# SENG 350
# - Software Architecture & Design

Shuja Mughal

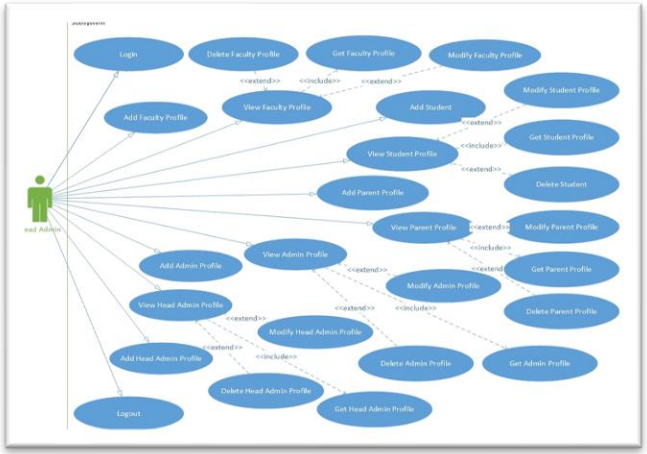
## **UML Diagrams**

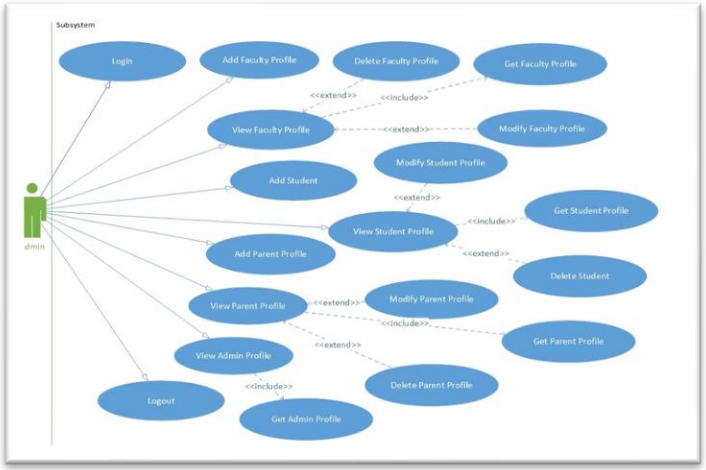
Fall 2024

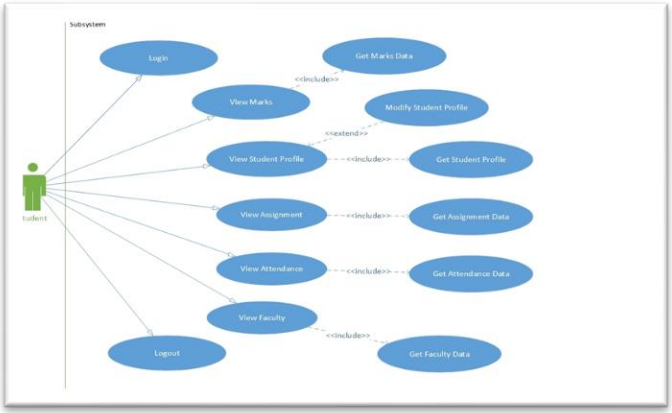
University of Victoria

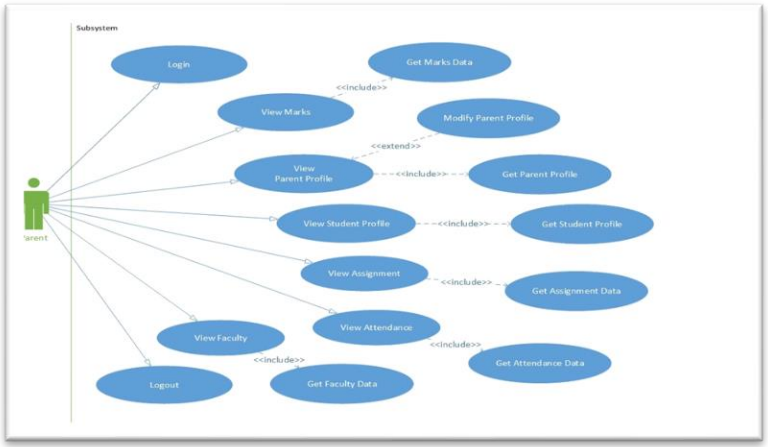
# Milestone 1 (Deadline: 1st Oct 2024)

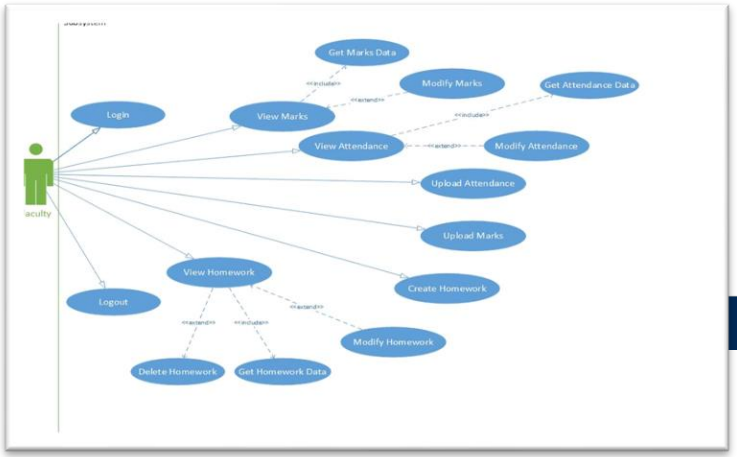
- Just a Reminder!
- The milestone 1 deadline is 1st Oct 2024 (11:00 am).
- Team Assessment 1 deadline is 1st Oct 2024 (11:00 am).



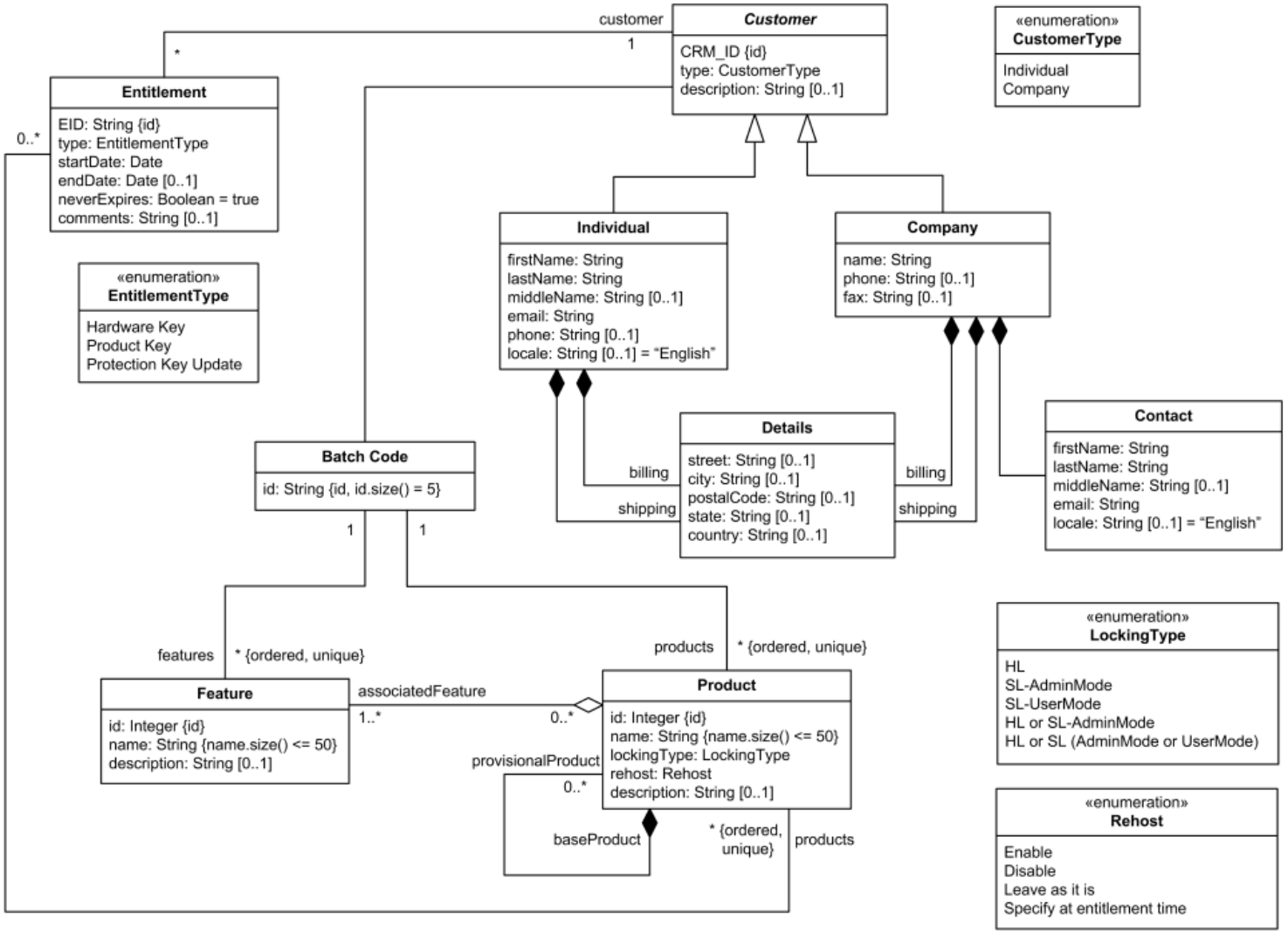




University of Victoria

# UML Diagrams

- Class diagrams
- Sequence diagrams
- Activity diagrams
- State transition diagrams
- Collaboration diagrams


# Class diagrams

University of Victoria

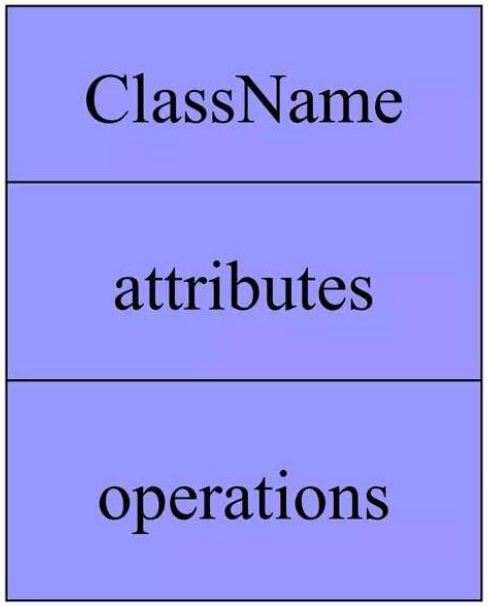
Jens H. Weber

# Class diagrams

- The purpose of a class diagram is to describe objects.
- An object is a concept, abstraction or thing that has meaning for a domain/application.
- Some objects have real-world counterparts, while others are conceptual entities.
- The choice of objects depends on the judgment and the nature of the problem.
- All objects have an identity and are distinguishable.

University of Victoria

# Classes

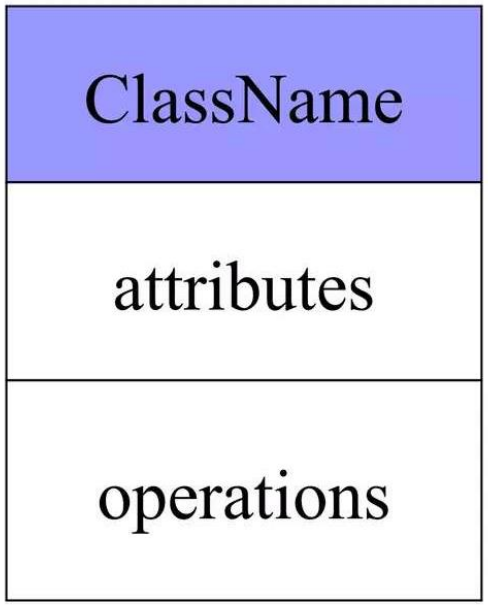



A *class* is a description of a set of objects that share the same attributes, operations, relationships, and semantics.

Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations in separate, designated compartments.

University of Victoria

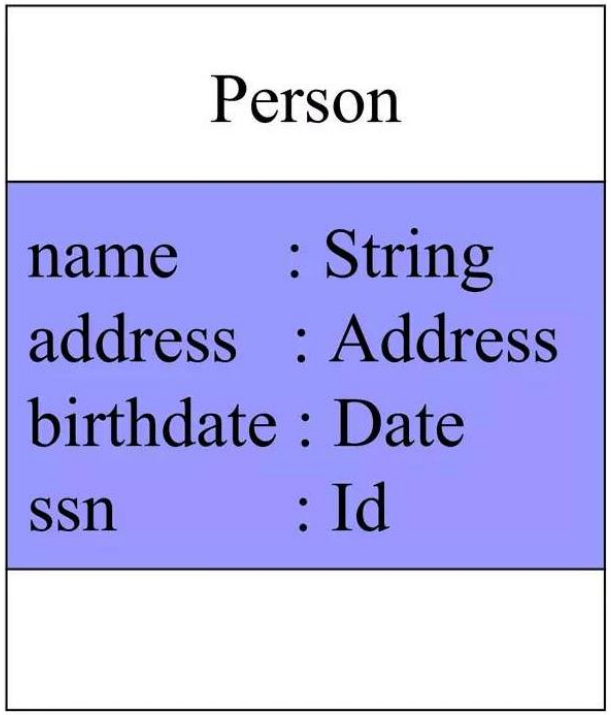
# Class Names




The name of the class is the only required tag in the graphical representation of a class. It always appears in the top-most compartment.

University of Victoria

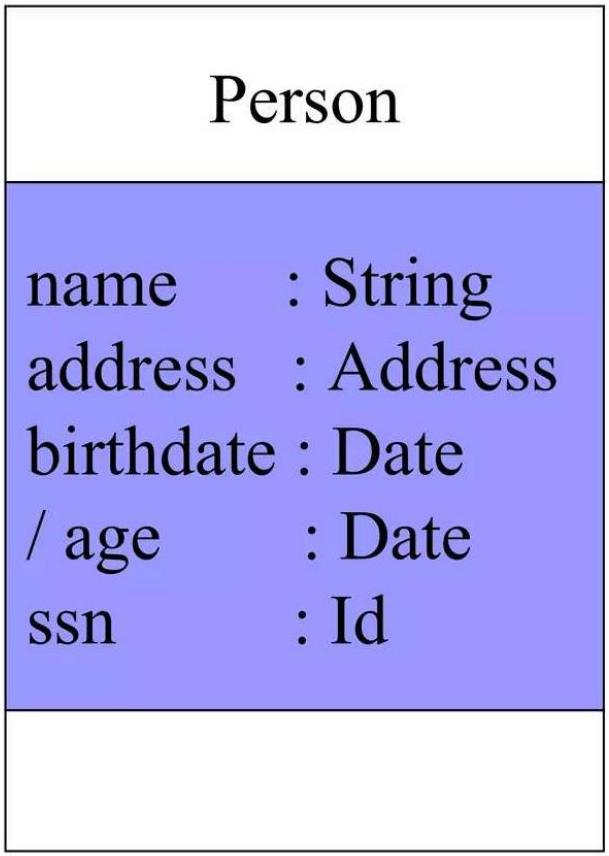
# Class Attributes




An *attribute* is a named property of a class that describes the object being modeled. In the class diagram, attributes appear in the second compartment just below the name-compartment.

University of Victoria

# Class Attributes (Cont'd)



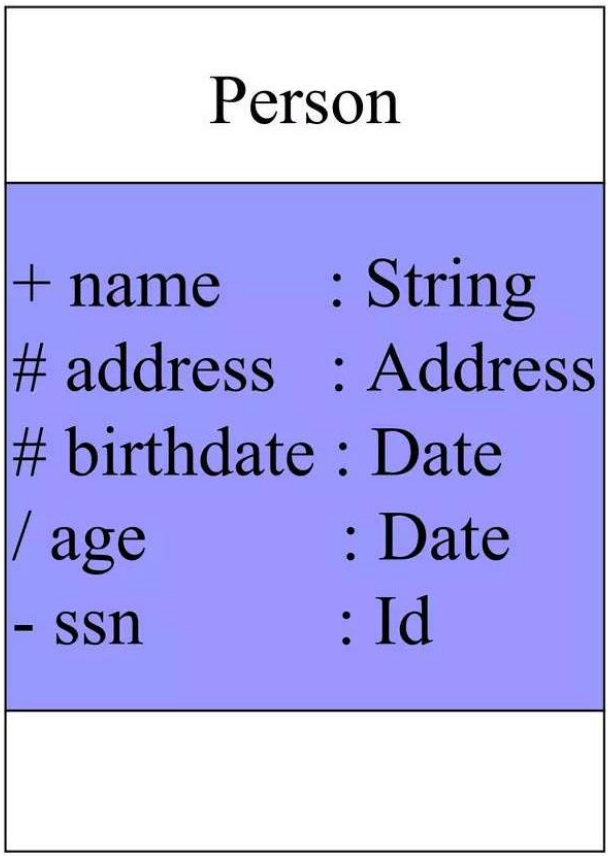

Attributes are usually listed in the form:

attributeName : Type

A *derived* attribute is one that can be computed from other attributes, but doesn't actually exist. For example, a Person's age can be computed from his birth date. A derived attribute is designated by a preceding '/' as in:

/ age : Date

# Class Attributes (Cont'd)

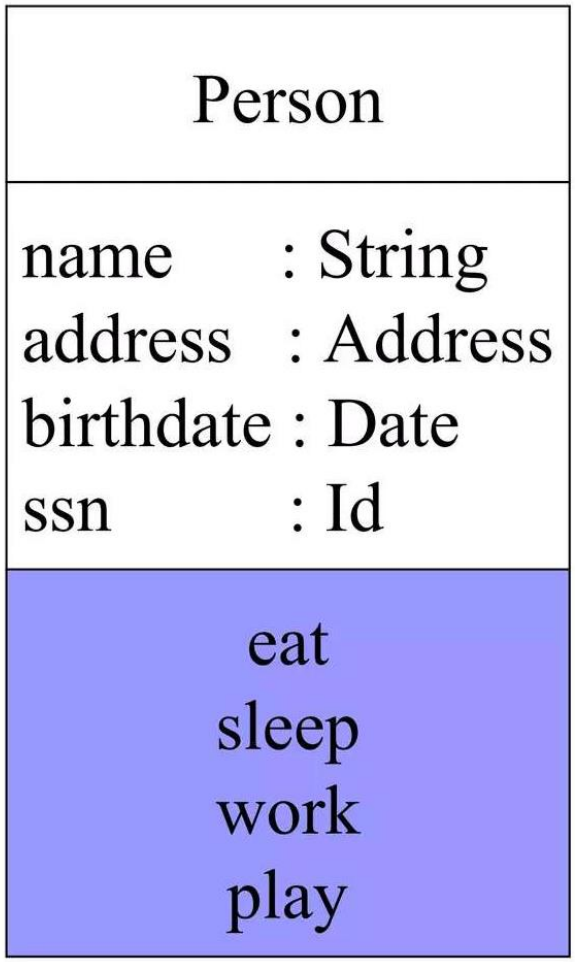



Attributes can be:

+ public
# protected
- private
/ derived

University of Victoria

# Class Operations




*Operations* describe the class behavior and appear in the third compartment.

# UML representation of classes/objects:

UML: Unified Modeling Language (OMG Standard): Visual Modeling language

Class/object representation

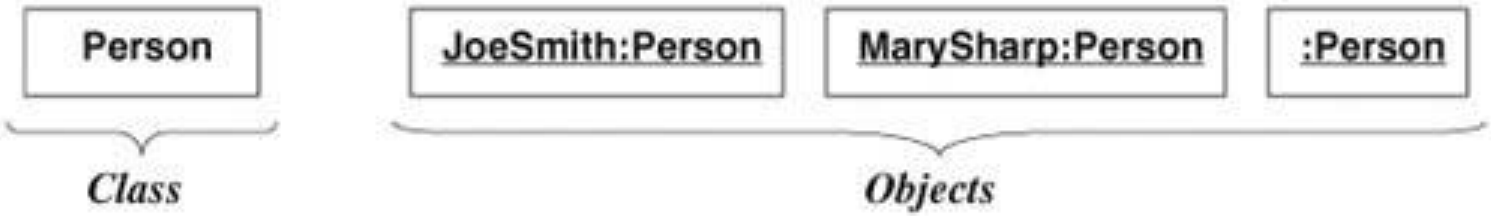



**Figure 3.1 A class and objects.** Objects and classes are the focus of class modeling.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

# UML representation

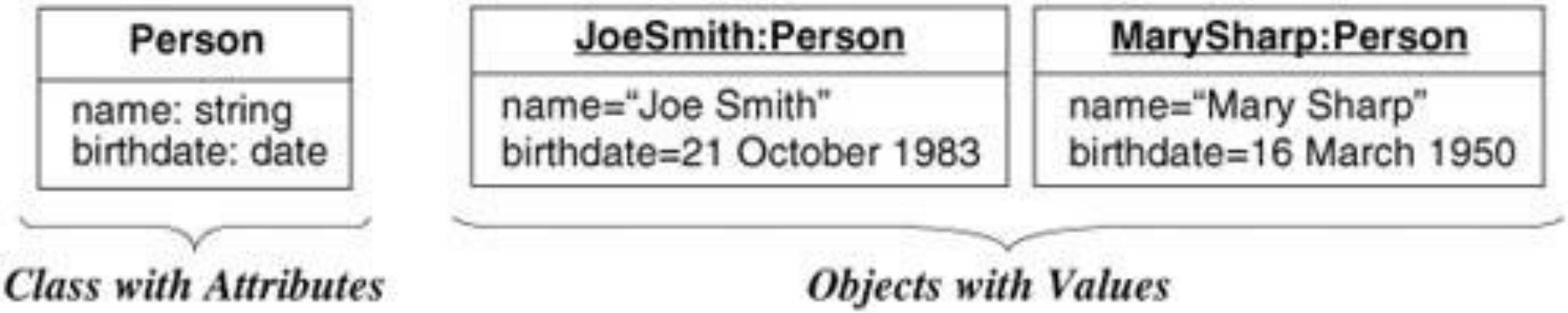



**Figure 3.2 Attributes and values.** Attributes elaborate classes.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

# Object identifiers

- Object identifiers are implicit.
- Objects belonging to the same and having the same attribute values may be different individual objects.

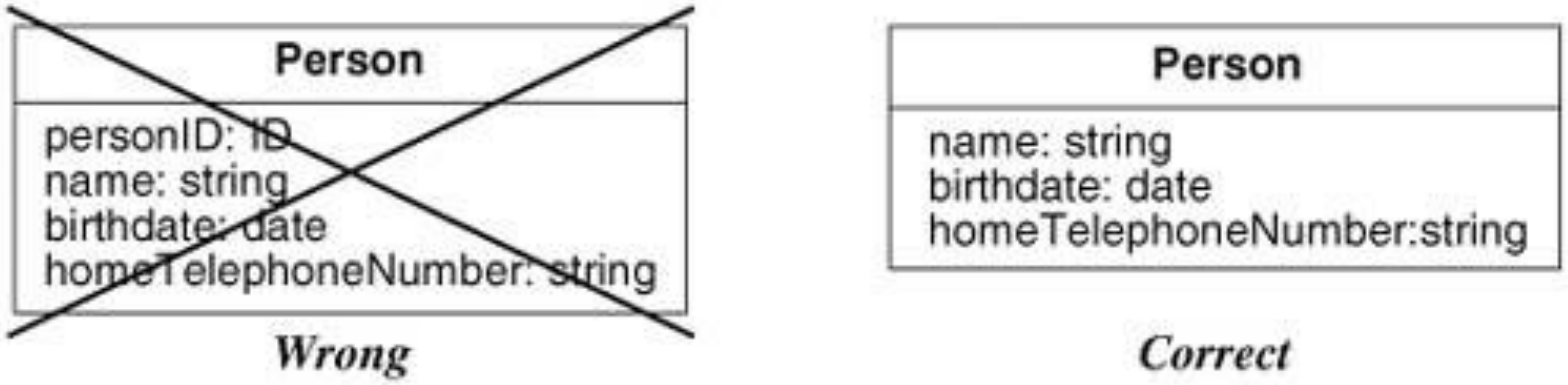



**Figure 3.3 Object identifiers.** Do not list object identifiers; they are implicit in models.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

# Operations/Methods

- An operation is a function or procedure that may be applied to or by objects in a class.
- Each operation has a target object as an implicit parameter.
- All objects in a class share the same operations.

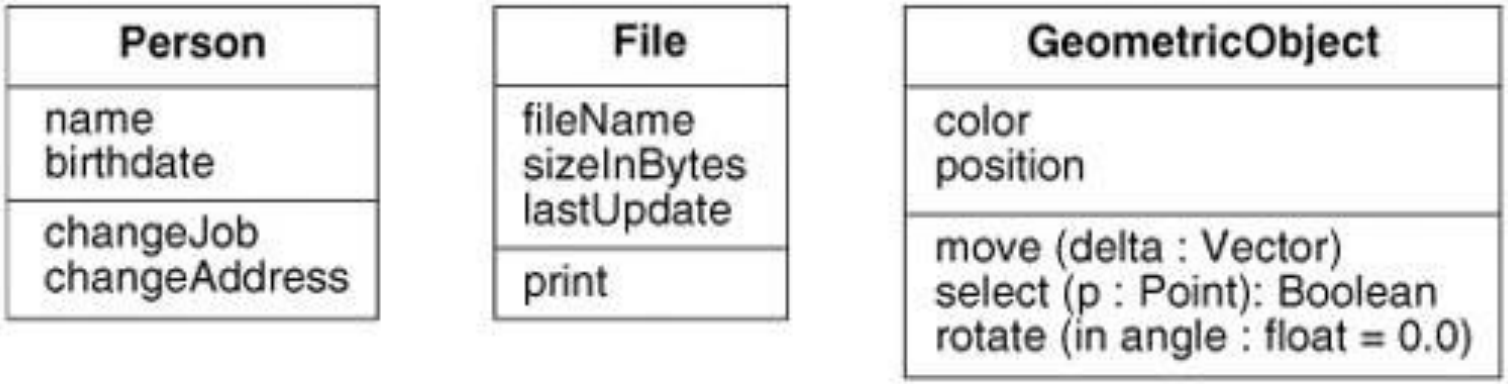



**Figure 3.4 Operations.** An operation is a function or procedure that may be applied to or by objects in a class.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

# Links and Association concepts

- A link is a physical or conceptual connection among objects.
- Most links relate to two objects, but some links relate to three or more objects.
- A link is an instance of an association.
- An association describes a group of links with common structure and semantics.
- Association is denoted by a line. Its name is optional if the model is unambiguous.

University of Victoria

# Examples

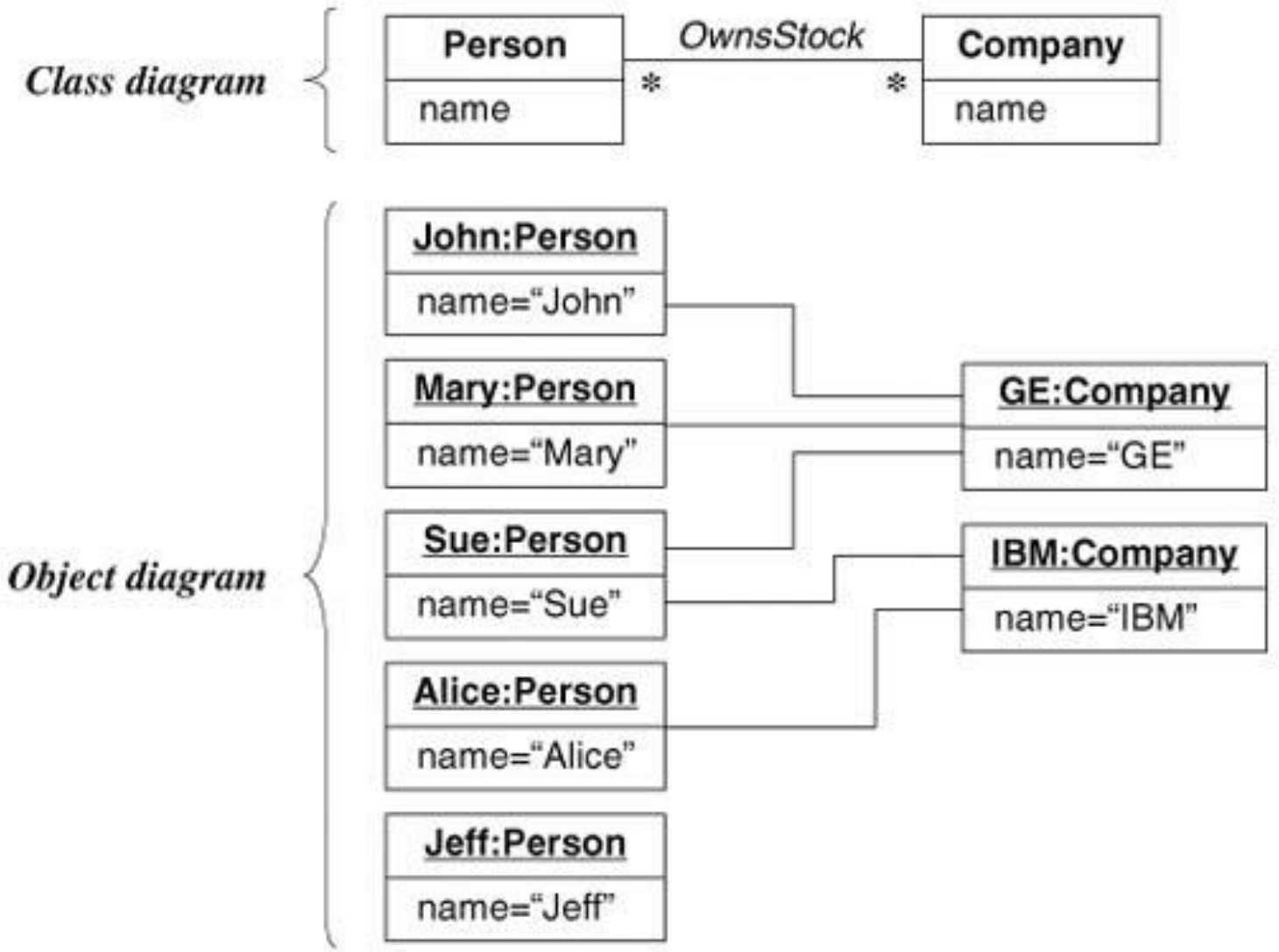



**Figure 3.7 Many-to-many association.** An association describes a set of potential links in the same way that a class describes a set of potential objects.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

# Associations and Multiplicity

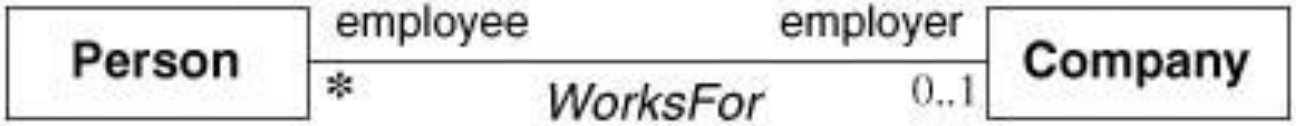
- **Associations** are inherently bi-directional. The association name is usually read in a particular direction, but the binary association may be traversed in either direction.

- **Multiplicity** specifies the number of instances of one class that may relate to a single instance of the associated class.
- UML diagrams explicitly list multiplicity at the end of association lines.
  - Intervals are used to express multiplicity:
  - 1 (exactly one)
  - 0..1 (zero or one)
  - 1..* (one or more)
  - 0..* (zero or more)
  - 3..5 (three to five inclusive)

University of Victoria

# Association Ends

- Associations have names at each end.
- They are called ‘Association Ends’.
- They may have names (which often appear in problem descriptions).




| employee | employer |
|---|---|
| Joe Doe | Simplex |
| Mary Brown | Simplex |
| Jean Smith | United Widgets |

**Figure 3.12 Association end names.** Each end of an association can have a name.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

# Association Ends

- The use of association end names is optional.
- However, association end names are useful for traversing associations.
- Association end names are necessary for associations between objects of the same class.

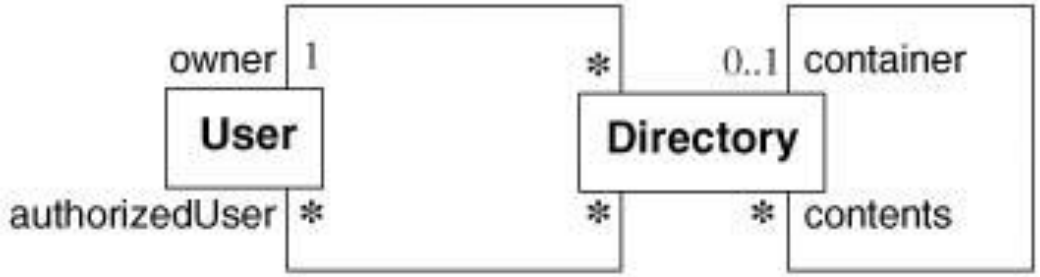



**Figure 3.13 Association end names.** Association end names are necessary for associations between two objects of the same class. They can also distinguish multiple associations between a pair of classes.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4.
© 2005 Pearson Education, Inc., Upper Saddle River, NJ.
All rights reserved.

University of Victoria

# Example of association ends use

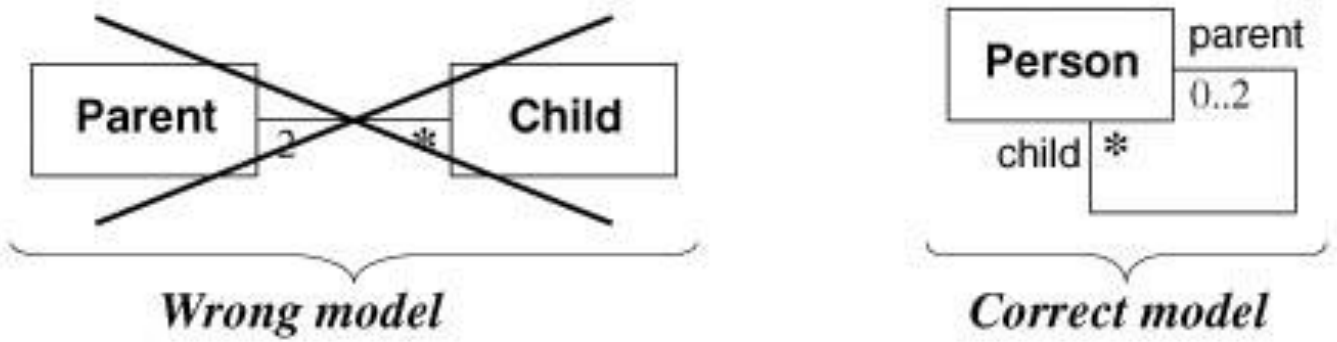



**Figure 3.14 Association end names.** Use association end names to model multiple references to the same class.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

# Association class

- UML offers the ability to describe links of association with attributes like any class.
- An association class is an association that is also a class.

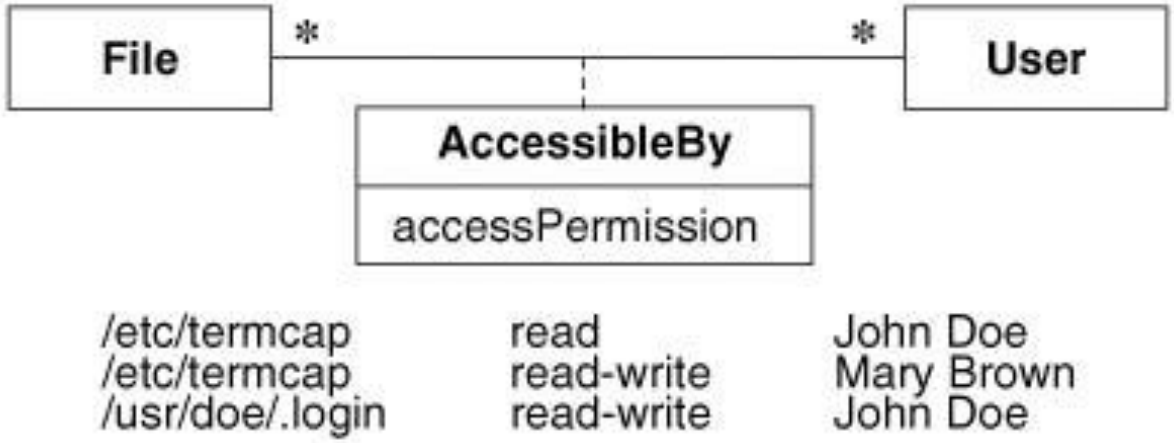



**Figure 3.17 An association class.** The links of an association can have attributes.

*Object-Oriented Moldeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

# Association class

Examples:

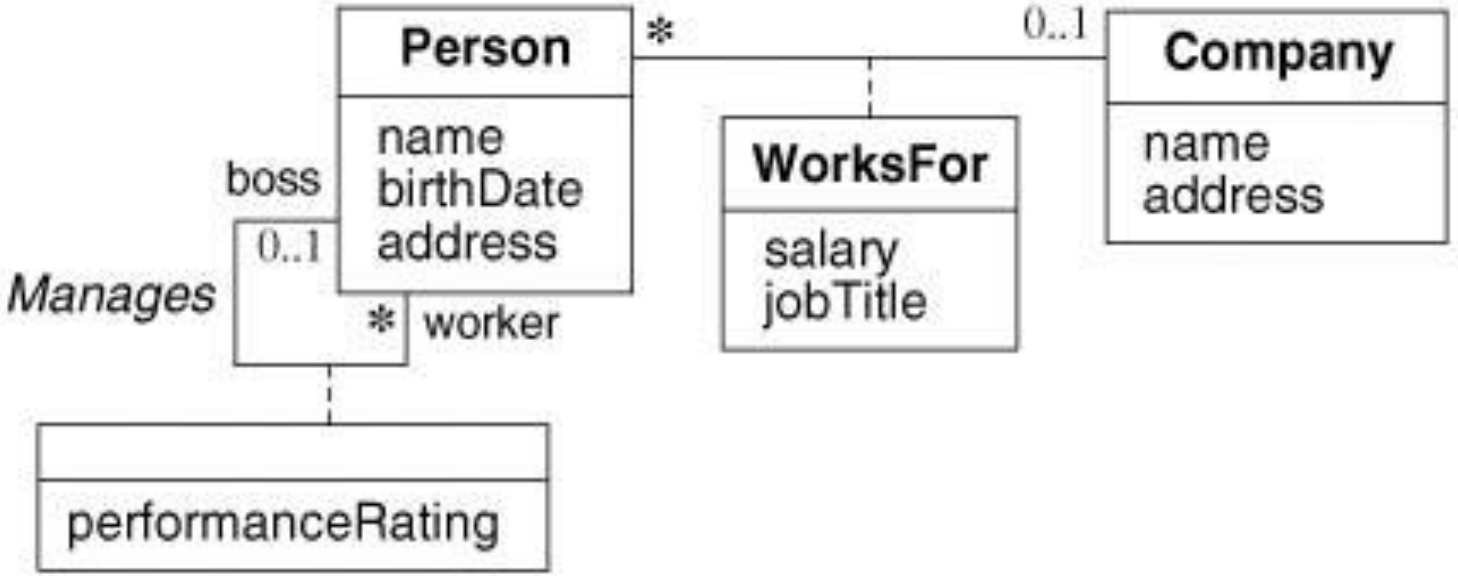



**Figure 3.18 Association classes.** Attributes may also occur for one-to-many and one-to-one associations.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

# Association class

Example

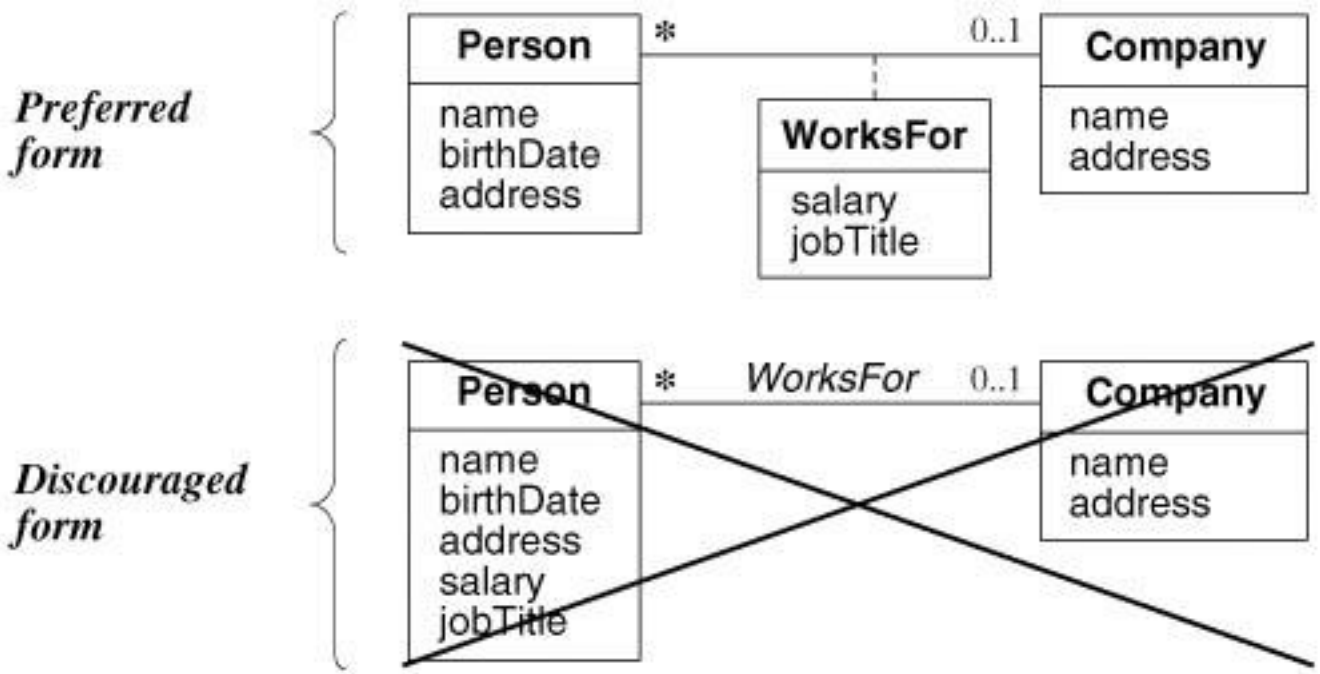



**Figure 3.19 Proper use of association classes.** Do not fold attributes of an association into a class.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh, ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

# Generalization/Inheritance

- Generalization is the relationship between a class (**superclass**) and one or more **variations** of the class (**subclasses**).
- Generalization organizes classes by **similarities** and differences, structuring objects' descriptions.
- A superclass holds **common** attributes, attributes and associations.
- The subclasses **add specific** attributes, operations, and associations. They i**nherit** the features of their superclass.
- Often, **Generalization** is called an "**IS A"** relationship
- **Simple generalization** organizes classes into a **hierarchy**.
- A subclass may **override** a superclass **feature (**attribute default values, operation) by **redefining a feature with the same name**.

University of Victoria

# Generalization/Inheritance

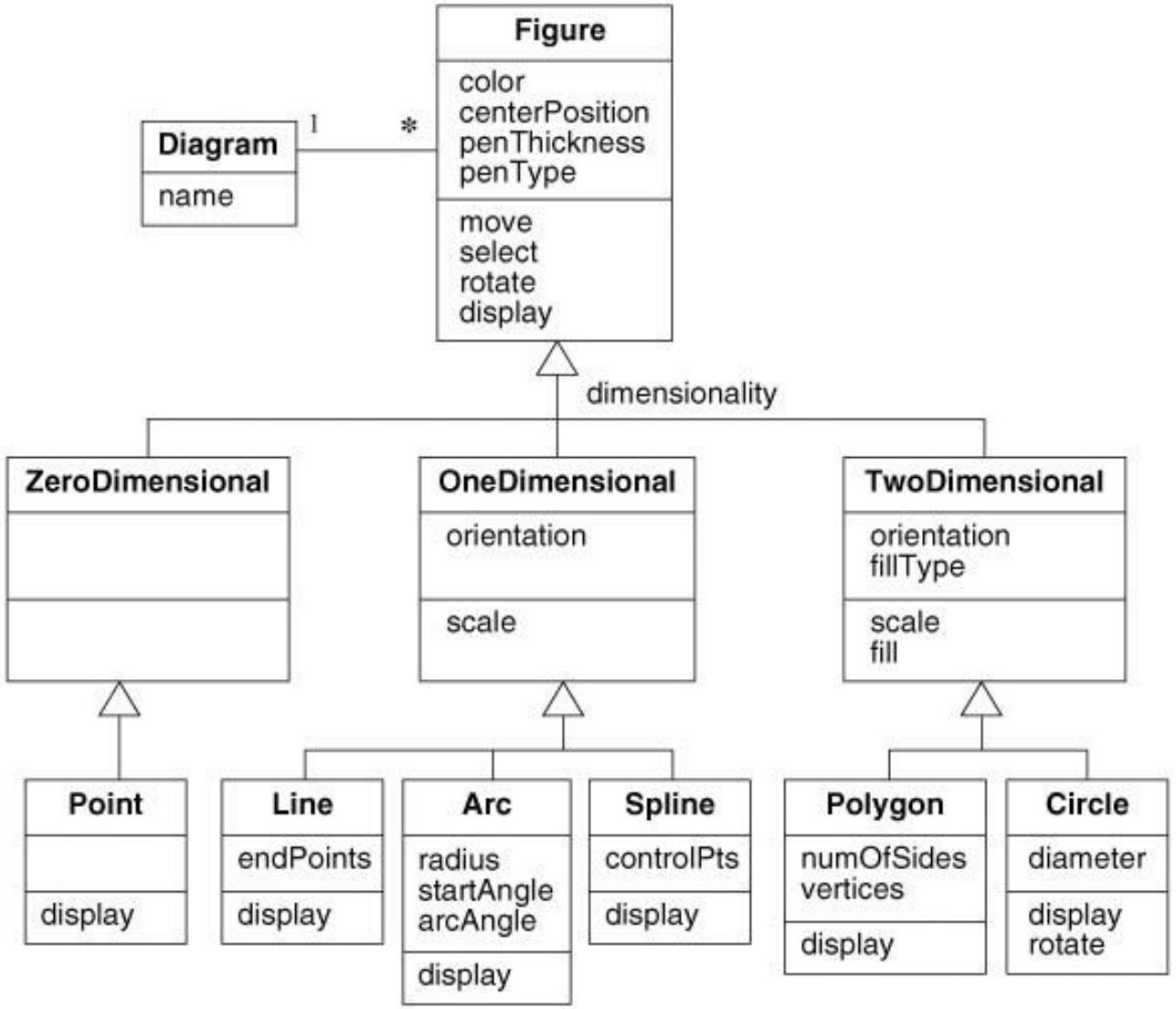



**Figure 3.25 Inheritance for graphic figures.** Each subclass inherits the attributes, operations, and associations of its superclasses.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh.
ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

# Use of generalization

**Used for three purposes:**

- Support of polymorphism:
  - Polymorphism increases the flexibility of software.
  - Adding a new subclass and automatically inheriting superclass behavior.
- Structuring the description of objects:
  - Forming classification and organizing objects according to their similarities.
- Enabling code reuse:
  - Reuse is more productive than repeatedly writing code from scratch.

University of Victoria

# Aggregation

- **Aggregation** is a **strong** form of **association.**
- **Aggregation is a transitive relation:**
  - if A is a part of B and B is a part of C, then A is also a part of C
- **Aggregation is an antisymmetric relation:**
  - If A is a part of B then B is not a part of A.

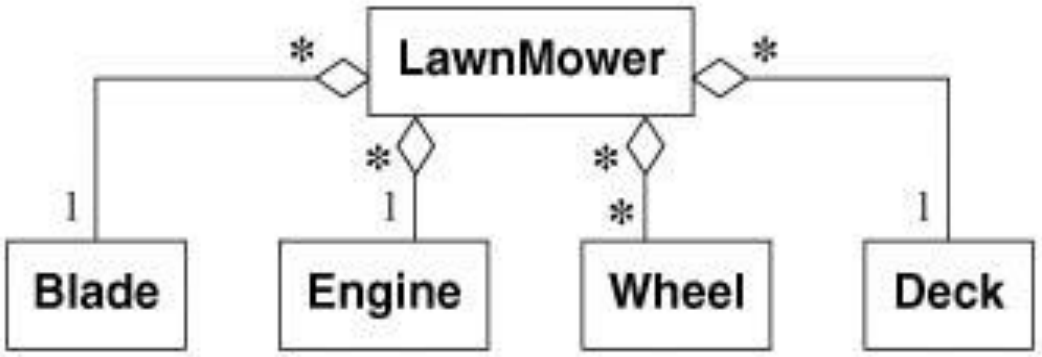



**Figure 4.9 Aggregation.** Aggregation is a kind of association in which an aggregate object is made of constituent parts.

*Object-Oriented Modeling and Design with UML*, Second Edition
by Michael Blaha and James Rumbaugh.
ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc.,
Upper Saddle River, NJ. All rights reserved.

University of Victoria

# Composition

- **Composition** is a form of aggregation with additional constraints.
- Composition is a strong form of aggregation where:
  - Deletion of a "whole" triggers automatic deletion of all its "parts." (Lifetime)

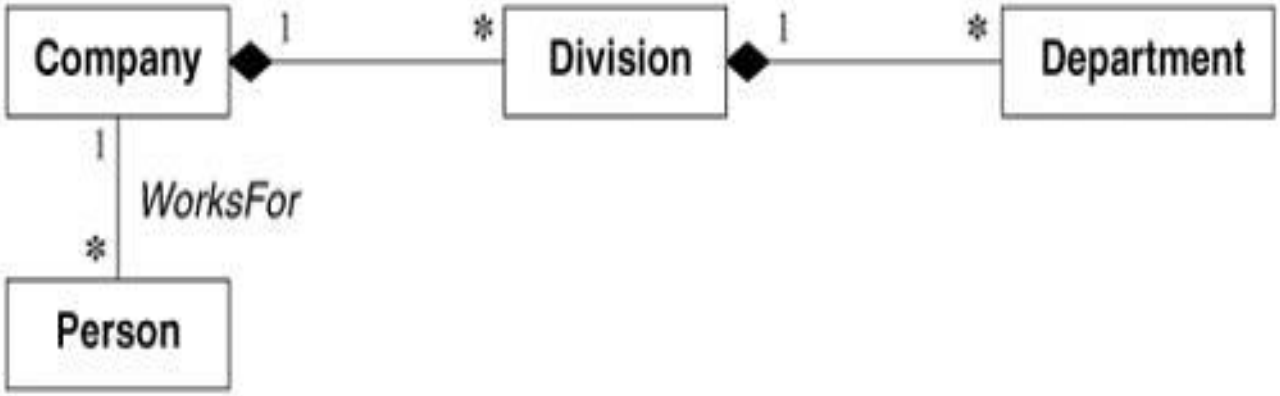



**Figure 4.10 Composition.** With composition a constituent part belongs to at most one assembly and has a coincident lifetime with the assembly.

*Object-Oriented Modeling and Design with UML*, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

University of Victoria

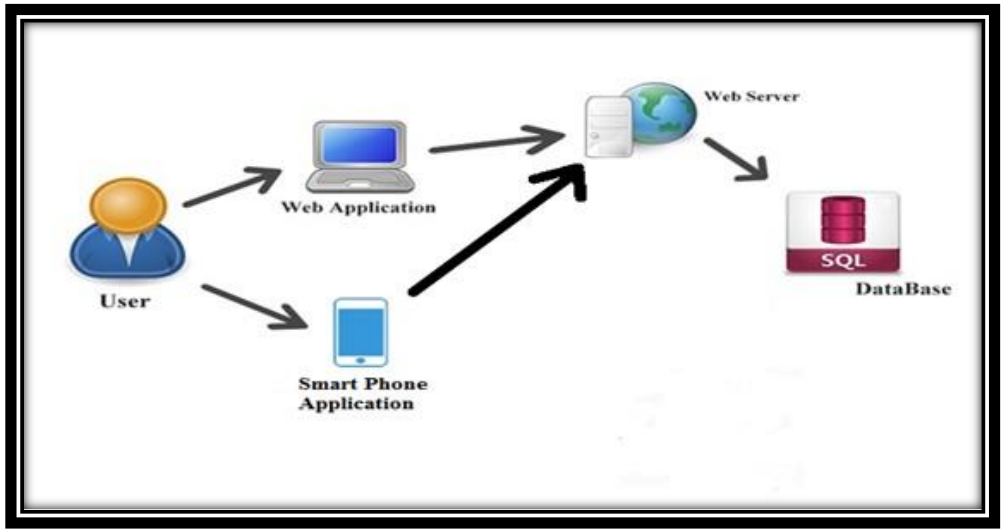
# Case Study

University of Victoria

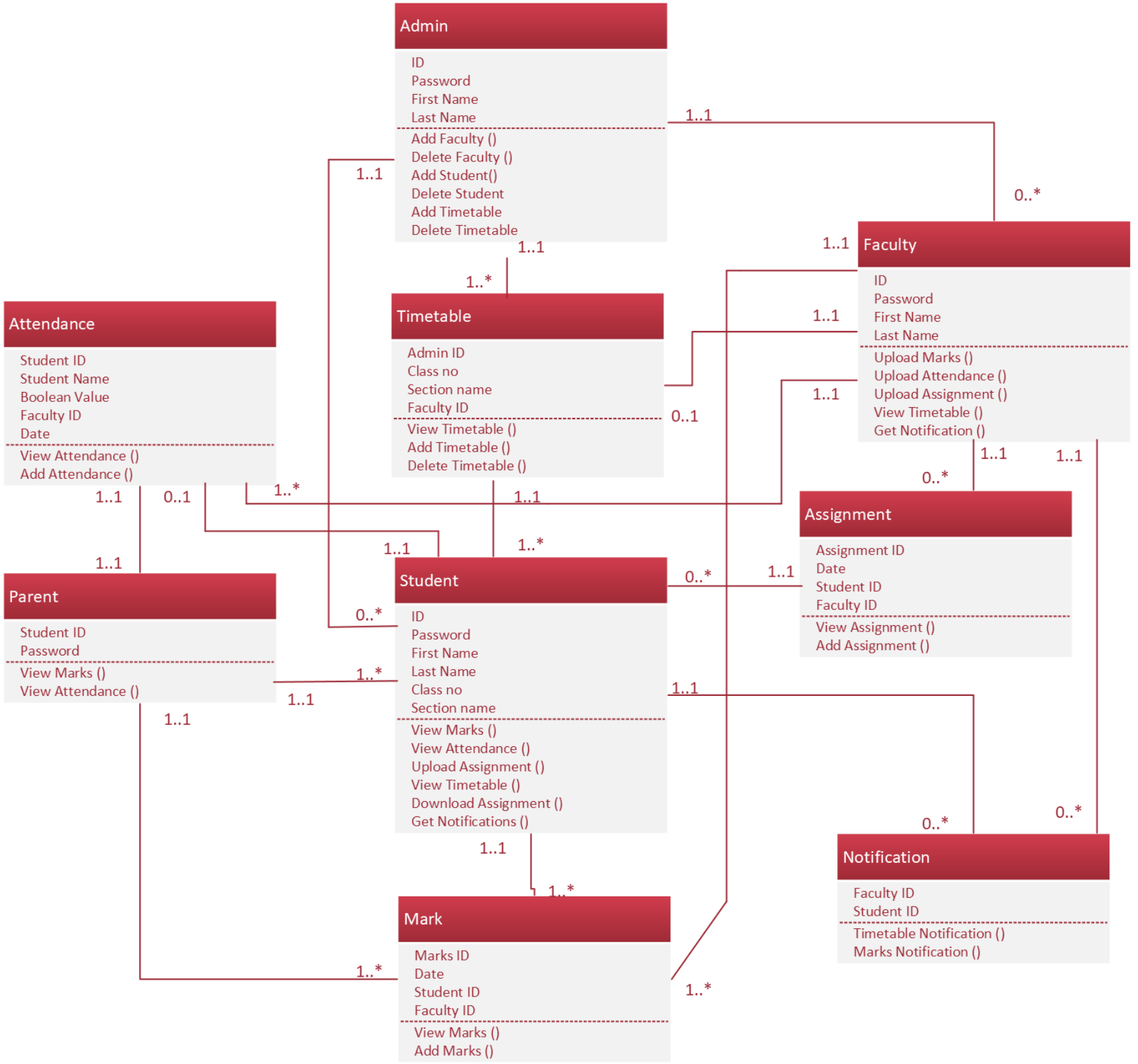
Jens H. Weber

# Case Study

Online School Portal (OSP) is a smartphone and web-based application that targets schools. It facilitates students, teachers, administration and parents. This system provides a platform that automates all the work, which is much more efficient than the legacy systems. In this software, each stakeholder can perform certain tasks to get their job done. Moreover, the application provides ease of use, implementing features of human-computer interaction. OSP includes a website and a smartphone application. The smartphone application and website are combined with a mutual server and a database.

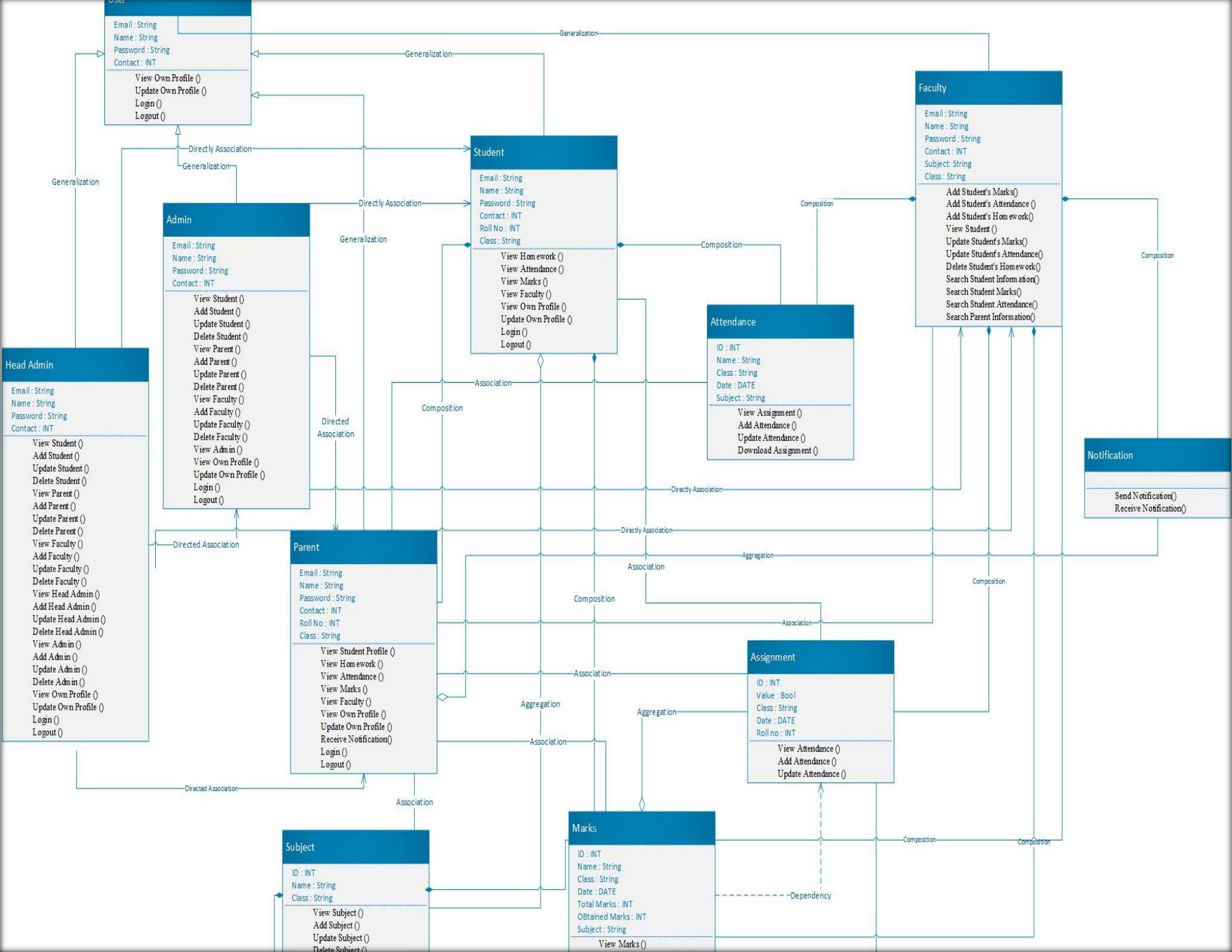



University of Victoria




# Case Study

University of Victoria


User
Email : String
Name : String
Password : String
Contact : INT
View Own Profile ()
Update Own Profile ()
Login ()
Logout ()
Generalization
Generalization
Generalization
Generalization
Directly Association
Directly Association
Student
Email : String
Name : String
Password : String
Contact : INT
Roll No : INT
Class : String
View Homework ()
View Attendance ()
View Marks ()
View Faculty ()
View Own Profile ()
Update Own Profile ()
Login ()
Logout ()
Faculty
Email : String
Name : String
Password : String
Contact : INT
Subject: String
Class : String
Add Student's Marks()
Add Student's Attendance ()
Add Student's Homework()
View Student ()
Update Student's Marks()
Update Student's Attendance()
Delete Student's Homework()
Search Student Information()
Search Student Marks()
Search Student Attendance()
Search Parent Information()
Admin
Email : String
Name : String
Password : String
Contact : INT
View Student ()
Add Student ()
Update Student ()
Delete Student ()
View Parent ()
Add Parent ()
Update Parent ()
Delete Parent ()
View Faculty ()
Add Faculty ()
Update Faculty ()
Delete Faculty ()
View Admin ()
View Own Profile ()
Update Own Profile ()
Login ()
Logout ()
Composition
Composition
Composition
Head Admin
Email : String
Name : String
Password : String
Contact : INT
View Student ()
Add Student ()
Update Student ()
Delete Student ()
View Parent ()
Add Parent ()
Update Parent ()
Delete Parent ()
View Faculty ()
Add Faculty ()
Update Faculty ()
Delete Faculty ()
View Head Admin ()
Add Head Admin ()
Update Head Admin ()
Delete Head Admin ()
View Admin ()
Add Admin ()
Update Admin ()
Delete Admin ()
View Own Profile ()
Update Own Profile ()
Login ()
Logout ()
Attendance
ID : INT
Name : String
Class : String
Date : DATE
Subject : String
View Assignment ()
Add Attendance ()
Update Attendance ()
Download Assignment ()
Association
Composition
Directed Association
Directed Association
Directly Association
Directly Association
Directed Association
Notification
Send Notification()
Receive Notification()
Aggregation
Association
Composition
Composition
Parent
Email : String
Name : String
Password : String
Contact : INT
Roll No : INT
Class : String
View Student Profile ()
View Homework ()
View Attendance ()
View Marks ()
View Faculty ()
View Own Profile ()
Update Own Profile ()
Receive Notification()
Login ()
Logout ()
Association
Association
Assignment
ID : INT
Value : Bool
Class : String
Date : DATE
Roll no : INT
View Attendance ()
Add Attendance ()
Update Attendance ()
Aggregation
Aggregation
Association
Directed Association
Association
Subject
ID : INT
Name : String
Class : String
View Subject ()
Add Subject ()
Update Subject ()
Marks
ID : INT
Name : String
Class : String
Date : DATE
Total Marks : INT
OBtained Marks : INT
Subject : String
View Marks ()
Dependency
Composition
Composition

# Upcoming Lecture

- Sequence diagrams
- Activity diagrams
- State transition diagrams
- Collaboration diagrams

University of Victoria