SENG2130 – Week 4
Class and Object Diagrams

Dr. Joe Ryan
SENG2130 – Systems Analysis and Design
University of Newcastle

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

FACULTY OF ENGINEERING AND BUILT ENVIRONMENT

www.newcastle.edu.au

UML diagram

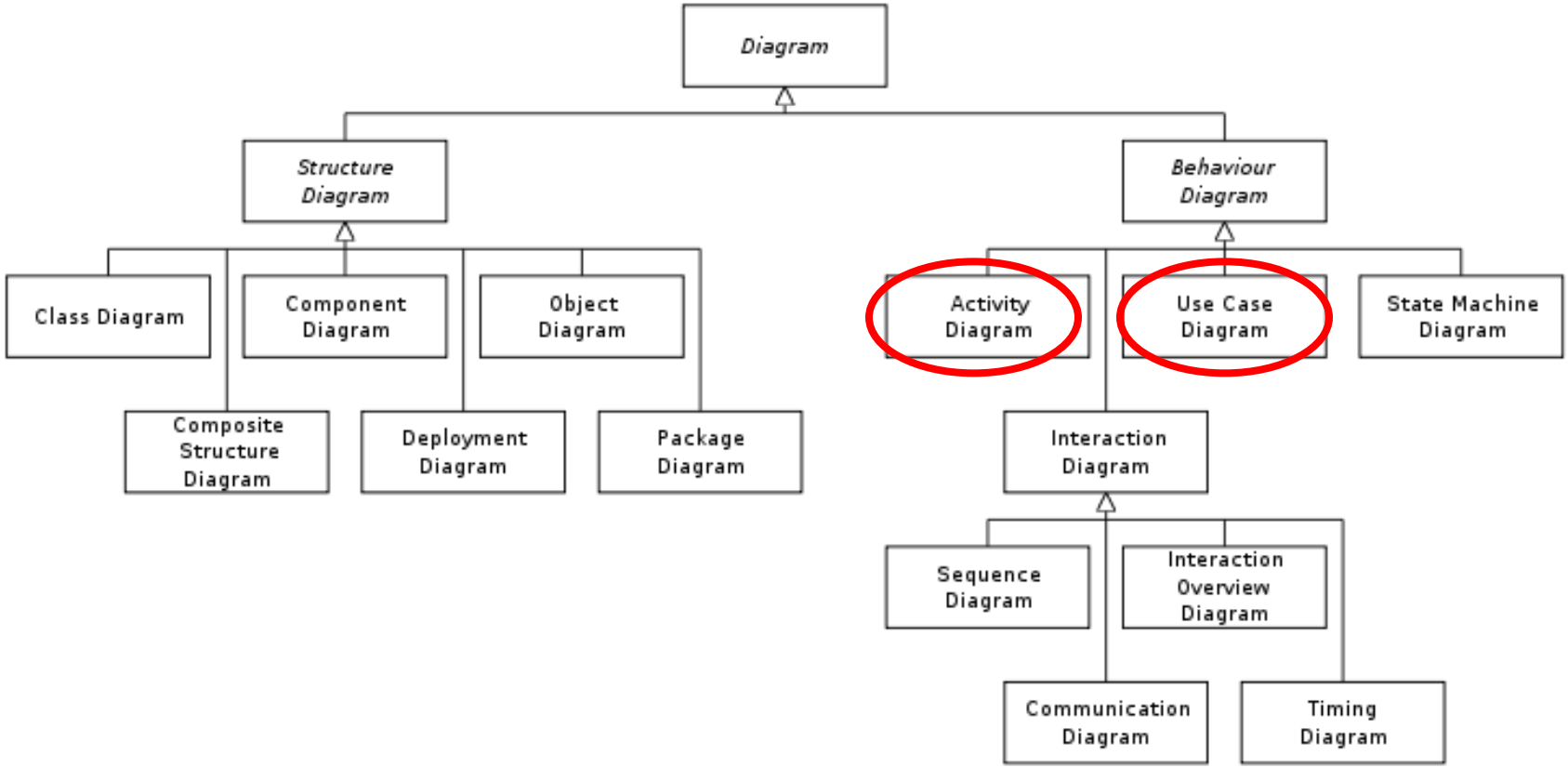
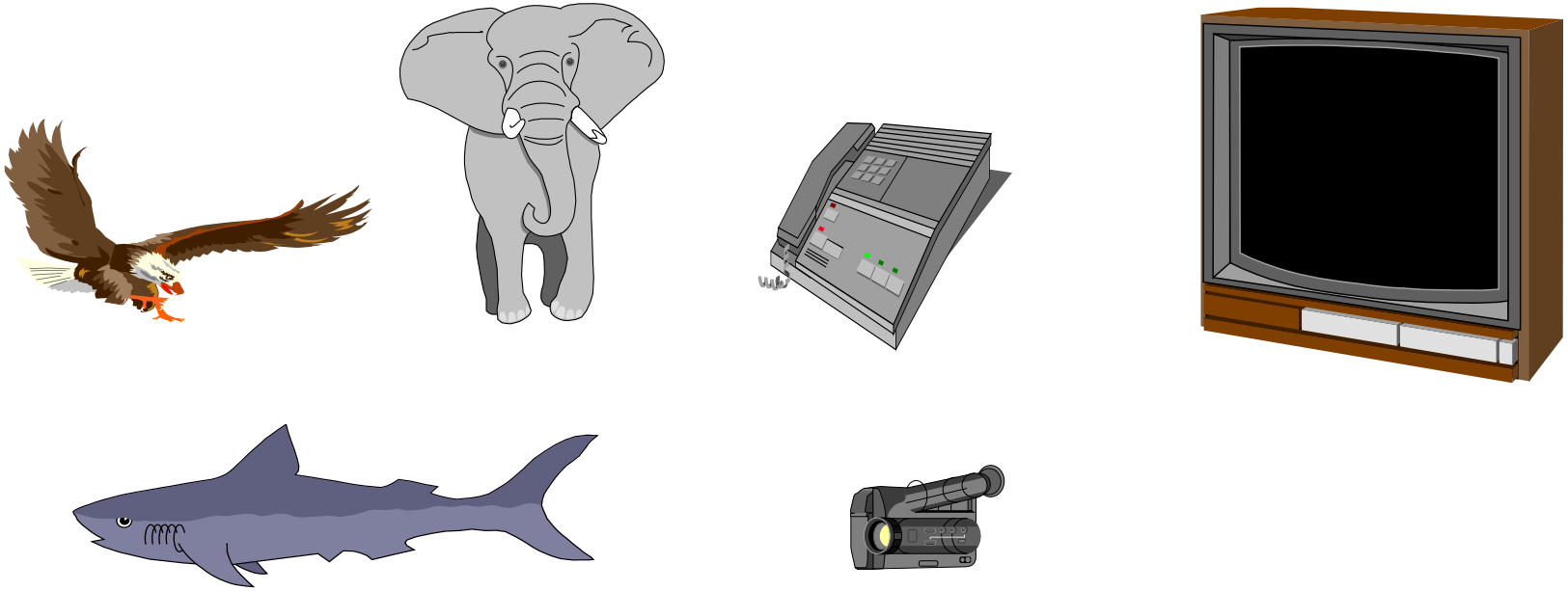


Diagram
Structure Diagram
Behaviour Diagram
Class Diagram
Component Diagram
Object Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Interaction Diagram
Sequence Diagram
Interaction Overview Diagram
Communication Diagram
Timing Diagram

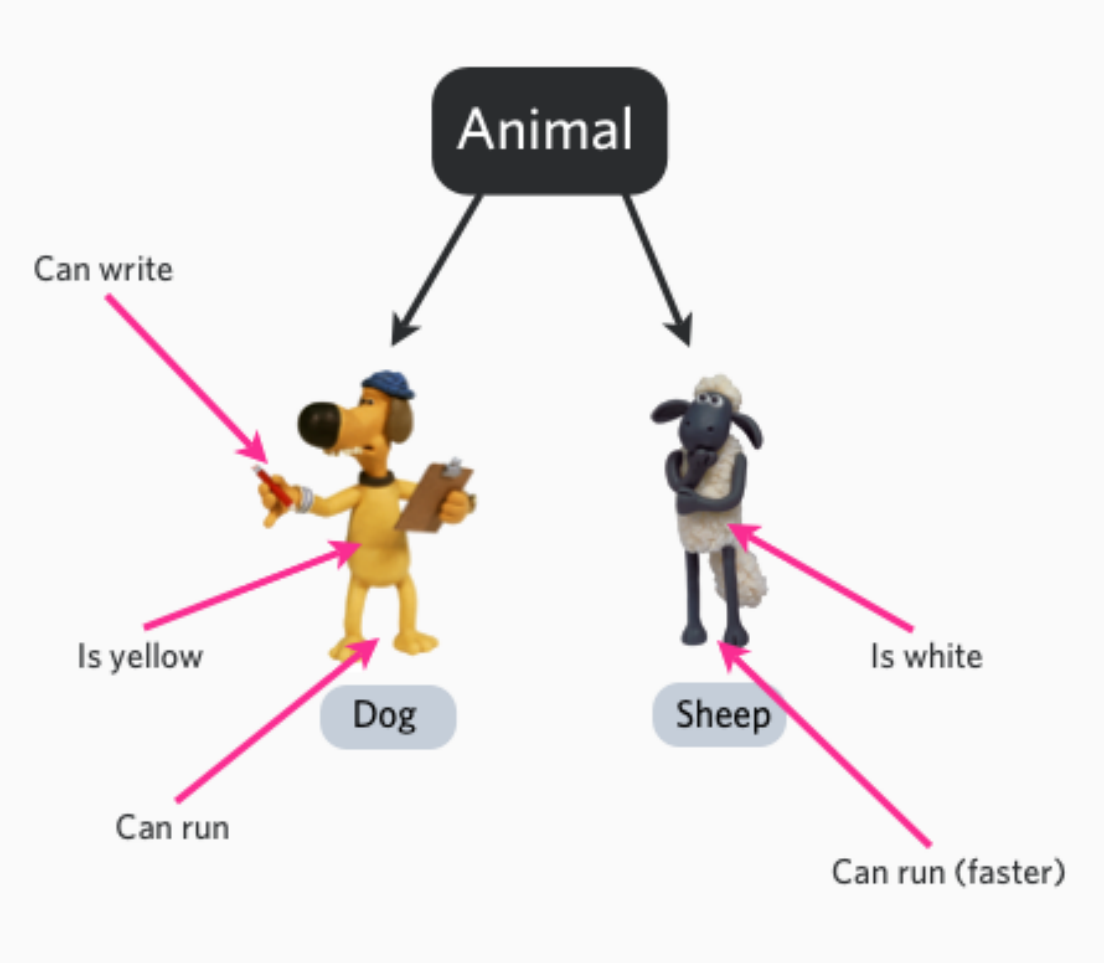
Classes and objects

- Objects are grouped into classes



- How many classes do you see?

Object Oriented Concept



Class

- A class is a description of a group of objects with common properties (attributes) and behavior (operations).

- An object is an instance of a class

Class

- Sample Class

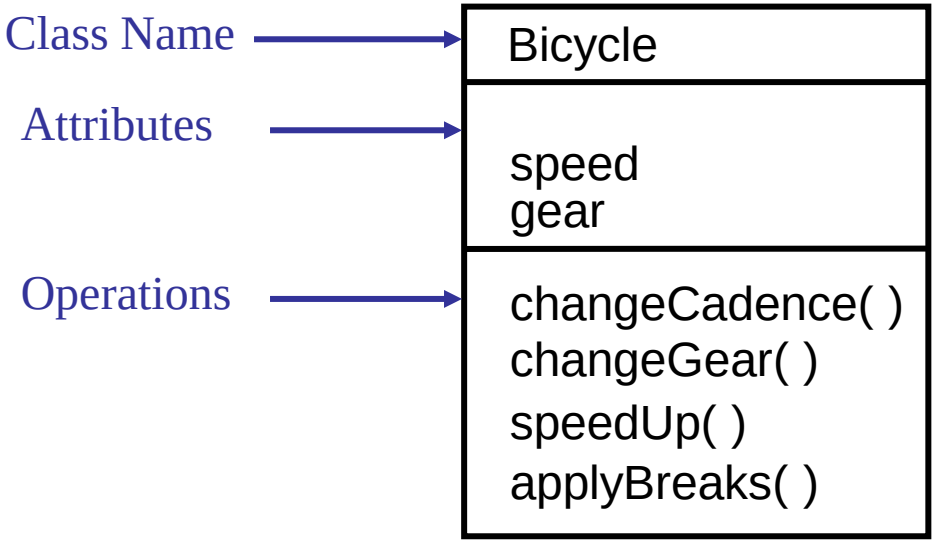
Class
Bicycle

Properties
cadence
speed
gear

Behavior
changeCadence
changeGear
speedUp
applyBreakes

Class

- A class is comprised of three sections
  - The first section contains the class name
  - The second section shows the properties (attributes)
  - The third section shows the behavior (operations)



Class

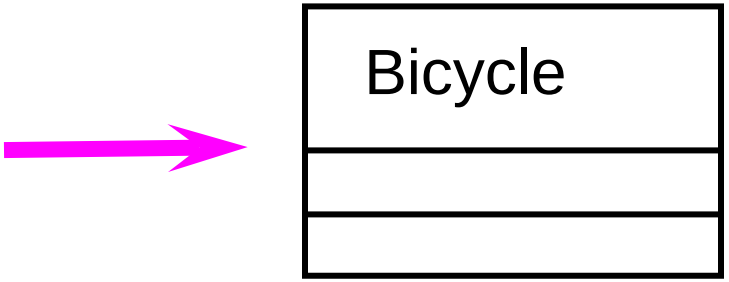
- The relationship between classes and objects
  - A class is an abstract definition of an object
    - It defines the properties and behavior of each object in the class
    - It serves as a template for creating objects

Bicycle Bike1

Objects

Bicycle Bike2

Class



Class

```
cadence:50 speed:10 gear:2
cadence:40 speed:20 gear:3
```

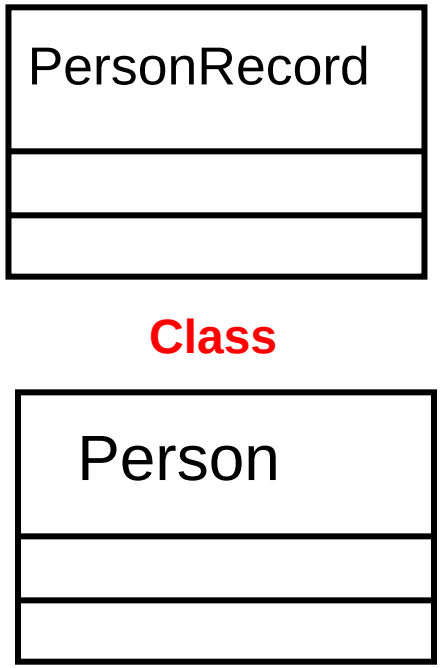
- A class is the blueprint from which individual objects are created.

```java
public class Bicycle
{
    int cadence;
    int speed;
    int gear;
    public Bicycle() {
        cadence = 0;
        speed = 0;
        gear = 1;
    }
    void changeCadence(int newValue) {
        cadence = newValue;
    }
    void changeGear(int newValue) {
        gear = newValue;
    }
    void speedUp(int increment) {
        speed = speed + increment;
    }
    void applyBrakes(int decrement) {
        speed = speed - decrement;
    }
    void printStates() {
        System.out.println("cadence:" +
            cadence + " speed:" +
            speed + " gear:" + gear);
    }
}
```

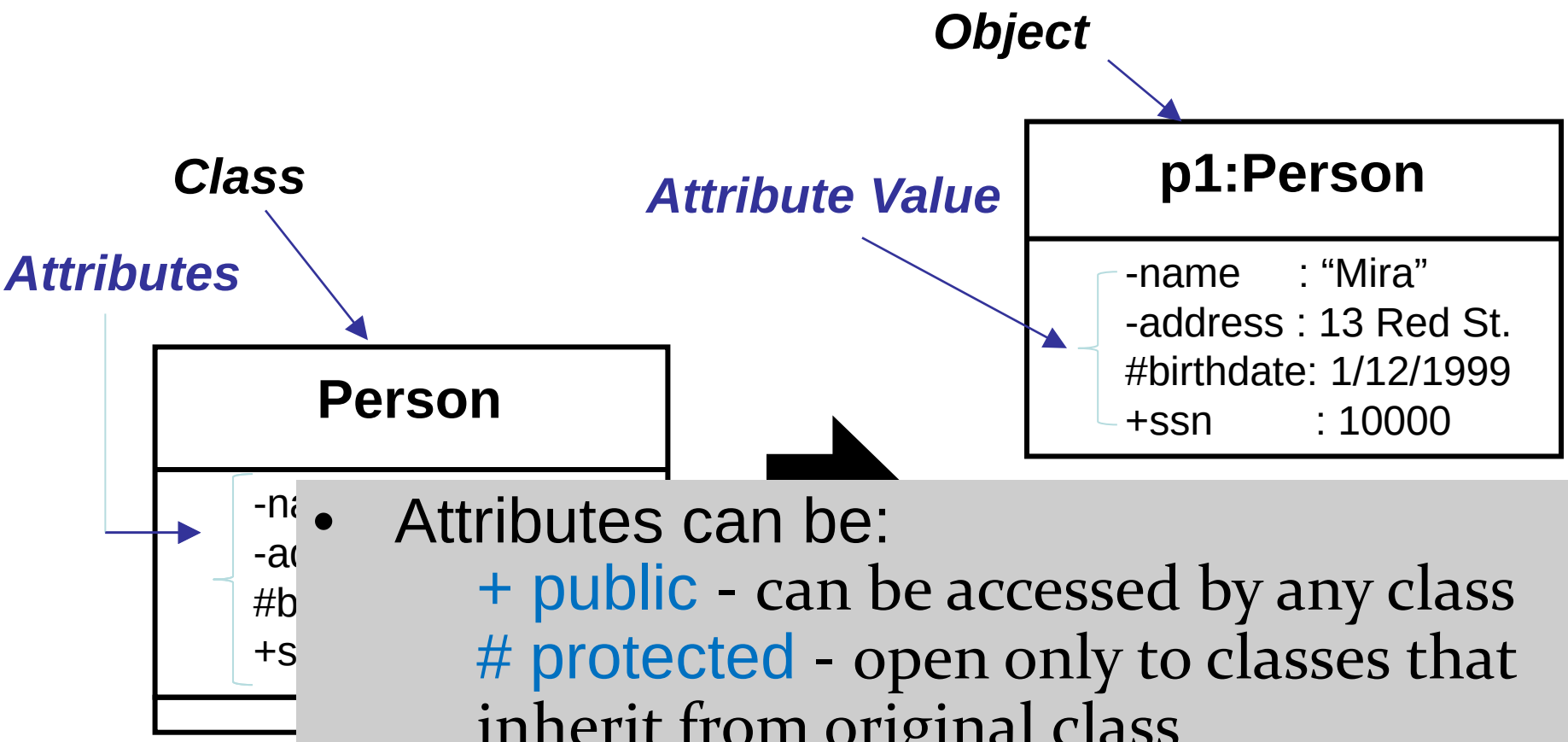
```java
public class BicycleDemo
{
    public static void main(String[] args) {

        // Create two different
        // Bicycle objects
        Bicycle bike1 = new Bicycle();
        Bicycle bike2 = new Bicycle();

        // Invoke methods on
        // those objects
        bike1.changeCadence(50);
        bike1.speedUp(10);
        bike1.changeGear(2);
        bike1.printStates();

        bike2.changeCadence(50);
        bike2.speedUp(10);
        bike2.changeGear(2);
        bike2.changeCadence(40);
        bike2.speedUp(10);
        bike2.changeGear(3);
        bike2.printStates();
    }
}
```

Class

- The rectangle is the icon for the class.
- The name of the class is, by convention, a word with an initial uppercase letter.
- If your class name has more than one word name, then join the words together and capitalize the first letter of the every word.



Class - attributes

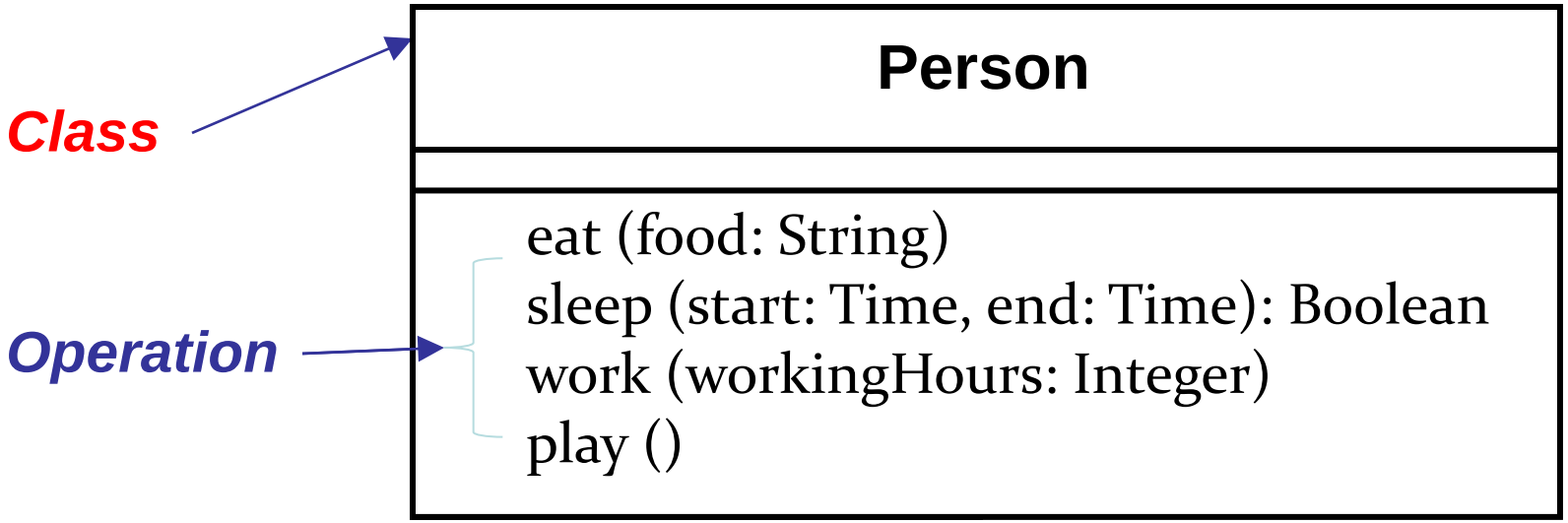


- Attributes can be:
  - + public - can be accessed by any class
  - # protected - open only to classes that inherit from original class
  - - private - only the original class can use the attribute or operation

Class

- Operation
  - What is an Op

- Operations describe the class behavior and appear in the third compartment.
- You can specify an operation by stating its signature: listing the name, type, and a return type in the case of functions.



Class

Person
- name : String - address : Address # birthdate: Date + ssn : Id
eat (food: String) sleep (start: Time, end: Time): Boolean work (workingHours: Integer) play ()

What is a Class Diagram

- A Class Diagram is a diagram describing the structure of a system
- shows the system's
  - classes
  - attributes
  - operations (or methods)
  - **relationships among the classes**.

Developing Class Models

- Class diagrams used for different purposes during different times in the development life-cycle
  - Models that support earlier modeling:
    - For domain analysis
    - For requirements specification

    vs.
  - Models that are close to code
- Class diagrams developed iteratively
  - Details added over time during lifecycle
  - missing parameters, multiplicities, other details

Essential Elements of a UML Class Diagram

- Class
  - Attributes
  - Operations
- **Relationships**
  - Associations
  - Generalization
  - Realization
  - Dependency
- Constraint Rules and Notes

Relationships

- Classes **A** and **B** are related if:
  - An object of class **A** sends a message to an object of **B**
  - An object of class **A** creates an instance of class **B**
  - An object of class **A** has an attribute of type **B** or collections of objects of type **B**
  - An object of class **A** receives a message with an argument that is an instance of **B** (maybe…)
    - Will it “use” that argument?
- Does an object of class **A** need to know about some object of class **B**?

More on Relationships

- Relationships should model the reality of the domain and allow implementation
- Relationships are between classes
  - A link connects two specific objects
  - Links are instances of associations
  - Note we could draw an object diagram to show objects and links
    - But often interaction diagrams are more useful for modeling objects

Relationships

- There are six kinds of relationships in UML
  - Associations
  - Aggregation
  - Compositions
  - Generalization
  - Realization
  - Dependency

Relationships - association

- Association
  - Commonly represented as direct or indirect references between classes
  - Association is shown by a solid line between classes.

Relationships - association

- Association
  - May have **an optional label** consisting of a name and a direction drawn as a solid arrowhead with no tail.
  - The direction arrow indicates the direction of association with respect to the name.

Relationships - association

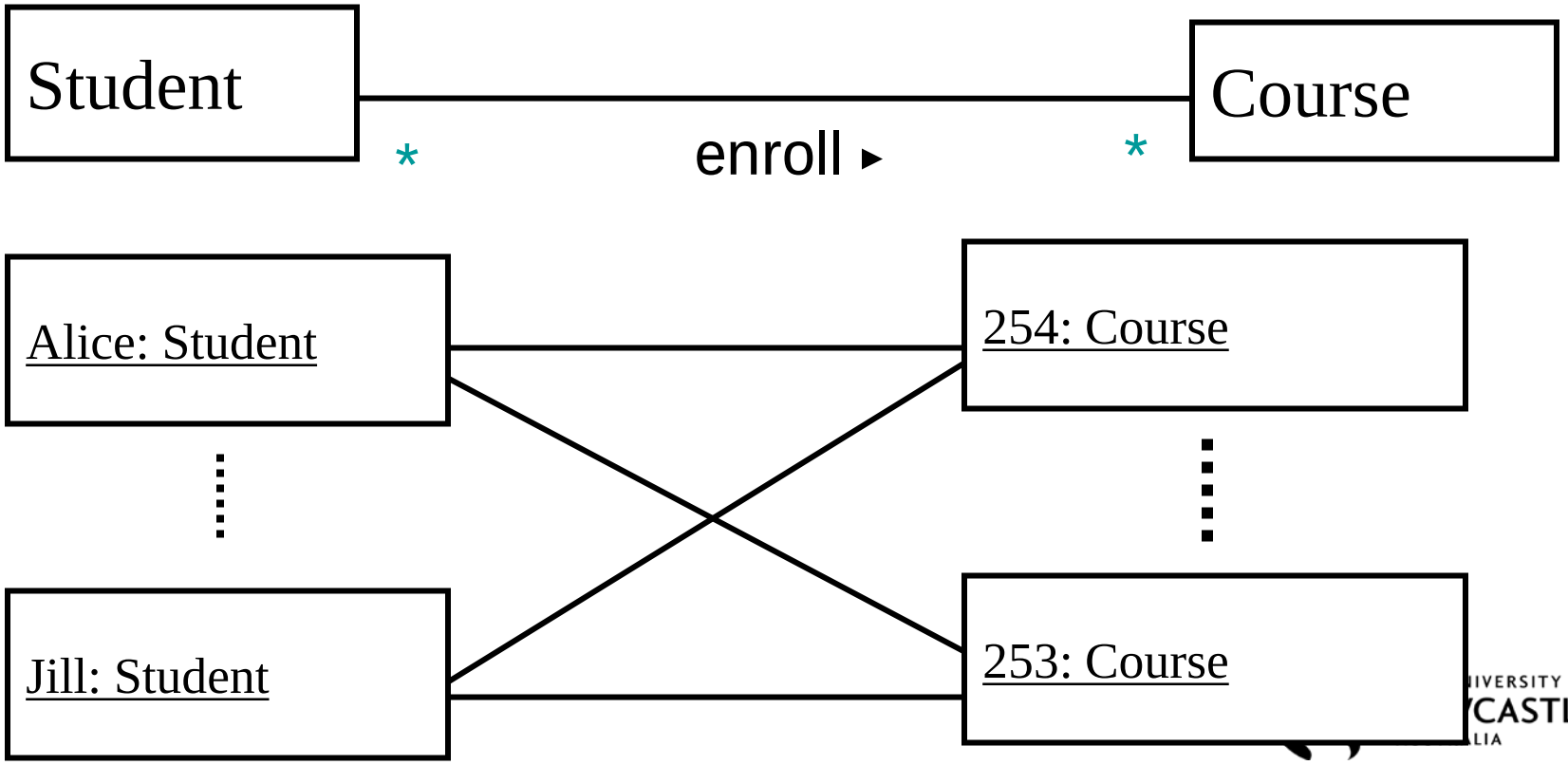
- Association
  - May have **an optional label** consisting of a name and a direction drawn as a solid arrowhead with no tail.
  - The direction arrow indicates the direction of association with respect to the name.

Relationships - association

- Association
  - Add multiplicity
  - A Student can take up to five Courses.
  - Student has to be enrolled in at least one course.
  - Up to 300 students can enroll in a course.
  - A class should have at least 10 students.

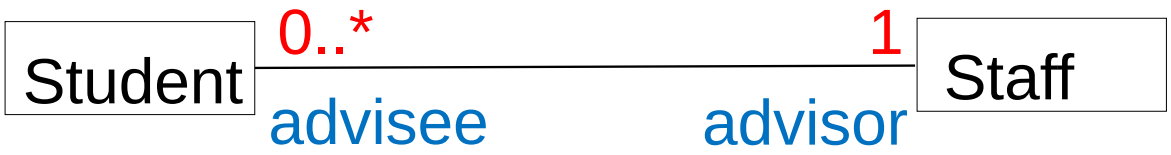
Relationships - association

- Association
  - A Student can take many Courses and many Students can be enrolled in one Course.



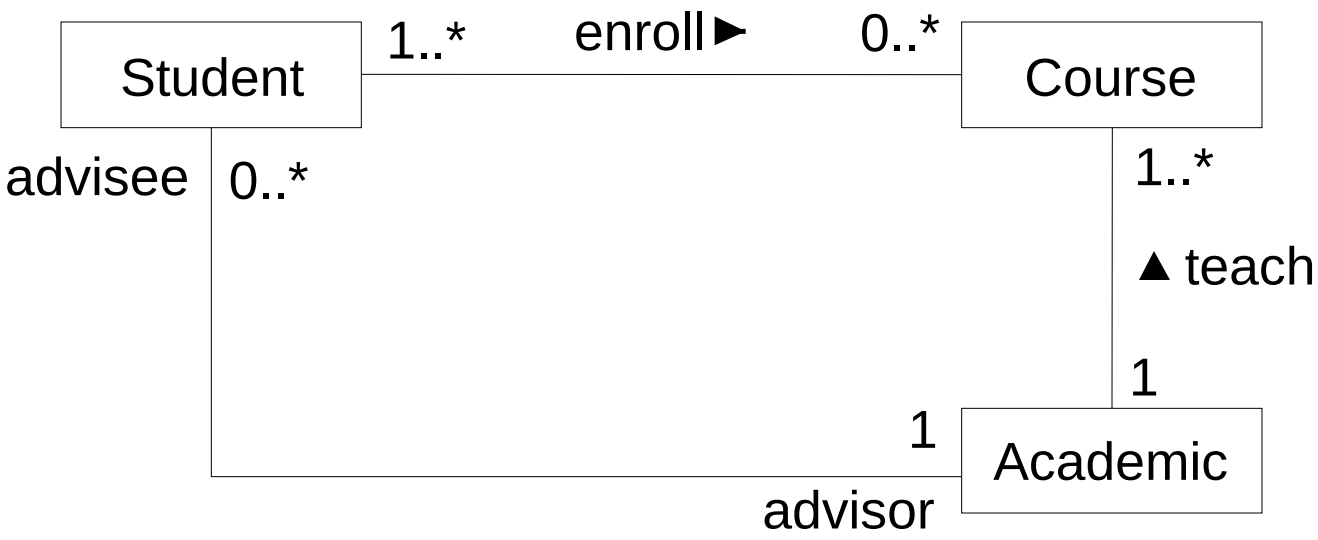
Relationships - association

- Association
  - May have an optional *role name* and an optional multiplicity specification.



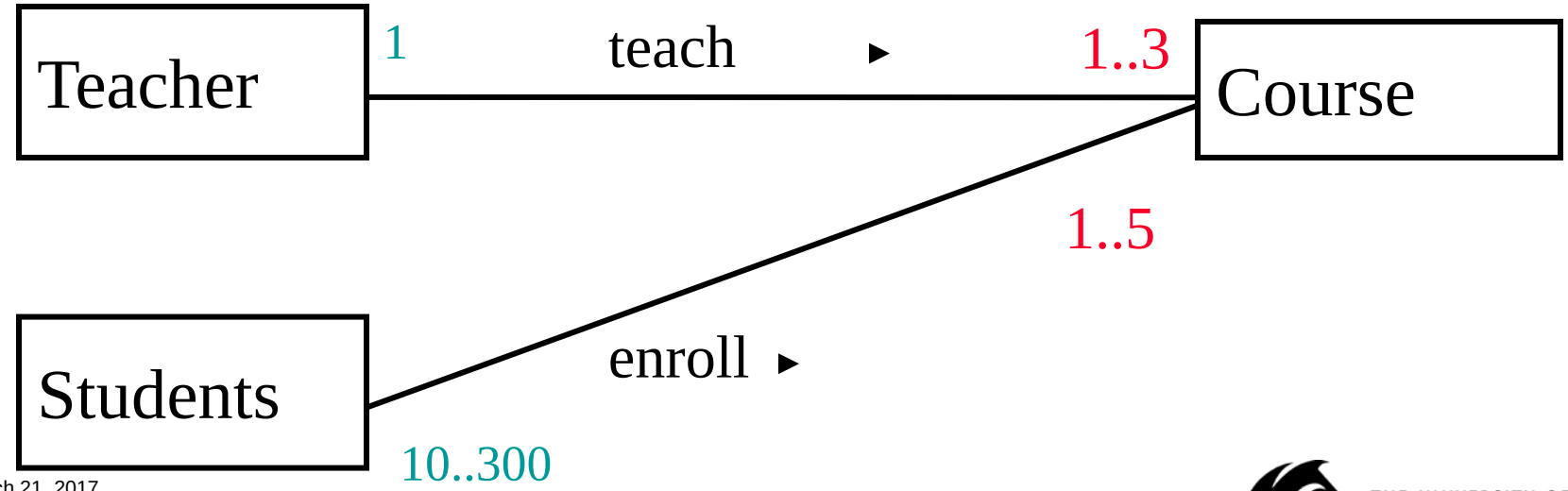
Relationships - association

- Association



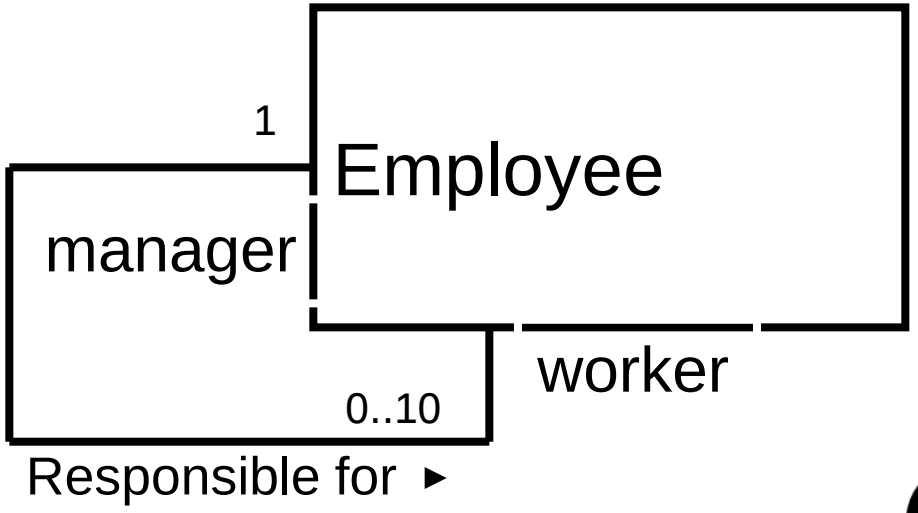
Relationships - association

- Association
  - A teacher teaches 1 to 3 courses (subjects)
  - Each course is taught by only one teacher.
  - A student can take between 1 to 5 courses.
  - A course can have 10 to 300 students.



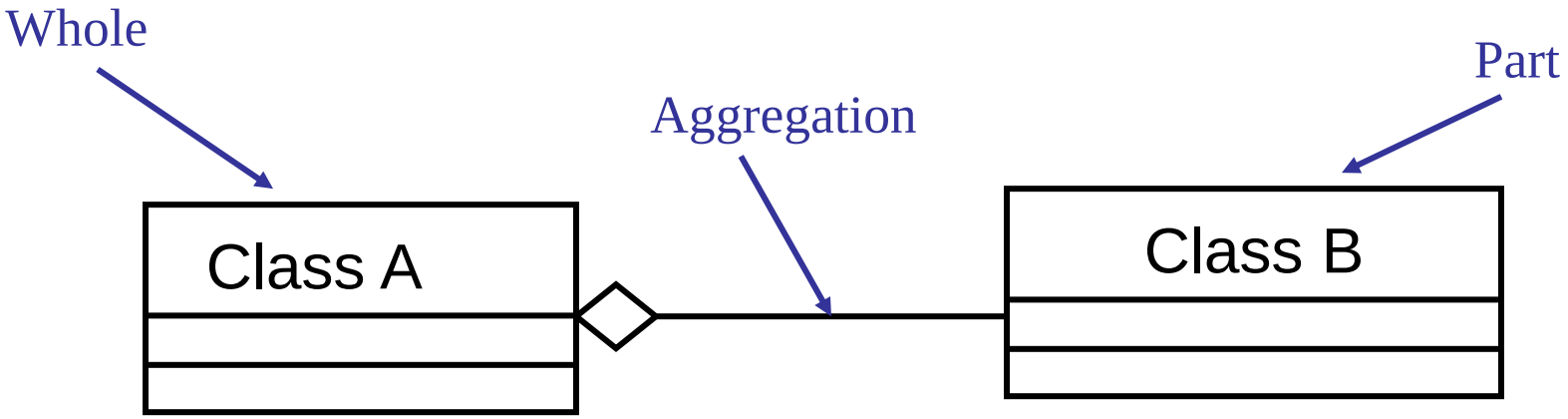
Relationships - association

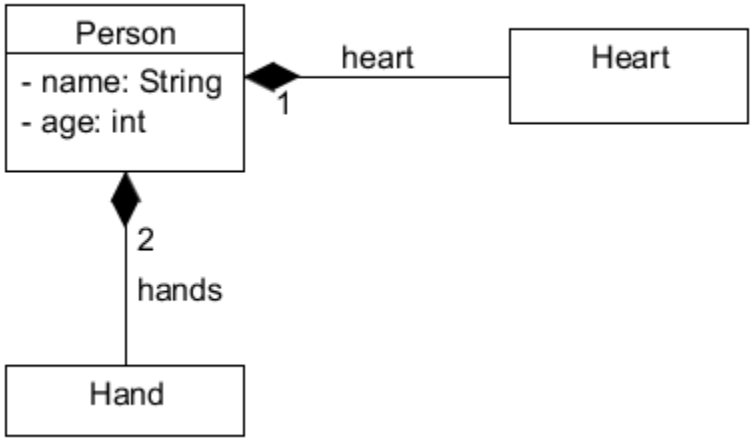
- Association - Self
  - An association that connects a class to itself is called **a self association**.
- A Company has Employees.
- A single manager is responsible for up to 10 workers.

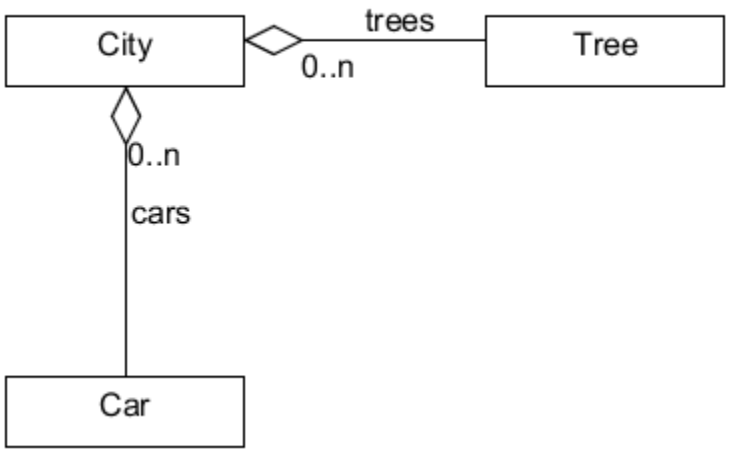


Relationships - aggregation

- Aggregation
  - Special form of association representing *has-a* or *part-whole* relationship.
  - Distinguishes the whole (aggregate class) from its parts (component class).
  - No relationship in the lifetime of the aggregate and the components (can exist separately).





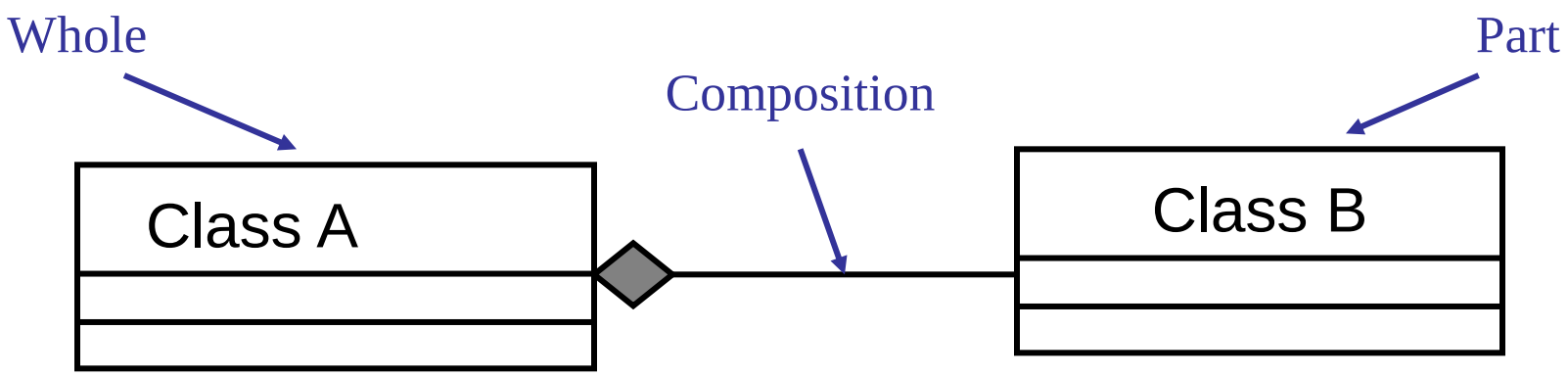



```
Public class City {
        private List<Tree> trees;
        private List<Car> cars;
        void setCity (List<Tree> trees, List<Car> cars) {
                this.trees ⇦ trees;
                this.cars ⇦ cars;
        }
        void checkCar() {
                cars.check();
        }
}
```

In aggregation (City, Tree, Car) "sub objects" (Tree, Car) will NOT be destroyed when City is destroyed.

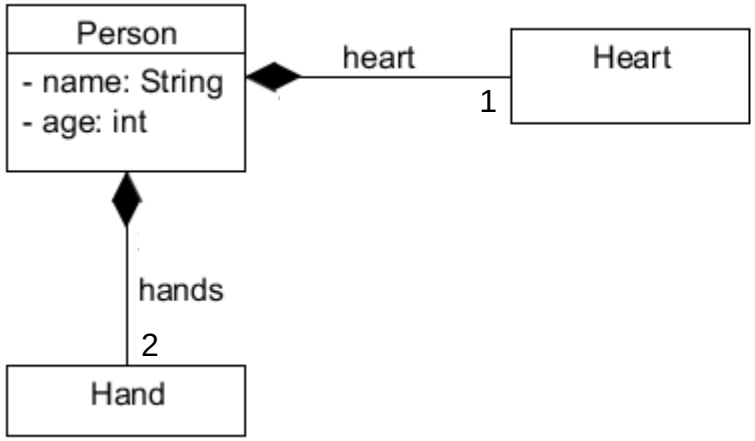
Relationships - composition

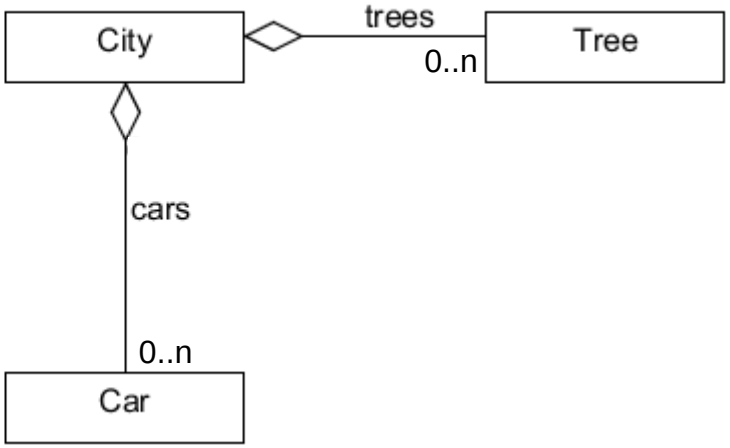
- Composition
  - **Stronger** form of aggregation
  - Implies exclusive ownership of the component class by the aggregate class
  - The lifetime of the components is entirely included in the lifetime of the aggregate (a component can not exist without its aggregate).

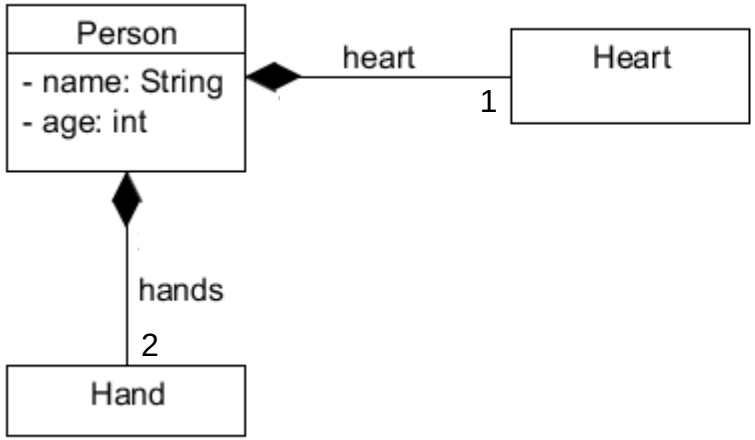


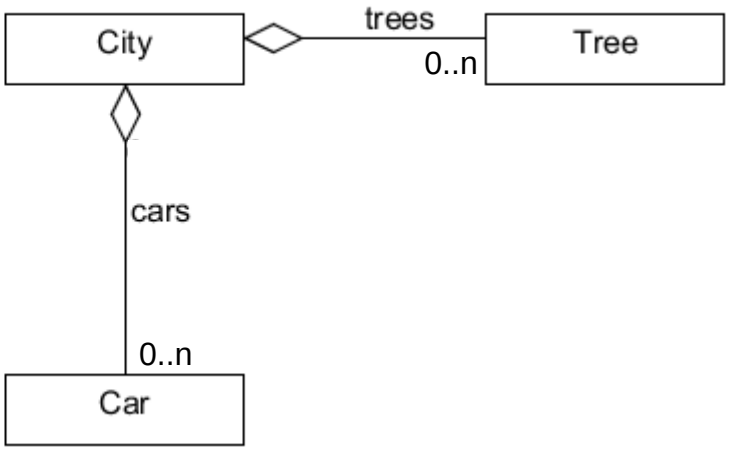
Relationships - composition

- Composition
  - The object only exists, or only makes sense inside the other, as a part of the other.
  - Person – heart: You don't create a heart and then pass it to a person.





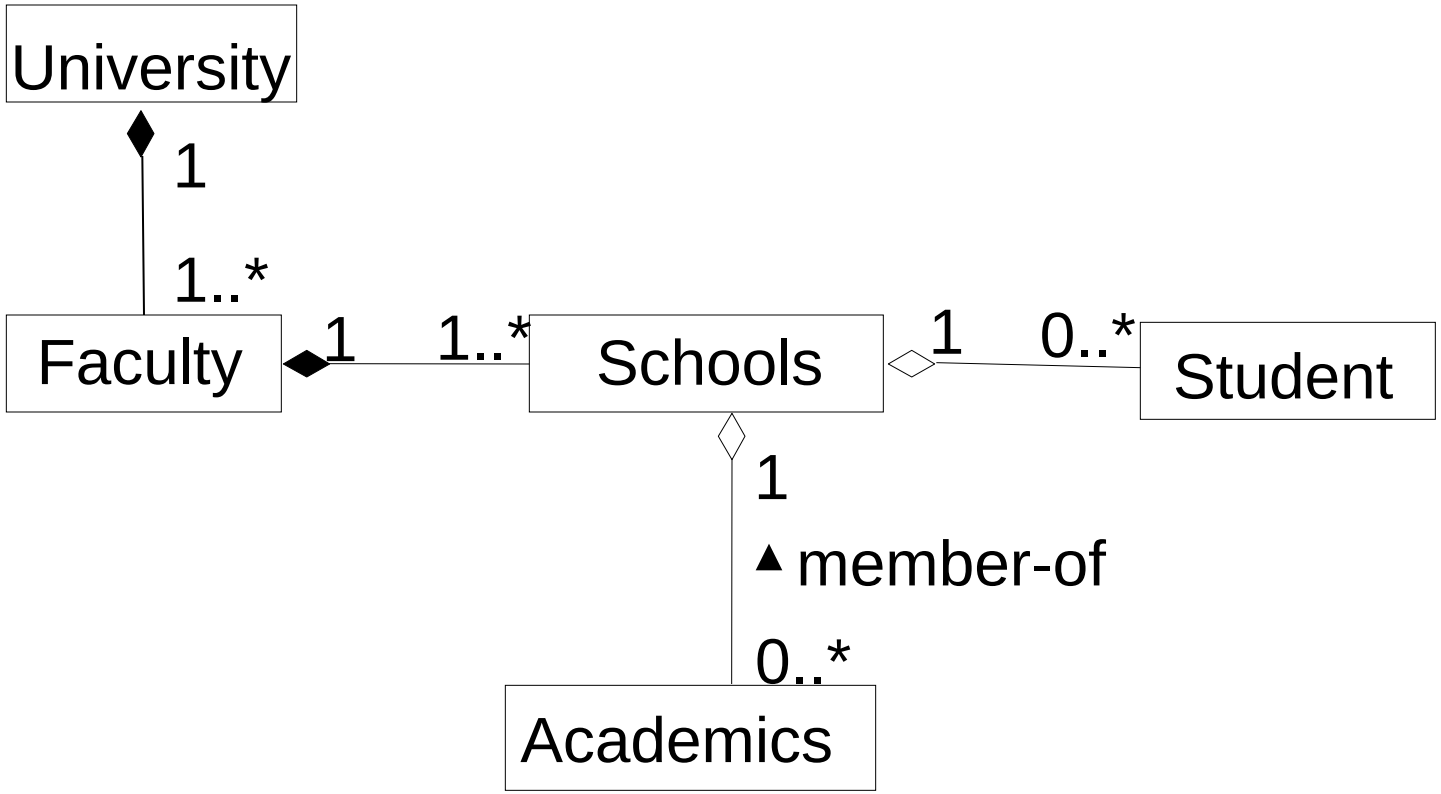




In composition (Person, Heart, Hand), "sub objects" (Heart, Hand) will be destroyed as soon as Person is destroyed.

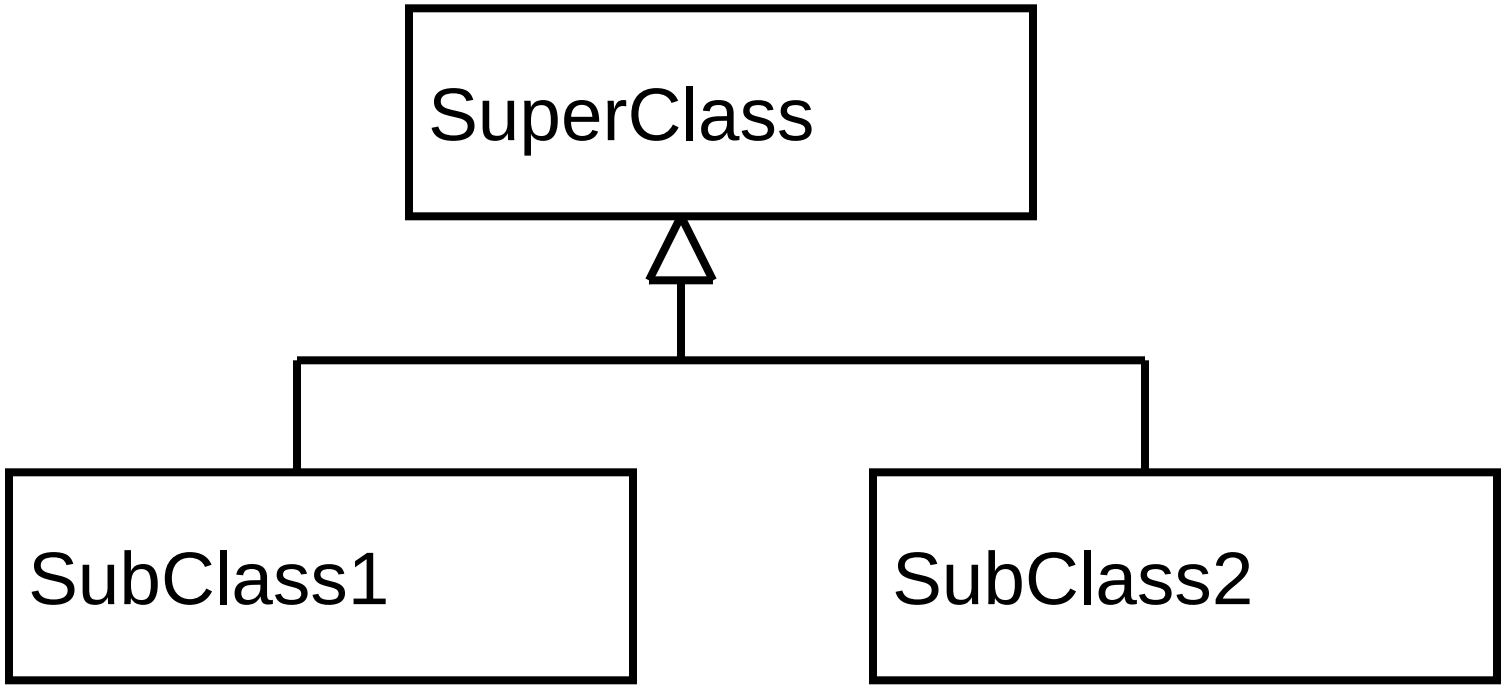
```
Public class Person {
        private Heart heart;
        private List<Hand> hands;
        Person () {
                heart = new Heart();
                hands = new Hand();
        }
        void checkHeart() {
                heart.check();
        }
}
```

Relationships – aggregation/composition



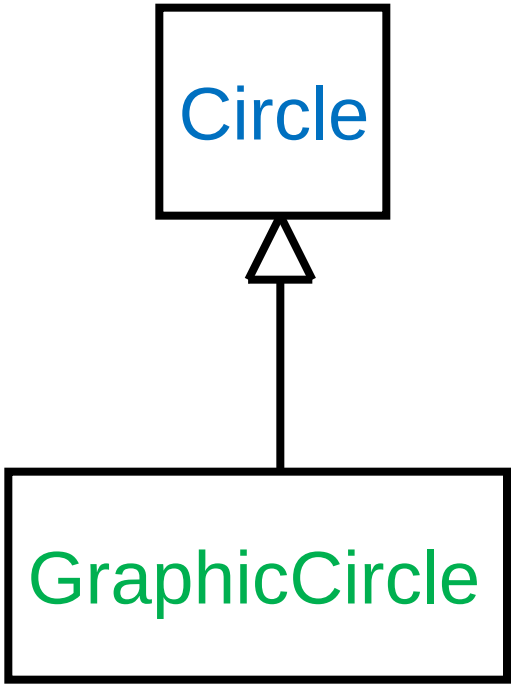
Relationships - generalization

- Child class is a special case of the parent class
- Generalization is an “is-a-kind of” relationship



Relationships - generalization

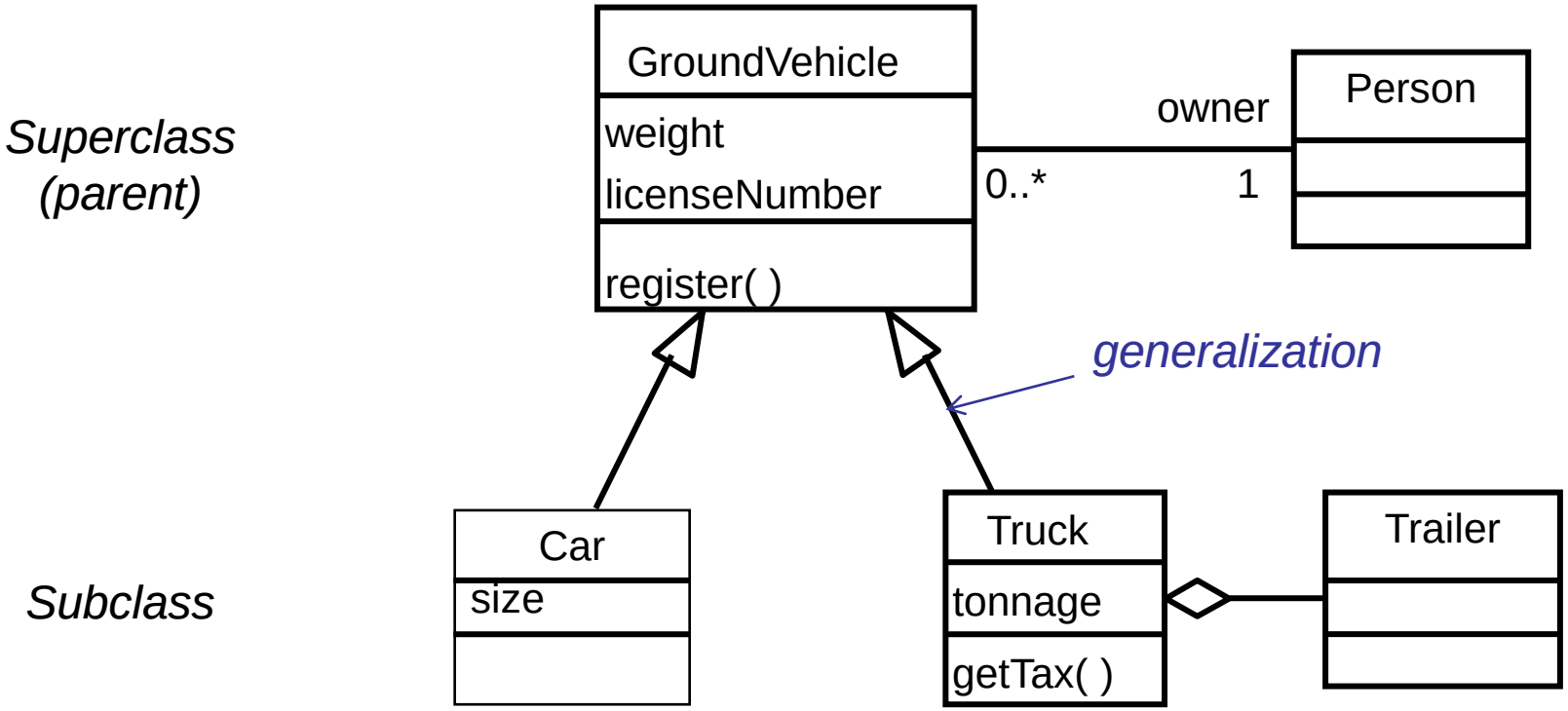
- Inheritance




```
public class Circle {

}




public class GraphicCircle extends Circle
{

}
```

Relationships - generalization

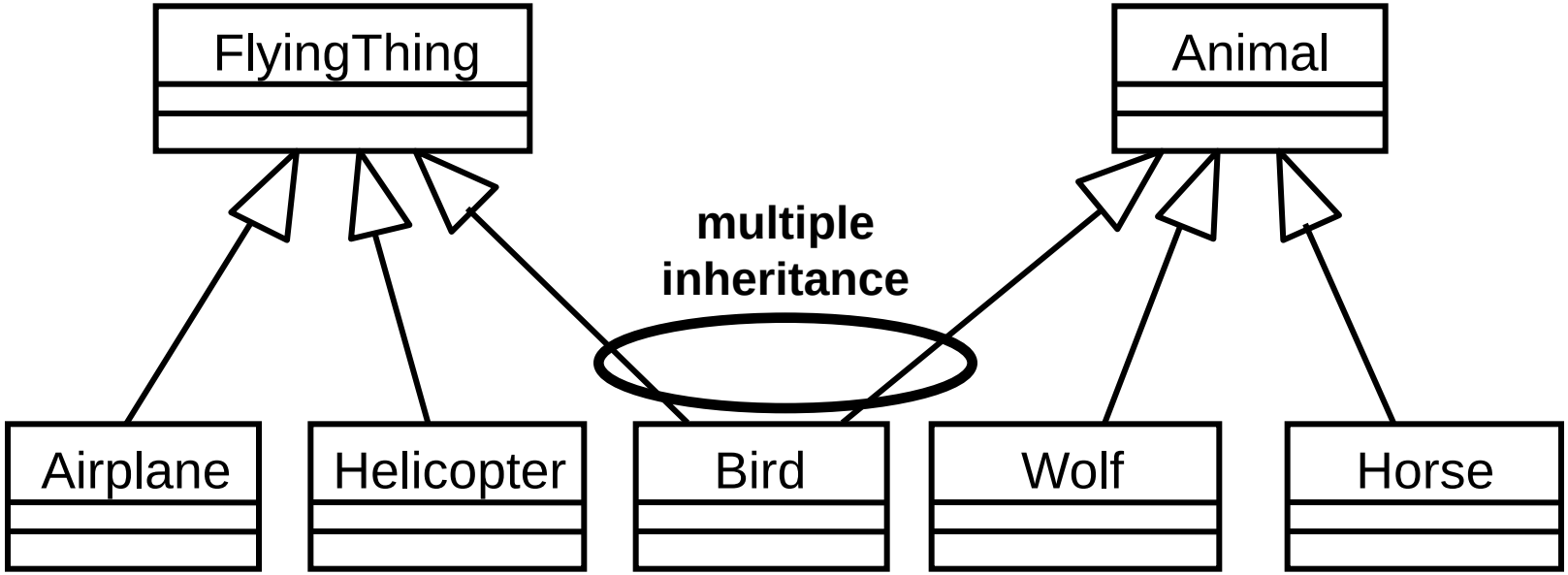
- Inheritance
  - Single inheritance
    - One class inherits from another



Relationships - generalization

Use multiple inheritance only when needed, and always with caution !

- Inheritance
  - Single inheritance
  - Multiple inheritance
    - A class can inherit from several other classes



Relationships - generalization

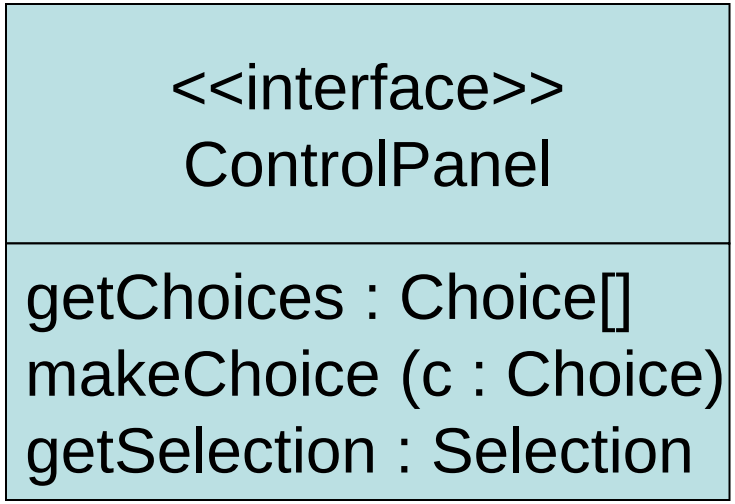
- What gets inherited?
  - A subclass inherits its parent's attributes, operations, and relationships
  - A subclass may:
    - Add additional attributes, operations, relationships
    - Redefine inherited operations (use caution!)
  - Common attributes, operations, and/or relationships are shown at the **highest applicable level** in the hierarchy

Relationships – realization: interface

<<interface>>
ControlPanel

An *interface* is a named set of operations that specifies the behavior of objects without showing their inner structure. It can be rendered in the model by a one- or two-compartment rectangle, with the *stereotype* <<interface>> above the interface name.

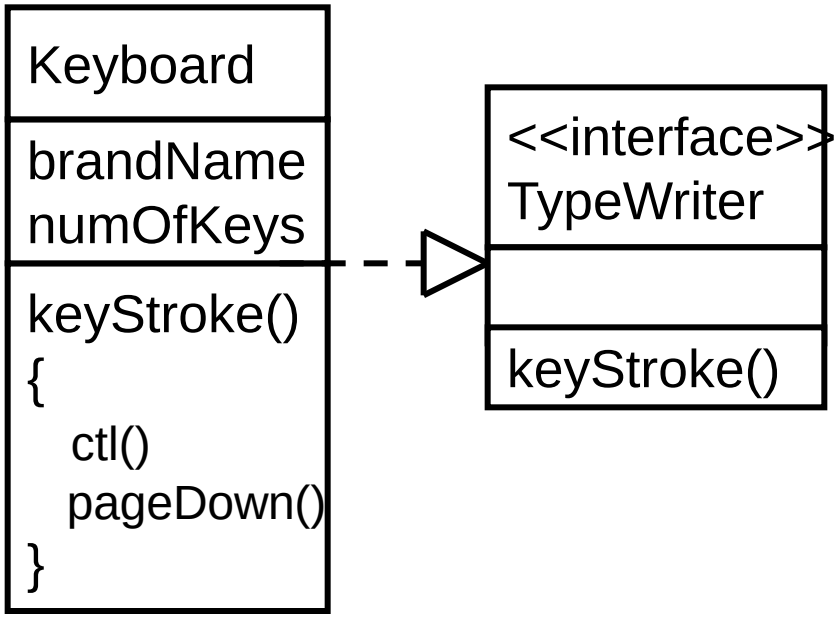
Relationships – realization: interface services



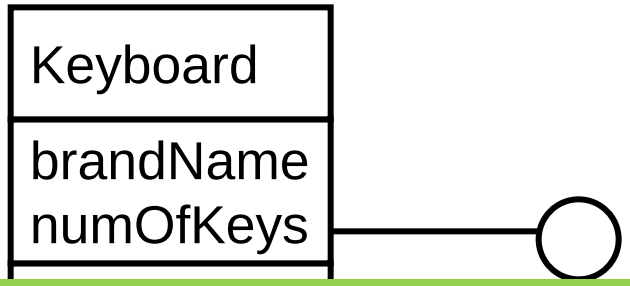
Interfaces do not get instantiated. They have no attributes or state. Rather, they specify the services offered by a related class.

Relationships - realization

- Interface is a set of operation the class carries out



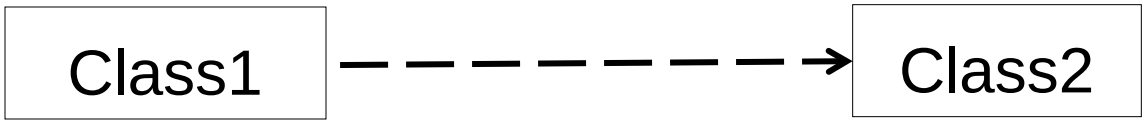
OR




```
public Interface TypeWriter {
        void keyStroke()
}
public class Keyboard implements
TypeWrite {
        public void keyStroke() {
        // implementation
                ctl()
                pageDown()
        }
}
```

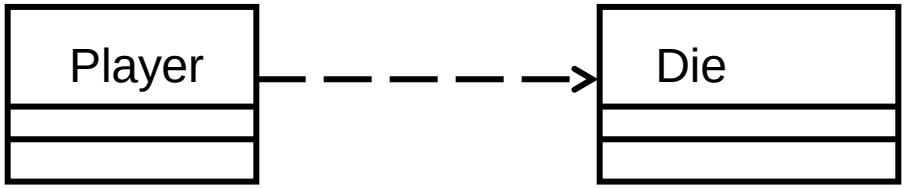
Relationships - dependency

- Relationship between the classes such that the proper operation of one class depends on the presence of the other class, and changes in one class would affect the other class.
- It is often confused as Association.
- It is normally created when you receive a reference to a class as part of a particular operation / method.



Relationships - dependency

- A relationship between two model elements where a change in one **<u>may</u>** cause a change in the other
- Non-structural, “using” relationship



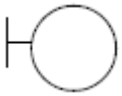

```
class Die { public void Roll()  { …  }  }
class Player
{
      public void TakeTurn (Die die)
      /*  Player is dependent on Die and Die’s Roll method does Player’s work  */
      {  die.Roll();   …. }
 }
```

Multiplicity

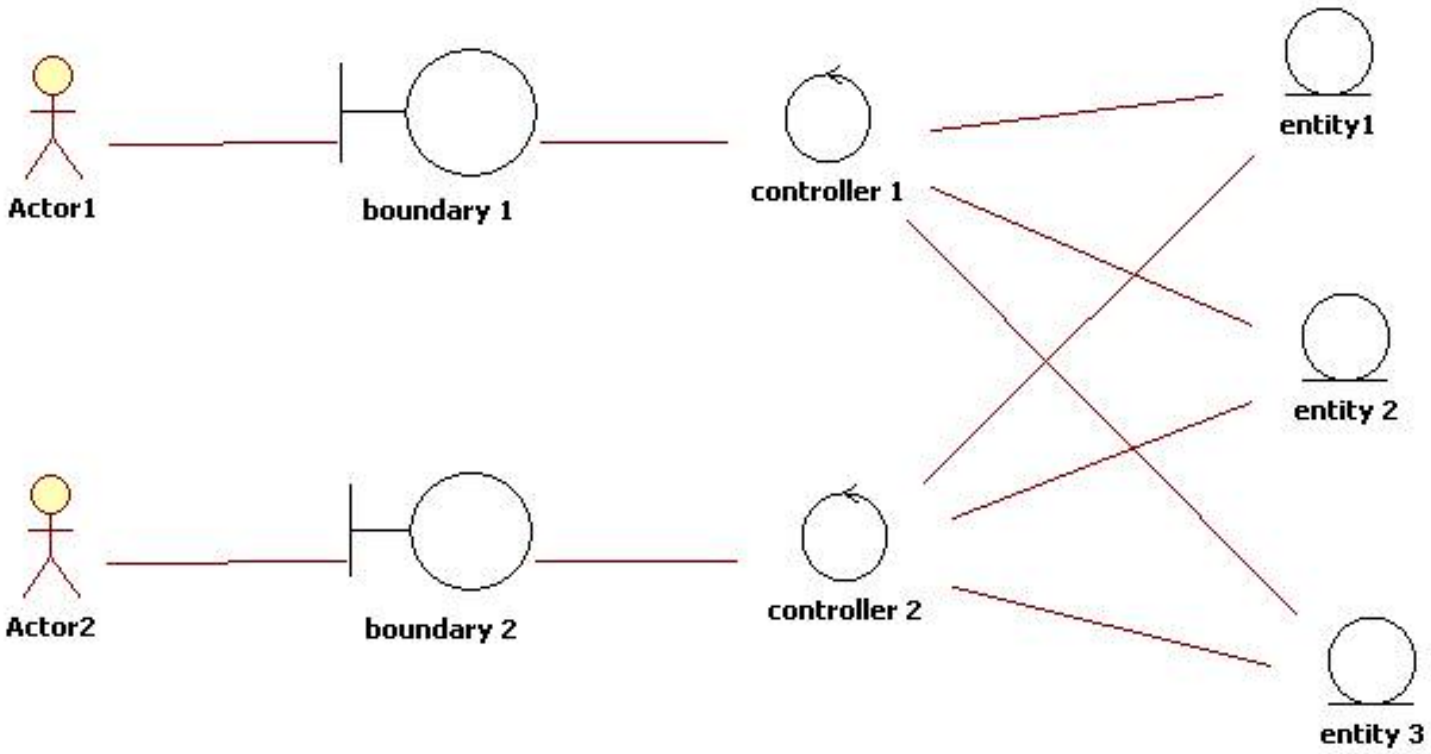
- The multiplicity specifies an integer interval, e.g.,
  - m..n closed (inclusive) range of integers
  - i singleton range
  - 0..* entire nonnegative integer, i.e., 0, 1, 2, …

- Multiplicity can be expressed as,
  - Exactly one - 1
  - Zero or one - 0..1
  - Many - 0..* or *
  - One or more - 1..*
  - Exact Number - e.g. 3..4 or 6
  - Or a complex relationship – e.g. 0..1, 3..4, 6..* would mean any number of objects other than 2 or 5

Class diagrams: Object types

- **Entity Objects** 
  - Represent the persistent information tracked by the system, real world entities, e.g., roles, invoices, databases, file
- **Boundary Objects** 
  - Represent the interaction between the user and the system, e.g., receiving/presenting data
  - Examples: printer interfaces, terminals, sensors, forms, GUI items
  - Each boundary class should be related to at least one actor
- **Control Objects** 
  - Represent the control tasks performed by the system

Class diagrams: Object types



Class diagrams: Object types

Year

DateManager

Month

Button

Day

LCDDisplay

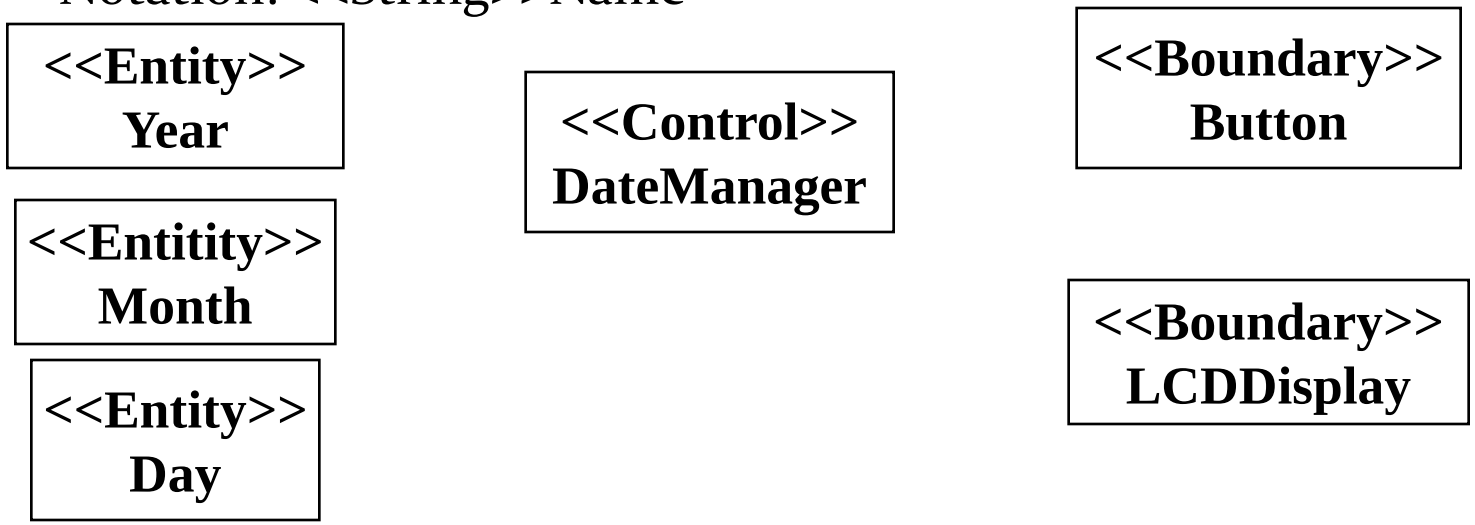
Entity Objects

Control Object

Boundary Objects

Class diagrams: Object types

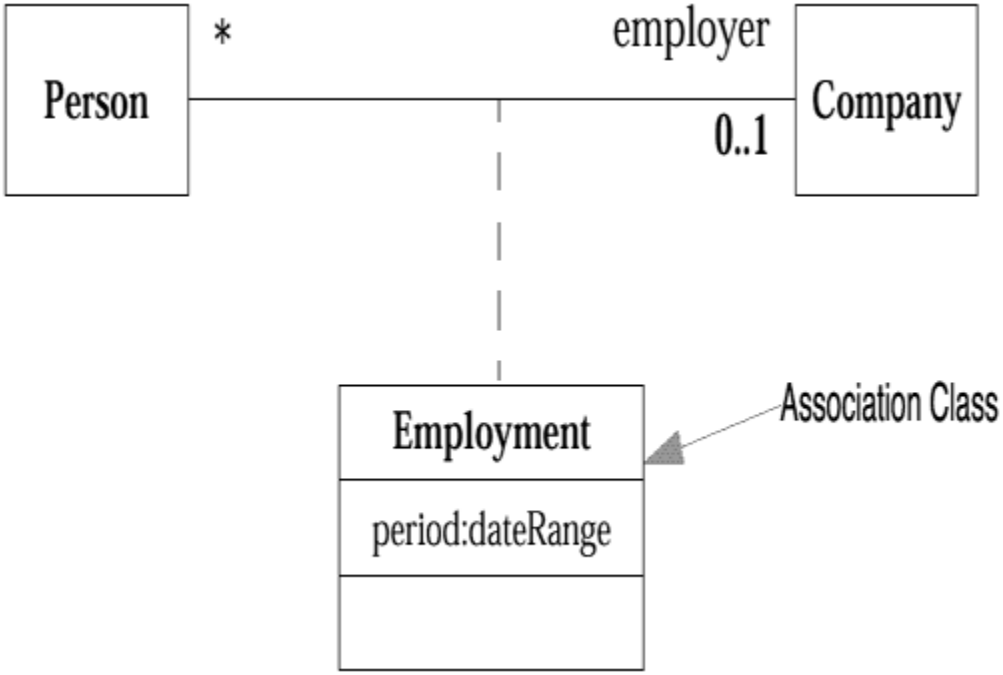
- UML provides the stereotype mechanism to introduce new types of modeling elements
  - A stereotype is drawn as a name enclosed by angled double-quotes (<<, >>) and placed before the name of a UML element (class, method, attribute, ….)
  - Notation: <<String>>Name



Entity Object Control Object Boundary Object

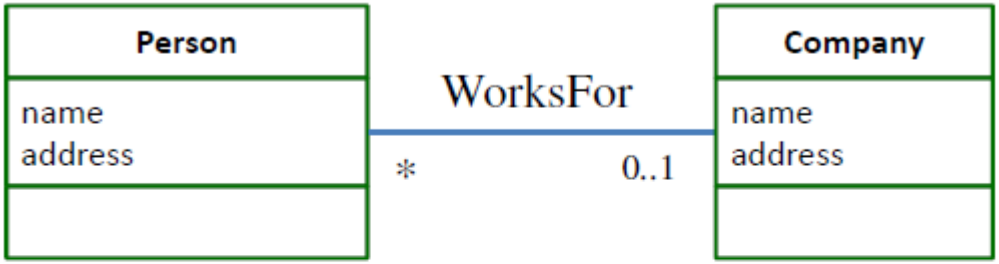
Association classes

- Sometimes, an attribute that concerns two associated classes cannot be placed in either of the classes.



Association classes

- Association classes allow you to add attributes, operations, and other features to associations

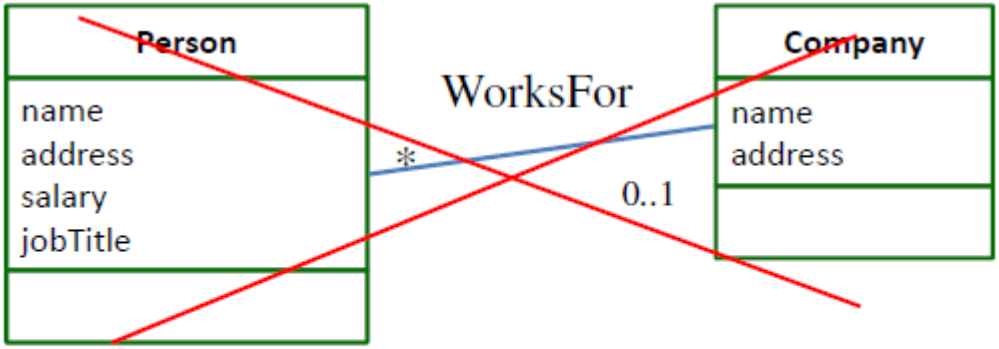


Salary?

Job Title?

Association classes

- Association classes allow you to add attributes, operations, and other features to associations

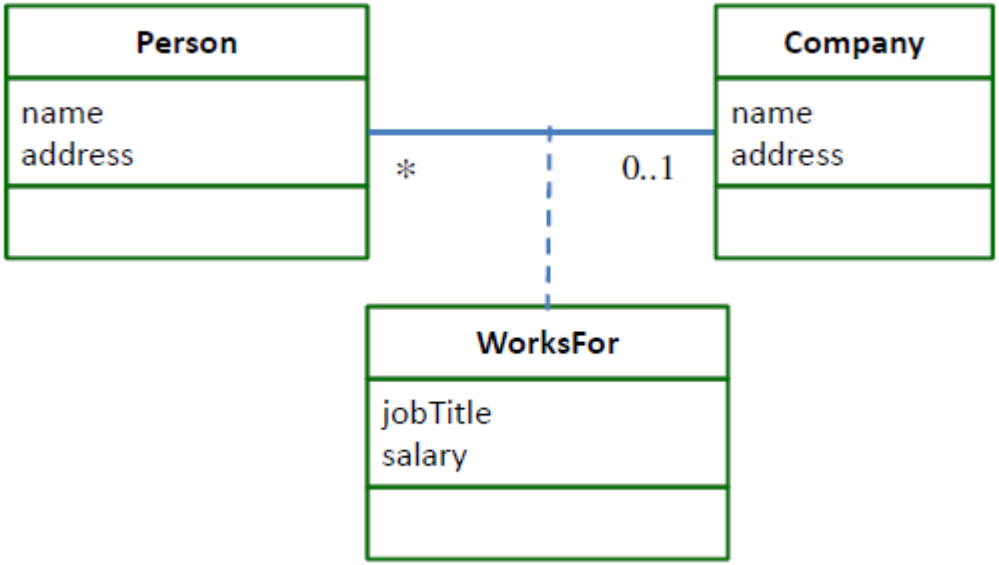


Salary?

Job Title?

Association classes

- Association classes allow you to add attributes, operations, and other features to associations



Do it yourself !!!

- Examples: Bank Accounts Management System
  - Example taken from the following book

T. C. Lethbridge and R. Laganière, *Object-Oriented Software Engineering: Practical Software Development using UML and Java*
2nd Edition, McGraw Hill, 2001.

Class diagrams

- Examples: Bank Accounts Management System

This system provides the basic services to manage bank accounts at a bank called OOBank. OOBank has many branches, each of which has an address and branch number. A client opens accounts at a branch. Each account is uniquely identified by an account number; it has a balance and a credit or overdraft limit. There are many types of accounts, including: A mortgage account (which has a property as collateral), a chequing account, and a credit card account (which has an expiry date and can have secondary cards attached to it). It is possible to have a joint account (e.g. for a husband and wife). Each type of account has a particular interest rate, a monthly fee and a specific set of privileges (e.g. ability to write cheques, insurance for purchases etc). OOBank is divided into divisions and subdivisions (such as Planning, Investments and Consumer), the branches are considered subdivisions of the Consumer Division. Each division has a manager and a set of other employees. Each customer is assigned a particular employee as his or her ‘personal banker’.

Class diagrams

- Exam nsure

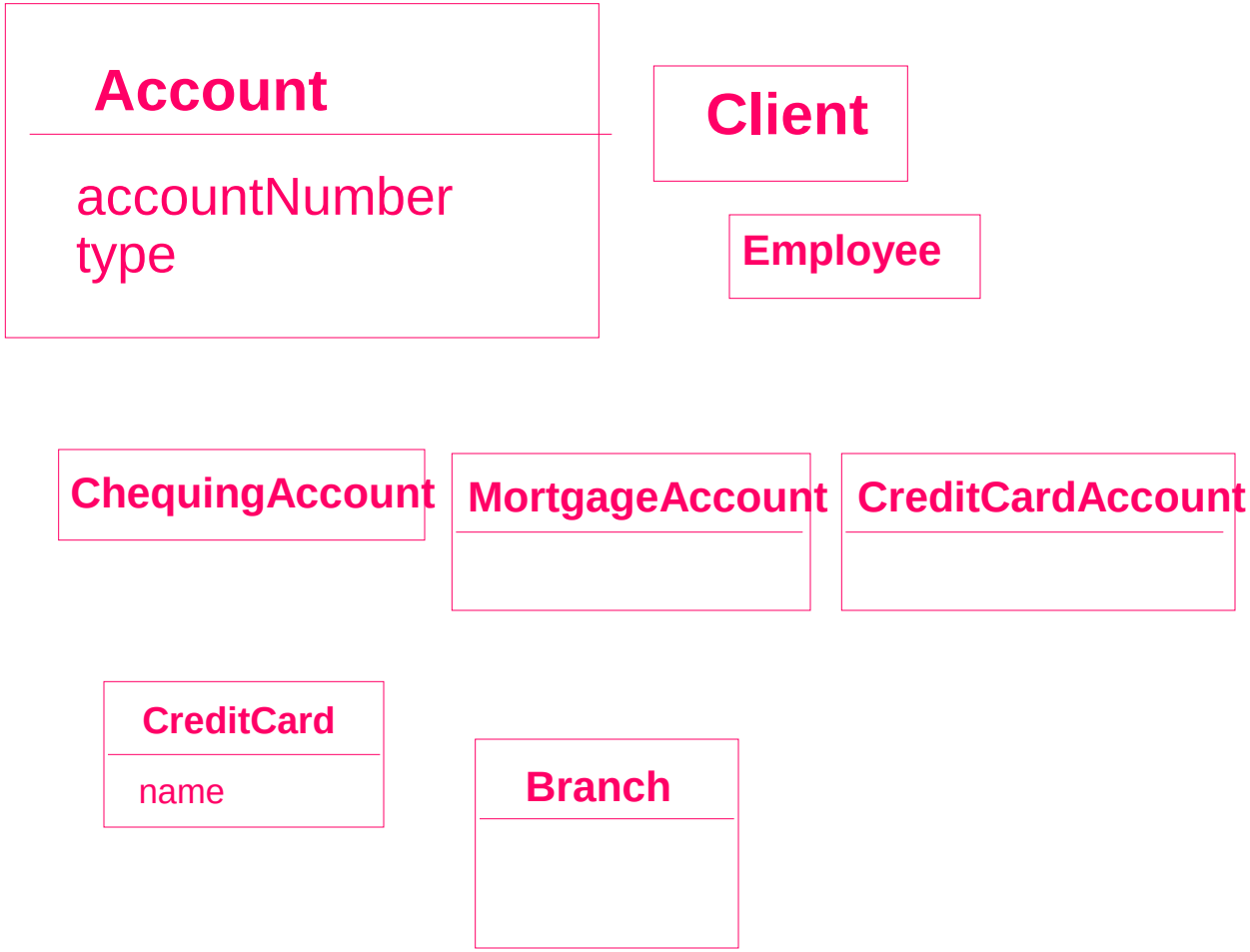
This *syste* ervices to manage bank accounts at a bank called *C* many branches, each of which has an *address* and *br* opens accounts at a branch. Each account is uniquel unt numb ce and a credit or overdra y types c ding: A mortgage account as collater count, and a credit card ac expiry da secondary cards attache have a *jo* for a *husband* and *wife*). E a particul *monthly fee* and a specific g. ability es, *insurance* for *purchas* ided into *divisions* (such as *Plannir* *Consumer* s are considered subdivi Division. Each division has a manager and a set of c customer is assigned a particular employee as his or h

Bank accounts
Bank
Branches
Client
Mortgage account
Property
Cheque account
Credit card
account
Card
Divisions
Manager
Employees

Address
Joint account
Privileges
Cheques
Insurance
Purchases

Class diagrams

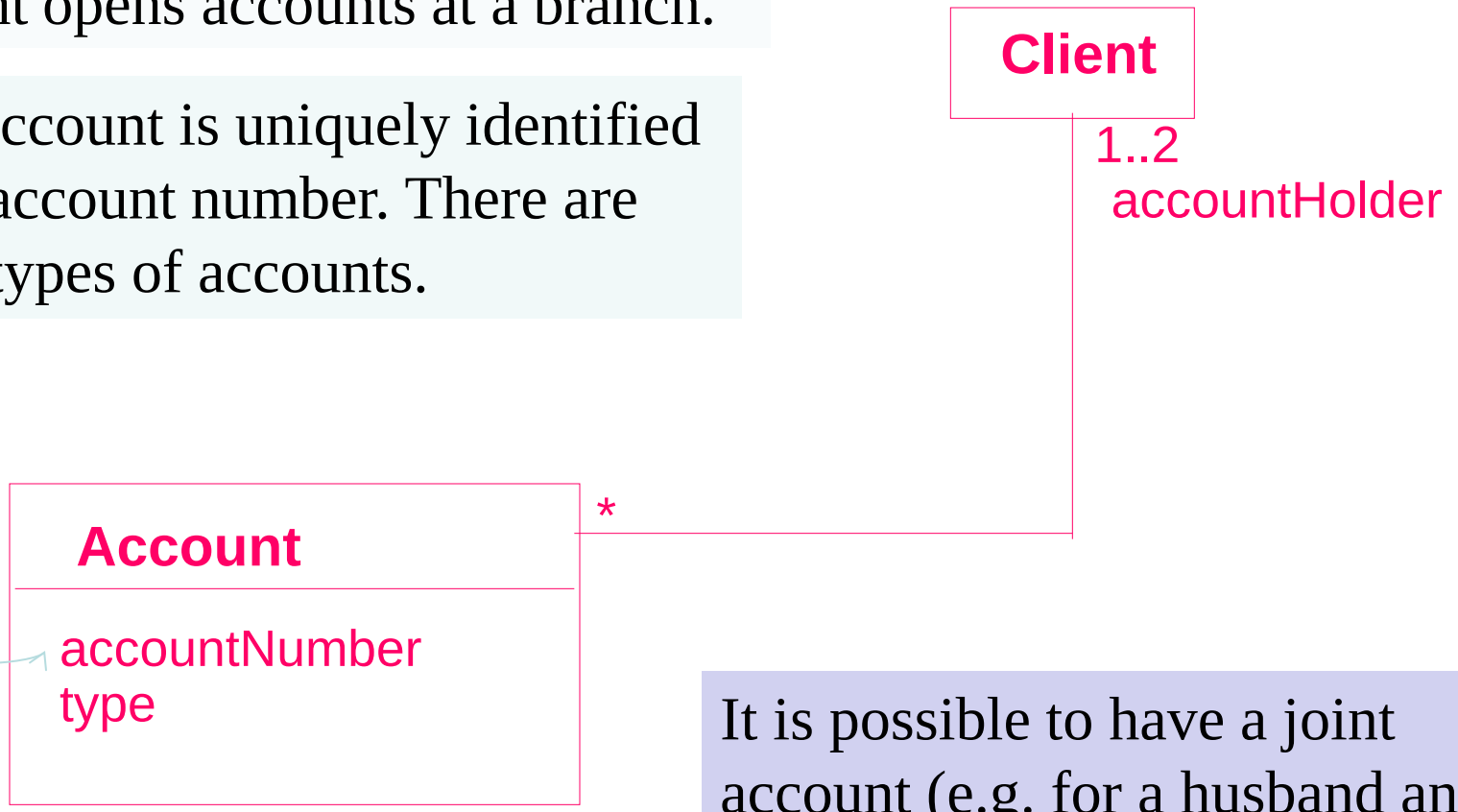
Bank accounts
Bank
Branches
Client
Mortgage account
Property
Cheque account
Credit card account
Card
Divisions
Manager
Employees



Class diagrams: draft class diagram

A client opens accounts at a branch.

Each account is uniquely identified by an account number. There are many types of accounts.

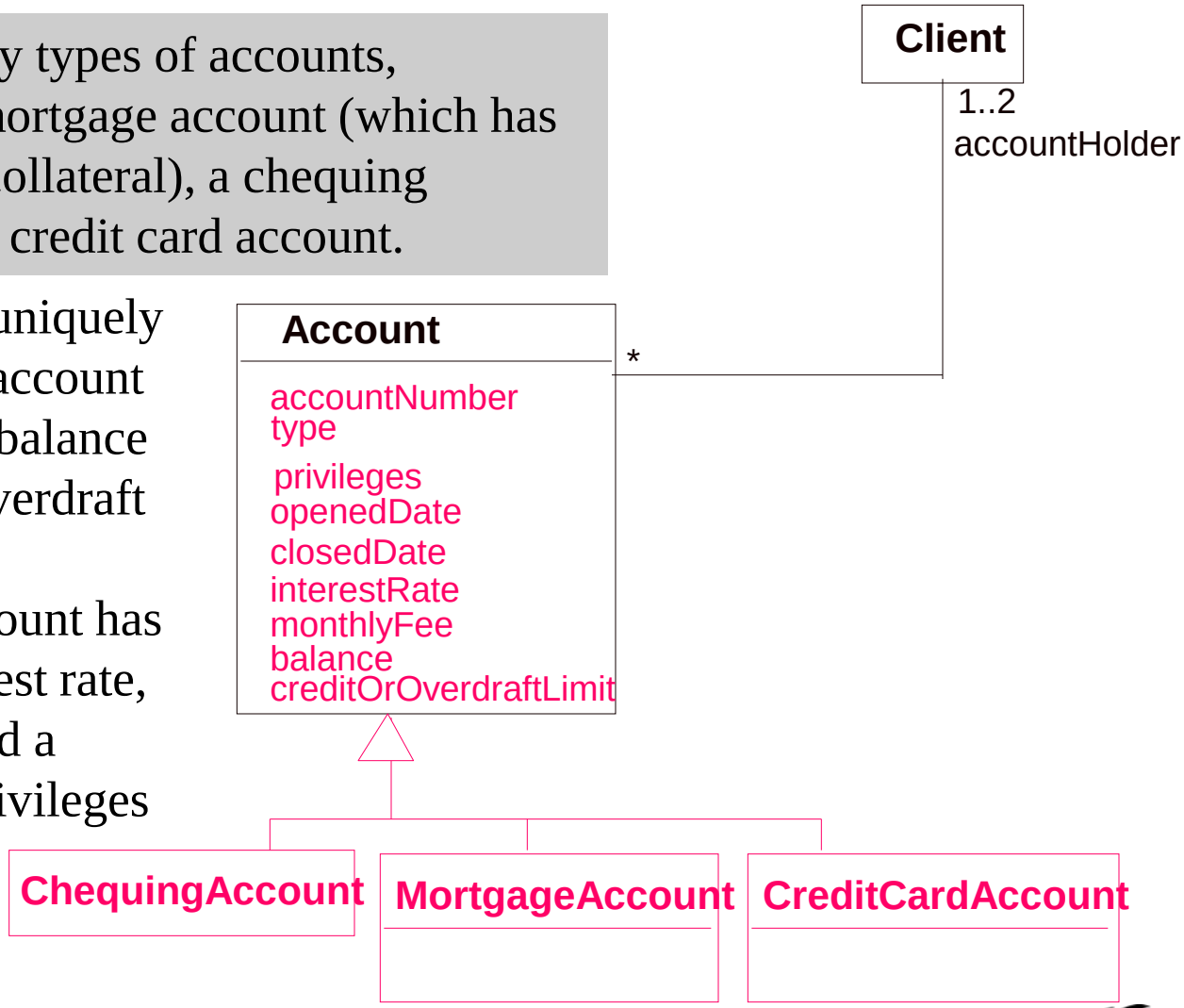


It is possible to have a joint account (e.g. for a husband and wife).

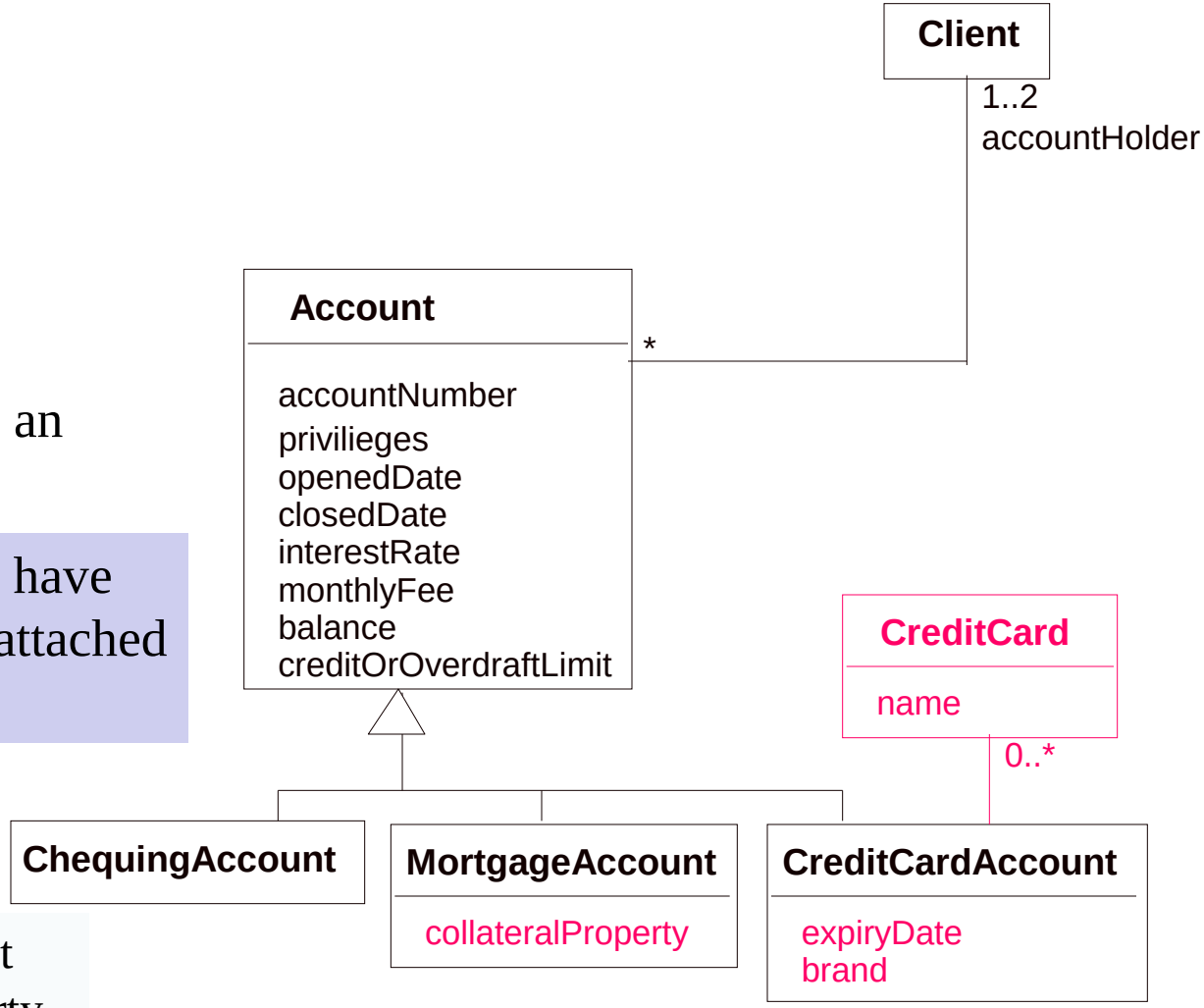
Class diagrams: draft class diagram

There are many types of accounts, including: A mortgage account (which has a property as collateral), a chequing account, and a credit card account.

Each account is uniquely identified by an account number; it has a balance and a credit or overdraft limit.
Each type of account has a particular interest rate, a monthly fee and a specific set of privileges



Class diagrams: draft class diagram



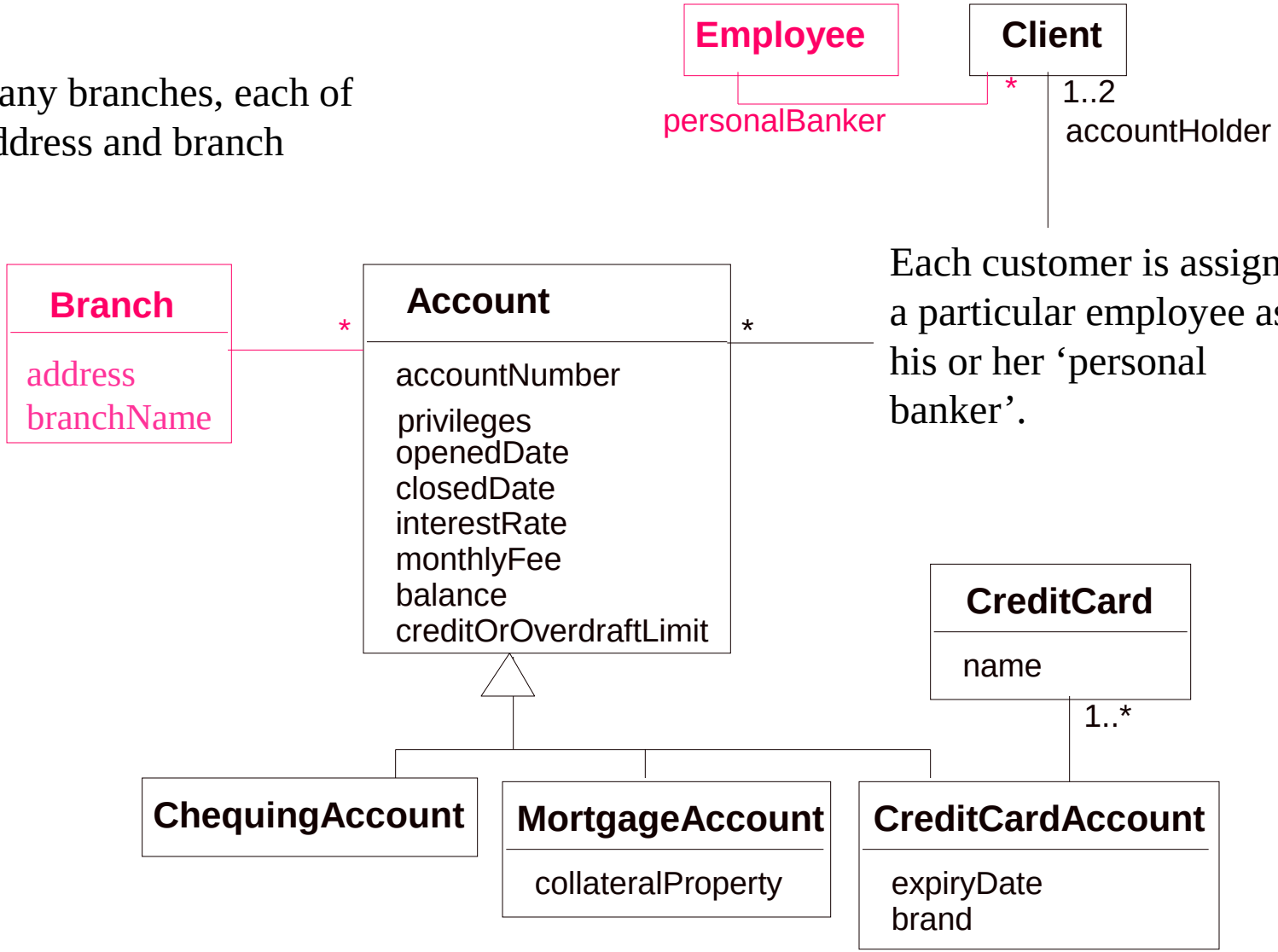
A credit card has an expiry date

A credit card can have secondary cards attached to it

A mortgage account (which has a property as collateral)

Class diagrams: draft class diagram

OOBank has many branches, each of which has an address and branch number.



Each customer is assigned a particular employee as his or her ‘personal banker’.

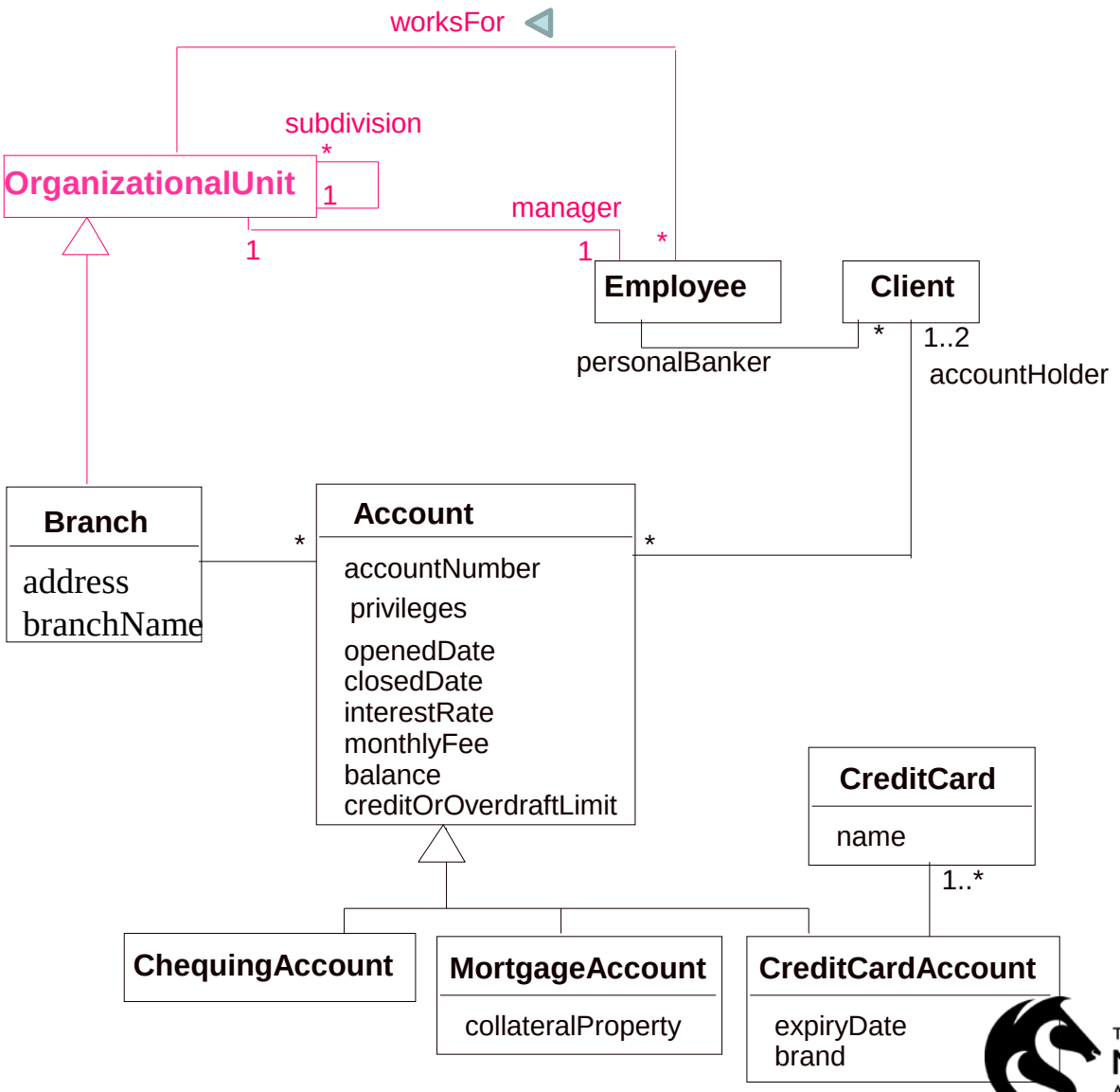
Class diagrams: draft class diagram

OOBank is divided into divisions and subdivisions (such as Planning, Investments and Consumer)

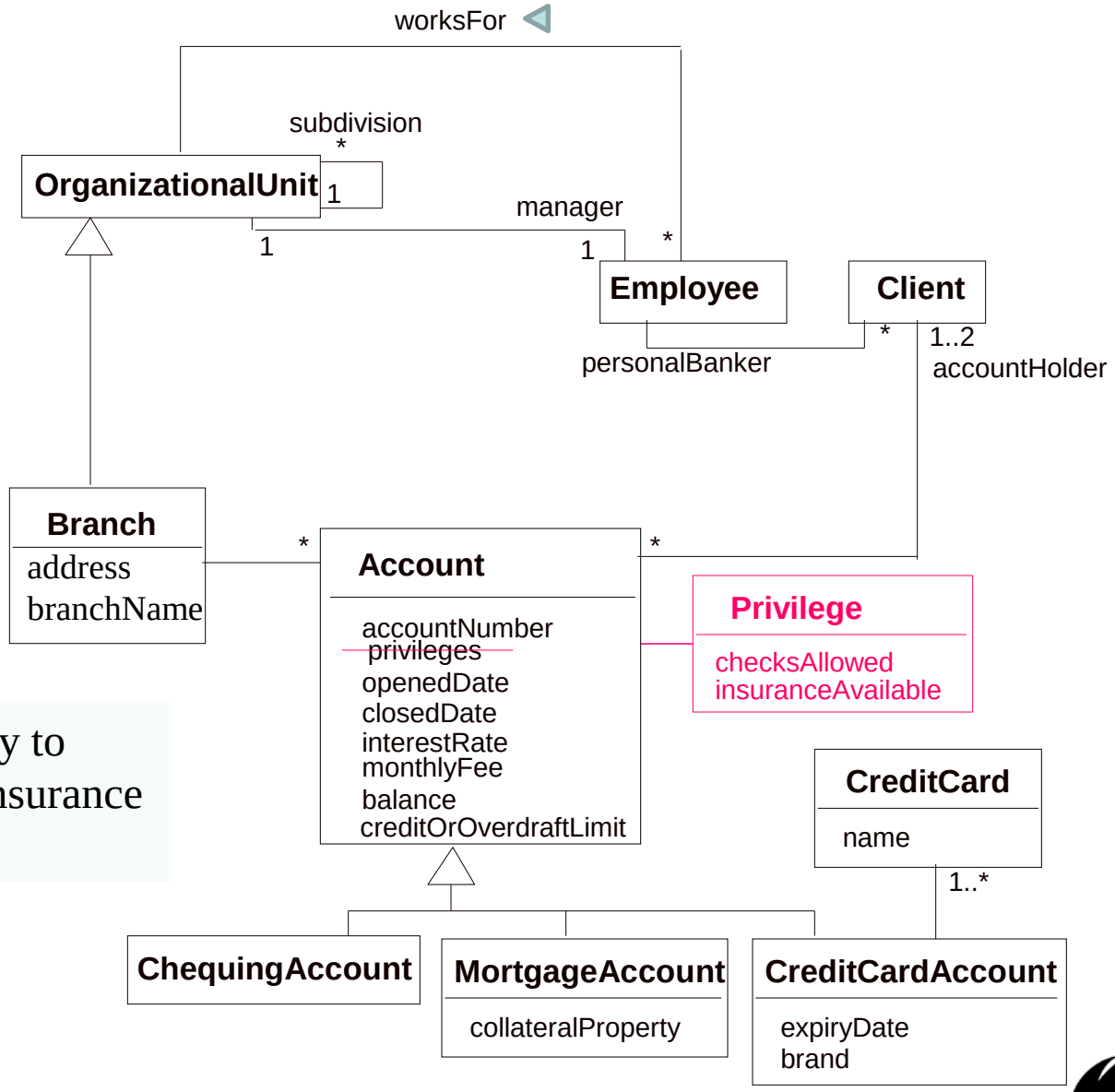
the branches are considered subdivisions of the Consumer Division.

Each division has a manager

and a set of other employees.

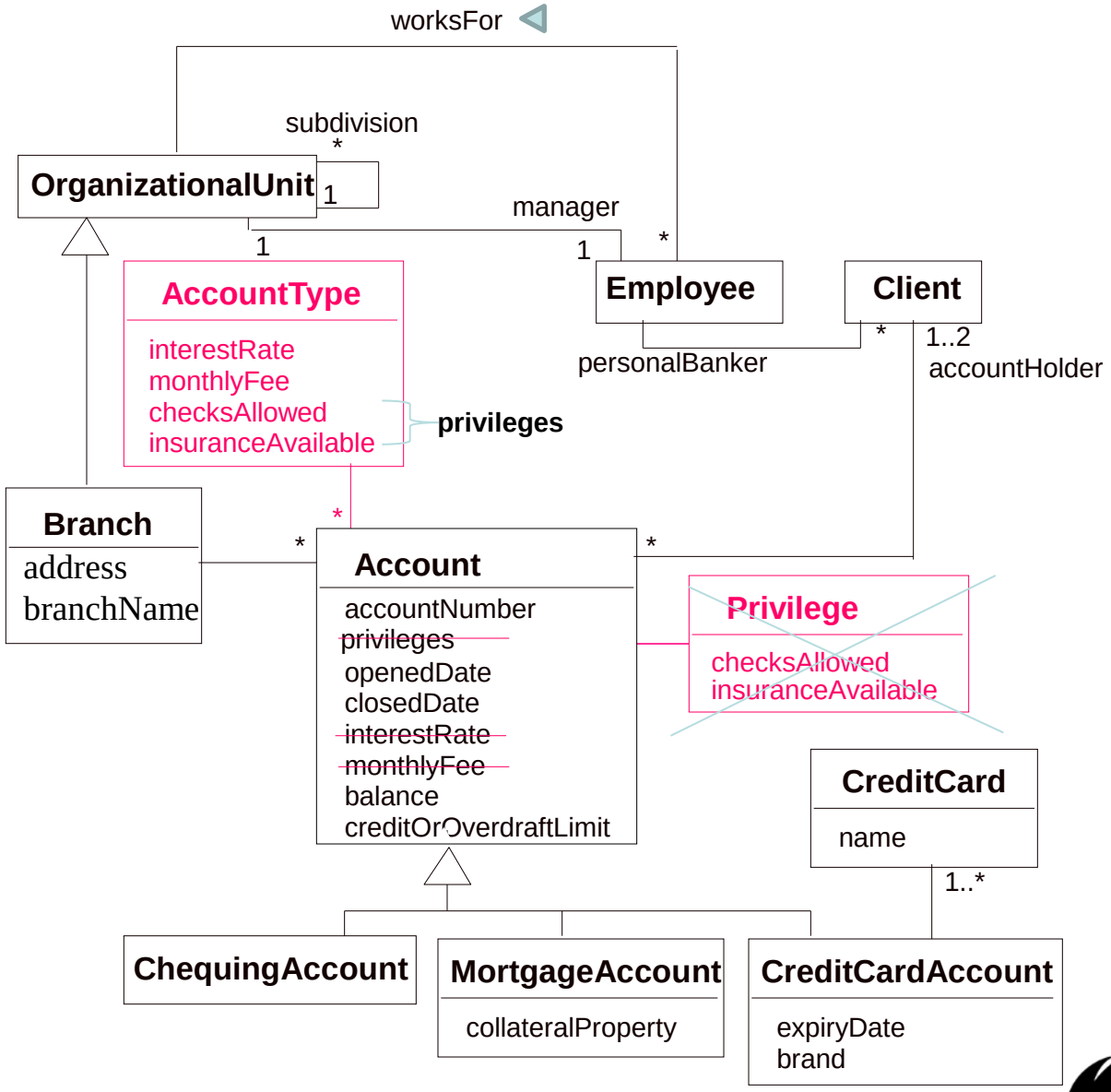


Class diagrams: draft class diagram(a)

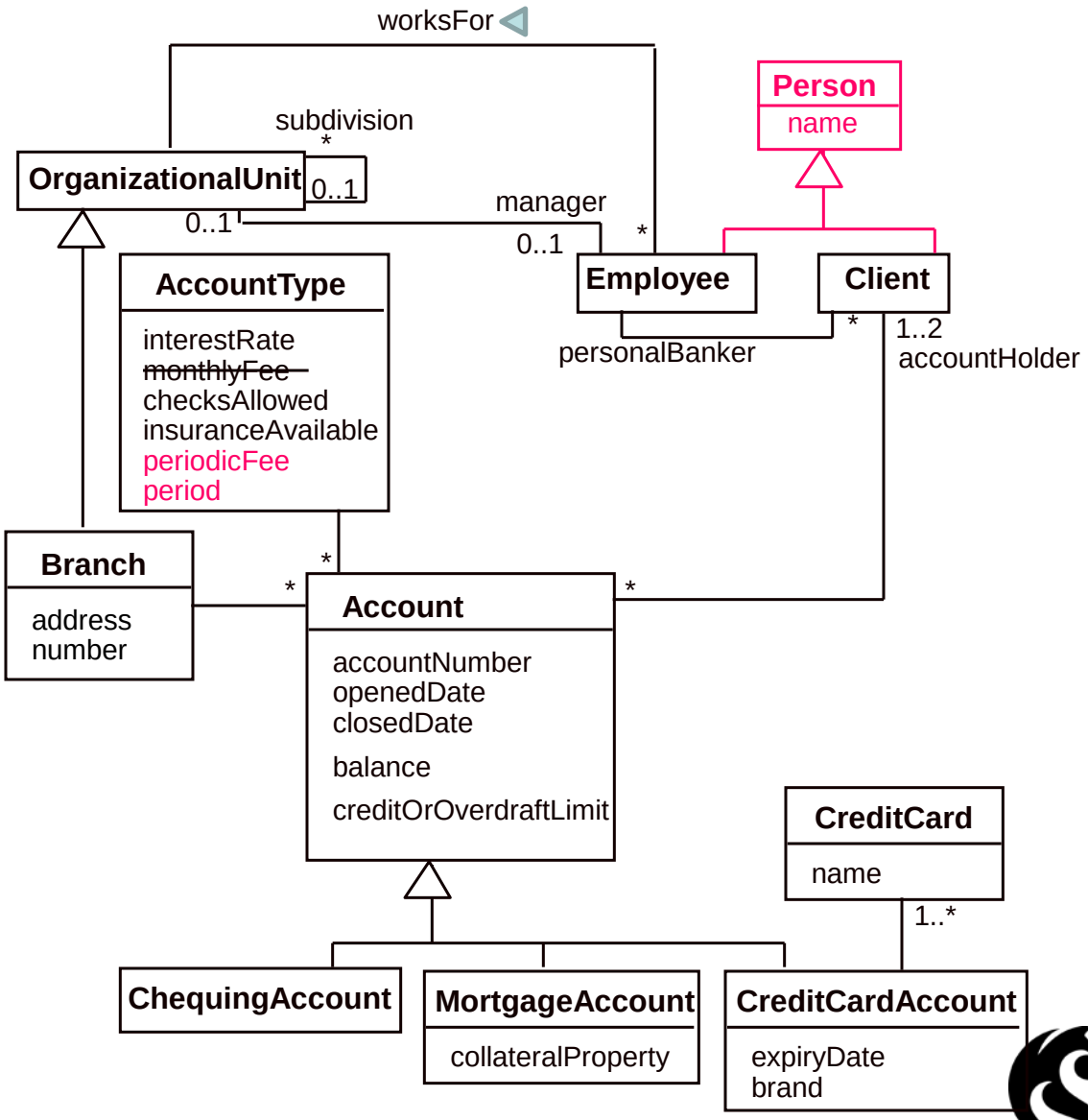


Privileges: ability to write cheques, insurance for purchases

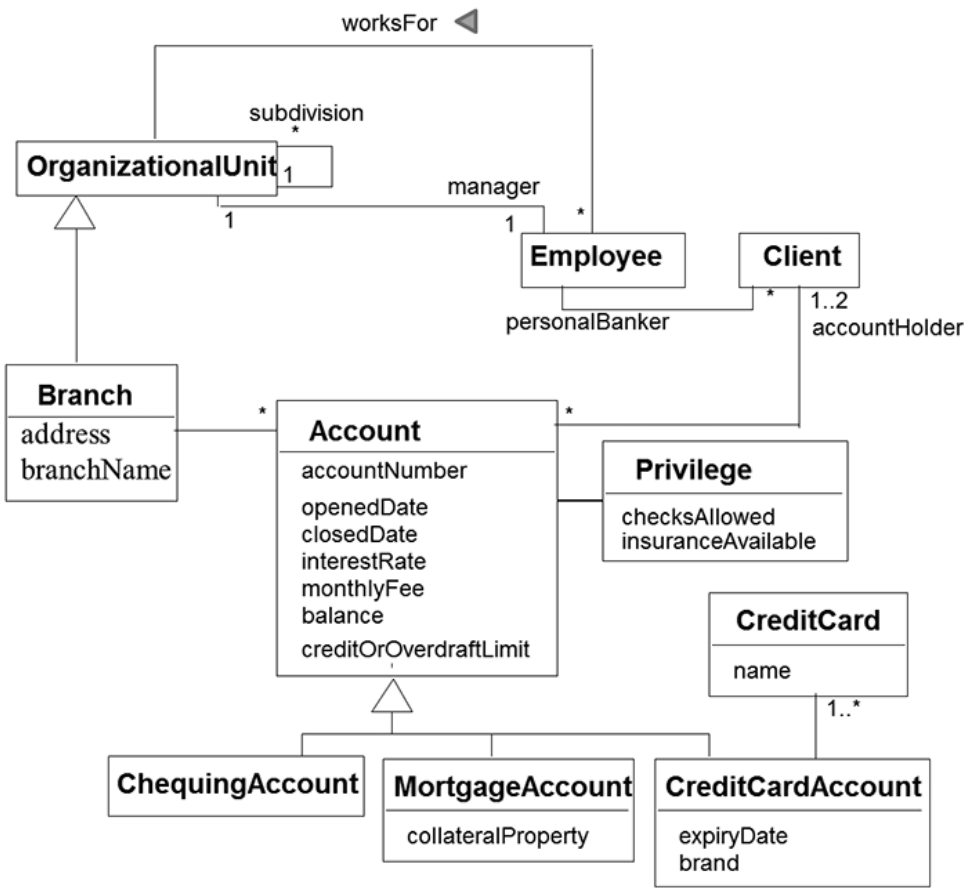
Class diagrams: draft class diagram(a)

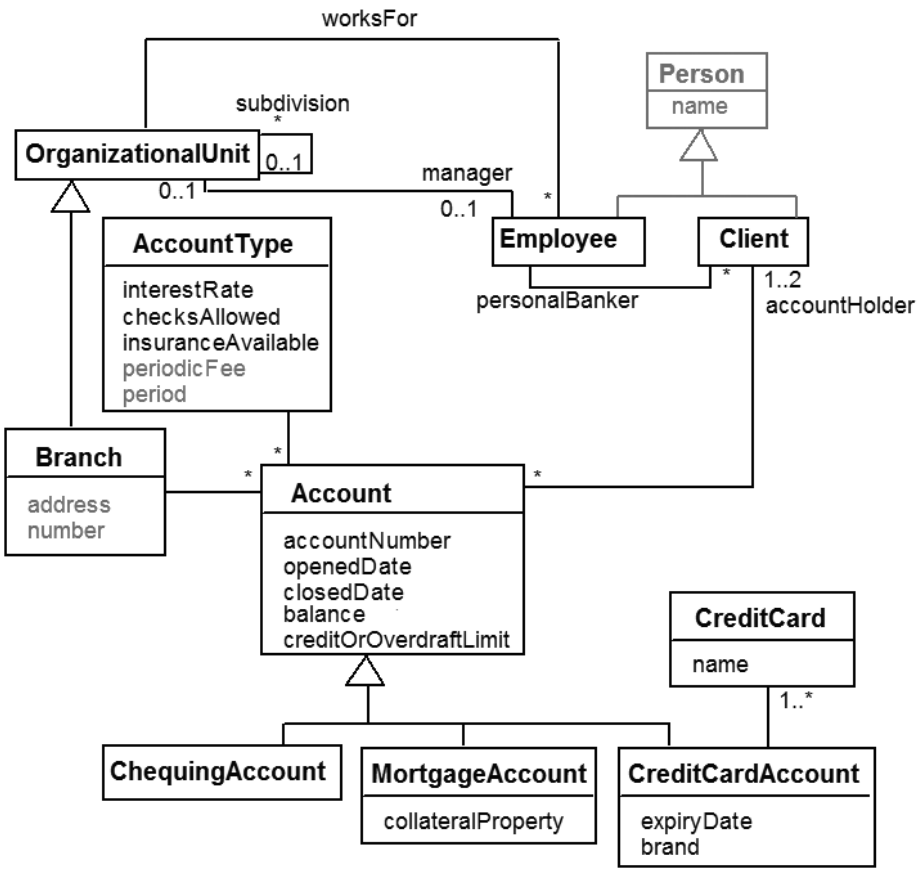


Class diagrams: draft class diagram(b)



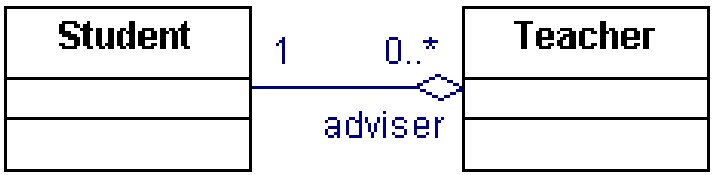
Class diagrams: draft class diagram(a/b)

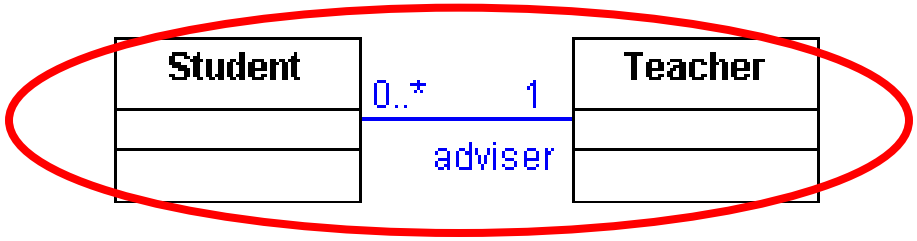


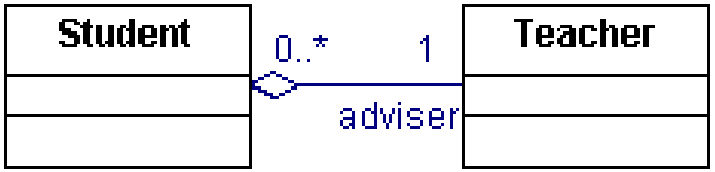


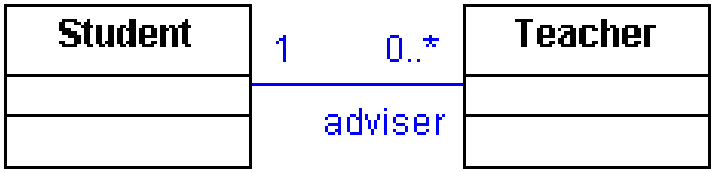
Class diagrams: Quiz

- Every student at TS College will be advised by one teacher. Some teachers advise many students, and some advise none. Which of the following class diagrams most clearly represents that student-teacher relationship?









UML Diagram

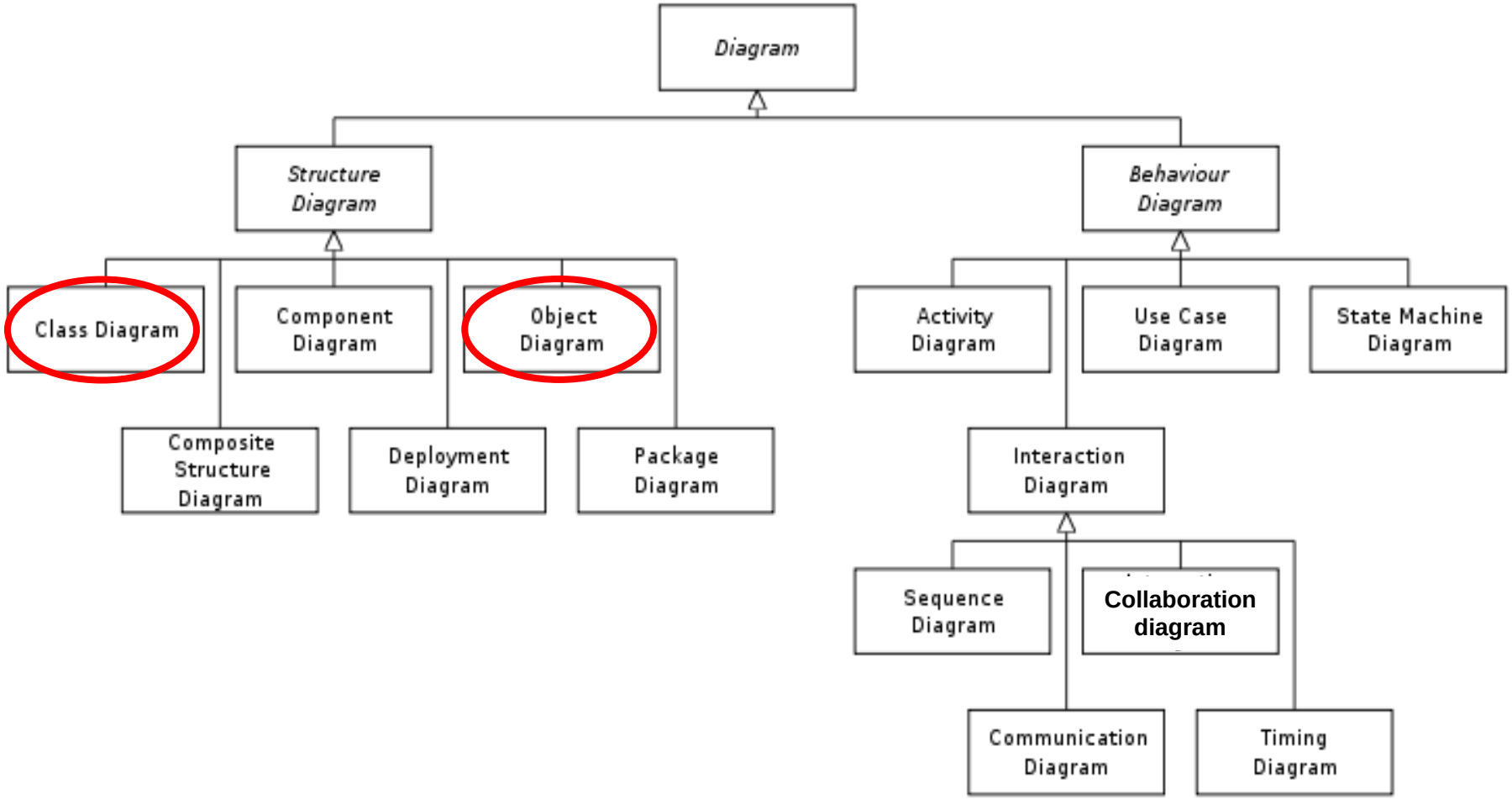
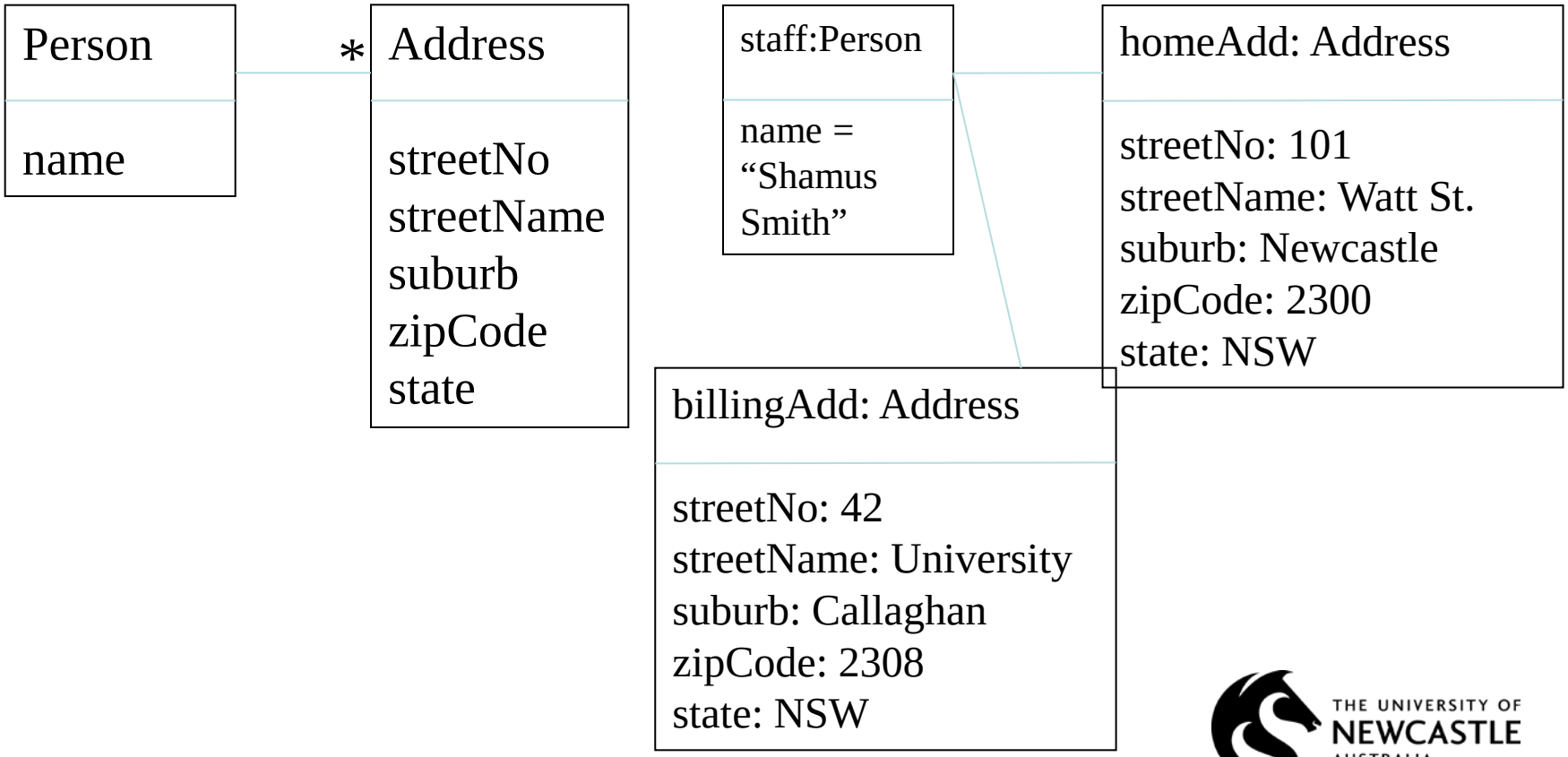


Diagram
Structure Diagram
Behaviour Diagram
Class Diagram
Component Diagram
Object Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Interaction Diagram
Sequence Diagram
Collaboration diagram
Communication Diagram
Timing Diagram

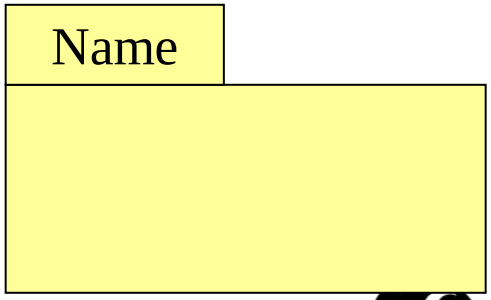
Class diagrams: Object diagram

- In UML, object diagram provide a snapshot of the *instances* in a system and the relationships between the instances.



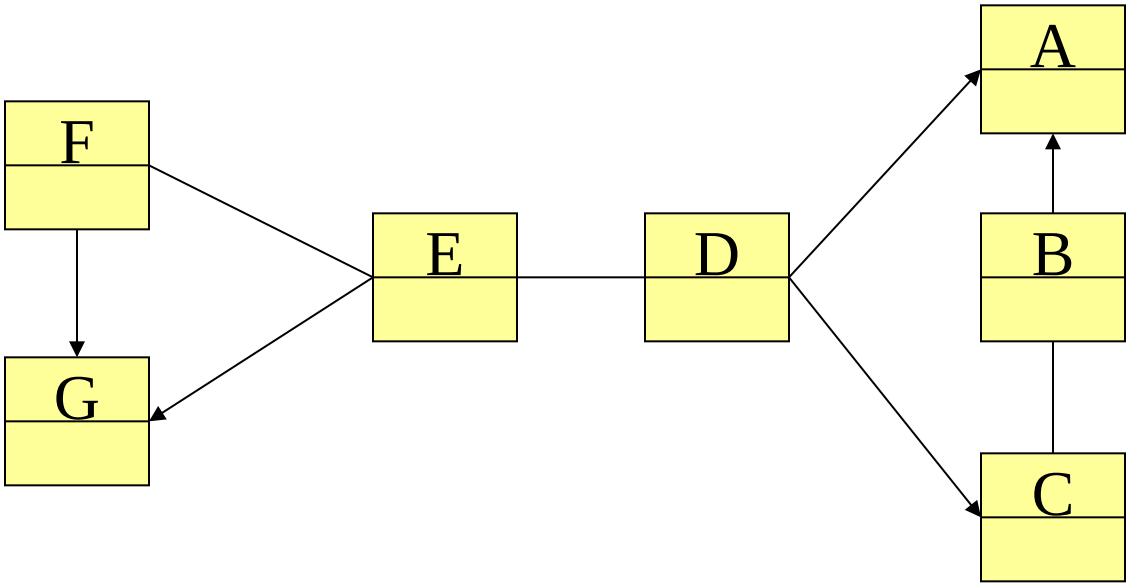
Class diagrams: UML Packages

- A package is a general purpose grouping mechanism.
  - Can be used to group any UML element (e.g. use case, actors, classes, components and other packages).
- Commonly used for specifying the logical distribution of classes.
- A package does not necessarily translate into a physical sub-system.



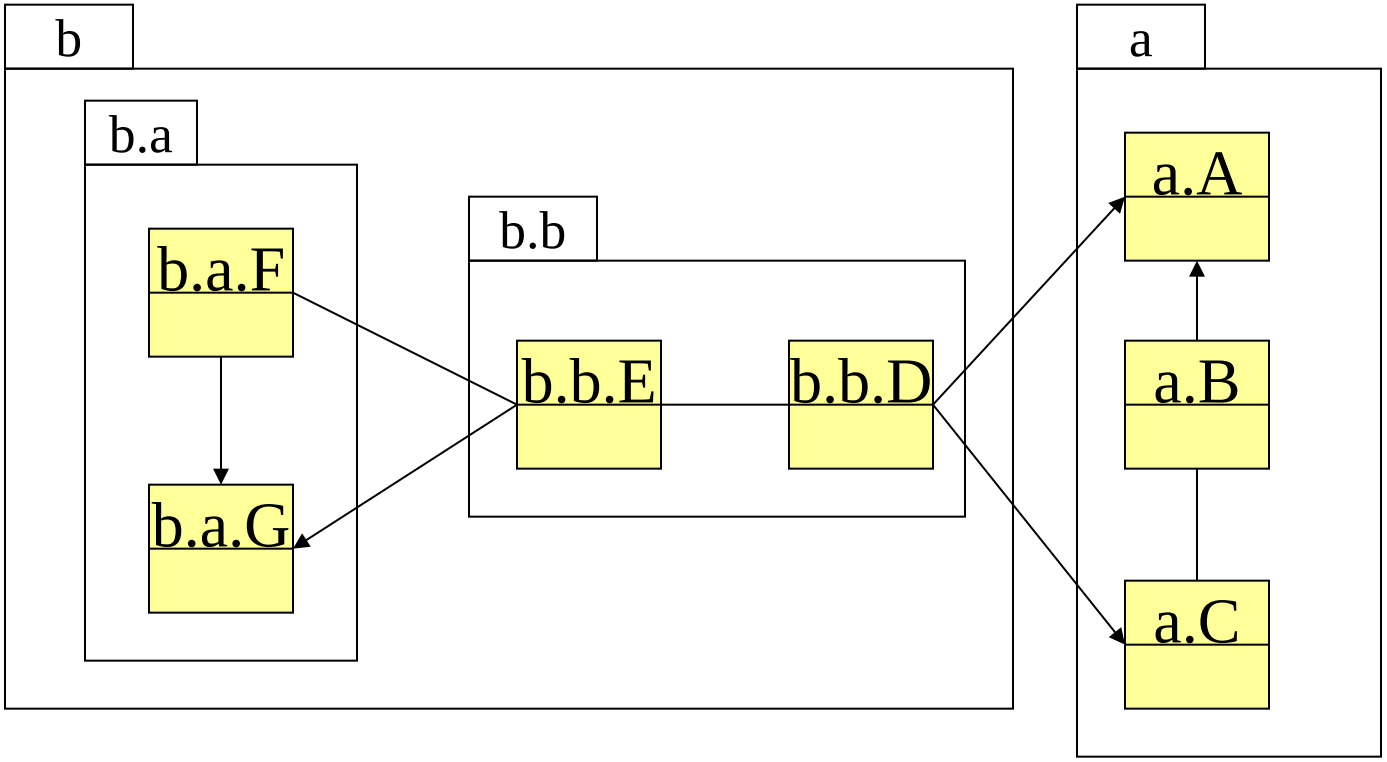
Class diagrams: UML Packages

- Add package information to class diagrams



Class diagrams: UML Packages

- Add package information to class diagrams

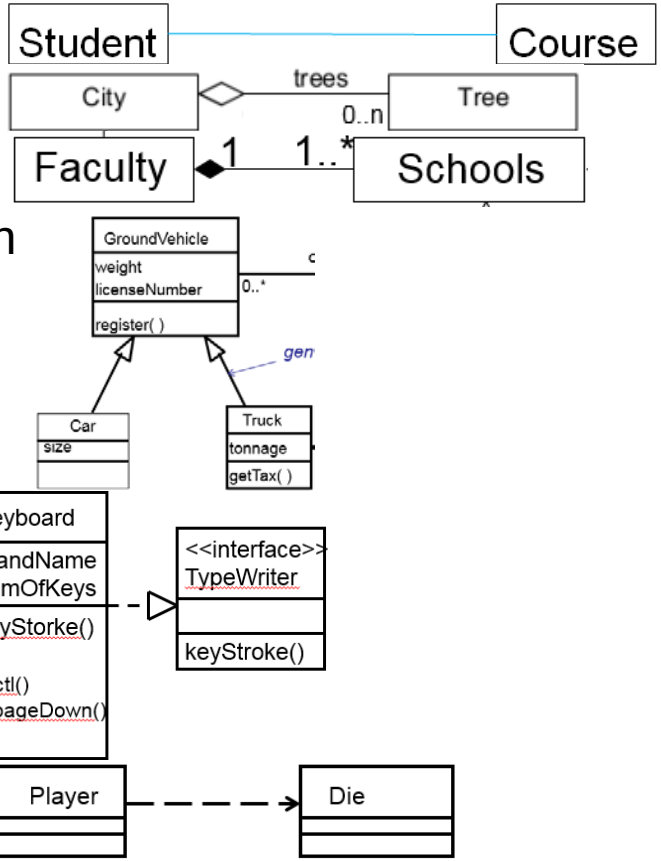


Summary

- Classes and objects
  - A class is a description of a group of objects with common properties (attributes) and behaviour (operations)
  - An object is an instance of a class
- Class diagram
  - A class diagram is a diagram describing the structure of a system
  - Classes, attributes, operations and relationships among the classes

Summary

- Relationships
  - Associations
  - Aggregation
  - Compositions
  - Generalization
  - Realization
  - Dependency



Next Week

Sequence diagrams