle, realization links to the circle form of notation are drawn without target arrows.

Associations

An association implies two model elements have a relationship - usually implemented as an instance variable in one class. This connector may include named roles at each end, multiplicity or cardinality, direction and constraints. Association is the general relationship type between elements. For more than two elements, When code is generated for class diagrams, associations become instance variables in the target class.

Generalizations

A generalization is used to indicate inheritance. Drawn from the specific classifier to a general classifier, the generalize implication is that the source inherits the target's characteristics.

Aggregations

Aggregations are used to depict elements which are made up of smaller components. Aggregation relationships are shown by a white diamond-shaped arrowhead pointing towards the target or parent class.

A stronger form of aggregation -a composite aggregation-is shown by a black diamond-shaped arrowhead and is used where components can be included in

a maximum of one composition at a time. If the parent of a composite aggregation is deleted, usually all of its parts are deleted with it; however a part can be individually removed from a composition without having to delete the entire composition. Compositions are transitive, asymmetric relationships and can be recursive.

Association Classes

An association class is a construct that allows an association connection to have operations and attributes. The following example shows that there is more to allocating an employee to a project than making a simple association link between the two classes: the role that the employee takes up on the project is a complex entity in its own right and contains detail that does not belong in theemployee or project class.

Dependencies

A dependency is used to model a wide range of dependent relationships between model elements. It would normally be used early in the design process where it is known that there is some kind of link between two elements but it is too early to know exactly what the relationship is.

Later in the design process, dependencies will be stereotyped (stereotypes available include «instantiate», «trace», «import» and others) or replaced with a more specific type of connector.

Traces

The trace relationship is a specialization of a dependency, linking model elements or setsof elements that represent the same idea across models. Traces are often used to track requirements and model changes. As changes can occur in both directions, the order of this dependency is usually ignored. The relationship's properties can specify the trace mapping, but the trace is usually bi-directional, informal and rarely computable.

Realizations

The source objects implements or realizes the destination. Realize is used to express traceability and completeness in the model -a business process orrequirement is realized by one or more use cases which are in turn realized by some classes, which in turn are realized by a component, etc. Mapping requirements, classes, etc. across the design of your system, up through the

levels of modeling abstraction, ensures the big picture of your system remembers and reflects all the little pictures and details that constrain and define it. A realization is shown as a dashed line with a solid arrowhead and the «realize» stereotype.

Nestings

A nesting is connector that shows that the source element is nested within the target element. The following diagram shows the definition of an inner class although in EA it is more usual to show them by their position in the Project View hierarchy.

Conclusion

Thus we have thoroughly studied and implemented the analysis class model.