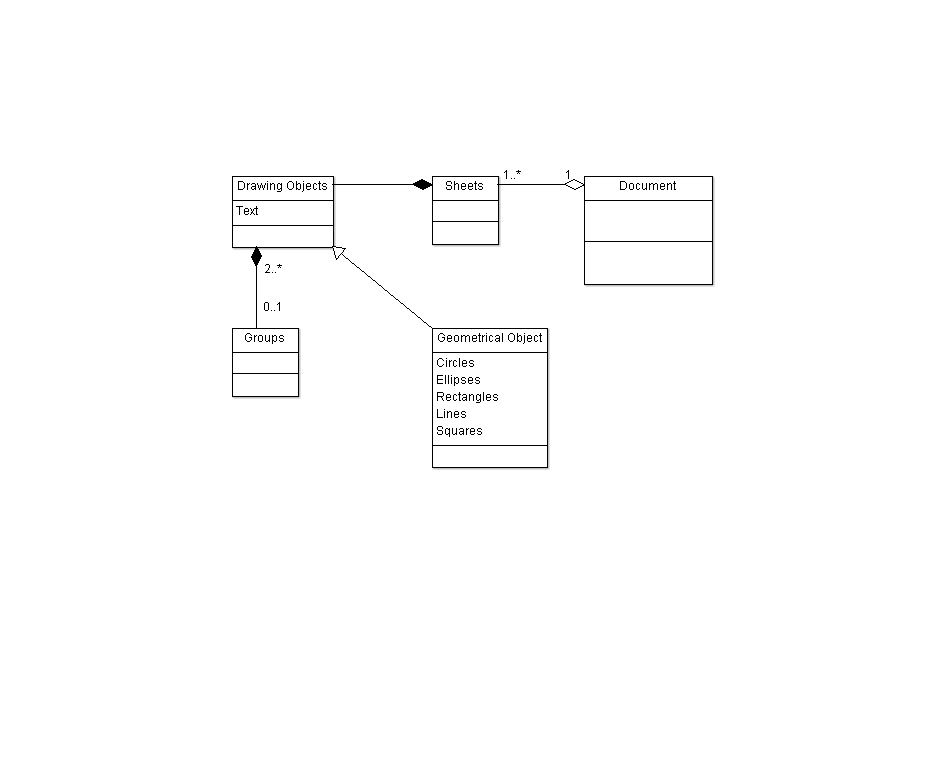
Due Date: 1/23/19

COMP-3700

HW 1

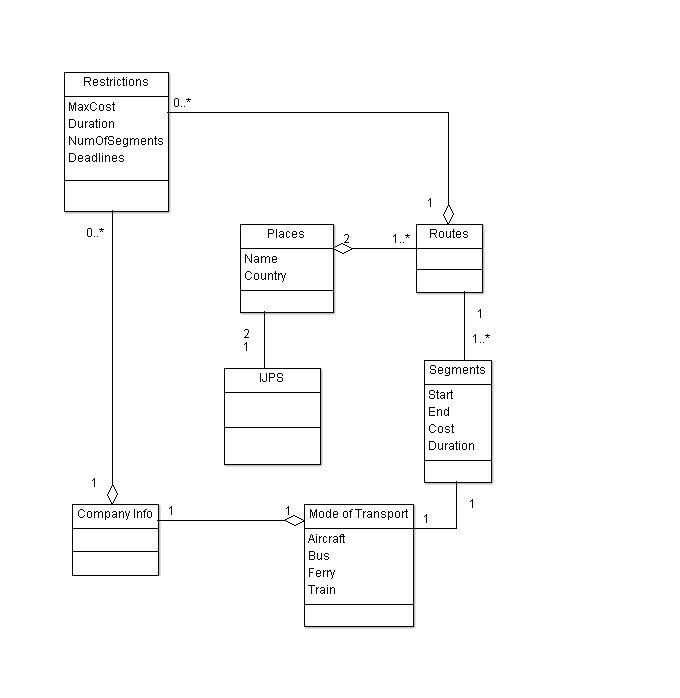
1. **Classical Categorization:** This is a type of categorization from the classical period in Greece. It was furthered by Aristotle in the use of establishing natural taxonomy (by using it to classify living beings), but the concept of grouping objects based on similar properties started with Plato. Categories should be clearly defined, mutually exclusive, and collectively exhaustive. Any entity of the given classification belongs to one (and only one) of the proposed categories. This is important for UML because it is the primary basis and original way of categorizing/classifying different objects, a method which we continually build upon today.

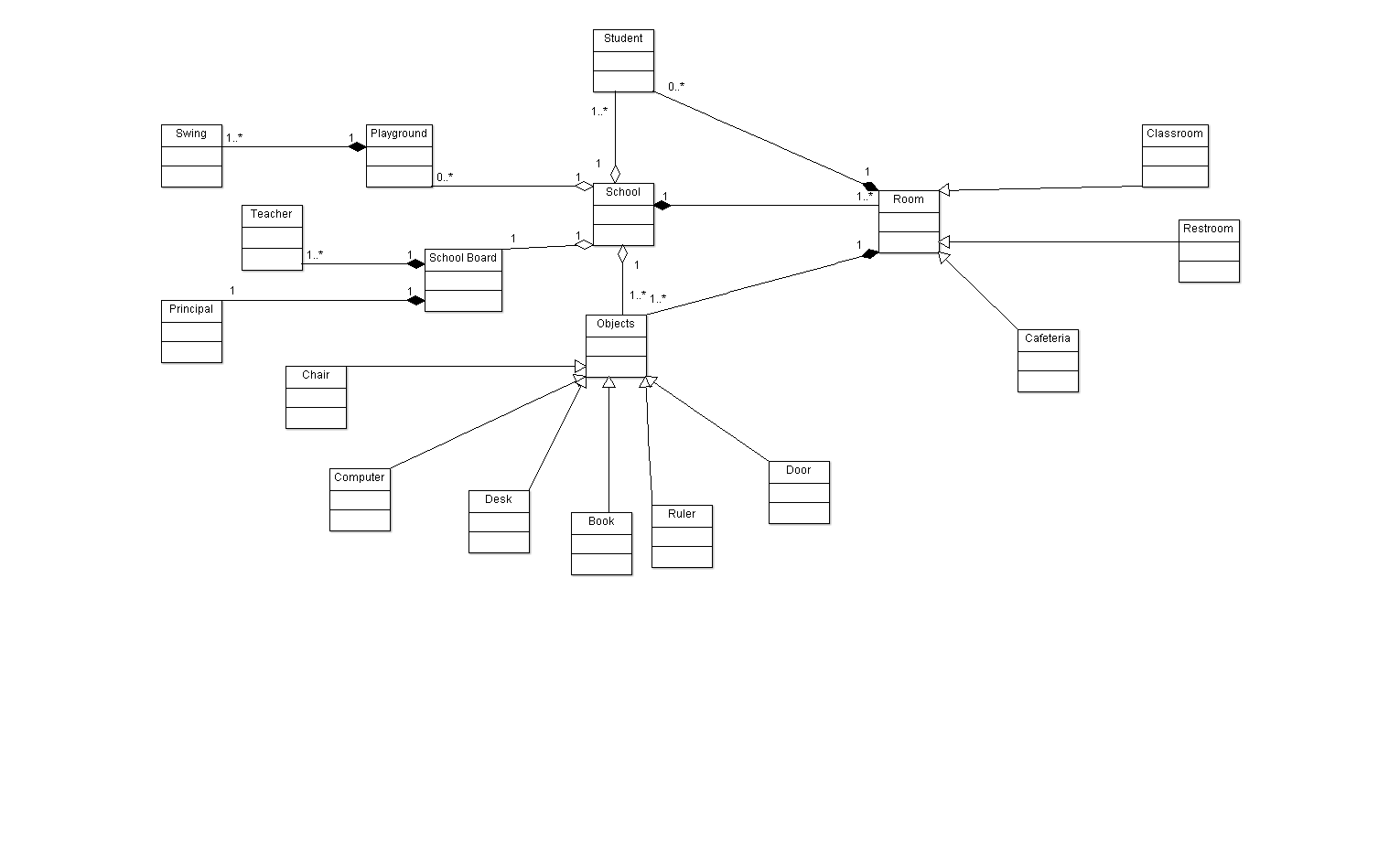
**Conceptual Clustering:** This is a modern variation of the classical approach that derives from attempts to explain how knowledge is represented. Concepts are generated by first formulating their conceptual descriptions and then classifying the entities by them. This brings up the idea of necessary and sufficient conditions. For something to be classified as the conceptual description, it is necessary for it to meet the conditions; but those conditions are not sufficient, because other objects can meet those conditions without being the conceptual description. Different clusters have different requirements, and objects have different levels of fitness for different clusters. This is important for UML because it is the way with which we add different objects to a class/cluster after it has already been formed.

**Prototype Theory:** A person has a “prototype” for what an object is naturally (i.e. some mental concept of an object based on previous experiences). The person would classify objects as being those objects based on how closely they match the preconceived mental concept (prototype). Prototype theory uses graded membership, so there are different levels of membership that are all on a hierarchy for the given object. The categories in the middle level are perceptually and conceptually the most salient. The lower-level hierarchy describes the hardest prototype to conceptualize while the higher-level hierarchy describes the simplest prototype to conceptualize. This is important for UML because it helps us to organize objects within a class while also deciding what the easiest to conceptualize object descriptions would be for our diagrams.

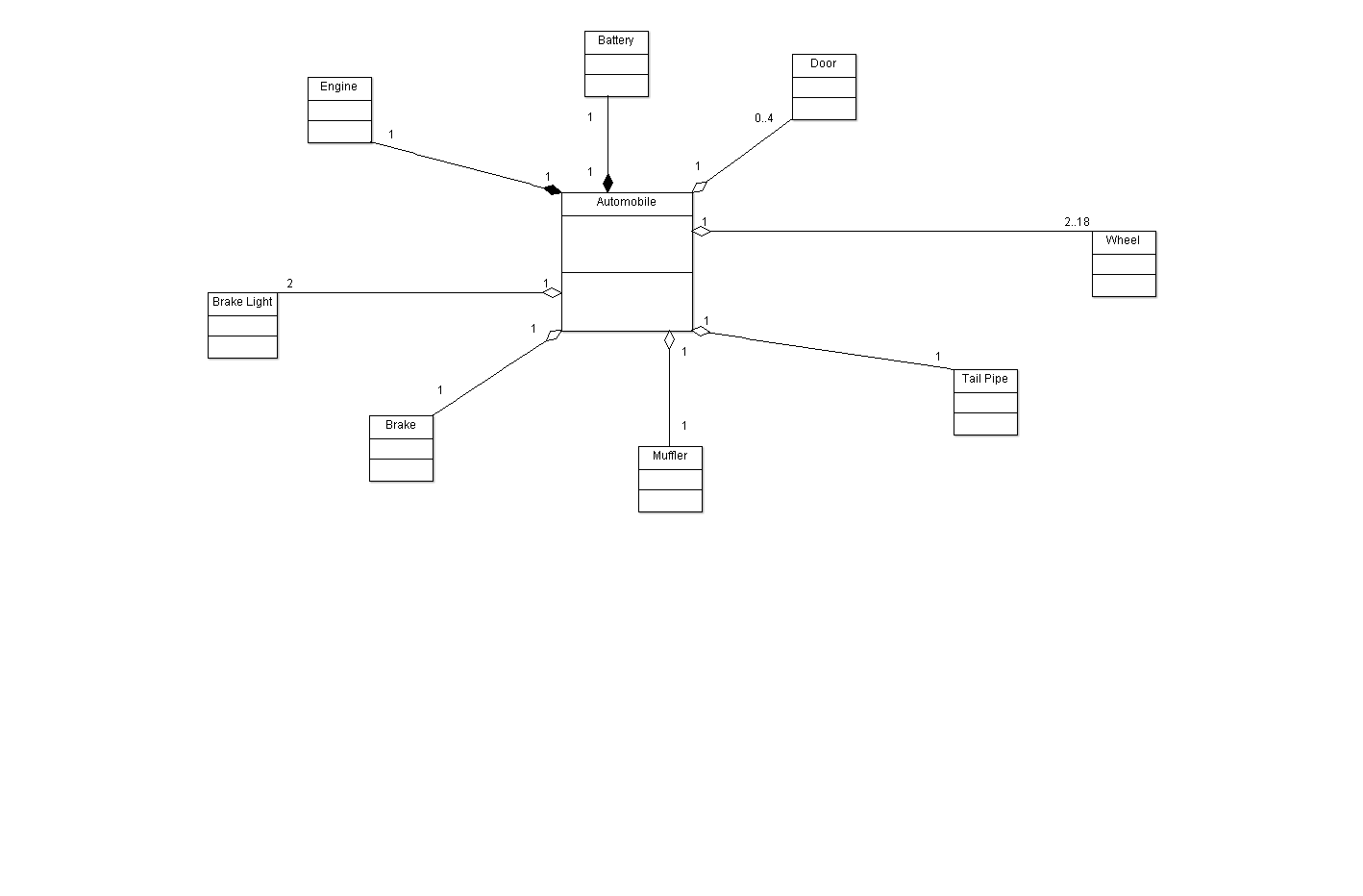
**Note:** All the following information came from the first reference.

2.)

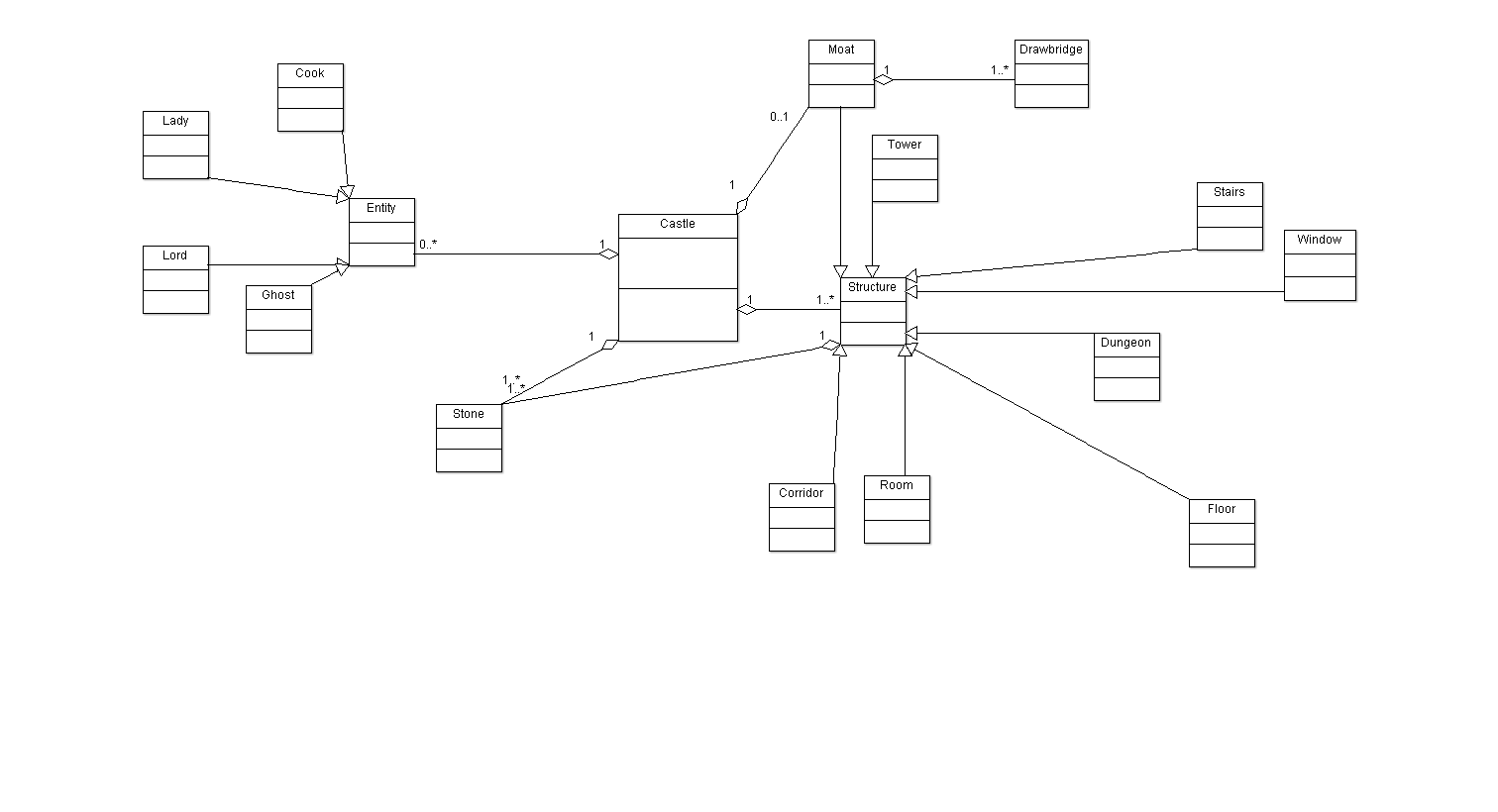
3.)

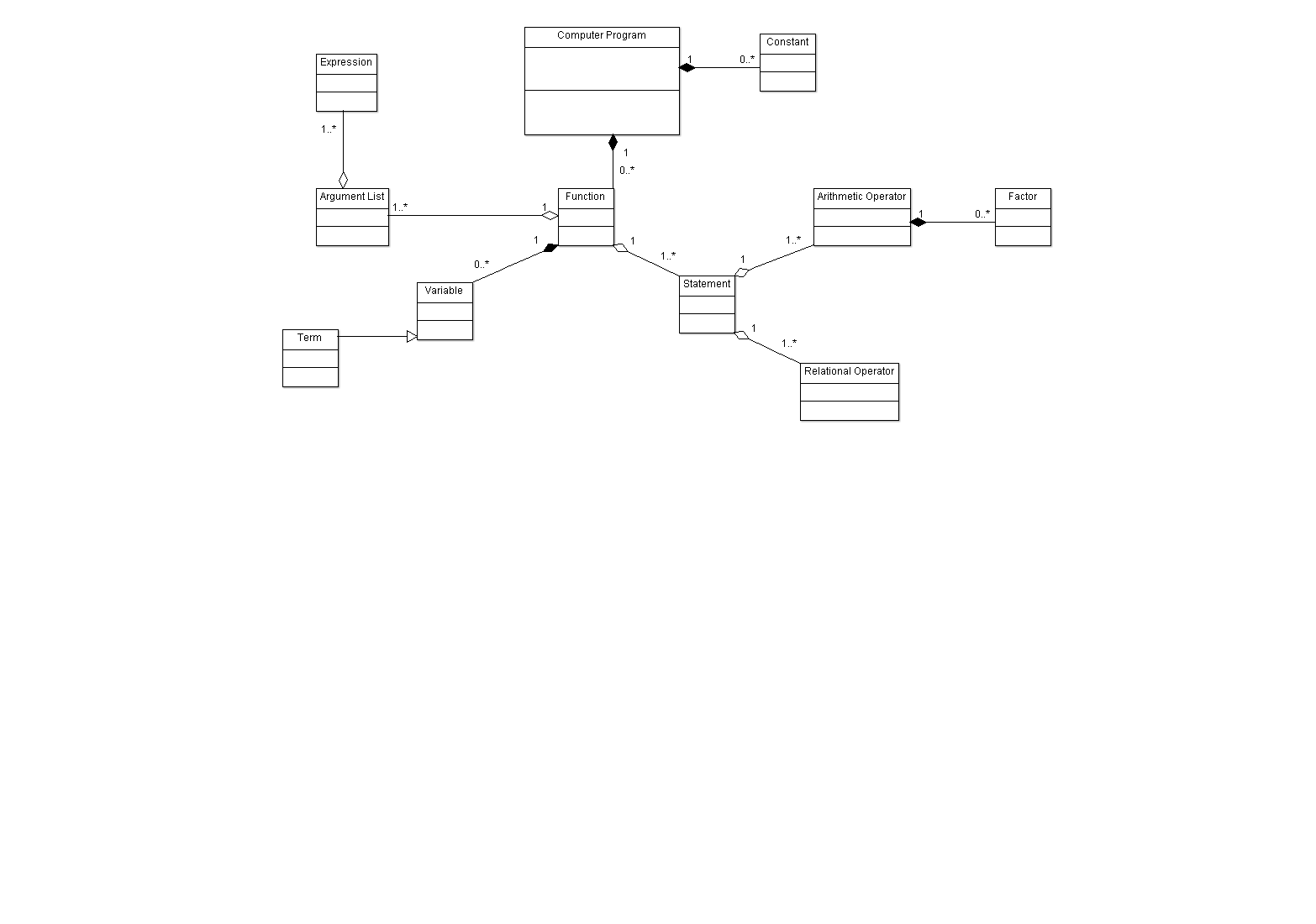


4.) a.)

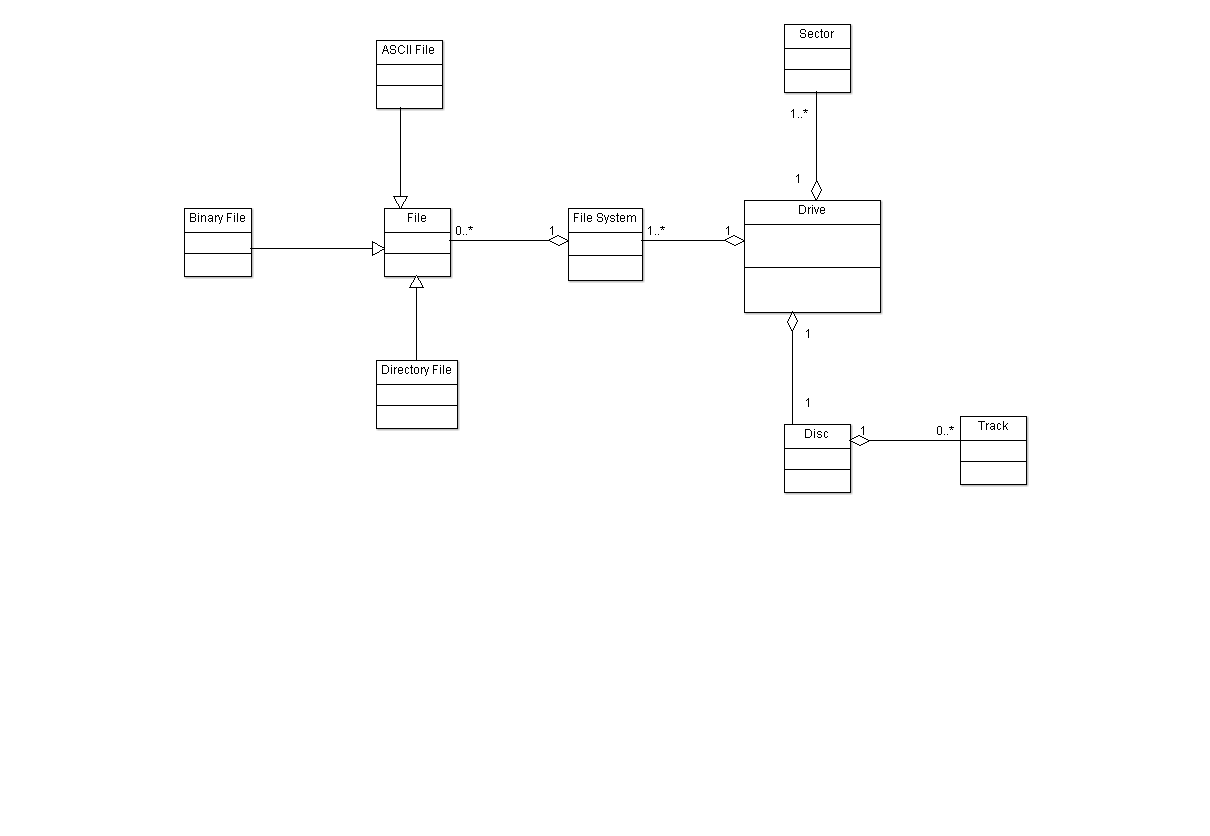


b.)

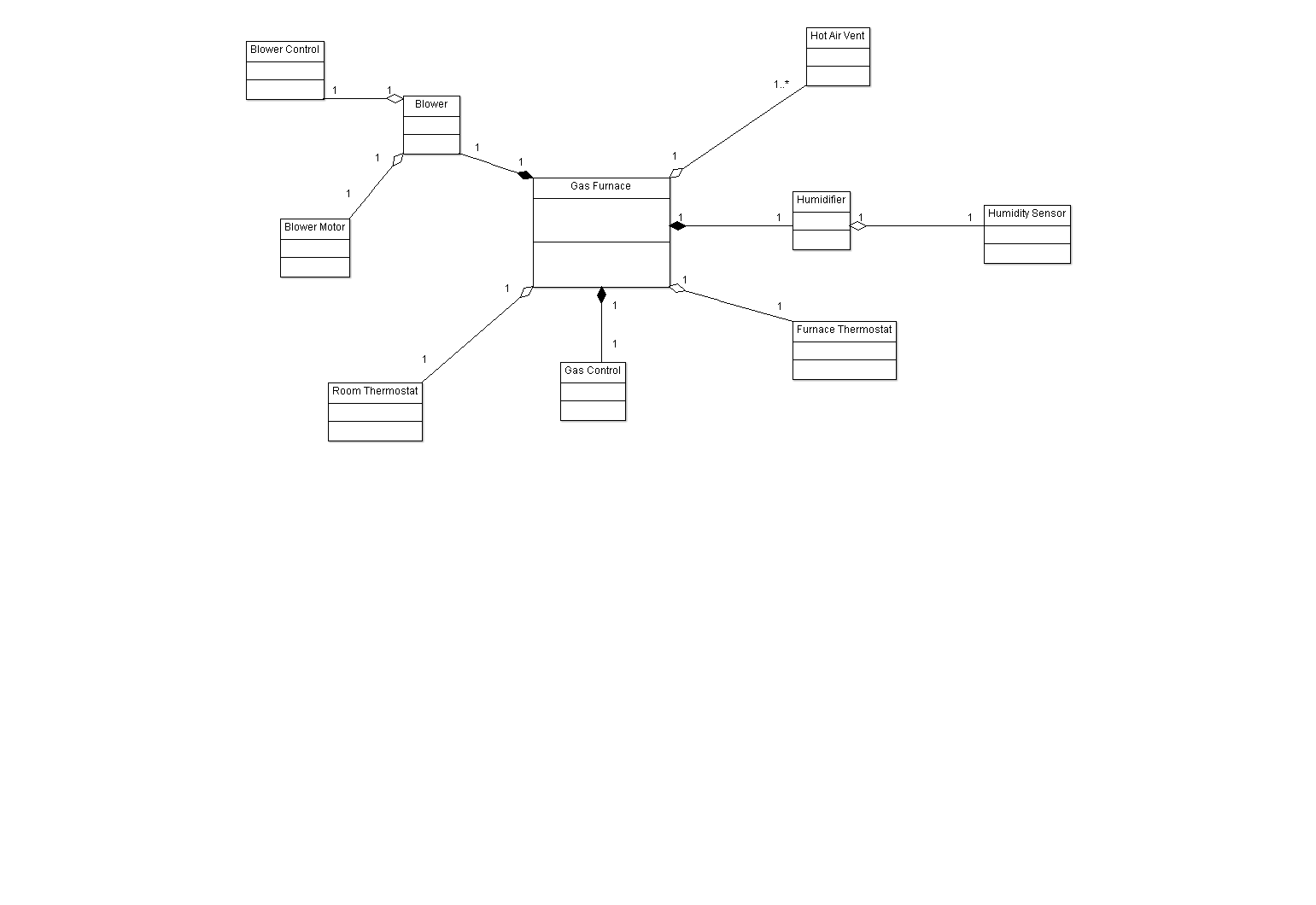
c.)

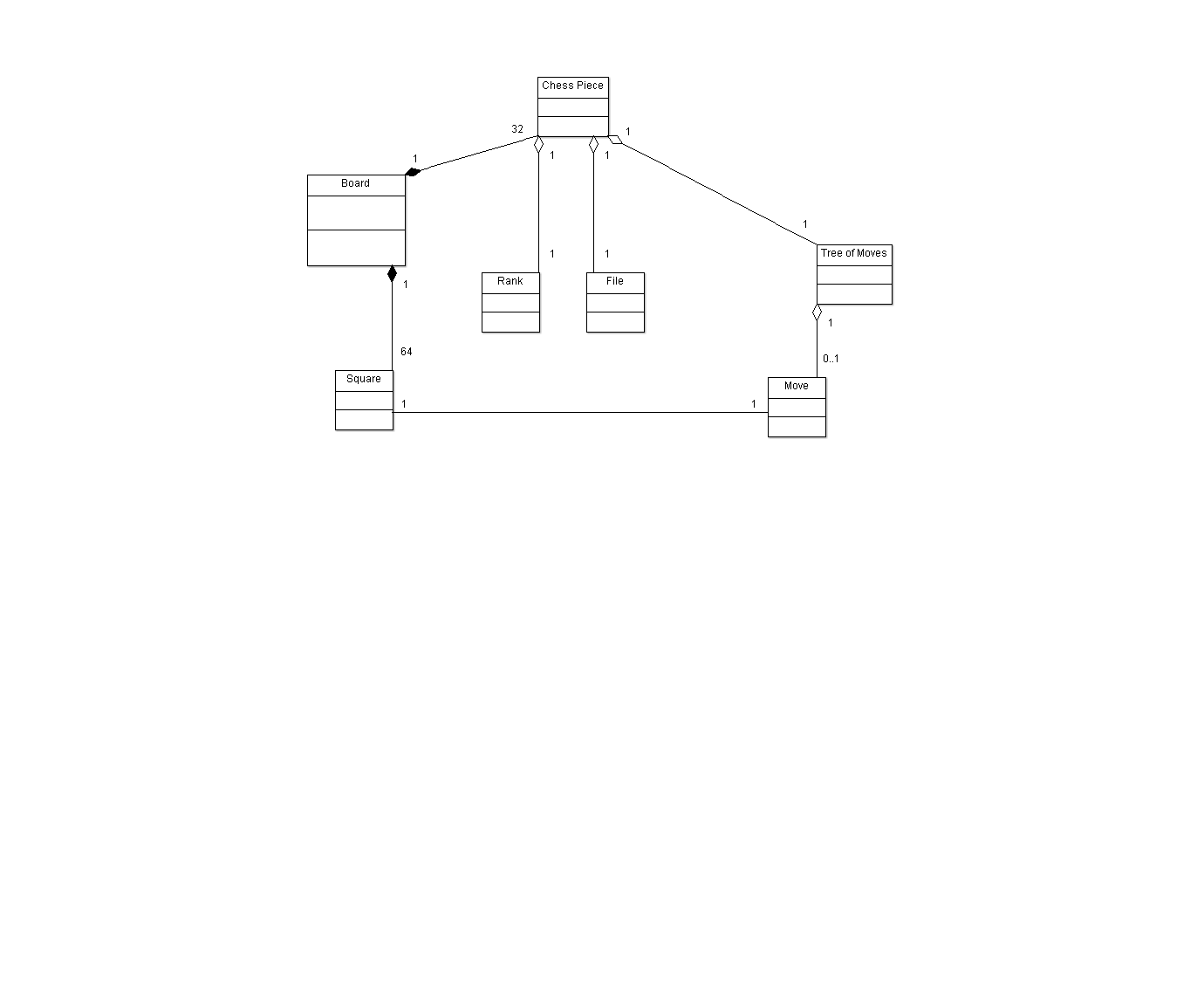


d.)

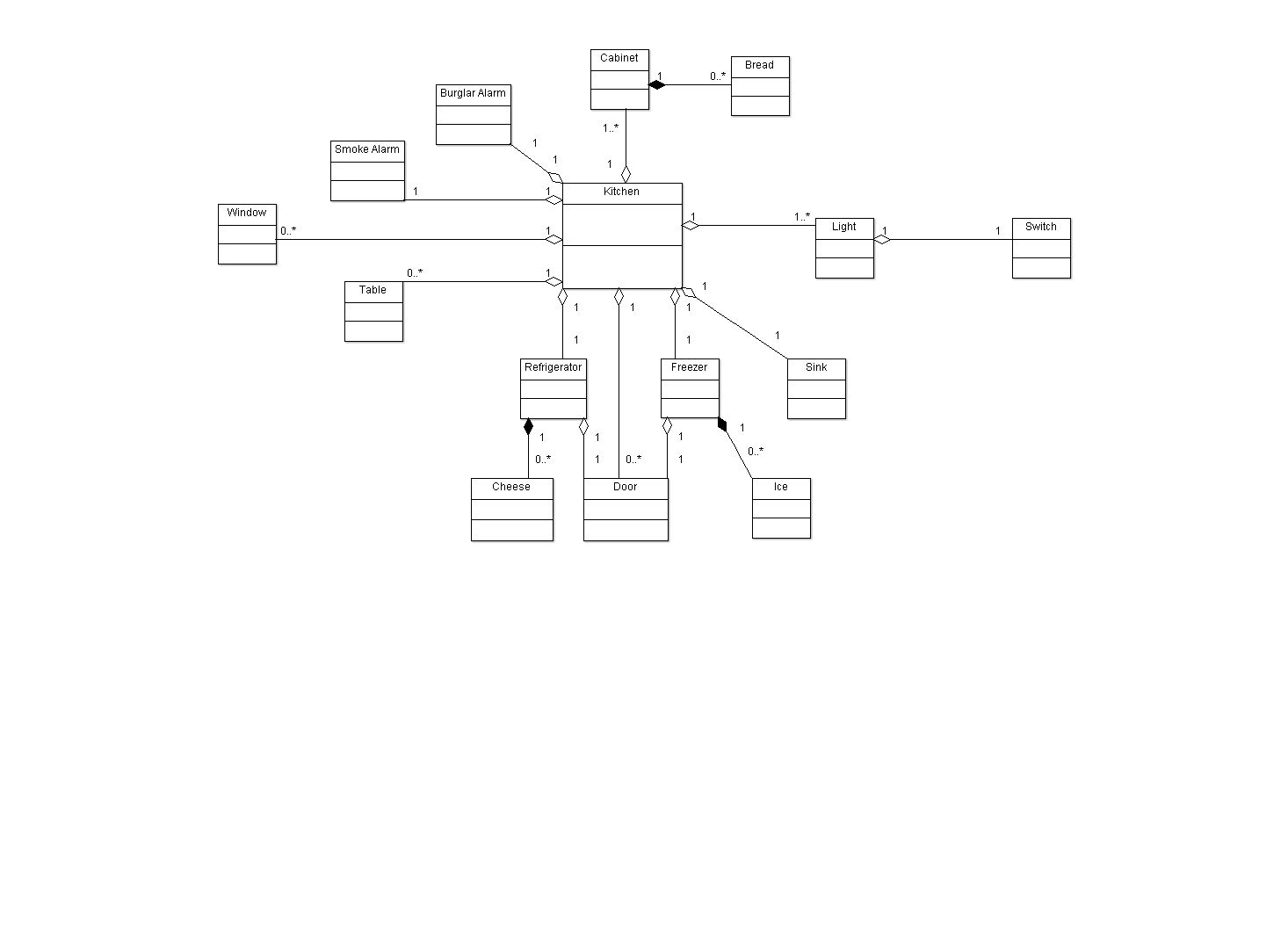


e.)

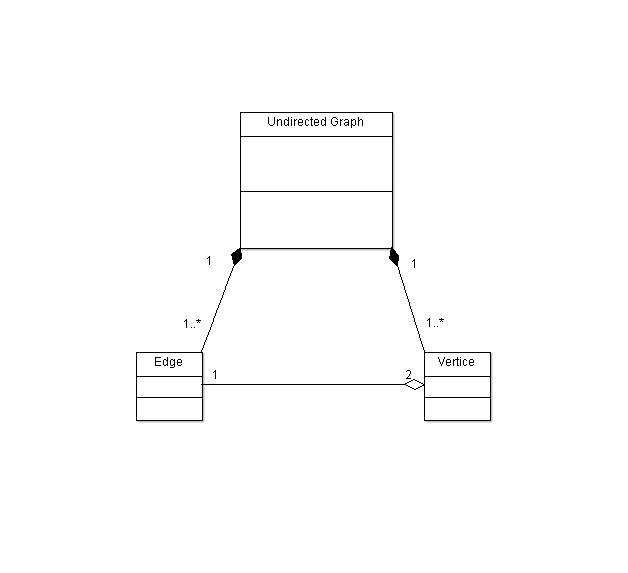
f.)

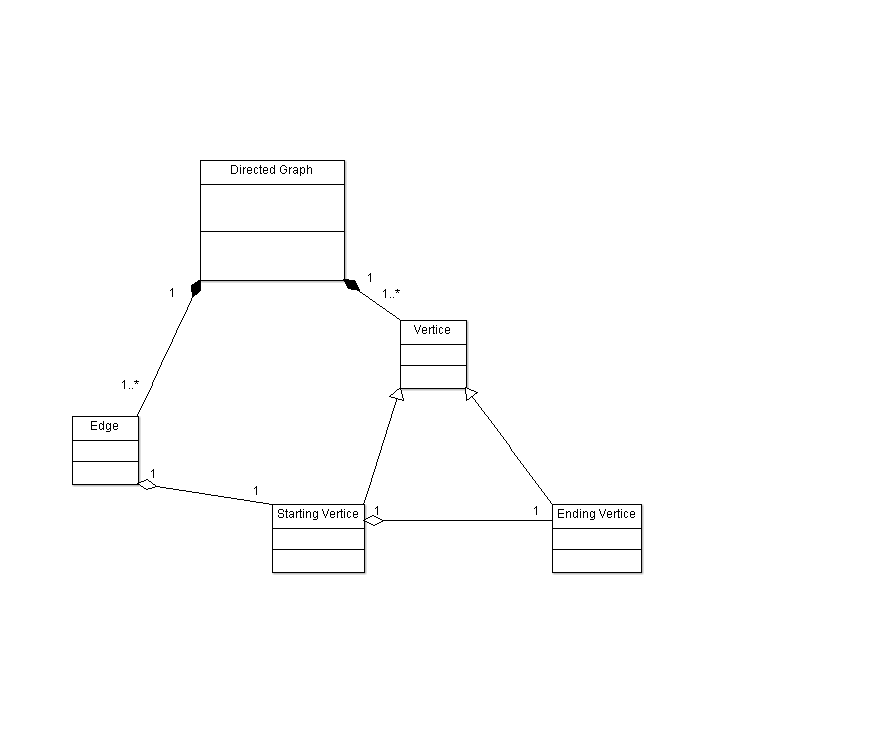


g.)



h.)

5.) a.)



b.)

6.) a.) This type of relationship between the two objects (a country and a capital city) is aggregation. The reason is because this follows a ‘has-a’ relationship (i.e. a country has a capital city).

b.) This type of relationship between the two objects (a dining philosopher and a fork) is association. The reason is because this follows a ‘one-to-many’ relationship (i.e. one dining philosopher uses one or many forks).

c.) This type of relationship between the two objects (a file and either an ordinary file or a directory file) is generalization. The reason is because this follows a ‘is-a’ relationship (i.e. a file is either an ordinary file or a directory file).

d.) This type of relationship between the two objects (files and records) is aggregation. The reason is because this follows a ‘has-a’ relationship (i.e. a file has a record).

e.) This type of relationship between the two objects (a polygon and an ordered set of points) is generalization. The reason is because this follows a ‘is-a’ relationship (i.e. a polygon is an ordered set of points).

f.) This type of relationship between the two objects (a drawing object is text, a geometrical object, or a group) is generalization. The reason is because this follows a ‘is-a’ relationship (i.e. a drawing object is one of the three objects).

g.) This type of relationship between the two objects (a person and a computer language) is association. The reason is because this follows a ‘one-to-one’ relationship (i.e. a person uses a computer language).

h.) This type of relationship between the two objects (modems/keyboards and input/output devices) is association. The reason is because this follows a ‘one-to-one’ relationship (i.e. a modem can only either be an input device or an output device, and the same holds true for a keyboard).

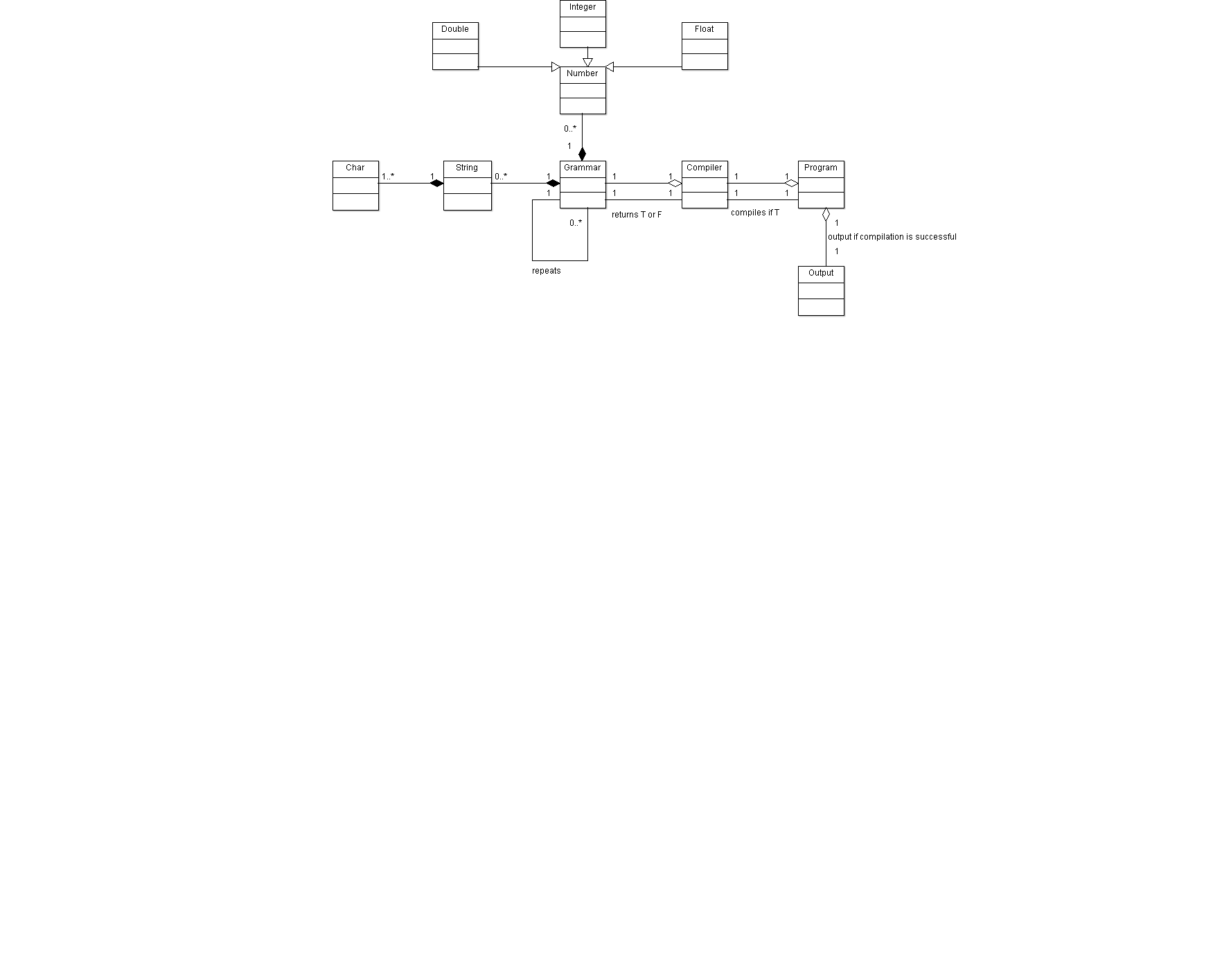
i.) This type of relationship between the two objects (classes and attributes) is aggregation. The reason is because this follows a ‘has-a’ relationship (i.e. one class has many attributes).

j.) This type of relationship between the two objects (a person and a team) is aggregation. The reason is because this follows a ‘has-a’ relationship (i.e. a team has a player, or multiple players).

k.) This type of relationship between the two objects (a route and a city) is aggregation. The reason is because this follows a ‘has-a’ relationship (i.e. a route has two cities).

l.) This type of relationship between the two objects (a student and a course) is aggregation. The reason is because this follows a ‘has-a’ relationship (i.e. a student has a course with a specific professor, or the course with a specific professor has a student (or many)).

**Note:** All the definitions for the following relationships used in this answer came from the second reference.

 7.)

**External References**

1. Boundless. “Boundless Psychology.” *Lumen Learning*, Lumen, courses.lumenlearning.com/boundless-psychology/chapter/classification-and-categorization/.
2. “Association, Aggregation, Composition, Abstraction, Generalization, Realization, Dependency.” *Java Tutorial Blog*, javapapers.com/oops/association-aggregation-composition-abstraction-generalization-realization-dependency/.