Software Modeling and Design

3700.002

Spring 2020

Solutions for Homework #1

Due: February 3, 2020 11:59PM (Monday)

Maximum points: 100

Maximum Bonus points: 24

(Individual Assignment)

Note: Submit a single pdf document with solutions to all the questions. Embed all diagrams and text in the word document and create the pdf. Use the tool of your choice to generate UML diagrams. Possible options are PlantUML, ARGOUML, Visio, Word, or any other UML development tool.

Q1: (10 Points): Problem 3.6 from textbook page number 52.

Figure A3.6 shows a class diagram for a family tree consistent with the exercise. Other models are also possible such as those showing divorce and remarriage. The cousin and sibling associations are logically redundant and can be derived. Chapter 4 discusses derived information. We used our semantic understanding of the exercise to determine multiplicity in our answer. In general, you can only partially infer multiplicity from examples. Examples can establish the need for many multiplicity but does not permit you to conclude that exactly 1 or 0..1 multiplicity applies.

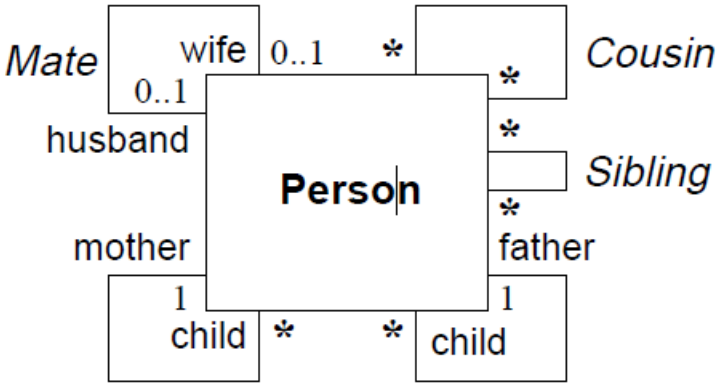

Figure A3.6 Class diagram for family trees

Q3: (10 Points): Problem 3.17 from textbook page number 56.

Figure A3.22 adds multiplicity and attributes to the class diagram for an athletic event scoring system. Age is a derived attribute that is computed from birthdate and the current date. (See Chapter 4.) We have added an association between Competitor and Event to make it possible to determine intended events before trials are held.

A class model cannot readily enforce the constraint that a judge rates every competitor for an event. Furthermore, we would probe this statement if we were actually building the scoring system. For example, what happens if a judge has scored some competitors for an event and gets called away? Are the scores of the judge discarded? Is the judge replaced? Or are partial scores for an event recorded? When you build applications, you must regard requirements as a good faith effort to convey what is required, and not necessarily as the literal truth.

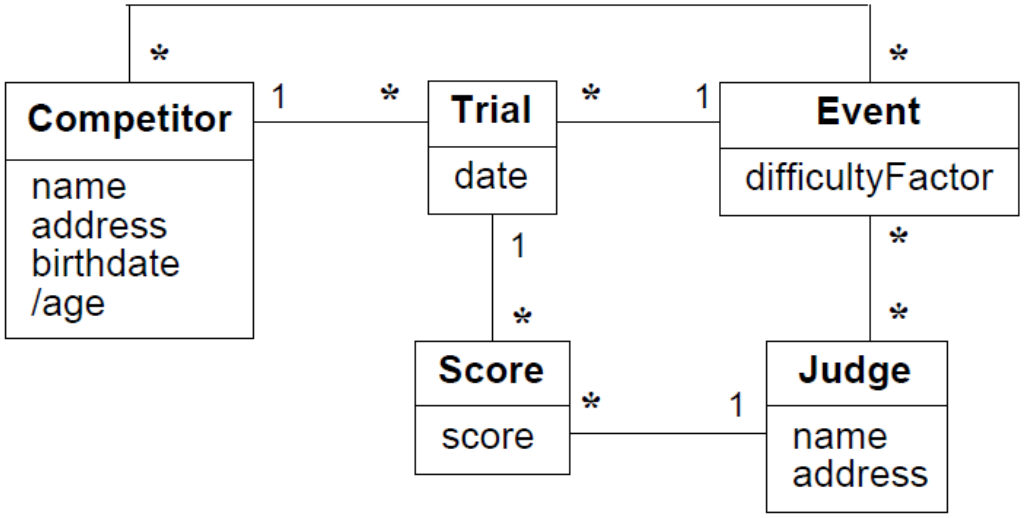

Figure A3.22 Portion of a class diagram for an athletic event scoring system

Q4: (10 Points): Problem 3.25 from textbook page number 57.

Figure A3.30 shows a class diagram for car loans in which pointers are replaced with associations. In this form, the arguably artificial restriction that a person has no more than three employers has been eliminated. Note that in this model an owner can own several cars. A car can have several loans against it. Banks loan money to persons, companies, and other banks.

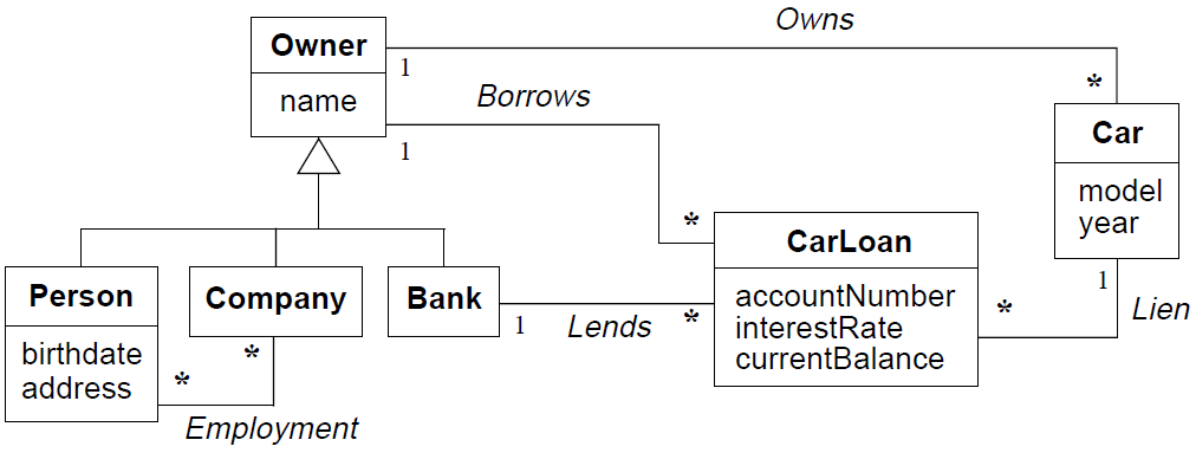

Figure A3.30 Proper class diagram for car loans

Q5: (6 Points): Problem 3.28 from textbook page number 58.

Figure A3.34 shows a class diagram for the dining philosopher's problem. The one-to-one associations describe the relative locations of philosophers and forks. The *InUse* association describes who is using forks. Other representations are possible, depending on your viewpoint. An object diagram may help you better understand this problem.

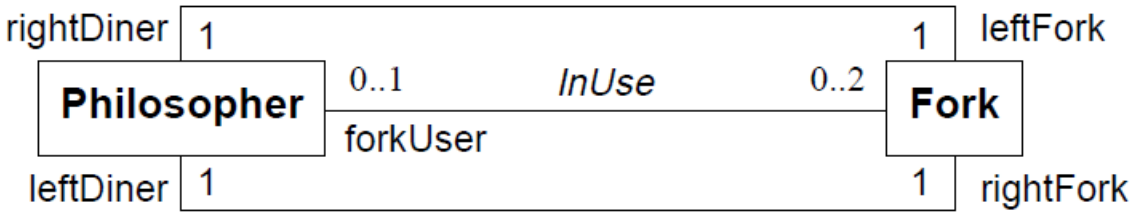

Figure A3.34 Class diagram for the dining philosopher problem

Q6: (3 Points): Problem 3.34 from textbook page number 59.

Figure E3.13 (a) states that a subscription has derived identity. Figure E3.13 (b) gives subscriptions more prominence and promotes subscription to a class. The (b) model is a better model. Most copies of magazines have subscription codes on their mailing labels; this could be stored as an attribute. The subscription code is intended to identify subscriptions; subscriptions are not identified by the combination of a person and a magazine so we should promote Subscription to a class. Furthermore, a person might have multiple subscriptions to a magazine; only the (b) model can readily accommodate this.

Q7: (7 Points): Problem 4.1 from textbook page number 83.

Figure A4.1 improves the class diagram in the exercise by changing some associations to aggregations. For a bill-of-material parts hierarchy, all parts except the root must belong to something. Note that none of the aggregations are composition—the parts do not have coincident lifetime with the assemblies.

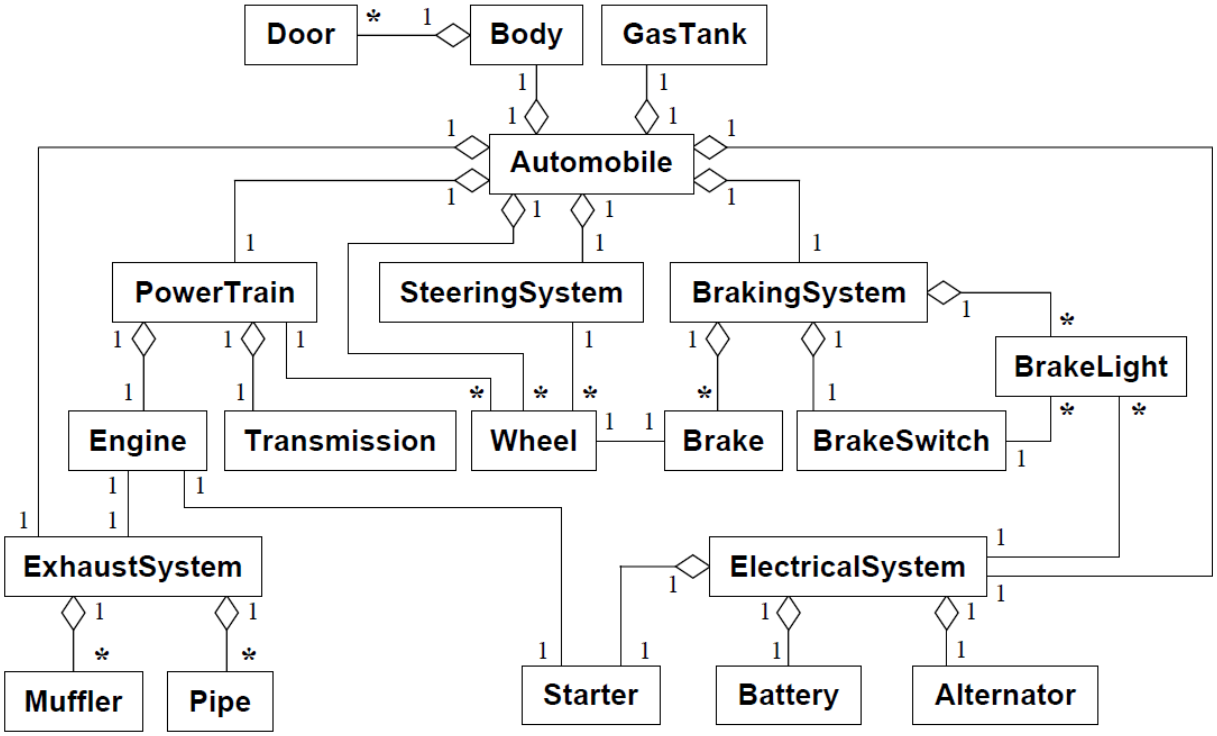

Figure A4.1 Portion of a class diagram of the assembly hierarchy of an automobile

Q8: (24 Points): Problem 4.3 from textbook page number 84.

a. *A country has a capital city.* Association. A capital city and a country are distinct things so generalization certainly does not apply. You could argue that a capital city is a part of a country and thus they are related by aggregation.

b. *A dining philosopher uses a fork.* Association. Dining philosophers and forks are completely distinct things and are therefore not in a generalization relationship. Similarly, neither object is a part of the other and the relationship is not aggregation.

c. *A file is an ordinary file or a directory file.* Generalization. The word "or" often is an indicator of generalization. File is the superclass and OrdinaryFile and DirectoryFile are subclasses.

d. *Files contain records.* Aggregation. The word "contain" is a clue that the relationship may be aggregation. A record is a part of a file. Some attributes and operations on files propagate to their constituent records.

e. *A polygon is composed of an ordered set of points.* Aggregation. The phrase "is composed of" should immediately make you suspicious that there is an aggregation. An ordered set of points is a part of a polygon. Some attributes and operations on a polygon propagate to the corresponding set of points.

f. *A drawing object is text, a geometrical object, or a group.* Generalization. Once again, the word "or" should prompt you to think of generalization. DrawingObject is the superclass. Text, GeometricalObject, and Group are subclasses.

g. *A person uses a computer language on a project.* Ternary association. Person, ComputerLanguage, and Project are all classes of equal stature. The association cannot be reduced to binary associations. None of these classes are a-kind-of or a-part-of another class. Thus generalization and aggregation do not apply.

h. *Modems and keyboards are input / output devices.* Generalization. The keyword "are" is the clue. Modem and Keyboard are the subclasses; InputOutputDevice is the superclass.

i. *Classes may have several attributes.* Association or aggregation. It depends on your perspective and the purpose of the model whether aggregation applies.

j. *A person plays for a team in a certain year.* Ternary association. Person, Team, and Year are distinct classes of equal stature.

k. *A route connects two cities. Association.* Either Route is a class associated with the City class, or Route is the name of the association from City to City.

l. *A student takes a course from a professor.* Ternary association. Student, Course, and Professor are distinct classes of equal stature.

Q9: (10 Points): Problem 4.4 from textbook page number 85.

Figure A4.3 shows a class diagram for a graphical document editor. The requirement that a Group contain 2 or more DrawingObjects is expressed as a multiplicity of 2..* on DrawingObject in its aggregation with Group. The fact that a DrawingObject need not be in a Group is expressed by the zero-one multiplicity. It is possible to revise this diagram to make a Circle a special case of an Ellipse and to make a Square a special case of a Rectangle. We presume that a DrawingObject belongs to a Sheet and has a coincident lifetime with it. Similarly, we presume that a Sheet belongs to one Document for its lifetime. Hence both are composition relationships.

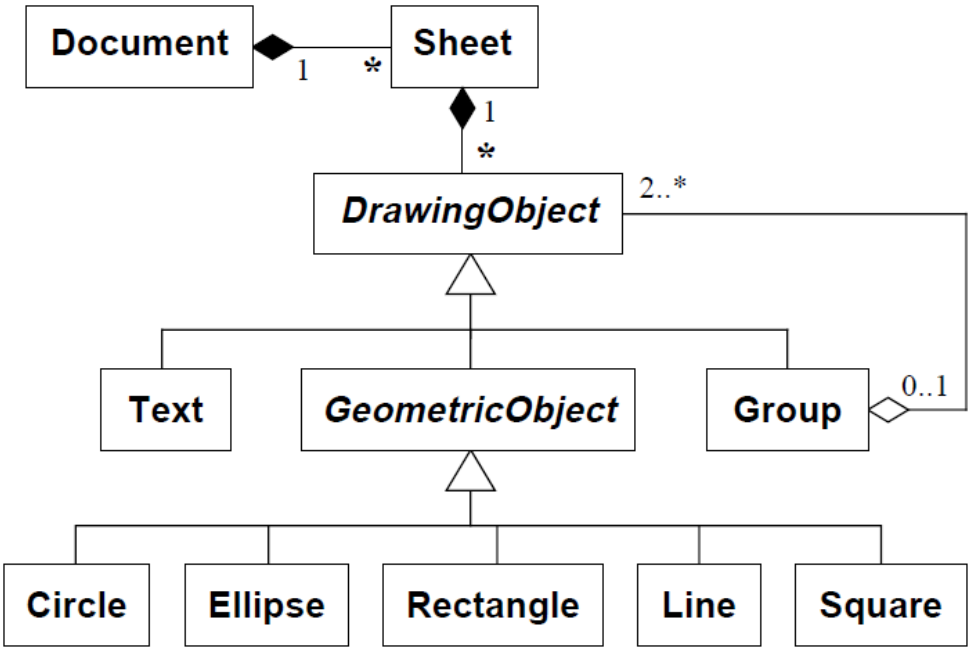

Figure A4.3 Class diagram for a graphical document editor that supports grouping

Q10: (10 Points): Problem 4.14 from textbook page number 86.

Figure A4.12 shows our answer to the exercise.

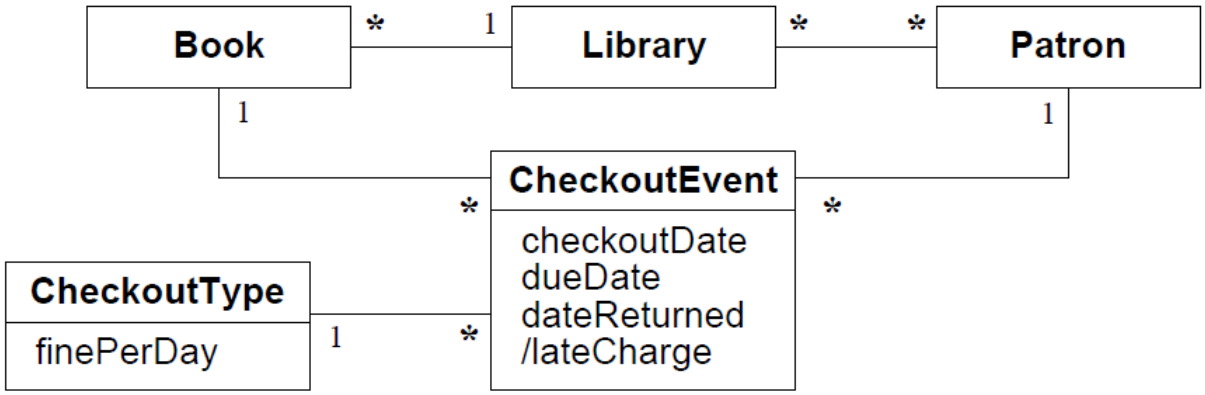

Figure A4.12 Class diagram for library book checkout system

Q11: (10 Points): Problem 4.18 from textbook page number 86.

Figure A4.17 shows a class model for the NASAA form. We could have made Recommendation an association class, but we promoted it to a class to handle the unusual situation where a broker has multiple suggestions for a security in the same call. Your model could vary a bit from ours, depending on the precise interpretation of the form.

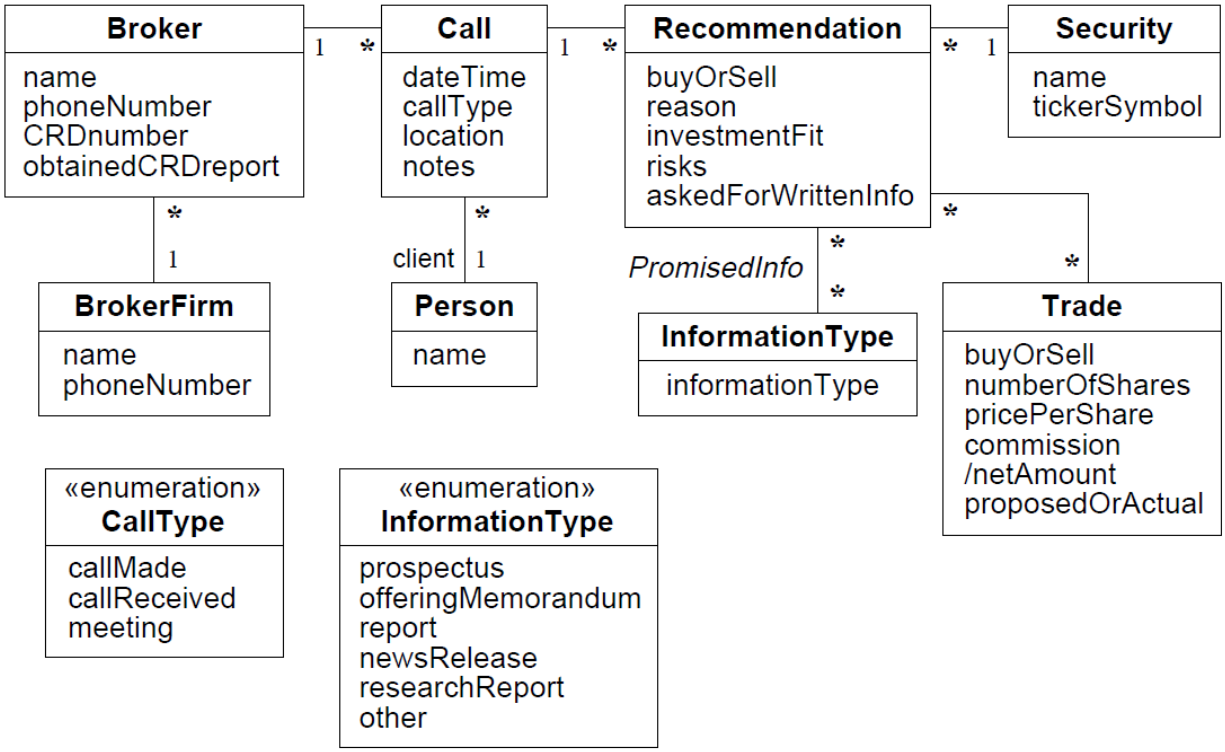

Figure A4.17 Class diagram for NASAA form

Q2: Bonus - (24 Points): Problem 3.13 from textbook page number 53.

a. Figure A3.11 shows one possible class diagram for a school.

A school has a principal, many students, and many teachers. Each of these persons has a name, birthdate, and may borrow and return books. Teachers and the principal are both paid a salary; the principal evaluates the teachers. A school board supervises multiple schools and can hire and fire the principal for each school. A school has many playgrounds and rooms. A playground has many swings. Each room has many chairs and doors. Rooms include restrooms, classrooms, and the cafeteria. Each classroom has many computers and desks. Each desk has many rulers.

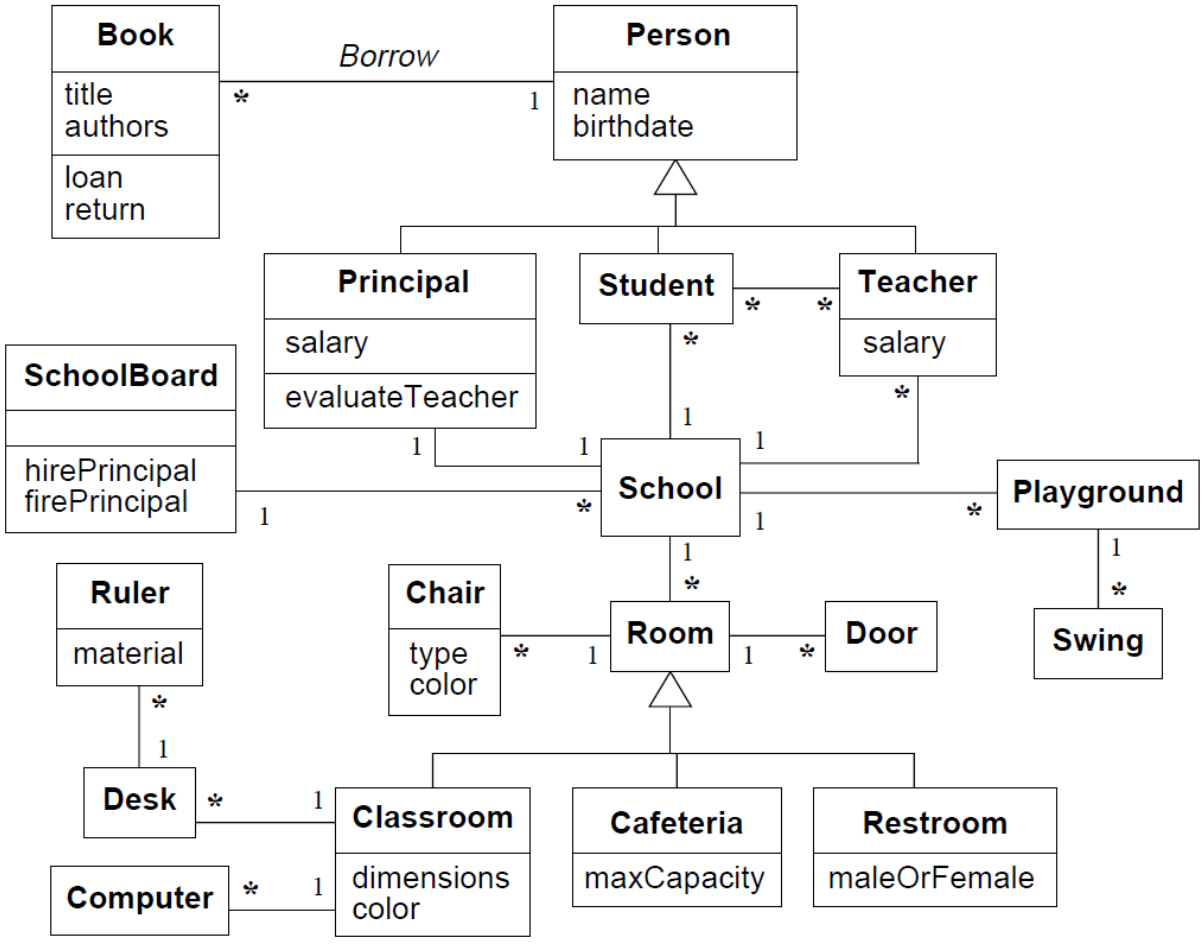

Figure A3.11 Class diagram for a school

b. Figure A3.12 shows a class diagram for an automobile.

An automobile is composed of a variety of parts. An automobile has one engine, one exhaust system, many wheels, many brakes, many brake lights, many doors, and one battery. An automobile may have 3, 4, or 5 wheels depending on whether the frame has 3 or 4 wheels and the optional spare tire. Similarly a car may have 2 or 4 doors (2..4 in the model, since UML2 multiplicity must be an interval). The exhaust system can be further divided into smaller components such as a muffler and tailpipe. A brake is associated with a brake light that indicates when the brake is being applied.

Note that we made manufacturer an attribute of automobile, rather than a class that is associated with automobile. Either approach is correct, depending on your perspective.

If all you need to do is to merely record the manufacturer for an automobile, the manufacturer attribute is adequate and simplest. If on the other hand, you want to record details about the manufacturer, such as address, phone number, and dealership information, then you should treat manufacturer as a class and associate it with automobile.

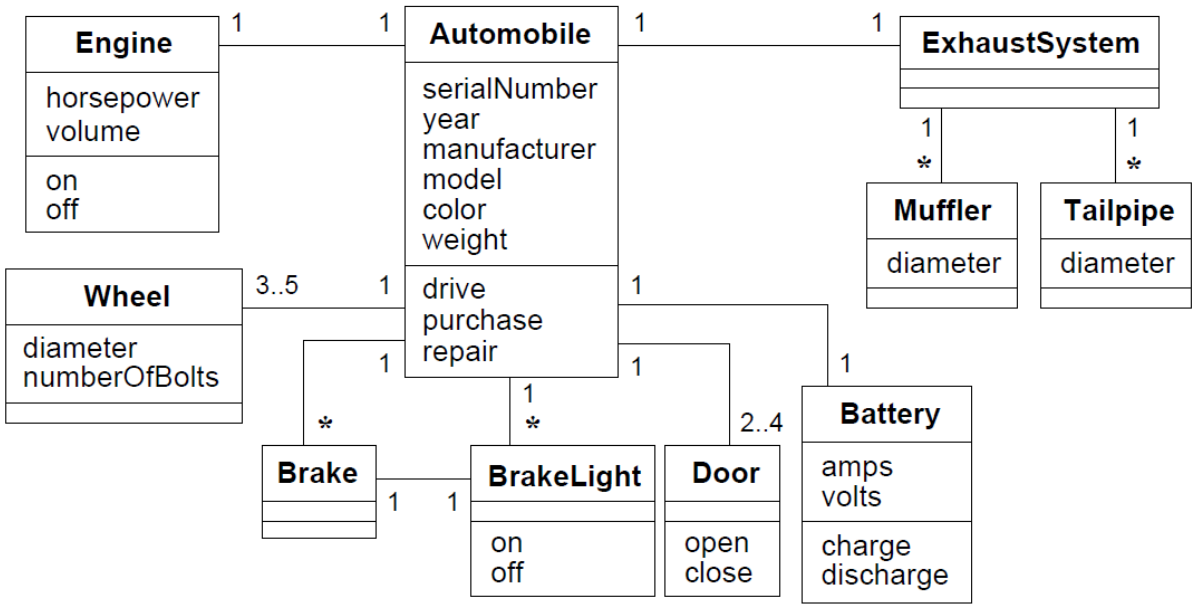

Figure A3.12 Class diagram for an automobile

c. Figure A3.13 shows a class diagram for a castle. A castle has many rooms, corridors, stairs, towers, dungeons, and floors. Each tower and dungeon also has a floor. The castle is built from multiple stones each of which has dimensions, weight, color, and a composition material. The castle may be surrounded by a moat. Each lord lives in a castle; a castle may be without a lord if he has been captured in battle or killed. Each lady lives in a castle with her lord. A castle may be haunted by multiple ghosts, some of which have hostile intentions.

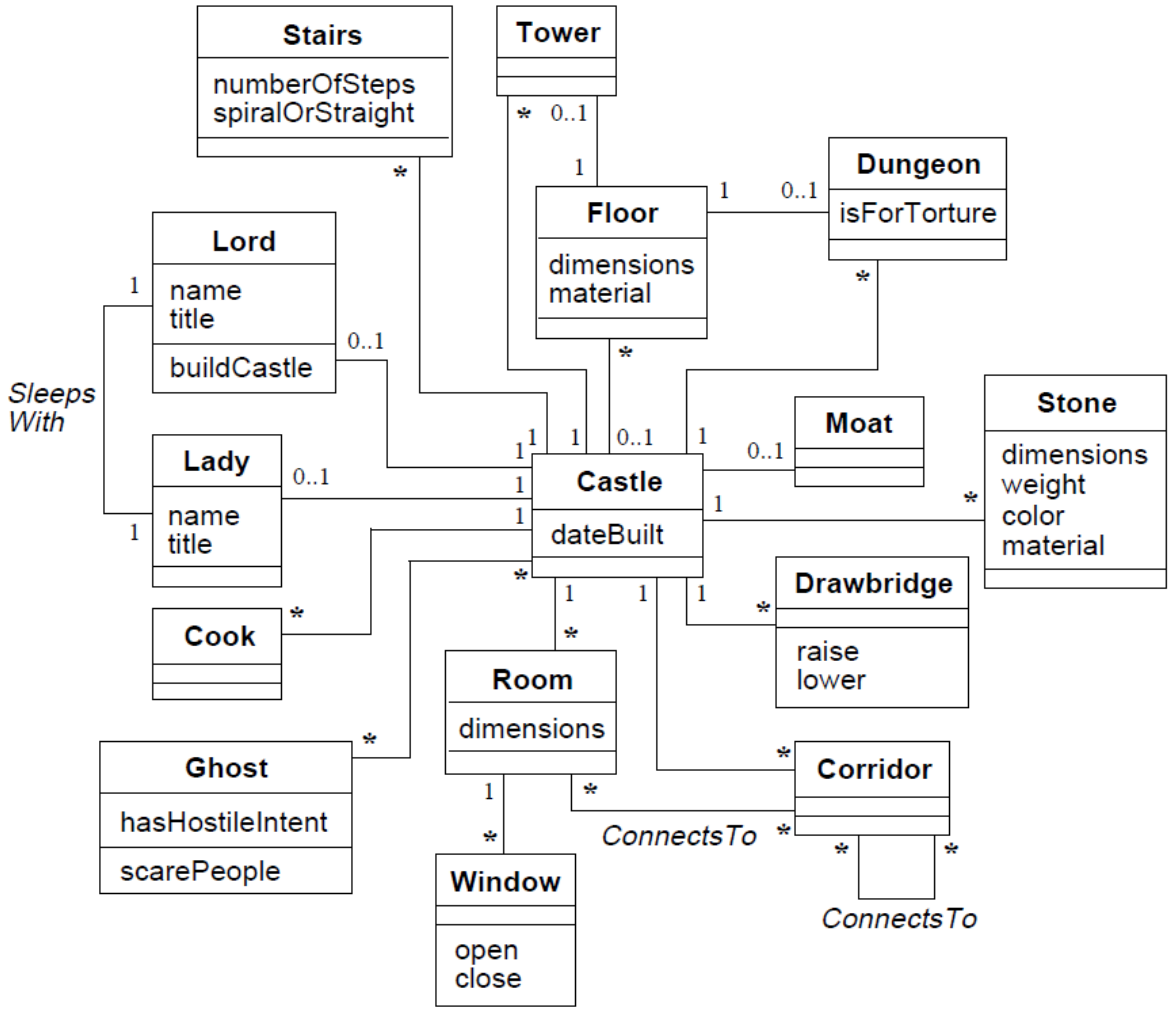

Figure A3.13 Class diagram for a castle

d. Figure A3.14 and Figure A3.15 show a class diagram for a program. Note that we have added quite a few classes.

In Figure A3.14 a program has descriptive properties such as its name, purpose, date last modified, and author. Important operations on programs include: compile, link, execute, and debug. A program contains global data definitions and many functions. Each function has data definitions and a main block. Each block consists of many statements.

Some types of statements are assignment, conditional, iteration, and procedure call. An assignment statement sets a target variable to the result of an expression. A conditional statement evaluates an expression; a then block is executed if the expression is true and an optional else block is executed if the expression is false. An iteration statement continues to execute a block until a loop expression becomes false. A procedure call invokes a function and may pass the result of zero or more expressions in its argument list.

Figure A3.15 details the structure of expressions and parallels the structure of the programming code that could be used to implement expressions. An expression may be enclosed by parentheses. If so, the parentheses are removed, and the remaining expression recursively defined. Otherwise an expression contains a relational operator such as ‘>’, ‘=’, or ‘<=’ or is defined as a term. A term is binary addition, binary subtraction, or a factor. A factor involves multiplication, division, or unary expressions. Unary expressions can be refined into unary positive or negative expressions or terminals. A terminal may be a constant, variable, or a function reference. The “many” multiplicity that appear on ParenthesizedExpression and the other subclasses indicate that expressions may be reused within multiple contexts. (See Exercise 3.3.)

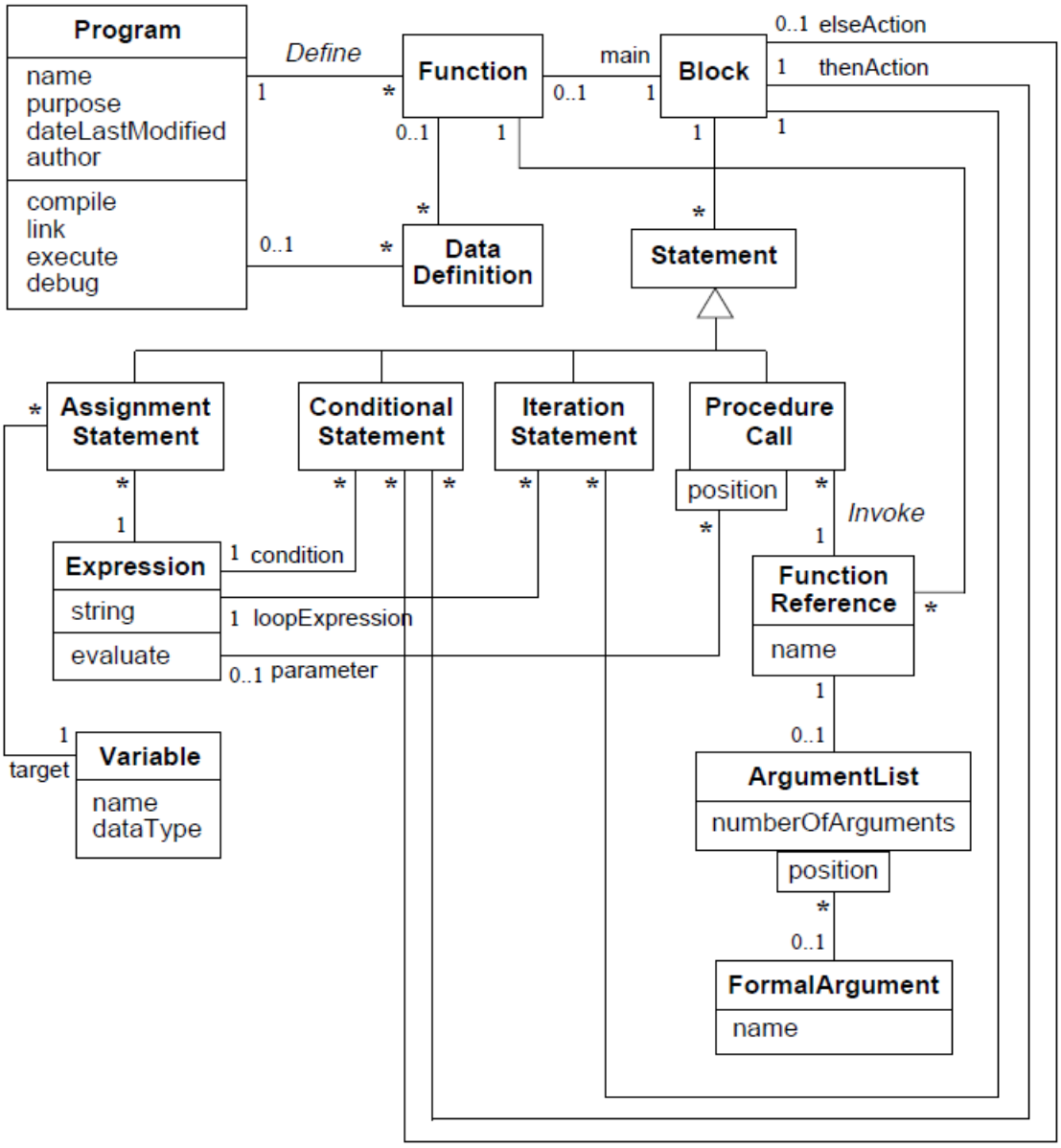

Figure A3.14 Class diagram for a computer program—part 1

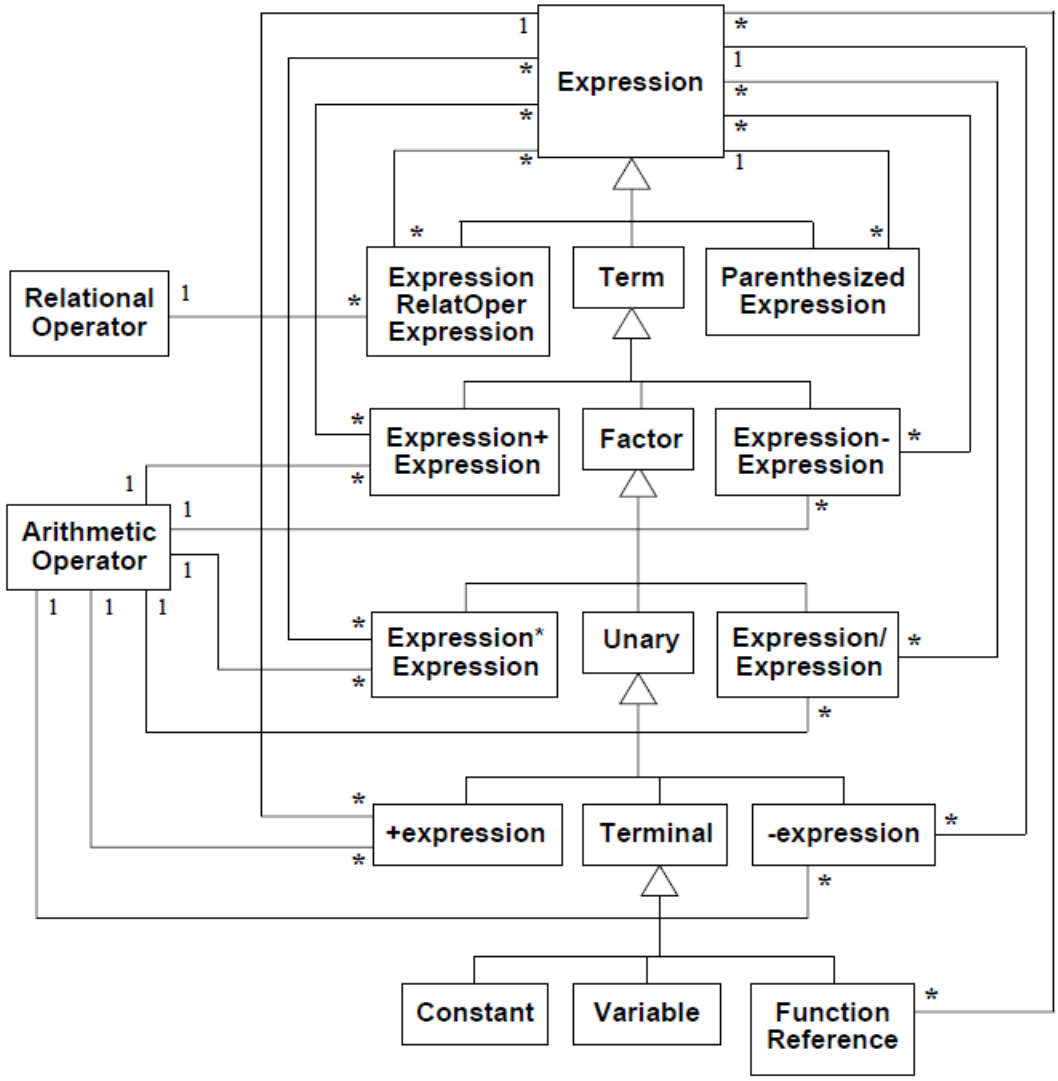

Figure A3.15 Class diagram for a computer program—part 2

e. Figure A3.16 shows a class diagram for a file system.

A drive has multiple discs; a hard drive contains many discs and a floppy drive contains one disc. (Platter may be a better name instead of Disc.) A disc is divided into tracks which are in turn subdivided into sectors. A file system may use multiple discs and a disc may be partitioned across file systems. Similarly, a disc may contain many files and a file may be partitioned across many discs.

A file system consists of many files. Each file has an owner, permissions for reading and writing, date last modified, size, and checksum. Operations that apply to files include create, copy, delete, rename, compress, uncompress, and compare. Files may be data files or directory files. A directory hierarchically organizes groups of presumably related files; directories may be recursively nested to an arbitrary depth. Each file within a directory can be uniquely identified by its file name. A file may correspond to many directory–file name pairs such as through UNIX links. A data file may be an ASCII file or binary file.

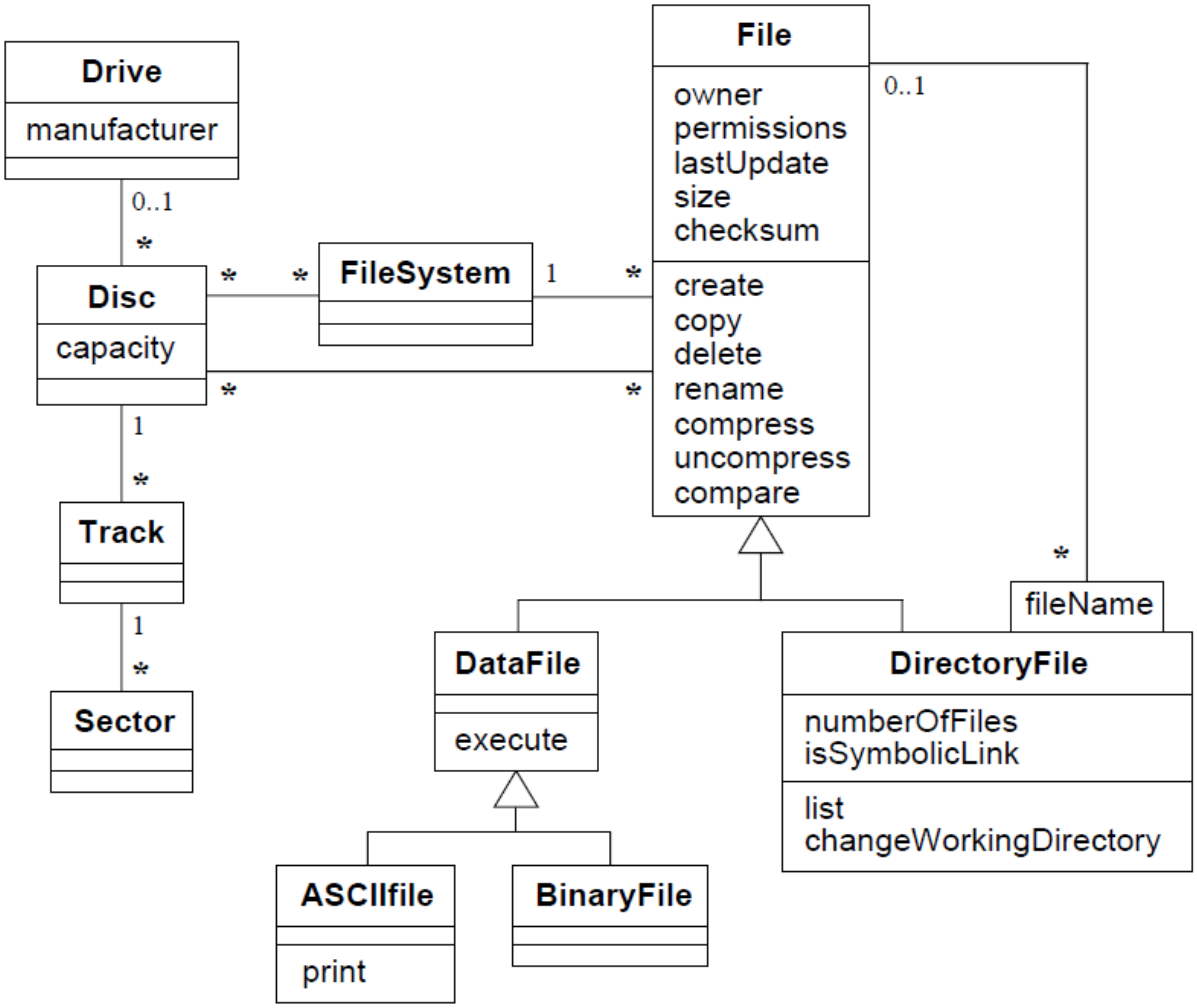

Figure A3.16 Class diagram for a file system

f. Figure A3.17 shows a class diagram for a gas fired, hot air heating system. A gas heating system is composed of furnace, humidification, and ventilation subsystems. The furnace subsystem can be further decomposed into a gas furnace, gas control, furnace thermostat, and many room thermostats. The room thermostats can be individually identified via the room number qualifier. The humidification subsystem includes a humidifier and humidity sensor. The ventilation subsystem has a blower, blower control, and many hot air vents. The blower in turn has a blower motor subcomponent.

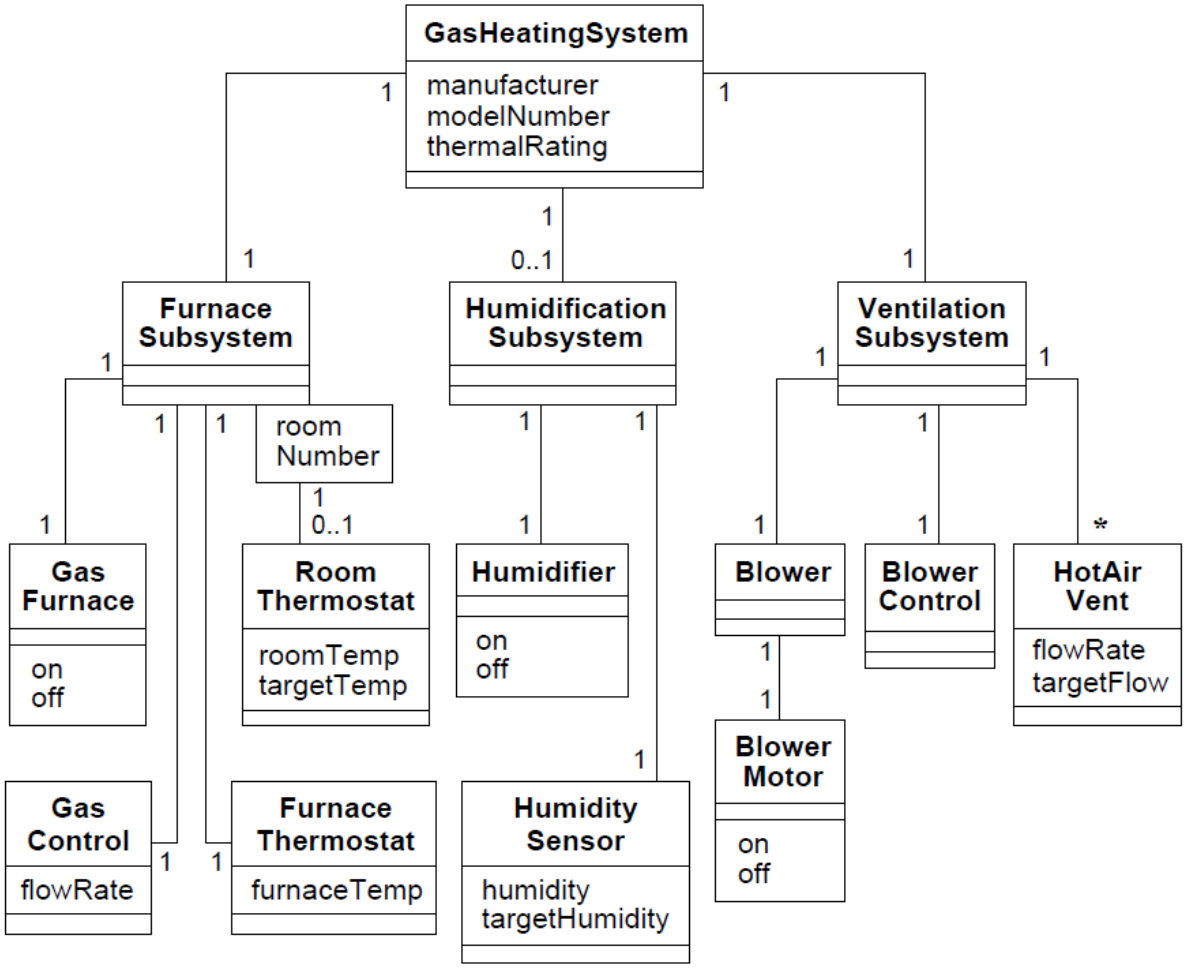

Figure A3.17 Class diagram for a gas-fired, hot-air heating system

g. Figure A3.18 shows a class diagram for a chess game. We assume that the purpose of this class diagram is to serve as a basis for a computerized chess game. A chess game involves many chess pieces of various types such as rooks, pawns, a king, and a queen. A chess game is also associated with a board and a sequence of moves. Each time the computer contemplates a move, it computes a tree of possible moves. There are various algorithms which can be used to evaluate potential moves and restrict the growth of the search tree. The human player can change the difficulty of the computer opponent by adjusting the depth of the strategy lookahead.

Each chess piece is positioned on a square or off the board if captured; some squares on the board are unoccupied. A move takes a chess piece and changes the position from an old position to a new position. A move may result in capture of another piece. The square for a move is optional since the chess piece may start or end off the board. Each square corresponds to a rank and file; the rank is the y-coordinate and the file is the xcoordinate.

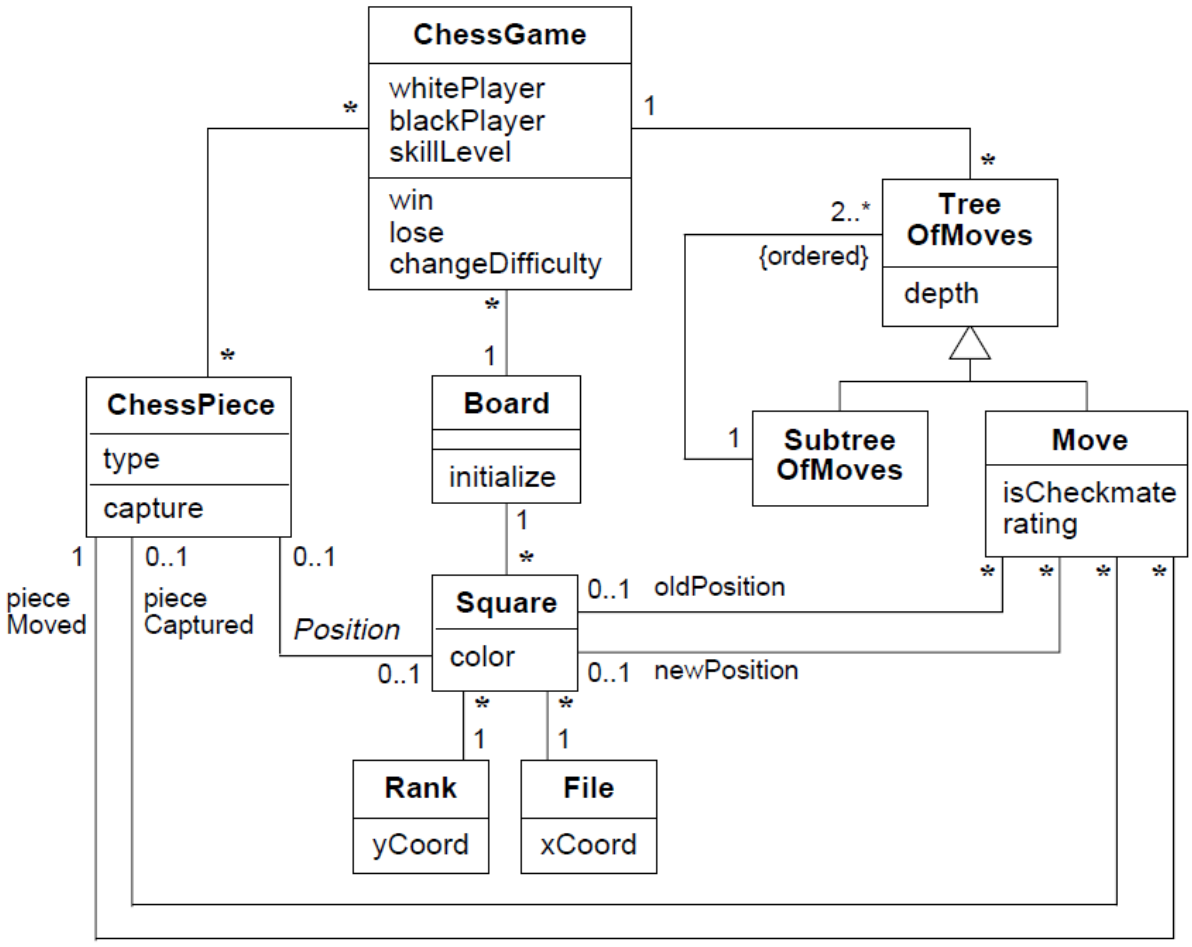

Figure A3.18 Class diagram for a chess game

h. Figure A3.19 shows a class diagram for a building. This diagram is simple and self-explanatory.

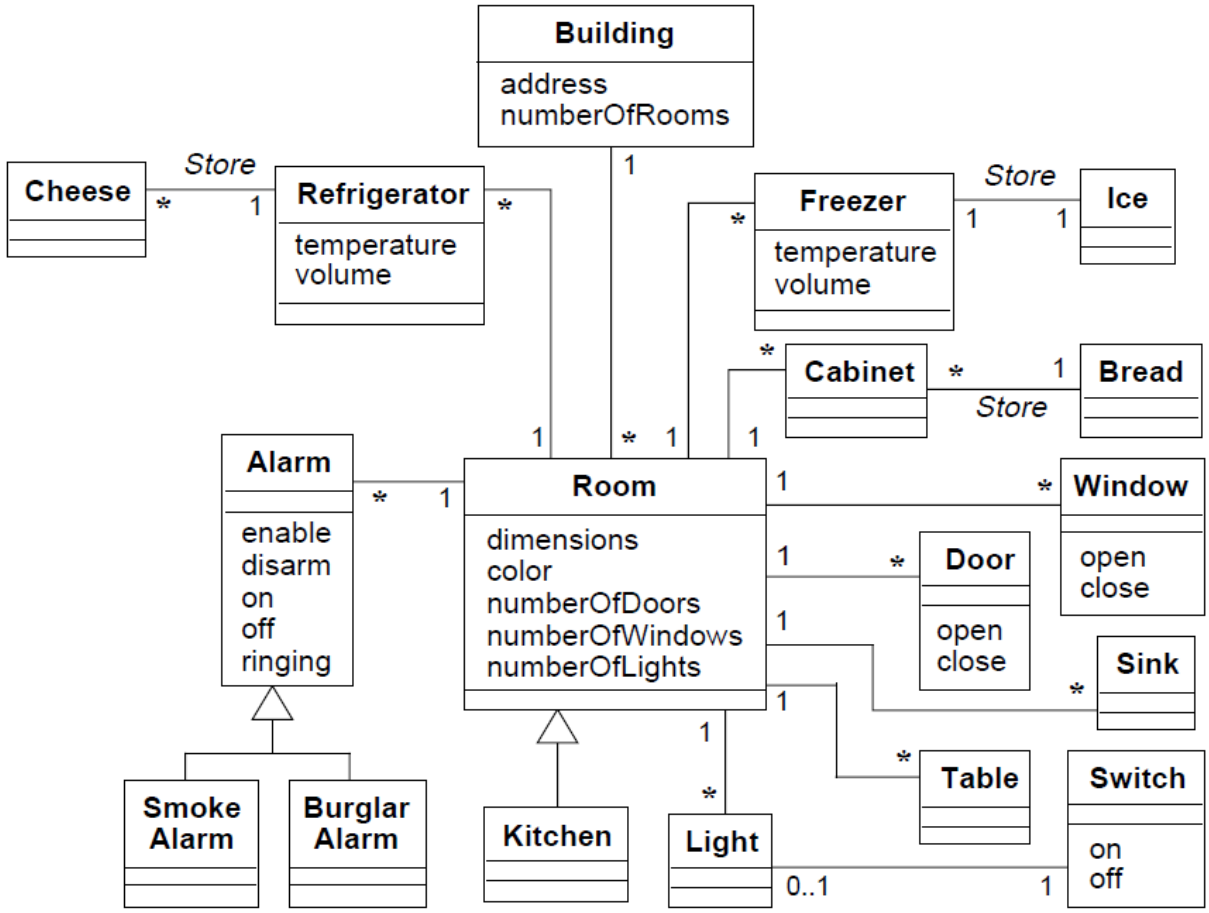

Figure A3.19 Class diagram for a building