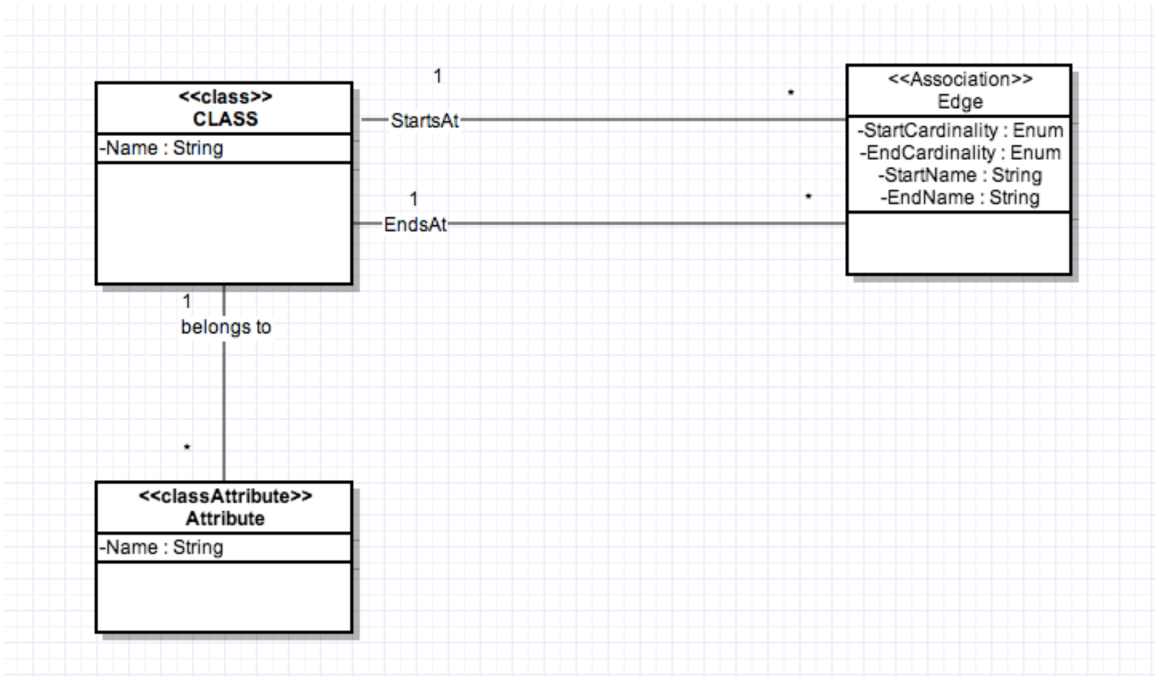
Homework – 3

Gaurav Nanda (gaurav324@gmail.com)

1. Metamodel for the above instances.



2. List of Constraints:
   - Name attribute of class “CLASS” and “ATTRIBUTE ” should be unique and not unique.

     ➢ Both instances have their names well defined and unique. i.e Classes have well defined names as A, B, C and D. Similarly attributes also have proper unique names.

   - At least one between the StartCardinality and EndCardinality should be greater than zero. i.e. We should not have an edge having zero cardinality on the both ends. One can argue that Edge can have (*, *) as their cardinalities, which is certainly fine. But defining an Edge as (0,0) does not make any sense, as you are diluting the purpose of defining edge itself.

     ➢ Each edge in above diagram has its cardinality as (1, *). So, we are good.

   - Attribute name has to be unique for each class. However, attribute name can be same for the different classes.

    - ➢ All the attributes defined inside all the classes are unique. Class A has only one attr1, Class B has –attr2. Class C and D don't even have any attributes. Therefore we are good.

- Start Node and the End Node of an edge must be different. i.e. Edge must not start and end at the same class. Though I am not sure whether this is always true (eg. Linked Lists can have start and end at same nodes), still listing this constraint, as I cannot infer this information from the given diagrams.

    - ➢ All the edges in the two class diagrams have edges starting and ending at different nodes.

3. table(Class, [ClassId, Name]).
   table(Edge, [EdgeId, StartsAt, StartName, StartCardinality, EndsAt, EndName, EndCardinality]).
   table(Attribute, [AttributeId, Name, belongsTo]).

   Instance-1

   Class (A#, A)
   Class (B#, B)

   Edge (E12#, A#, End1, 1, B#, End2, *)

   Attribute (Attr1#, attr1, A#)
   Attribute (Atrr2#, attr2, B#)

   Instance-2

   Class (C#, C)
   Class (D#, D)

   Edge (E56#, C#, End5, 1, D#, End6, *)
   Edge (E34#, C#, End3, 1, D#, End4, *)

Here, with the introduction of ClassId, EdgeId and AttributeId, we can enforce additional constraints:

- o Ids (ClassId, EdgeId and AttributeId) are unique and non-null.
- o We can also check that foreign keys constraints are also valid.
  - ➢ StartsAt and EndsAt attributes of edge must be primary keys of Class table. This would also check that StartsAt and EndsAt are not dangling and pointing to some valid node.
  - ➢ belongsTo attribute of Attribute must be primary key of Class table.

4. a) Metamodel defined in (1) certainly conforms to itself. If I try to draw a metamodel for (1), that is turning out to be the same.

   For instance, meta meta model would again have "Class", "Edge" and "Attribute" as three different classes. "Class" and "Attribute" would have just one attribute "name". "Edge" would have [startCardinality, endCardinality, startName, endName] which is just the same as the metamodel itself.

   b)
   Class(C1#, Class)
   Class(C2#, Attribute)
   Class(C3#, Edge)

   Edge(E1#, C1#, StartsAt, 1, C3#, null, *)
   Edge(E2#, C1#, EndsAt, 1, C3#, null, *)
   Edge(E3#, C1#, belongsTo, 1, C2#, null, *)

   Attribute(Attr1#, name, C1#)
   Attribute(Attr2#, name, C2#)
   Attribute(Attr3#, StartCardinality, C3#)
   Attribute(Attr4#, EndCardinality, C3#)
   Attribute(Attr5#, StartName, C3#)
   Attribute(Attr6#, EndName, C3#)

   c)
   1. All of the names are not null and are unique.
   2. Each edge is having (1, *) cardinality, which is greater than zero.
   3. (Attribute name, ClassId) combination is unique in each instance of Attribute table.
   4. Each of the "Edge" is starting and ending at different nodes. For example, "startsAt" starts at Class and ends at Edge.

   (Additional constraints)
   5. All ClassId, EdgeId and AttrId are not null and unique.
   6. (Validates foreign key) ClassIDs referred in the Edge table are valid entries present in Class table. Similarly, ClassIDs referred in the Attribute are valid entries present in the Class table.