Lecture 5 – Analysis. Domain modeling. Class diagrams and object diagrams

Bogumiła Hnatkowska

Lecture objectives

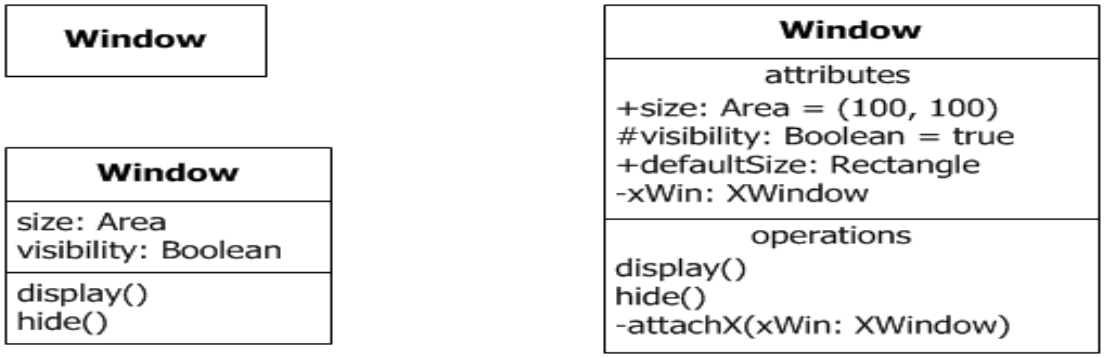
- To present areas in which class diagrams can be applied
- To present the basic elements of class diagrams and object diagrams
- To practice with class diagram’s basic elements

Class diagrams – what they are for?

- Class diagram – a graph that shows a collection of static model elements (classes), their contents and relationships between them
- Potential area of application:
  - Domain modeling – for presenting entities existing in problem domain, and relationships between them (this model helps in understanding the domain)
  - Software modeling – for presenting static view of an application when the application is to be built in object-oriented paradigm (this model can be further automatically translated into source code being its documentation)
  - Database modeling – for presenting conceptual data models (this model can be further automatically translated into SQL scripts)

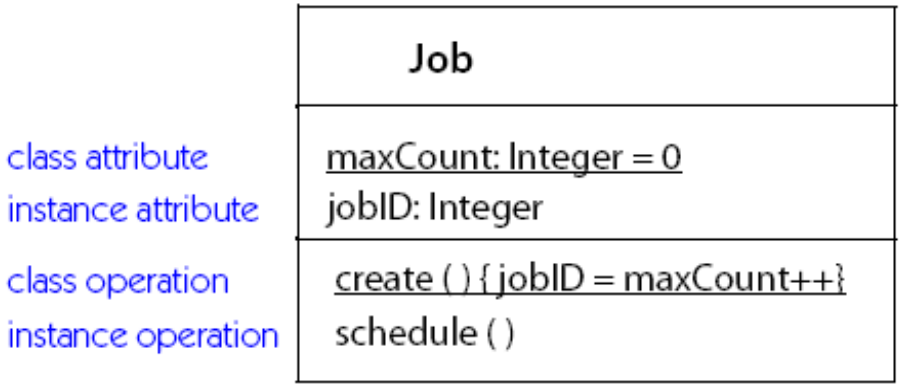
What is a class?

- Class – descriptor for a set of objects that share the same features: properties (structural), and operations (behavioural).
- Property – represents an attribute (typically of data type) and/or a member end of association (typically of a class type).



What is a static feature?

- A feature which characterize the whole class (class feature), not specific instances
- A class can have static attributes and/or operations




```
public class Job {
    static int maxCountInteger = 0;
    int jobId;

    Job() {
        jobId = maxCountInteger++;
    }
    void schedule() {}
}
```

What is an attribute?

- Attribute is the description of a named element of a specified type in a class; each object of the class separately holds a value of the type.
  - Name
  - Visibility (the same for other elements):
    - - (private)
    - + (public)
    - # (protected)
    - ~ (package)
  - Multiplicity
  - Initial values
- Examples:

+size: Area = (100,100)	Public with initial value
#visibility: Boolean = invisible	Protected with initial value
<u>maximum-size: Rectangle</u>	Static without initial value
-colors : Saturation [3]	Private; An array of 3 saturations
points : Point [2..*]	An array of 2 or more points
name : String [0..1]	If the name is missing, it is a null value.

What is an operation?

- **Operation** is a specification of a transformation or a query that an object (class instance) may be called to execute.
- **Method** describes how an operation is implemented.
- An operation has:
  - Name.
  - Parameter list.
  - Return type.
  - Visibility
  - [Stereotypes]
- Examples:
  +display (): Location
  +hide ()
  «constructor» +create ()
  -attachXWindow(xwin:Xwindow)

What is an abstract class?

- Abstract class – a class without direct instances (every instance of the abstract class shall be an instance of one of its specializations)


```
abstract class Animal {

}
```

Relationships

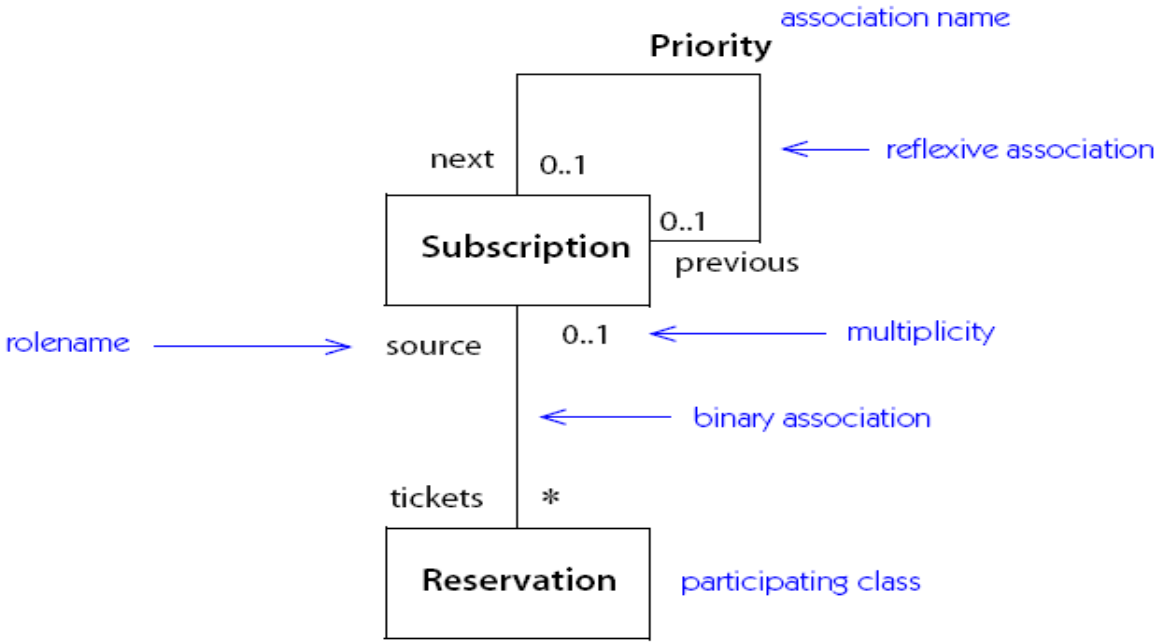
- Associations
  - Binary (linking two classes)
  - N-ary (linking more than two classes)
  - Aggregation
  - Composition
- Association classes
- Generalizations

Association

- Association – the semantic relationship between two or more classes that involves connections among their instances.
- Association can be described by a name
- Association end – connection of an association to a class:
  - Role name
  - Visibility
  - Multiplicity

Binary association

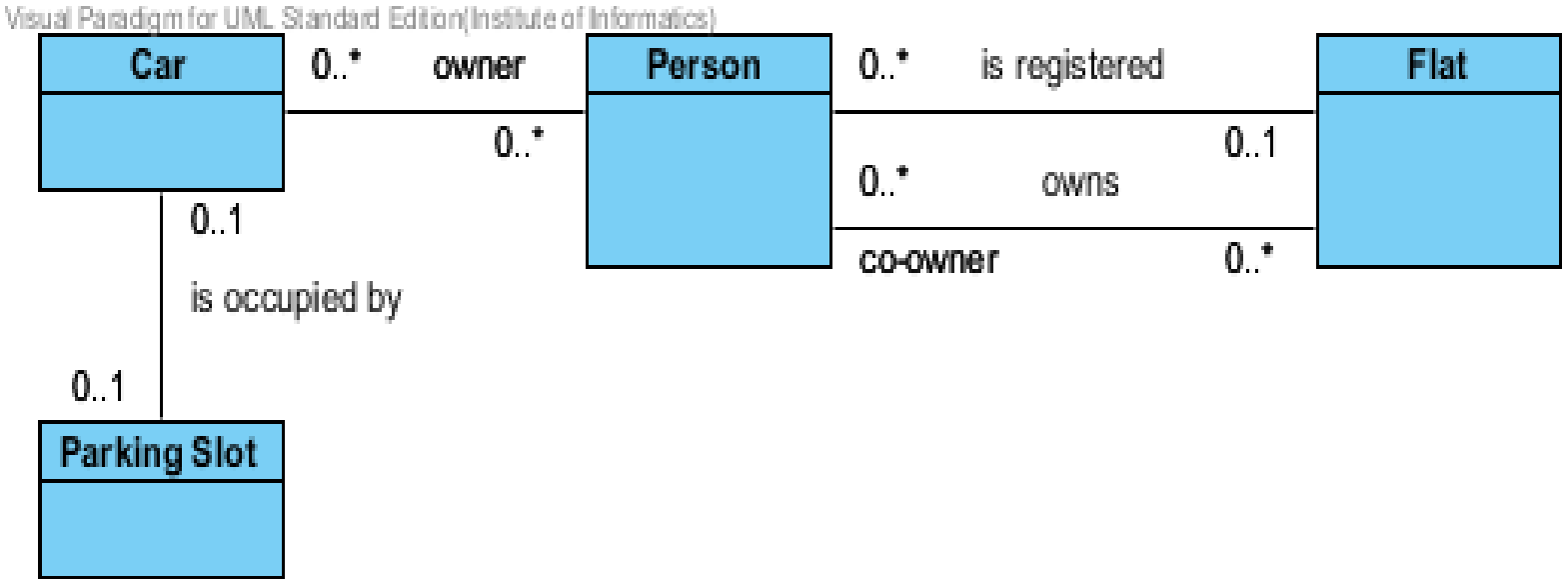
- Binary association – a line or path connecting the participating classes.
- Self-association – association with both ends linking the same class



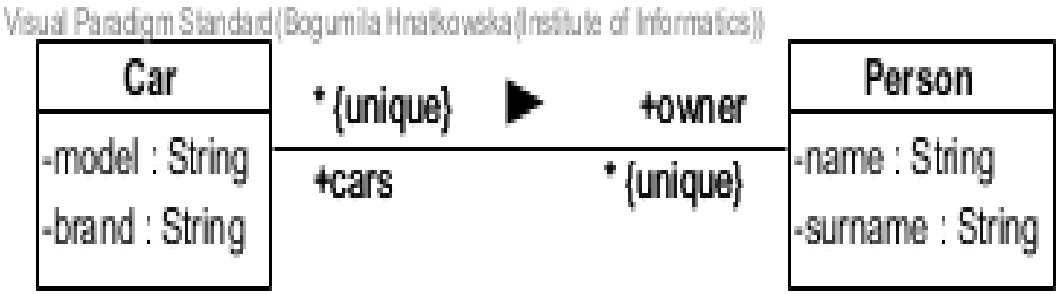
Binary associations – exercise

- A person (owner) can have many cars. A car can have many owners.
- A car may stay on one parking slot, but the parking slot may be also empty.
- A person can have many flats. A flat can be owned by many co-owners.
- A person can be registered in at most one flat, and many people can be registered in the same flat.

Binary association – solution



Binary association – code example




```
public class Car {
  private String _model;
  private String _brand;
  public Set<Person> owner =
     new HashSet<Person>();
}
```

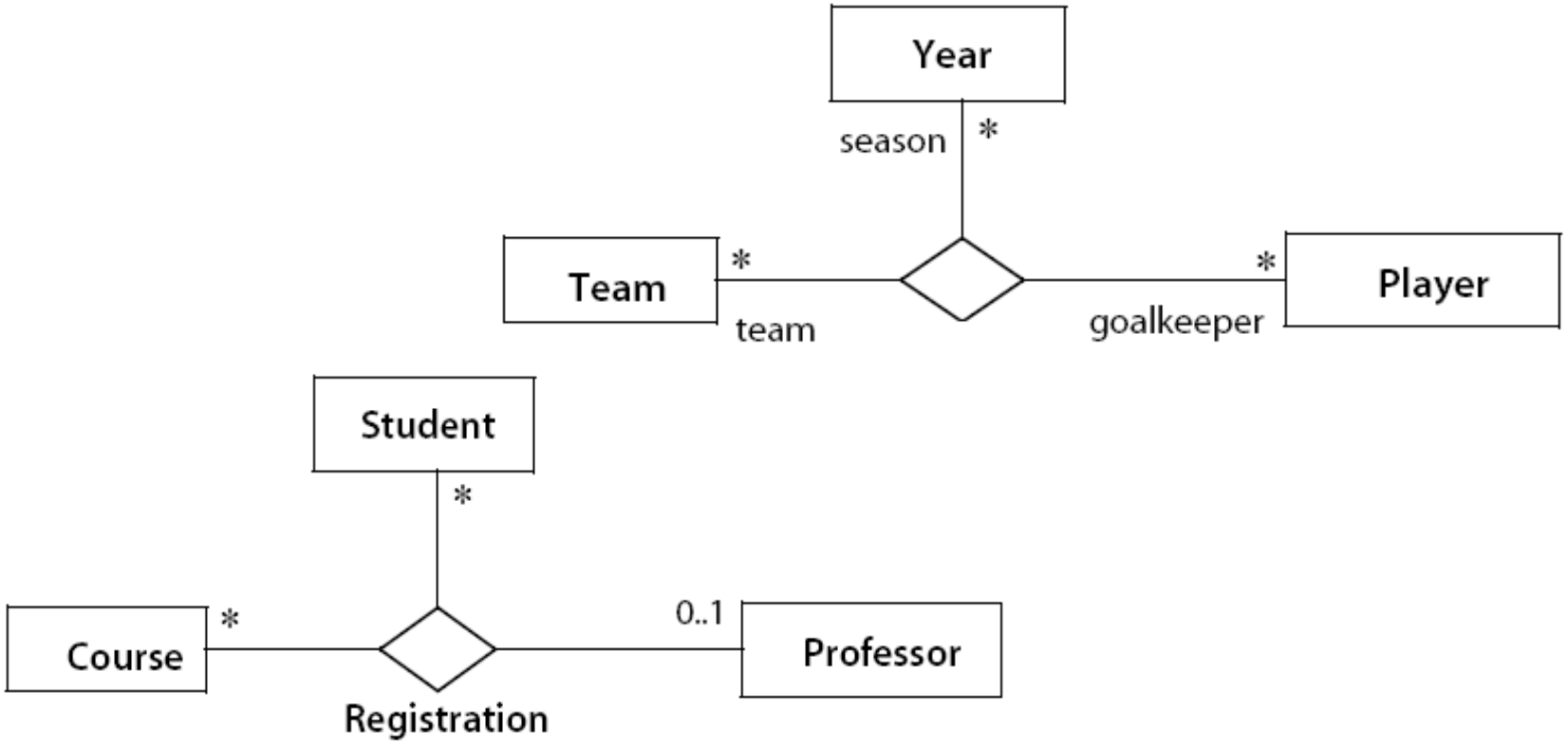
```
public class Person {
  private String _name;
  private String _surname;
  public Set<Car> cars =
     new HashSet<Car>();
}
```

Binary associations – exercise

- A library (name, address) has many rooms (identified by a number, area in m2)
- Each room can store zero or more shelves
- A shelf (identified by a number) has many books; a book stays on at most 1 shelf
- A book can have many authors (0 if the author is unknown)
- One author can write more than 1 book

N-ary associations

- Association among three or more classes.
- Each instance of the association is an n-tuple of values.

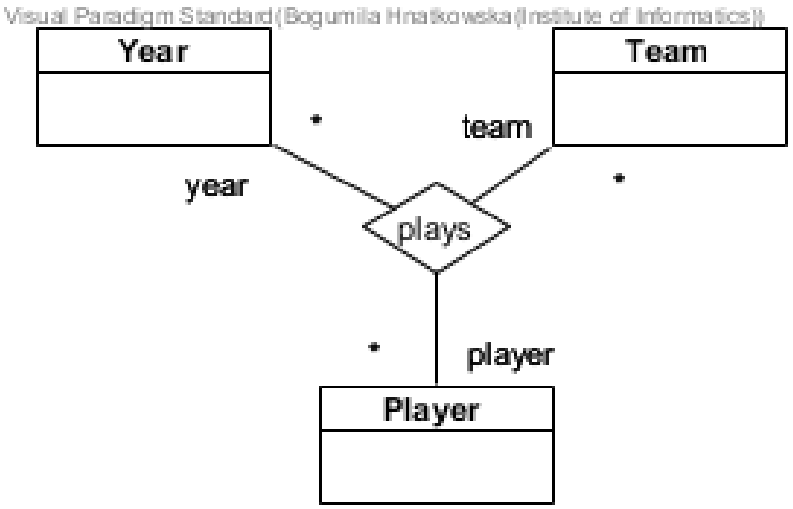


N-ary association – exercise

- An university offers many faculties to students
- Within one university a student can study on many faculties (at least one)
- A student can study at many universities

- A task included in a schedule is assigned to one person
- A person in a schedule can have many tasks assigned

N-ary association - implementation



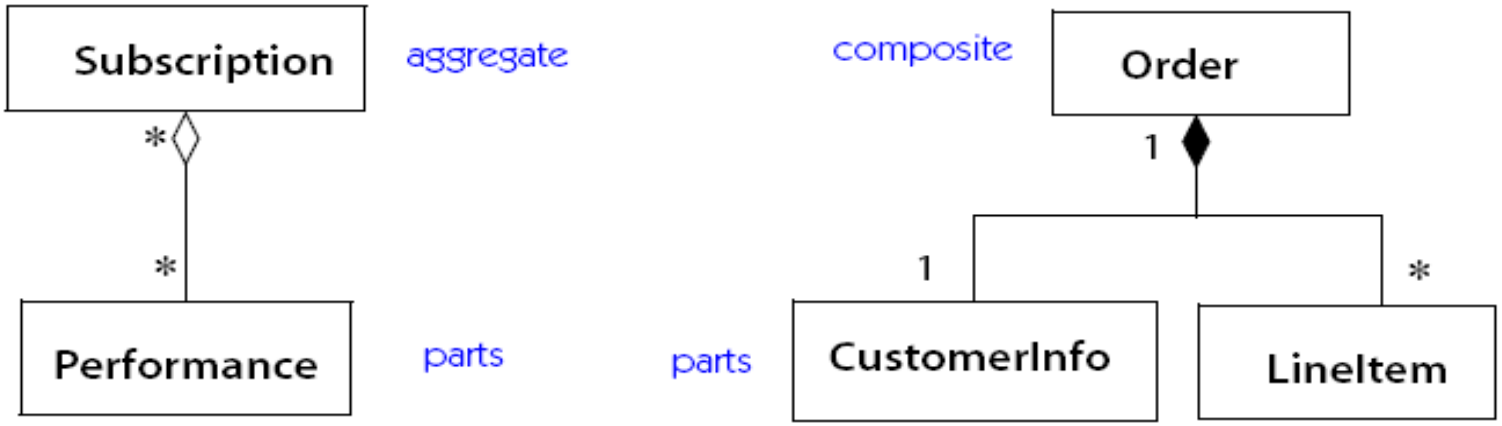

```
public class Player {
 // can contain only those plays
 // which contained  a specific Player
 // public List<plays> plays;
}
public class Year {
  // as above
}
public class Team {
  // as above
}
```

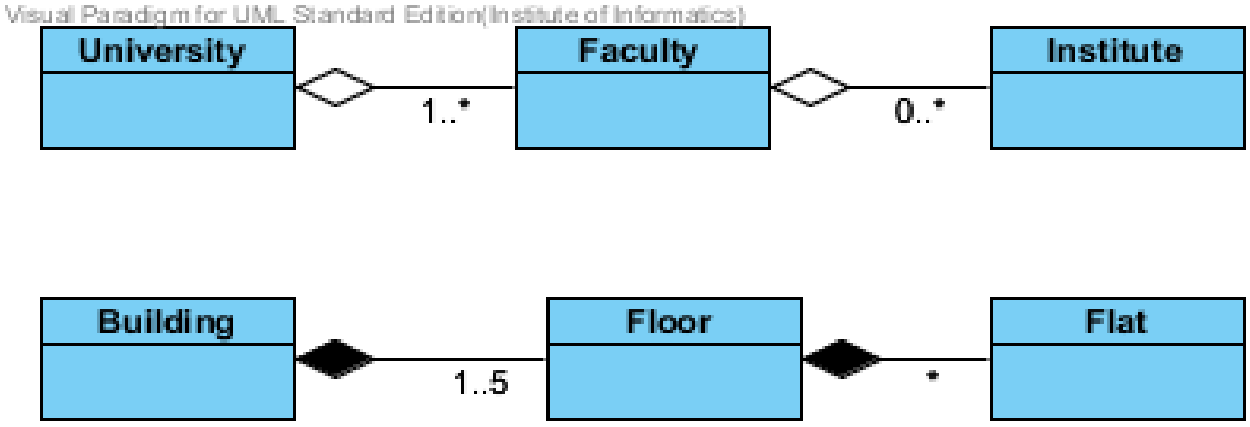
```
public class plays {
   // attributes mustn't be nulll
   public Player player;;
   public Team team;
   public Year year;
}
```

Aggregations and compositions

- **Aggregation** – an association that represents a „has-a relationship between an aggregate and its parts. There is no life duration relationship between the aggregage and the part (the part can live longer than the whole)
- C**omposition** – an association that represents „a part of" relationship between a composite and its parts; a stronger form of aggregation in which the composite has sole responsibility for managing its parts, such as their allocation and de-allocation. There is life duration relationship between the whole and the part (while destroying the whole also parts are destroyed). An object may belong to at most one composition.
- **Aggregation/Composition** is meaningful only for binary associations.

Aggregations and compositions, cont.



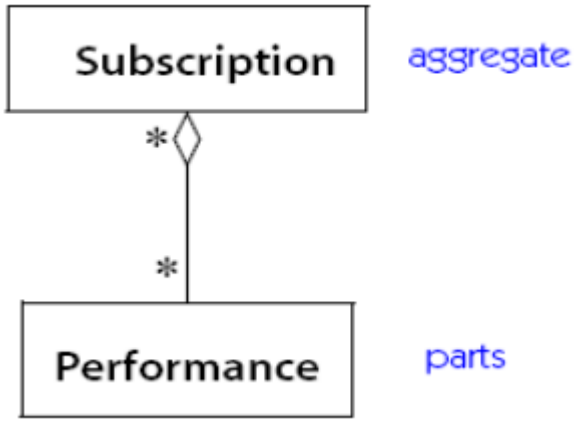


Aggregation and compositions, exercise

- Select a proper relations linking:
  - Player and Team
  - Invoice and Invoice Line
  - Book, Chapter, Subchapter

Aggregation – implementation

- See implementation of association



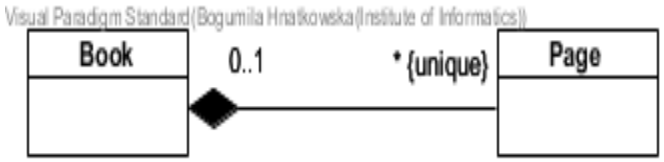
Composition – implementation

Note – difficult to be fully mimic in Java or c#

```
class Book {
   private String title;
   private static int pageNr = 1;
   private List<Page> pages = new ArrayList<>();

   public Book(String title){
      this.title = title;
   }

   // Book manages pages
   public void addPage(String content){
      pages.add(new Page(pageNr++,
         content, this));
   }
```

```
   // Book manages pages
   public Page removePage(int pageNr){
      Page page = pages.stream()
      .filter(p -> p.nr == pageNr).findFirst().orElse(null);
      if (page != null) {
         pages.remove(page);
         page.book = null;
      }
      return page;
   }

public List<Page> getPages(){
      return Collections.unmodifiableList(pages);
   }
```

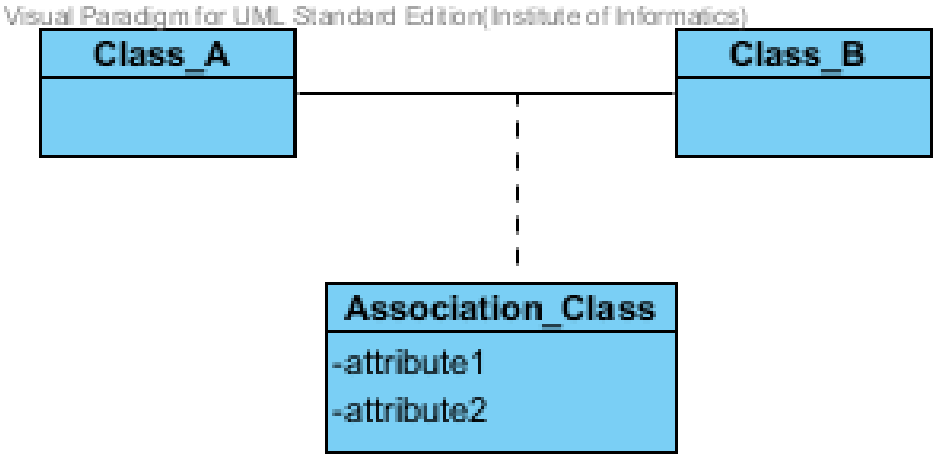
Composition – implementation



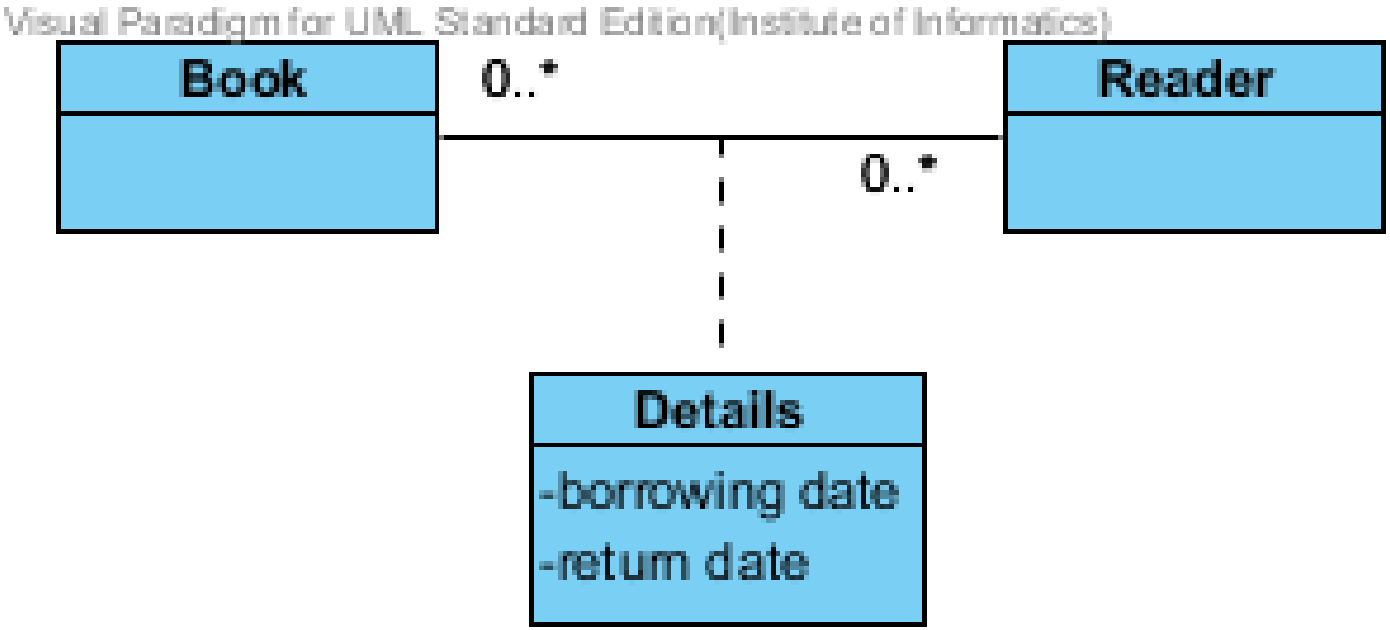

```
// Ummutable class
public class Page {
   private int nr;
   private String content;
   private Book book;

   private Page(int nr, String content, Book book){
      this.nr = nr; this.content = content;
      this.book = book;
   }

   public String getContent(){ return content; }
   public int getNr(){ return nr; }
  }
}
```

Association class

- Association class = class + association
- Used when association has some properties



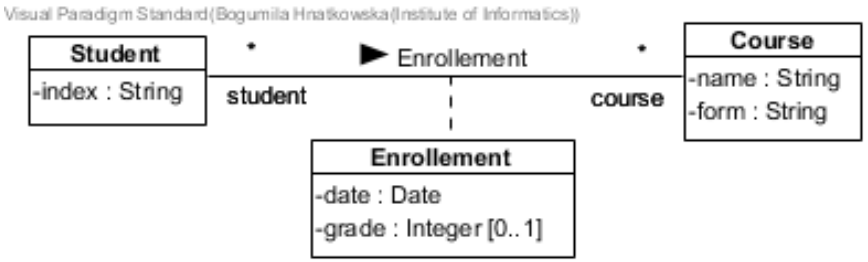
Association class – example



Association class – exercise

- Student enrolls for a course on specific date and time
- Two cities can be connected with many roads each of which has a name and a length
- Two people can be married playing different roles (wife, husband); we would like to remeber the marriage date and place

Association class – implementation




```
public class Enrollement {
  private Date date;
  private Integer grade;
  Student student;
  Course course;
}
```

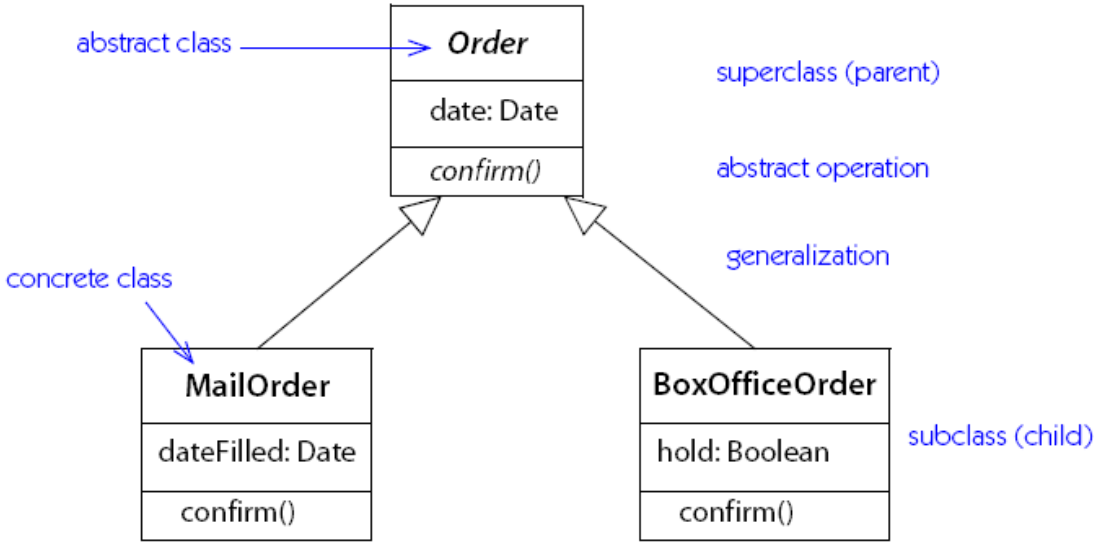
```
public class Course {
  private String name;
  private String form;
  public ArrayList<Enrollement> enrollements;
  // we can add e.g. Set<Course> getCourses(){}
}
```

```
public class Student {
  private String index;
  public ArrayList<Enrollement> enrollements;
}
```

Generalization

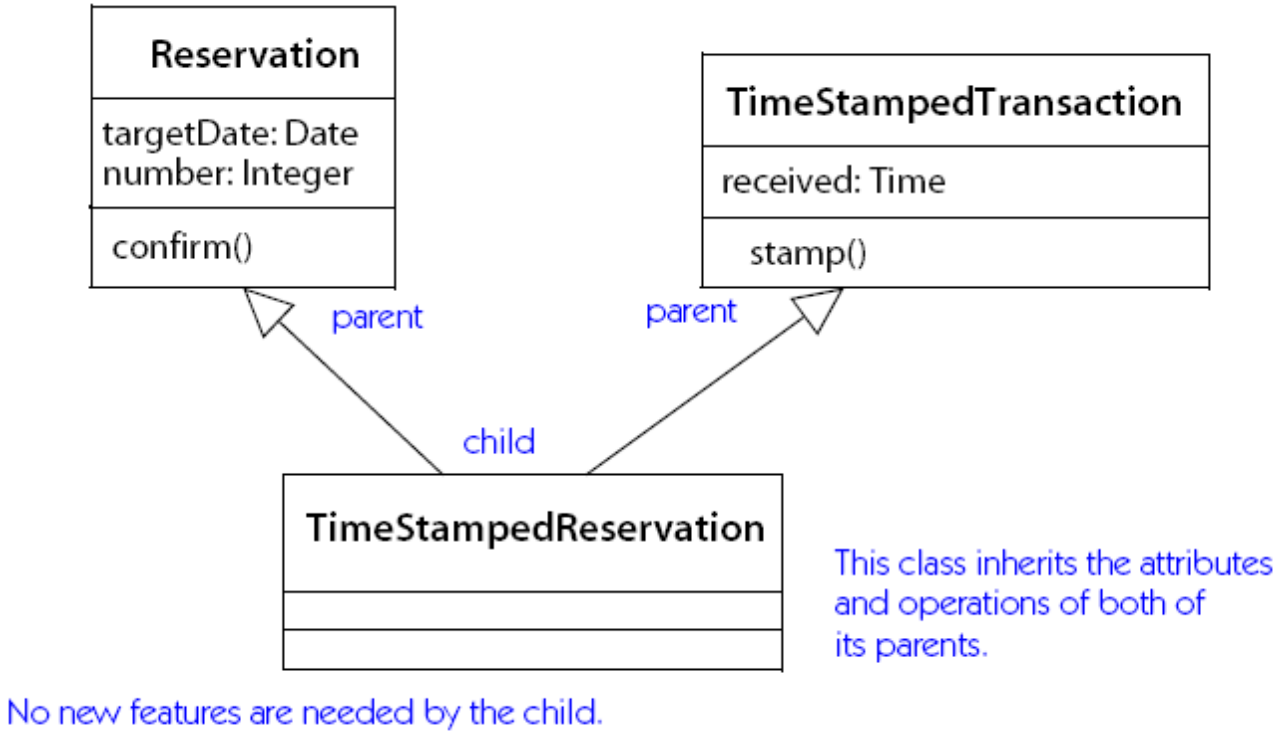
- Generalization – „is-a” relationship between more general and more specific description that builds on it and extends it.
- The more specific description is fully consistent with the more general one.
- Generalization relationship constitutes acyclic graph.
- Generalization enables polymorphism.
- Kinds of inheritance:
  - Simple
  - Multiple

Generalization, cont.




```
abstract class Order {}
class MailOrder extends Order {}
class BoxOfficeOrder extends Order {}
```

Generalization, cont.

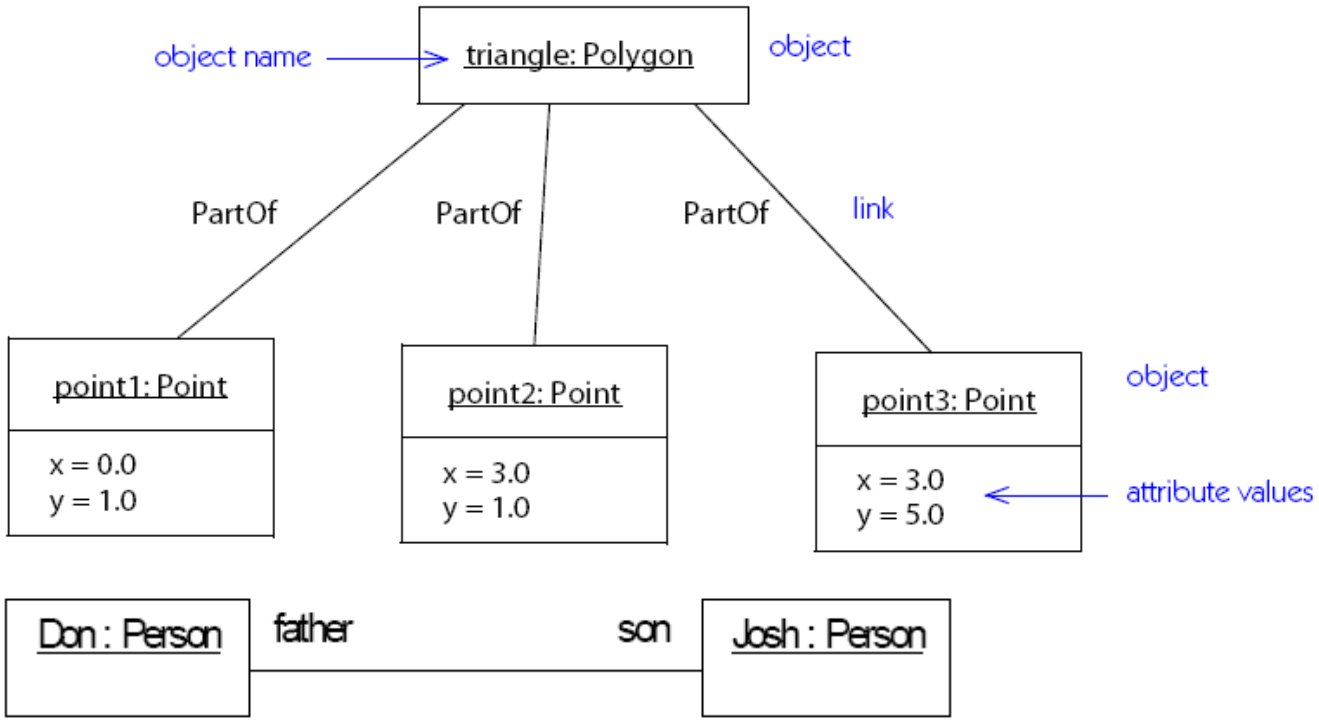


Generalization – exercise

- Use generalization relationship to connect entities:
  - Number, Integer, Real, Prime
  - Team member, Programmer, Tester, Analyst
  - Alcohol, Vodka, Wine, Red Wine, White Wine, Dry Wine, Sweet Wine, Beer

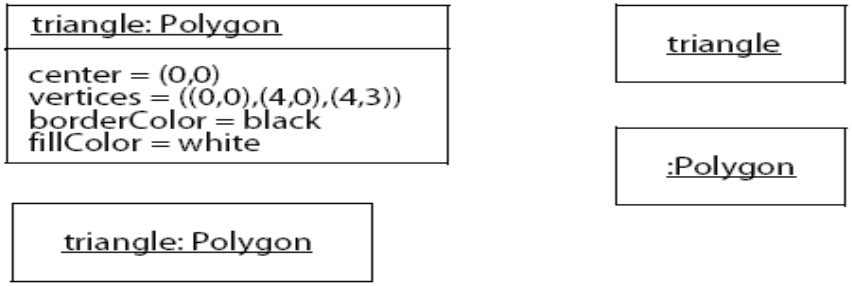
Object diagrams

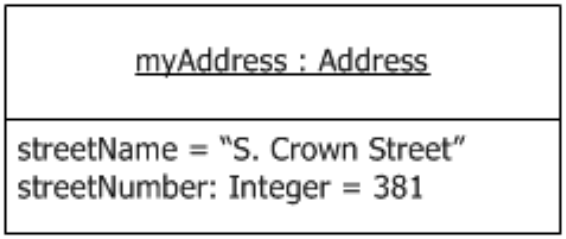
- Object diagram – a snapshot of a system at a point in time; Contains objects and links.



What is an object?

- Object – an instance of a class – a discrete entity with a well defined boundary and identity that encapsulates state and behavior.

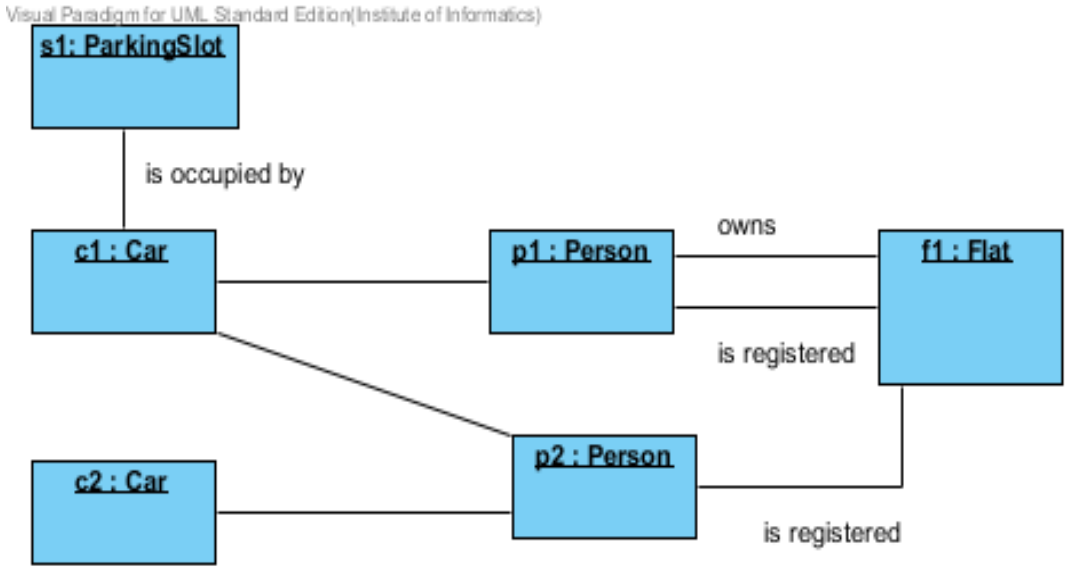




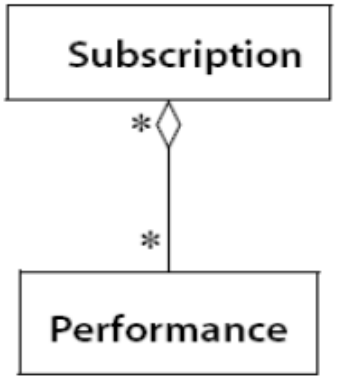

```
Address myAddress =
        new Address( s: "S. Crown Street",  i: 381);
Area medium = new Area();
```

Link – association instance

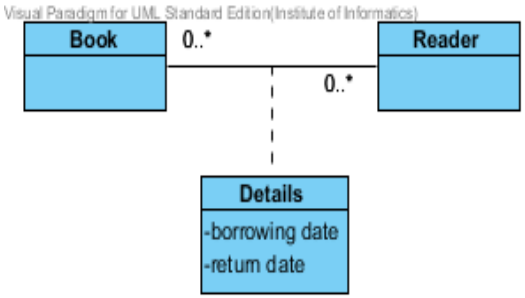
- Link – an instance of an association
- Link is represented in the same way the association is – as a line (for binary associations), as a diamond with outgoing lines (for n-ary associations)



Interpretation of association



- Is it possible for a specific performance to be linked with the same subscription many times?
- According to the UML semantics NO unless at least one end of association has the property isUnique = false



- NOTE. Even when all ends of the Association Class have isUnique=true, it is possible to have several links associating the same set of instances of the end Classes.

Revision

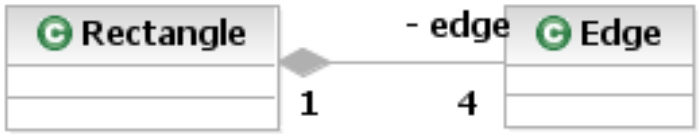
- What can class diagram be used for?
- What are the basic elements of class diagram?
- What kind of relationships can be modeled on class diagrams?
- What is the main difference between aggregation and composition?

Exercises

- Model the following expressions. Name the associations, if it is possible define roles and multiplicities. Write exemplary model instantiations:
  - Each person has two parents (mother and father); a given woman may have many children with different men and a given man may have many children with different women
  - There are many different kind of furniture in the class (tables, chairs, wardrobes, etc.). Each furniture at one time can stand only in one room
  - The course is planned in a given room. However, each year the room can be changed. Students enroll to particular courses. The max number of students is 100 when the min is 10. The course is provided by a teacher. A teacher can teach many courses.

Exercises, cont.

- Write exemplary models instantiations (object diagrams)



- Give a possible interpretation for a diagram
- Translate above given diagrams into a source code