Revision of OOD, Composition, Implementation hiding, UML

Dr. Abdallah Karakra |Comp 2311 | Masri504

4/12/2023

جامعة بيرزيت
BIRZEIT UNIVERSITY

Index

Chapter 10

- Section 10.2.
- Section 10.3.
- Section 10.4.

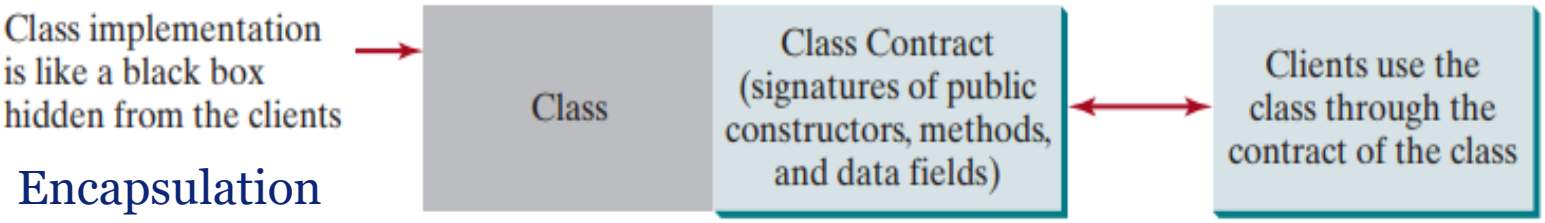
CHAPTER 10

Object-Oriented Thinking

Class Abstraction and Encapsulation

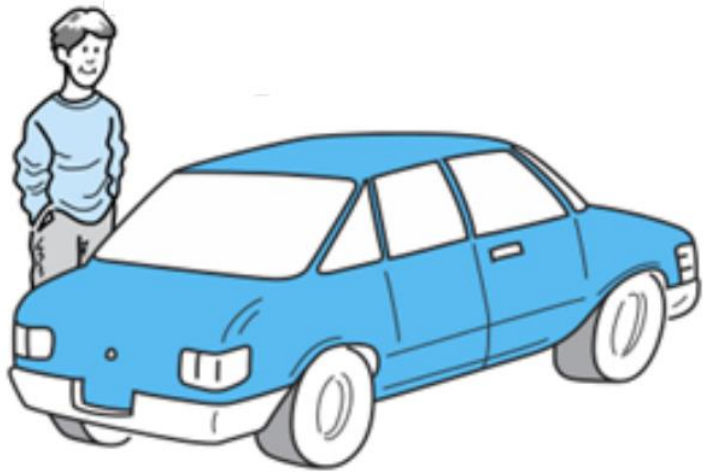
Class abstraction is separation of class implementation from the use of a class. The details of implementation are encapsulated and hidden from the user. This is known as class encapsulation

- Class abstraction means to separate class implementation from the use of the class.
- The creator of the class provides a description of the class and let the user know how the class can be used.
- The user of the class does not need to know how the class is implemented. The detail of implementation is encapsulated and hidden from the user
- A description and collection of public constructors, methods, and field that are accessible from outside the class is called the Class's Contract

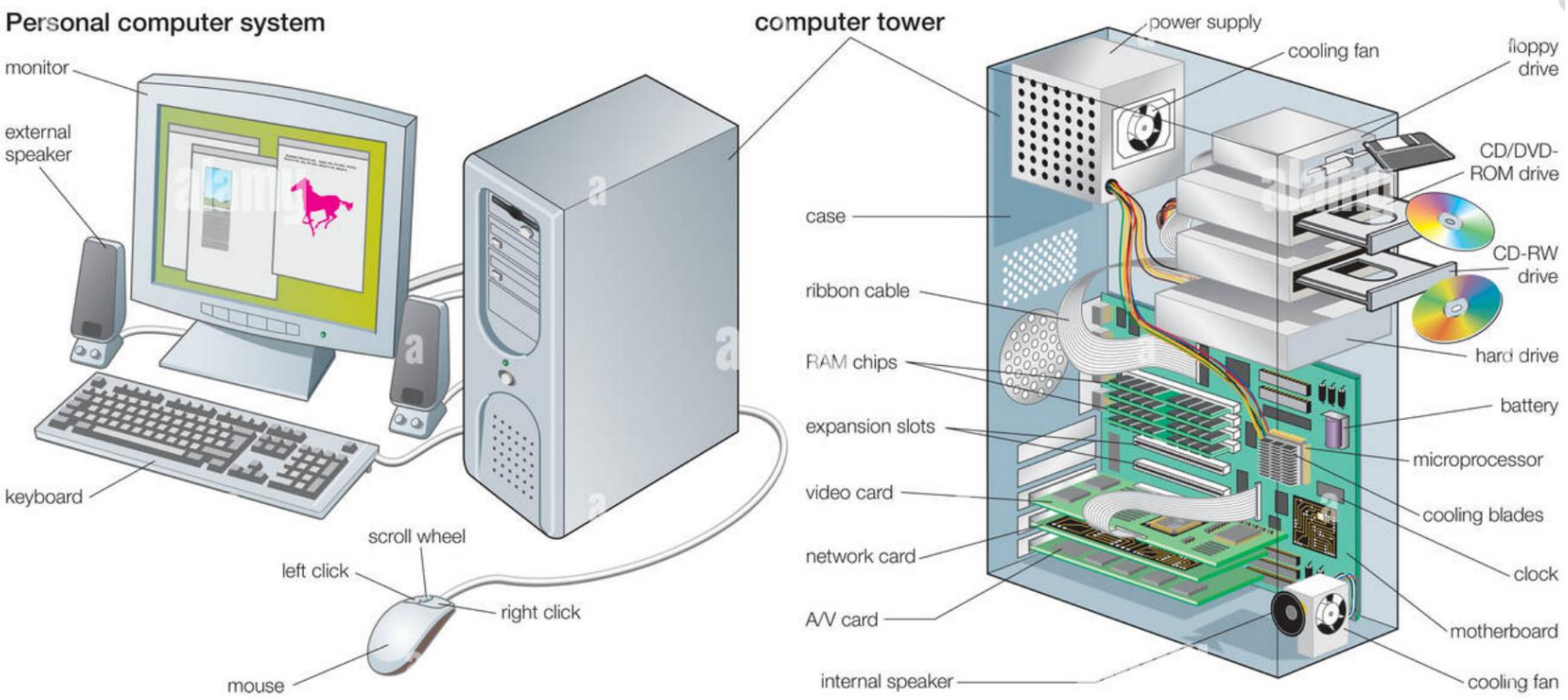


Example: Building a Car from an OOP Perspective

- Car has many components such as carburetor, starter, alternator, engine, AC system, gas tank, transmission system, etc
- Each component can be viewed as an object that has properties and methods
- To get the components to work together, you only need to know how the component is used and how it interacts with one another (abstraction) – OOP way of Thinking
- You don't need to know how each component works internally – you can build the car without knowing (encapsulation).



Example: Building a computer-system from an OOP Perspective ?



https://www.alamy.com/stock-photo-the-components-of-a-personal-computer-system-24066746.html

Class Relationships

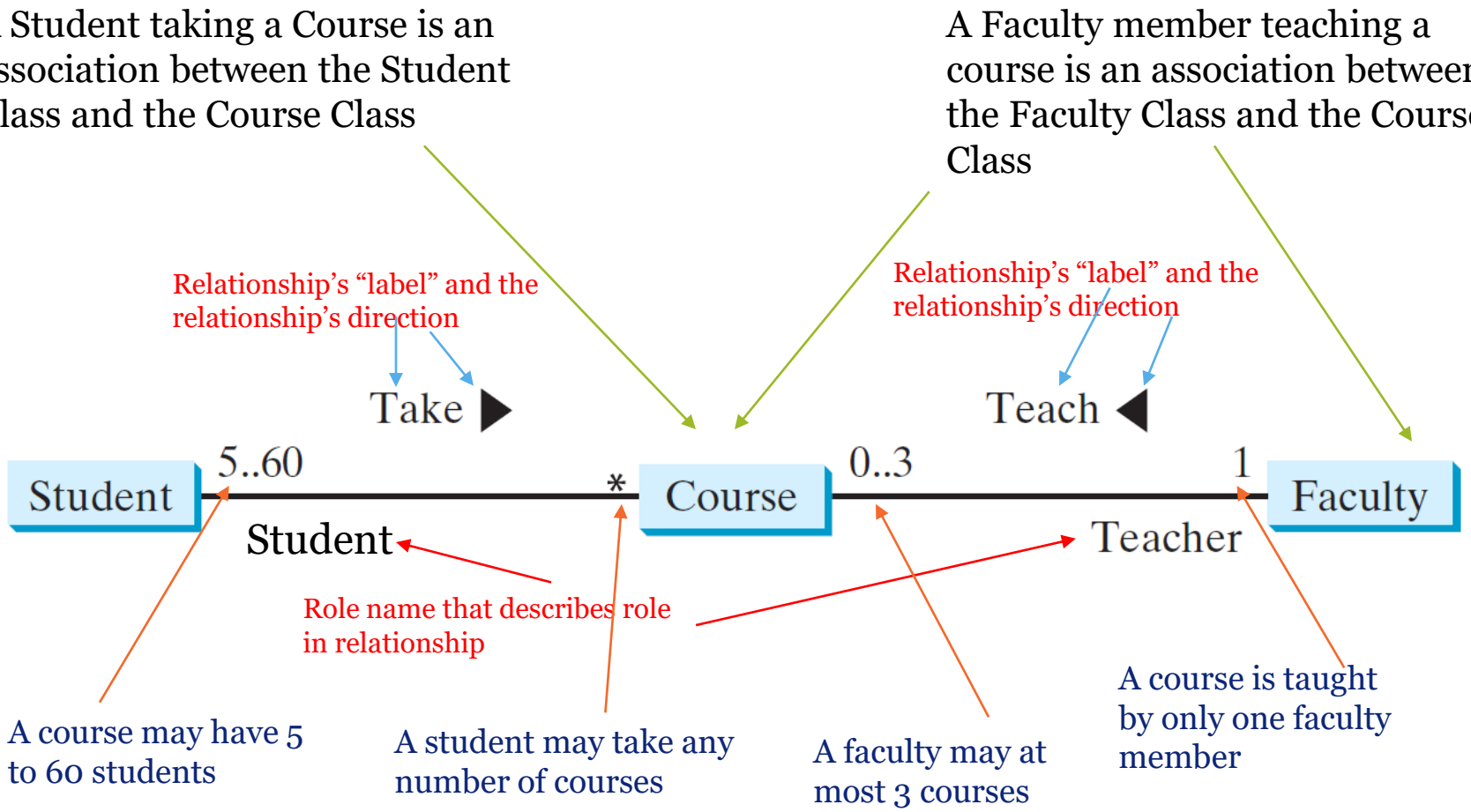
In order to design classes, you need to explore the relationships between them.

There are four common relationships between classes:

- Association : a general binary relationship that describes an activity between two classes
- Aggregation : an association that represents an ownership relationship between two objects
- Composition : an association that represents an "exclusive" ownership relationship between two objects
- Inheritance : the ability to generate a specialized class (subclass) from a general class (superclass). Will cover this in a later chapter.

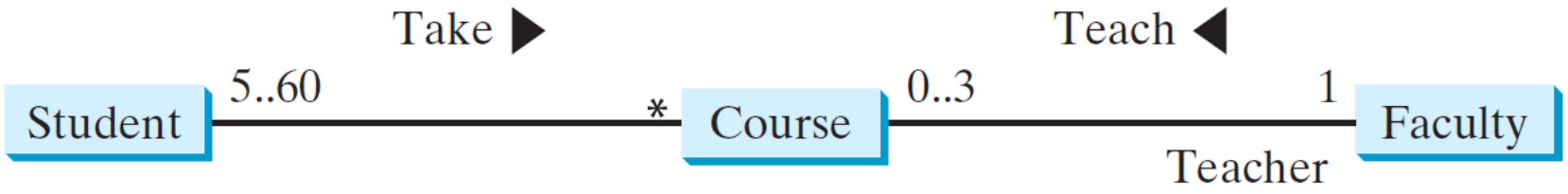
Association – UML Example

Association – a general binary relationship that describes an activity between two classes



Association – Implementation

Key Point The association relations are implemented using data field and methods in classes.



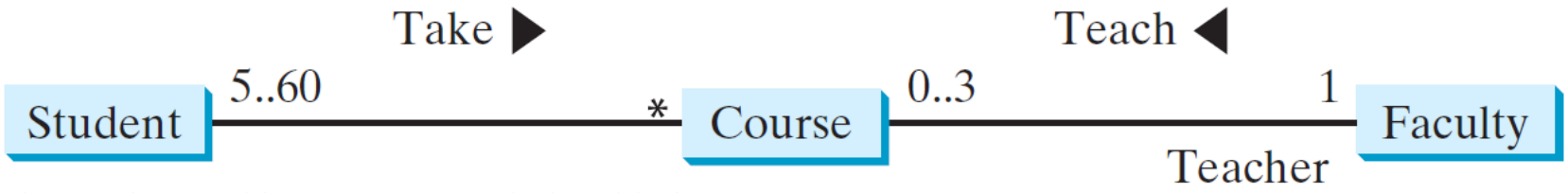

```
public class Student {
  private Course[]
    courseList;

  public void addCourse(
    Course c) { ... }
}
```

```
public class Course {
  private Student[]
    classList;
  private Faculty faculty;

  public void addStudent(
    Student s) { ... }

  public void setFaculty(
    Faculty faculty) { ... }
}
```

```
public class Faculty {
  private Course[]
    courseList;

  public void addCourse(
    Course c) { ... }
}
```

Association – Implementation

The association relations are implemented using data field and methods in classes.



The Student-Taking-A-Course relationship is implemented using addCourse method in Student class and addStudent method in the Course class

The Faculty-Teaches-A-Course relationship is implemented using addCourse method in Faculty class and setFaculty method in the Course class

```
public class Student {
  private Course[]
    courseList;

  public void addCourse(
    Course c) { ... }
}
```

```
public class Course {
  private Student[]
    classList;
  private Faculty faculty;

  public void addStudent(
    Student s) { ... }

  public void setFaculty(
    Faculty faculty) { ... }
}
```

```
public class Faculty {
  private Course[]
    courseList;

  public void addCourse(
    Course c) { ... }
}
```

The Student class can store a list of courses the student is taking

The Course class can use a list to store the students taking the course

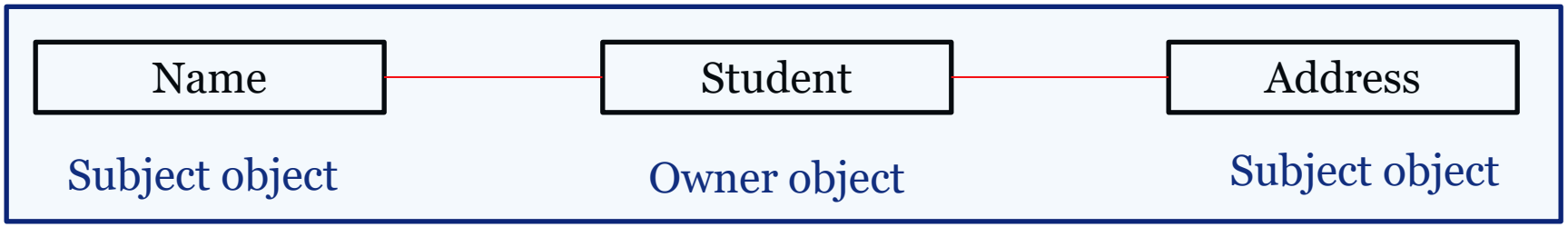
The Course class can use a data field to store the faculty who teaches the course

The Faculty class can use a list to store the courses the faculty is teaching

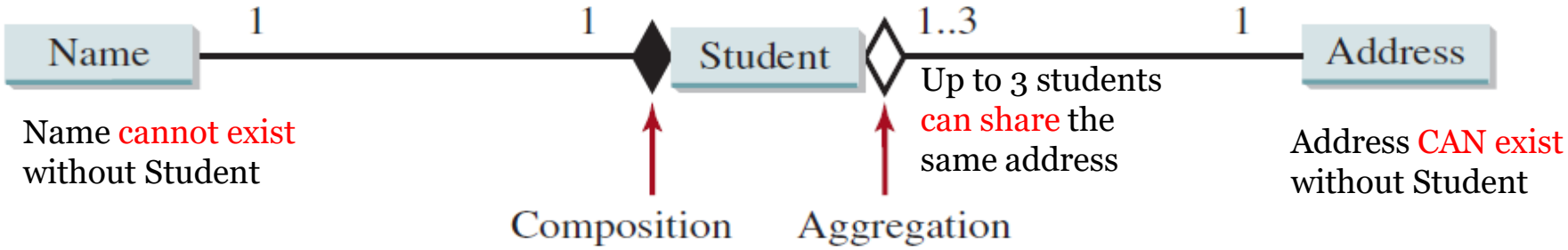
Aggregation and Composition

Aggregation – an association that represents an ownership relationship between two objects

- Owner object is called the "aggregating object" and it's class is called the "aggregating class"
- Subject object is called the "aggregated object" and it's class is called the "aggregated class"



If the subject object CAN NOT exist without the Owner object, this aggregation is called "Composition"



Aggregation and Composition - Implementation

The Student-Has-A-Name relationship is implemented using the data field "name" in the Student class – the aggregating class

The Student-Has-An-Address relationship is implemented using the data field "address" in the Student class – the aggregating class

```
public class Name {
  ...
}
```

Aggregated class

```
public class Student {
  private Name name;
  private Address address;
  ...
}
```

Aggregating class

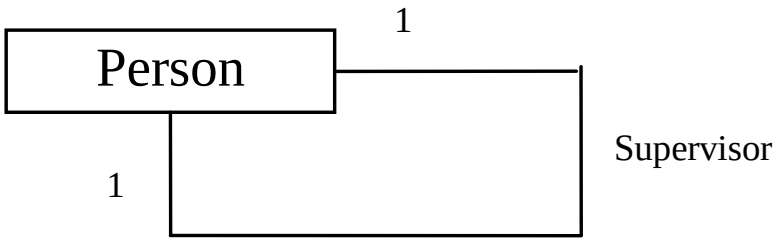
```
public class Address  {
  ...
}
```

Aggregated class

NOTE: Since aggregation and composition relationships are represented using classes in similar ways, many textbooks don't differentiate them and call both compositions.

Aggregation Between Same Class Person

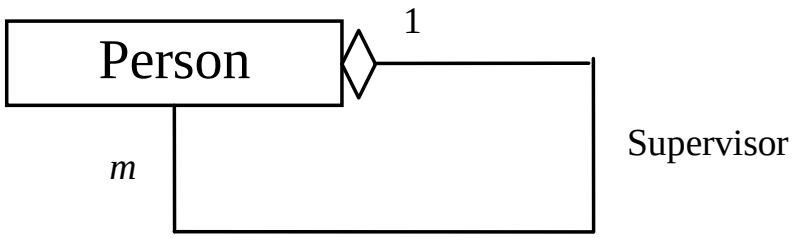
Aggregation may exist between objects of the same class. For example, a person may have a supervisor.




```
public class Person {
  // The type for the data is the class itself
  private Person supervisor;
  ...
}
```

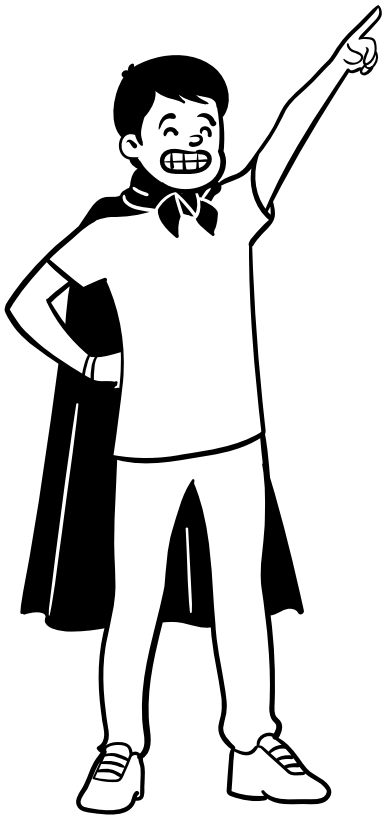
The Person-Has-A-Supervisor relationship implementation

What happens if a person has several supervisors?



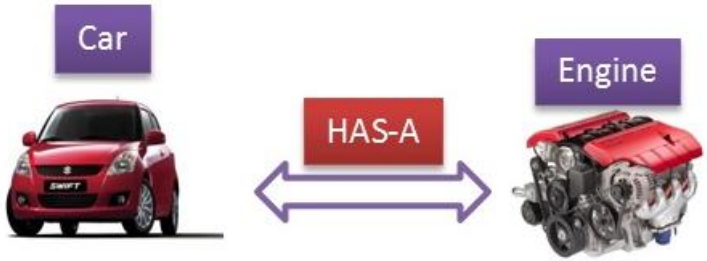

```
public class Person {
  ...
  private Person[] supervisors;
}
```

Can use an array to implement the Person-Has-Many-Supervisors relationship

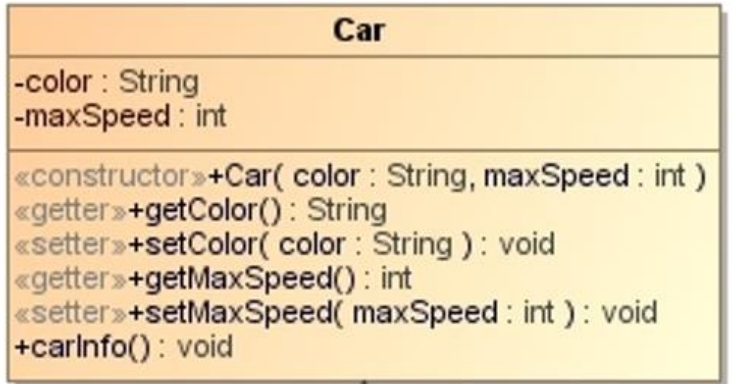


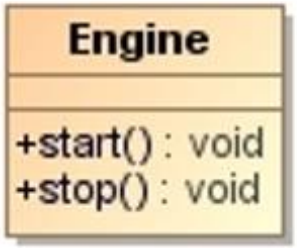
Next slides provide more examples

Examples (Car & Engine)



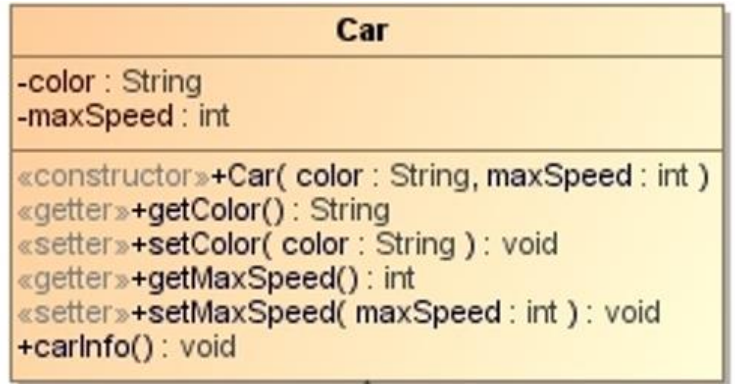
Composition (engine just for one car)

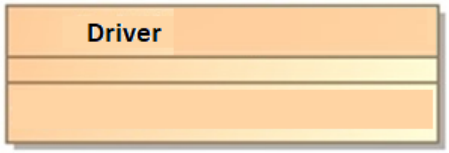




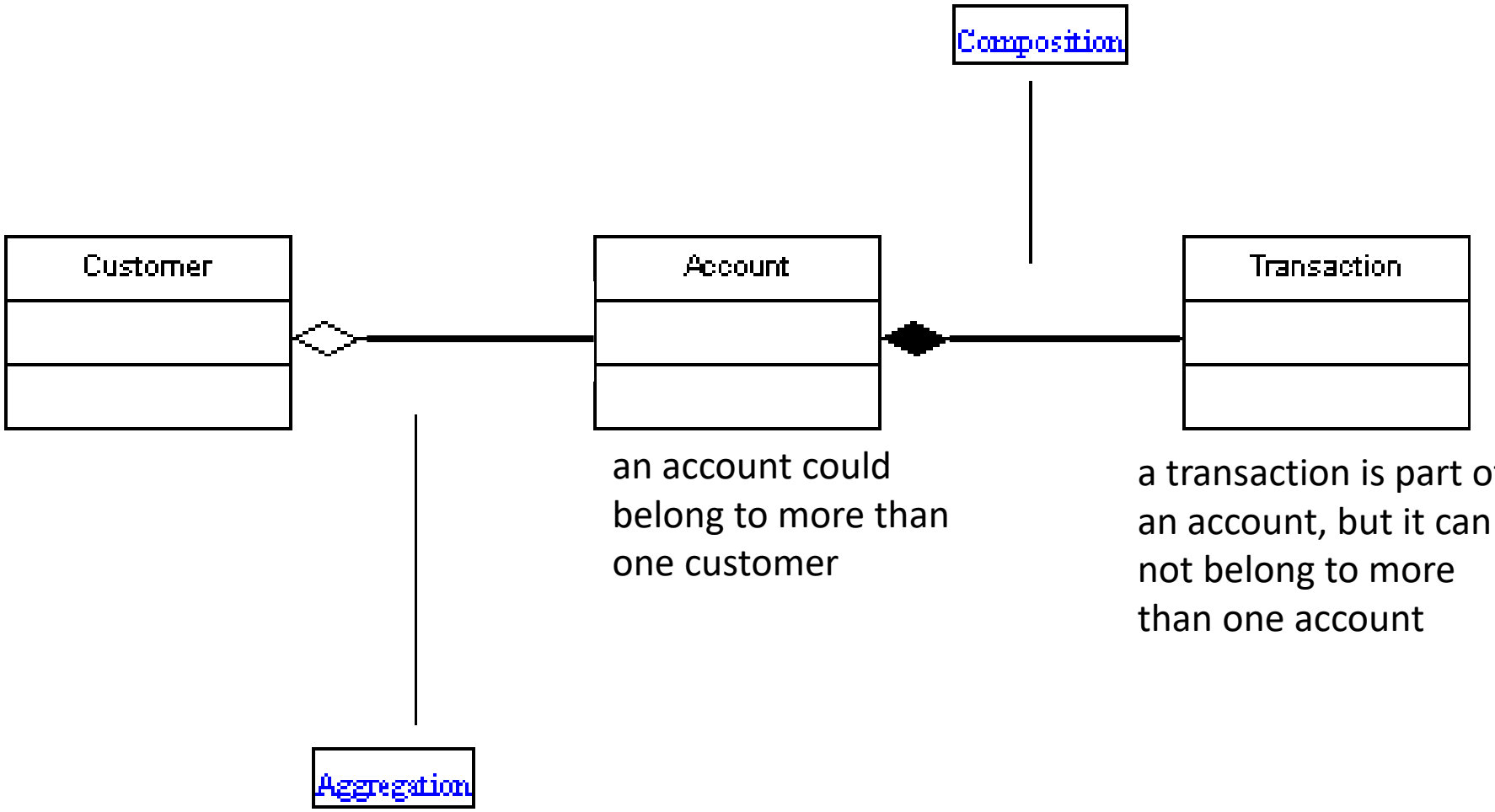
Examples (Car & Driver)

Aggregation (shared between more than one driver)





Example (Aggregation and Composition)



an account could belong to more than one customer

a transaction is part of an account, but it can not belong to more than one account

Revision of OOD, Composition, Implementation hiding, UML

BIRZEIT UNIVERSITY