CSE 470
Software Engineering
Class Diagram

Imran Zahid
Lecturer
Computer Science and Engineering, BRAC University

BRAC UNIVERSITY
Inspiring Excellence

Class Diagram from Scenario

Problem Statement to Class Diagram

Example:
"Develop a graphic editor that can draw different geometric shapes such as line, circle and triangle. User can select, move or rotate a shape. To do so, editor provides user with a menu listing different commands. Individual shapes can be grouped together and can behave as a single shape."

Step 1 - Identify Classes

Extract nouns in the problem statement.

Develop a graphic editor that can draw different geometric shapes such as line, circle and triangle. User can select, move or rotate a shape. To do so, editor provides user with a menu listing different commands. Individual shapes can be grouped together and can behave as a single shape.

Step 1 - Identify Classes

Eliminate irrelevant classes.

- Editor - Very broad scope
- User – Out of system boundary
- commands – Broad scope

Step 1 - Identify Classes

Add more classes by analyzing requirements

Group - required to behave as a shape

Individual shapes can be grouped together and can behave as a single shape

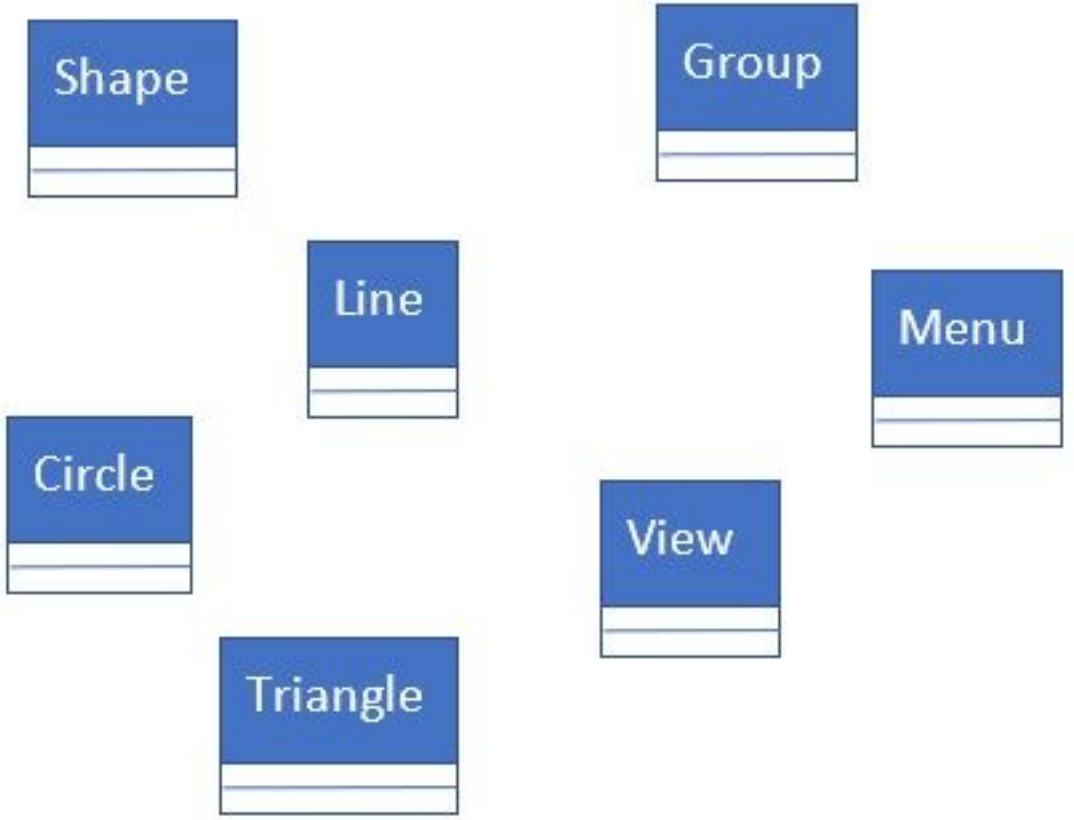
View – editor must have a display area

Step 1 - Identify Classes

Following classes have been identified:

- Shape
- Line
- Circle
- Triangle
- Menu
- Group
- View

Step 1 - Identify Classes



Step 2 - Identify Associations

Extract verbs connecting objects.

"Individual shapes can be grouped together"

Group consists of lines, circles, triangles

Group can also consists of other groups

(Composition)

Step 2 - Identify Associations

View contains shapes

- View contains lines
- View contains circles
- View contains triangles
- View contains groups

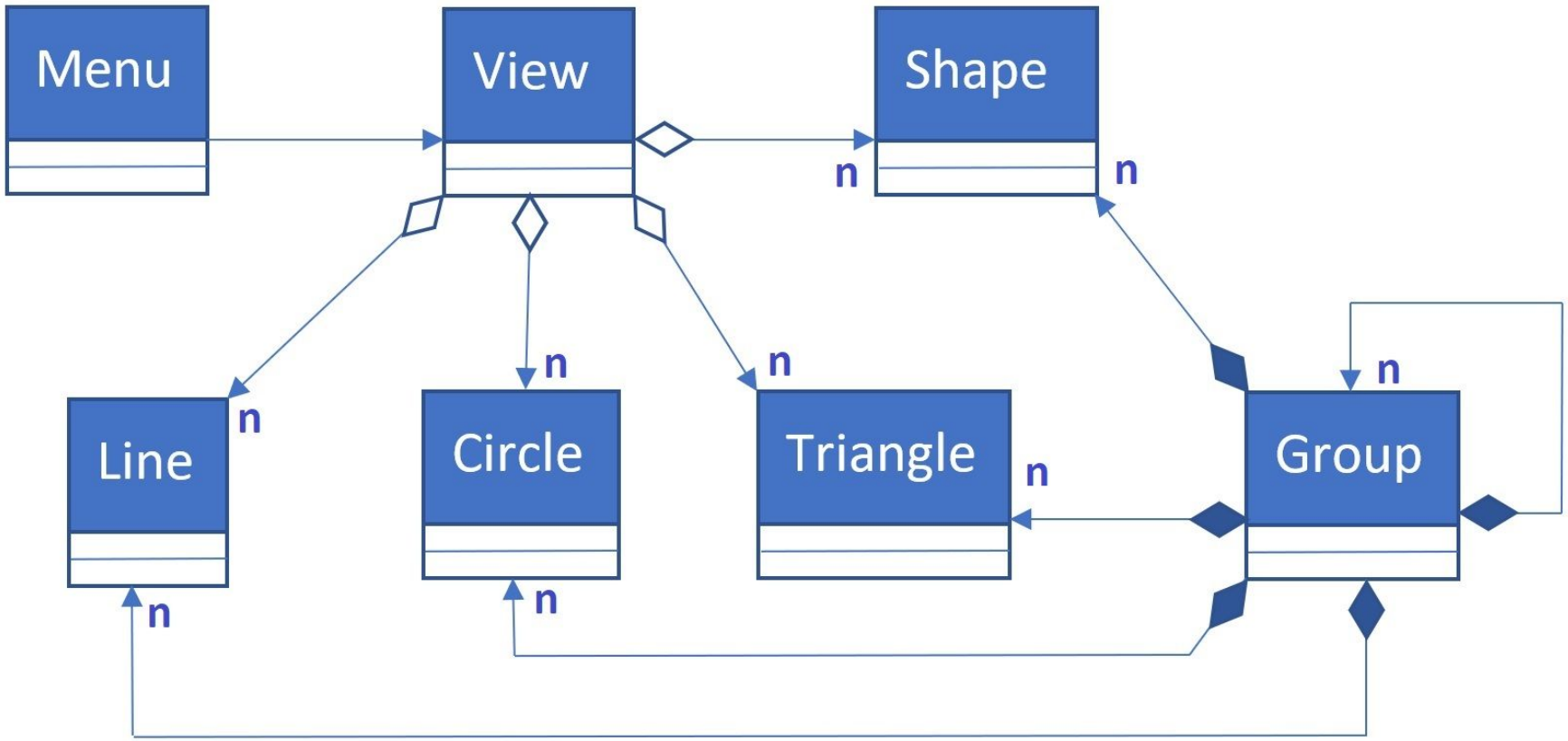
(Aggregation)

Step 2 - Identify Associations

Menu sends message to View

(Simple One-Way Association)

Step 2 - Identify Associations

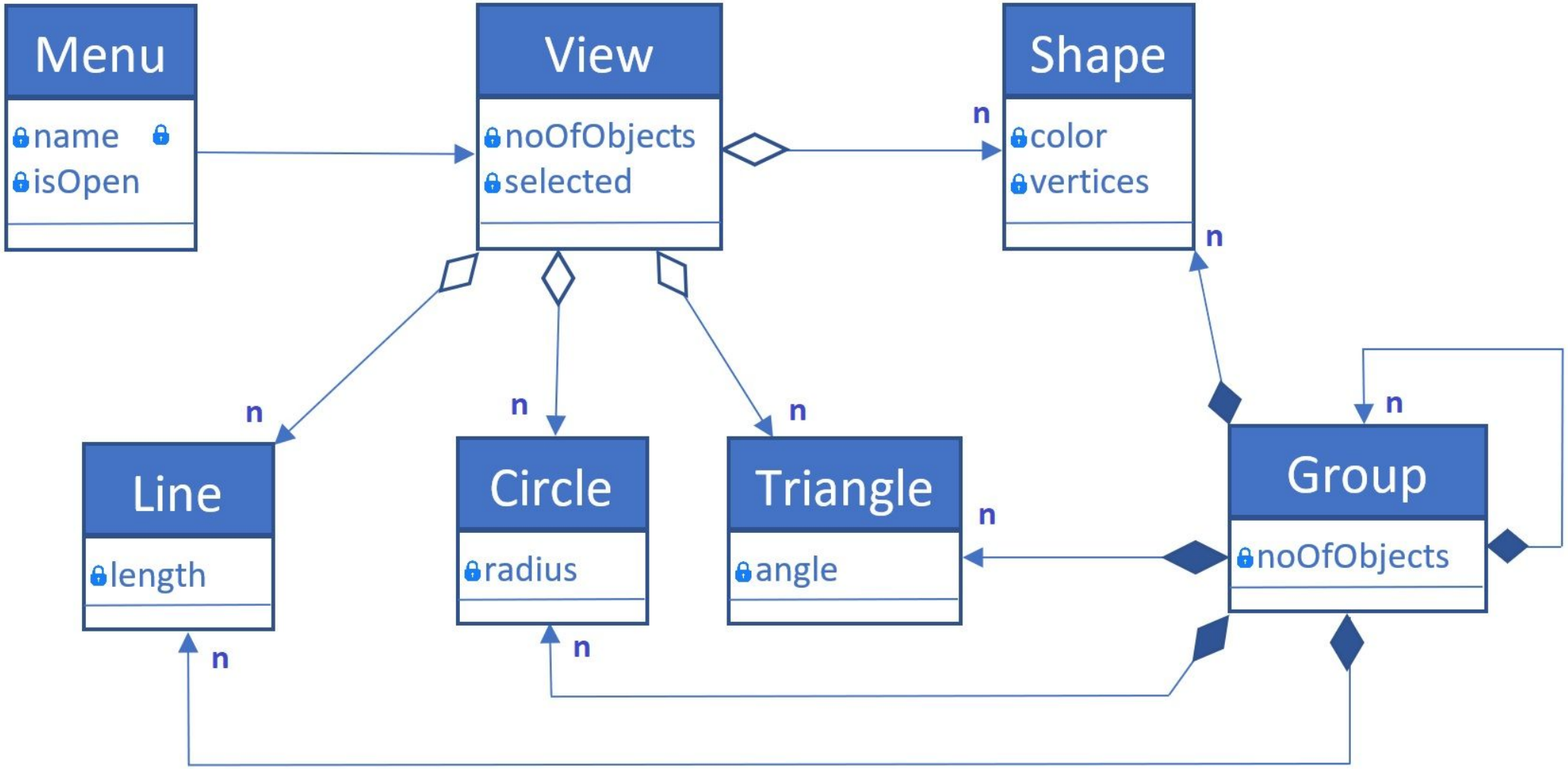


Step 3 - Identify Attributes

Extract properties of the object from the domain knowledge.

- **Shape**
  - Color
  - Vertices
- **Line**
  - Color
  - Vertices
  - Length
- **Circle**
  - Color
  - Vertices
  - Radius
- **Triangle**
  - Color
  - Vertices
  - Angle
- **Menu**
  - Name
  - isOpen
- **Group**
  - noOfObjects
- **View**
  - noOfObjects
  - selected

Step 3 - Identify Attributes



Step 4 - Identify Operations

Extract verbs connected with an object.

Develop a graphic editor that can draw different geometric shapes such as line, circle and triangle. User can select, move or rotate a shape. To do so, editor provides user with a menu listing different commands. Individual shapes can be grouped together and can behave as a single shape.

Step 4 - Identify Operations

Eliminate irrelevant operations.

Develop - out of system boundary

Provides - have broad semantics

Behave – broad semantics

Step 4 - Identify Operations

Extract properties of the object from the domain knowledge.

Shape

- Draw
- Select
- Move
- Rotate

Line

- Draw
- Select
- Move
- Rotate

Circle

- Draw
- Select
- Move
- Rotate

Triangle

- Draw
- Select
- Move
- Rotate

Menu

- Open
- Select
- Move
- Rotate

Group

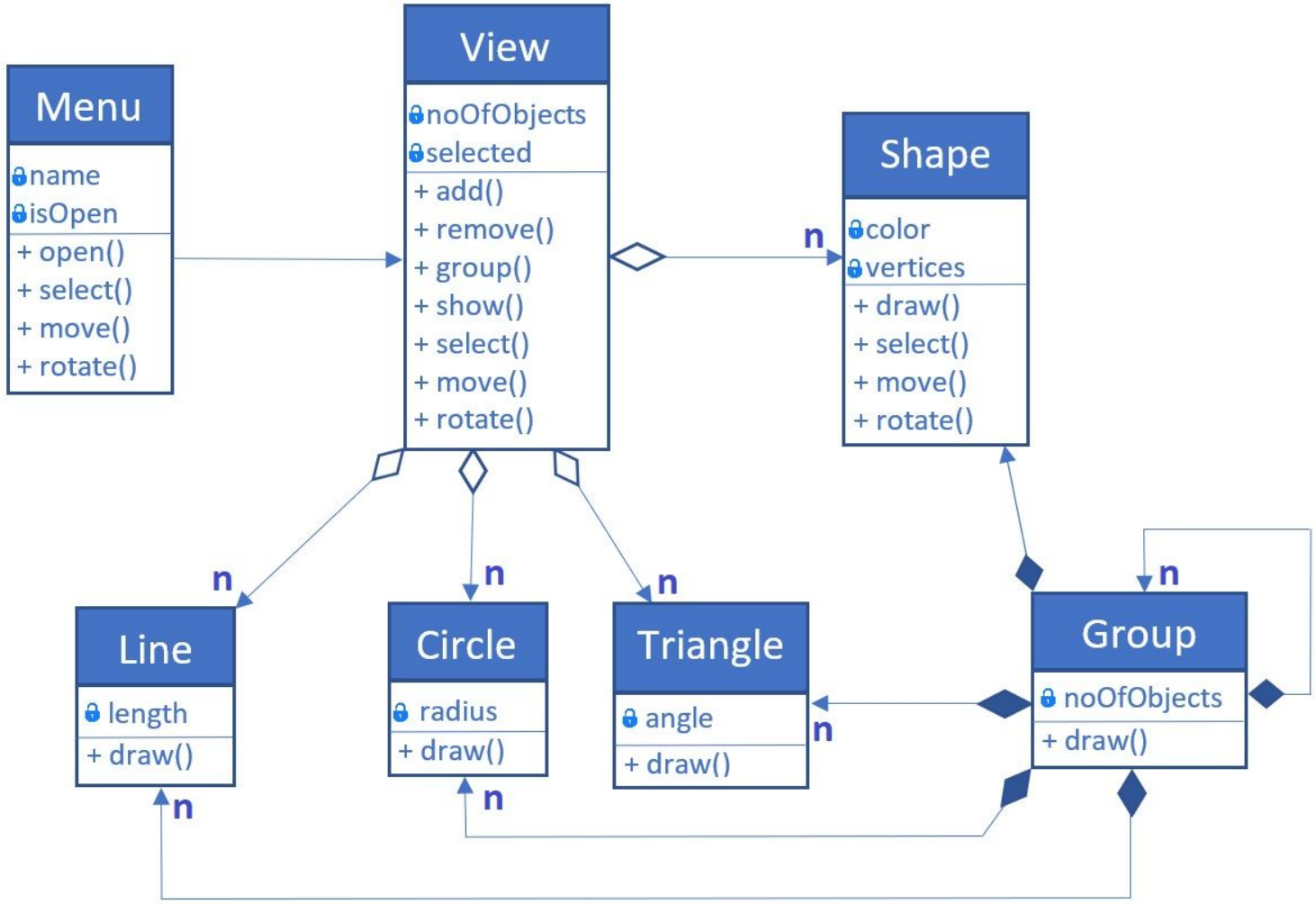
- Draw
- Select
- Move
- Rotate

Step 4 - Identify Operations

Extract operations using domain knowledge

- **View**
  - Add
  - Remove
  - Group
  - Show

Step 4 - Identify Operations



Step 5 - Identify Inheritance

Search for “is a kind of” by looking at keywords like “such as”, “for example”, etc

"…shapes such as line, circle and triangle…"

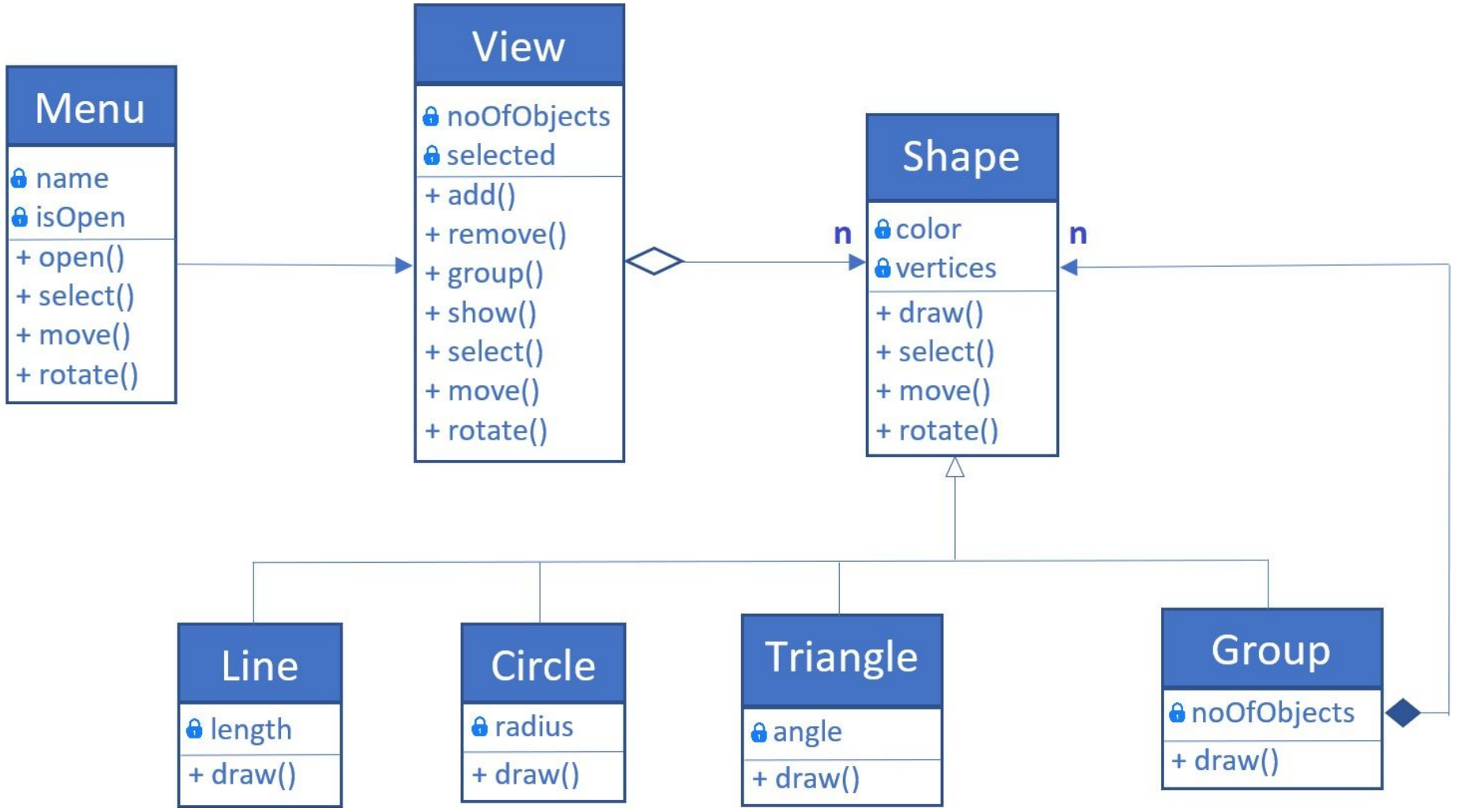
Line, Circle and Triangle inherits from Shape

By analyzing requirements

"Individual shapes can be grouped together and can behave as a single shape"

Group inherits from Shape

Final Class Diagram

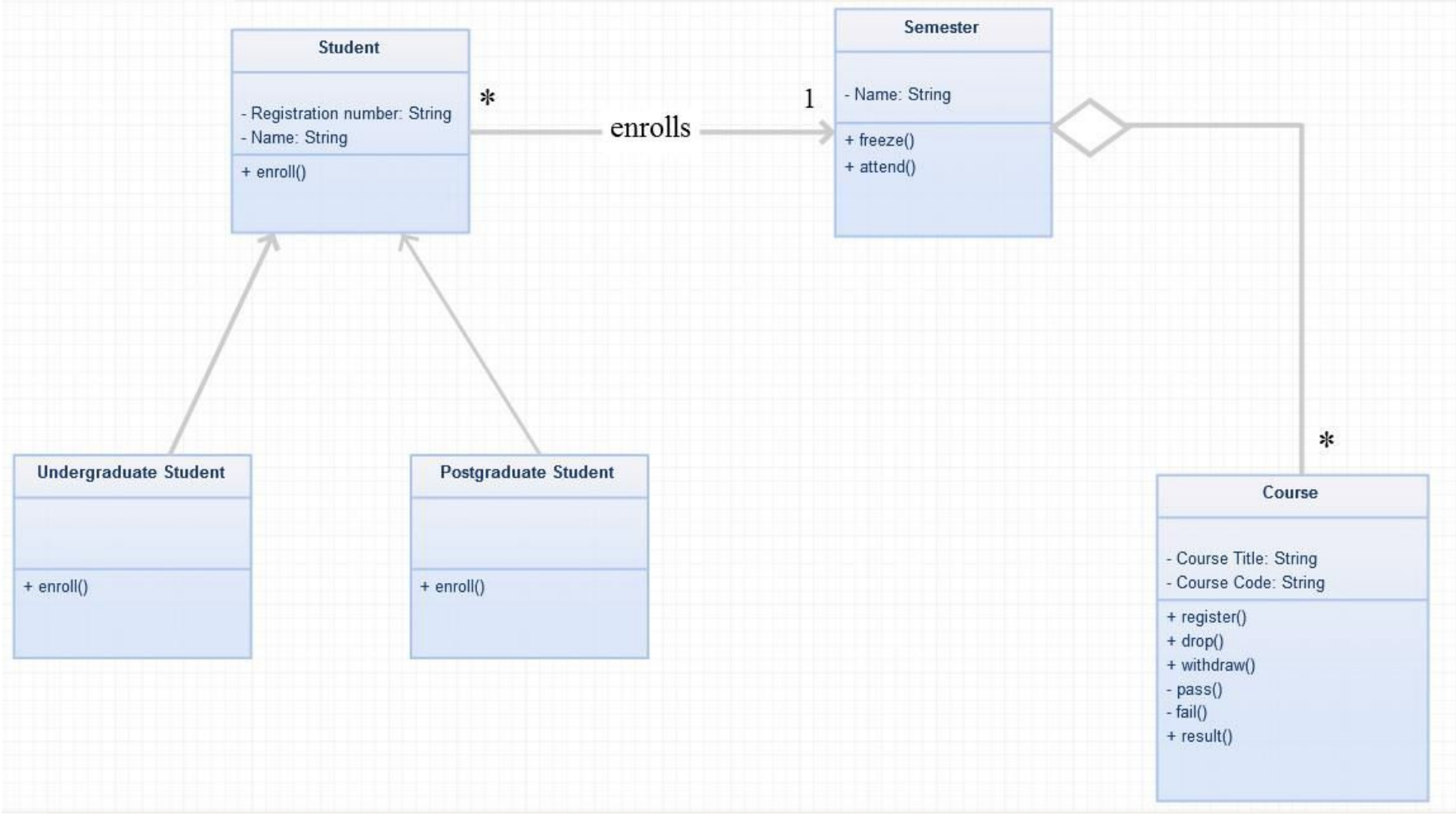


Example

Develop a Student Registration System in which student gets enroll in a semester. The semester contains courses. Each course has a title and course Code. The course can be registered, dropped, withdraw, passed and failed by the students. Also a semester may be freezed or attended. There are two kinds of students; Undergraduate Students (for BCS & MCS) and Postgraduate Students (for MPhil & PhD). Each type of students enroll in different ways.

John is a Postgraduate student who wants to enrol for the Spring ‘25 semester.

Example


Student
- Registration number: String
- Name: String
+ enroll()
*
enrolls
1
Semester
- Name: String
+ freeze()
+ attend()
Undergraduate Student
+ enroll()
Postgraduate Student
+ enroll()
*
Course
- Course Title: String
- Course Code: String
+ register()
+ drop()
+ withdraw()
- pass()
- fail()
+ result()

Example

Develop an application for a university to provide an interface of viewing course information for staff, students and managers.

The lecturers and courses details are listed in a prospectus in which courses or lecturers can be added or removed. Courses at the college comprise a number of units at two levels, with students normally taking level one units in the first year of their course and progressing level two units in the second. Part time students may take two or more years to complete all their units at a given level. Level one units are assessed on coursework grades, but level two units are assessed with examinations. Some courses also include a compulsory foundation unit where particular technical skills need to be established.

The system will provide lists of lecturers, courses and units, and access to more detailed information about which units are taught on which courses. Individual lecturer records may be queried for information such as their office room number, e-mail address and subject specialisms, as well as reporting the units that a lecturer teaches. Each course in the prospectus can be identified by a unique course code, and consists of a number of units at each level. Every course has one lecturer acting as course leader, who is responsible for administering that course and updating unit descriptions. Management functions of the prospectus allow lecturers and courses to be added to or removed from the system.

Example

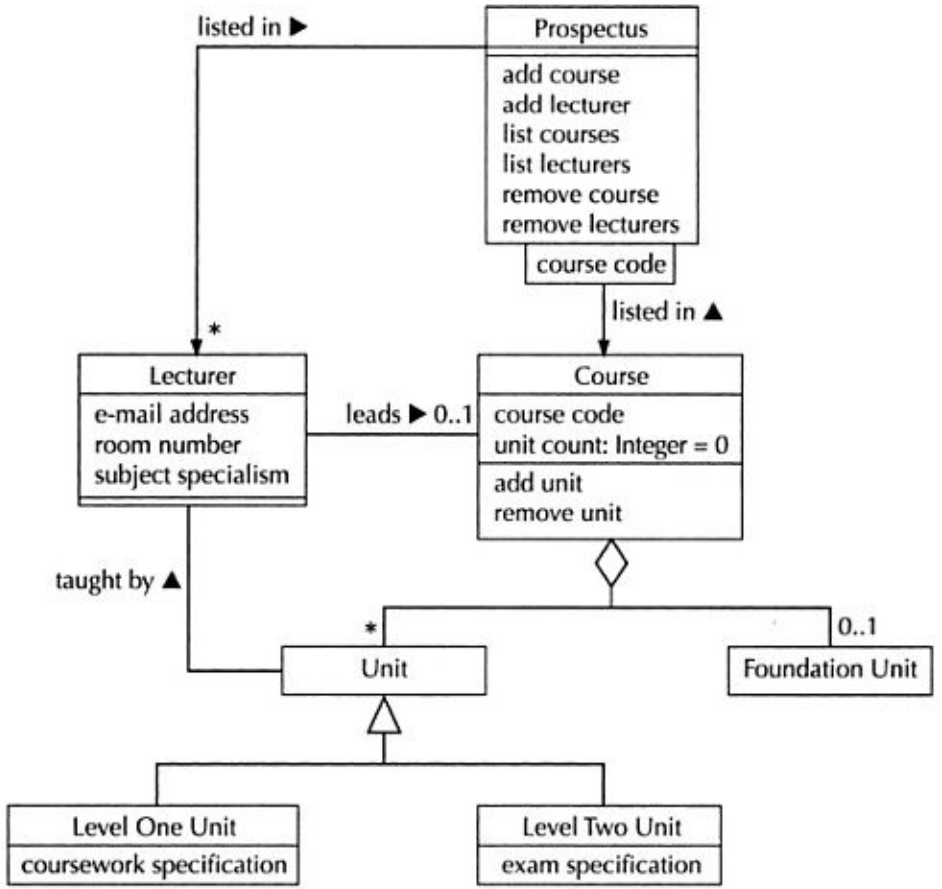
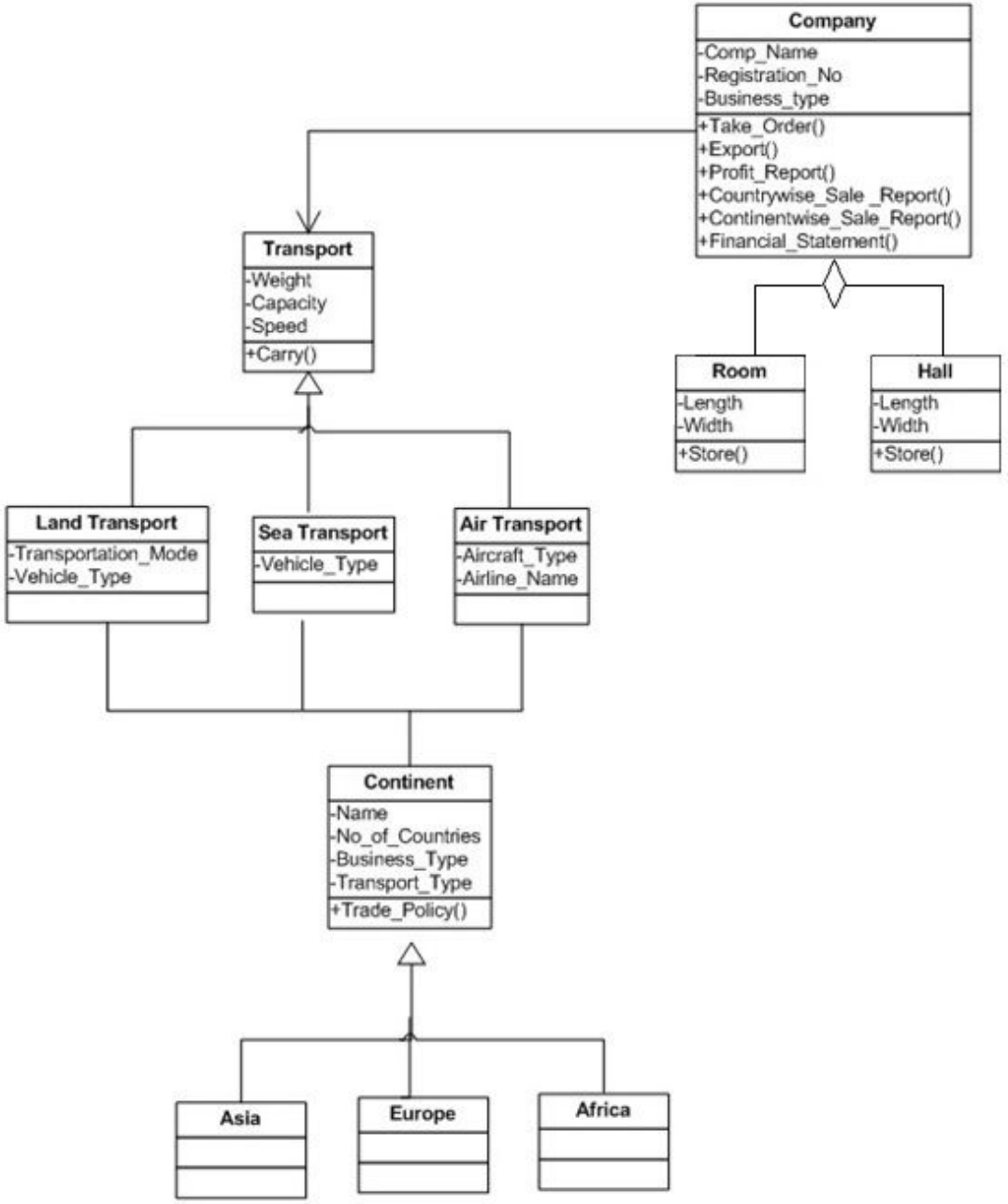


Fig. : The class diagram showing associations and their multiplicity, also aggregations, inheritance and some attributes and operations.

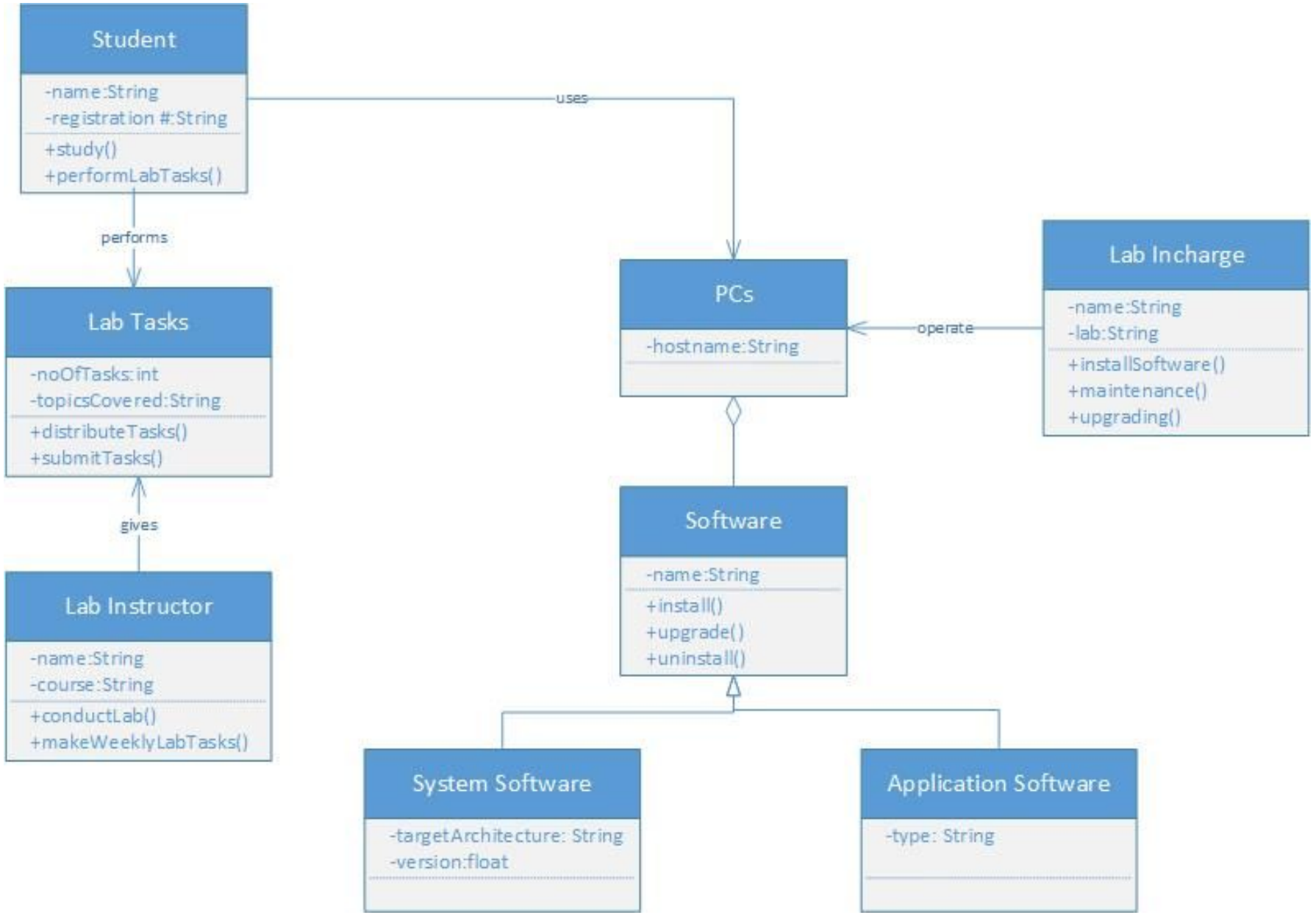
Example

Let’s assume a firm called “Banfood Internationals”, that is running an export trade business of vegetables from Bangladesh to 3 continents Africa, Asia and Europe. The company is using three kinds of transport i.e. Land transport, Sea transport, and Air transport to carry goods for shipments. It uses Land and Sea transport from Bangladesh to South East Asia, Middle East, Far East, and Africa. Moreover, it uses sea and air transport for European countries. It also has cold rooms and halls available to rent in for customers’ small and large orders respectively


Company
-Comp_Name
-Registration_No
-Business_type
+Take_Order()
+Export()
+Profit_Report()
+Countrywise_Sale _Report()
+Continentwise_Sale_Report()
+Financial_Statement()
Transport
-Weight
-Capacity
-Speed
+Carry()
Room
-Length
-Width
+Store()
Hall
-Length
-Width
+Store()
Land Transport
-Transportation_Mode
-Vehicle_Type
Sea Transport
-Vehicle_Type
Air Transport
-Aircraft_Type
-Airline_Name
Continent
-Name
-No_of_Countries
-Business_Type
-Transport_Type
+Trade_Policy()
Asia
Europe
Africa

Example

Develop a UML class diagram that models the working of computer labs in your department. It relates Students, Lab Instructor, weekly Lab Tasks, PCs, and Software installed on the PCs. There are two types of software installed i.e. System Software and Application Software. All the PCs are connected through LAN with a server which is being operated by a lab in charge. The lab in charge is also responsible for installation and maintenance of System and Application Software for all the systems in the lab.


Student
-name:String
-registration #:String
+study()
+performLabTasks()
uses
performs
Lab Tasks
-noOfTasks:int
-topicsCovered:String
+distributeTasks()
+submitTasks()
gives
Lab Instructor
-name:String
-course:String
+conductLab()
+makeWeeklyLabTasks()
PCs
-hostname:String
operate
Lab Incharge
-name:String
-lab:String
+installSoftware()
+maintenance()
+upgrading()
Software
-name:String
+install()
+upgrade()
+uninstall()
System Software
-targetArchitecture: String
-version:float
Application Software
-type: String

BRAC UNIVERSITY
Inspiring Excellence

Thank you