UML Class Diagram

The Basics of Class Diagrams

Unified Modeling Language

- A standard notation for describing *software models and code*
- Unifies the notation of Booch, OMT (Rumbaugh et al), and OOSE (Jacobson et al)

Many Kinds of UML Diagrams

UML has 20+ different kinds of diagrams.

Each diagram shows a different kind of information (or different *view*) of application.

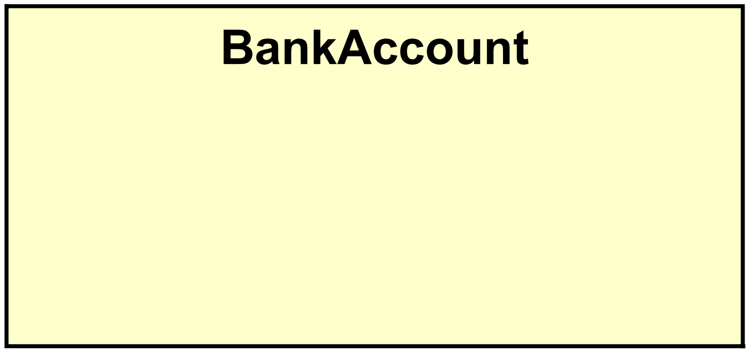
- Class diagram
- Sequence diagram
- State Machine diagram (*aka State Chart Diagram*)
- Object diagram
- Interaction diagram
- Activity diagram
- Package Diagram
- many others!

These 3 are the most common and most important to know.

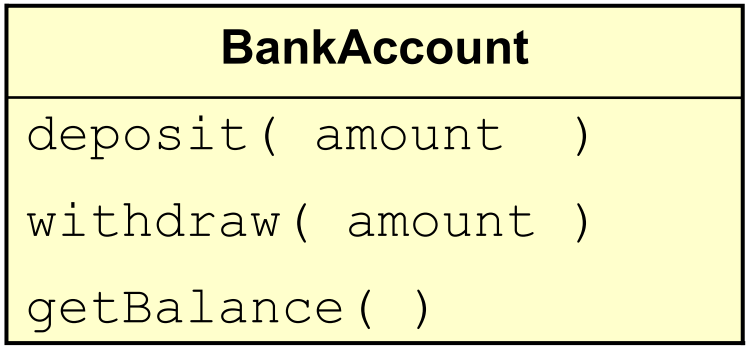
Class Diagram

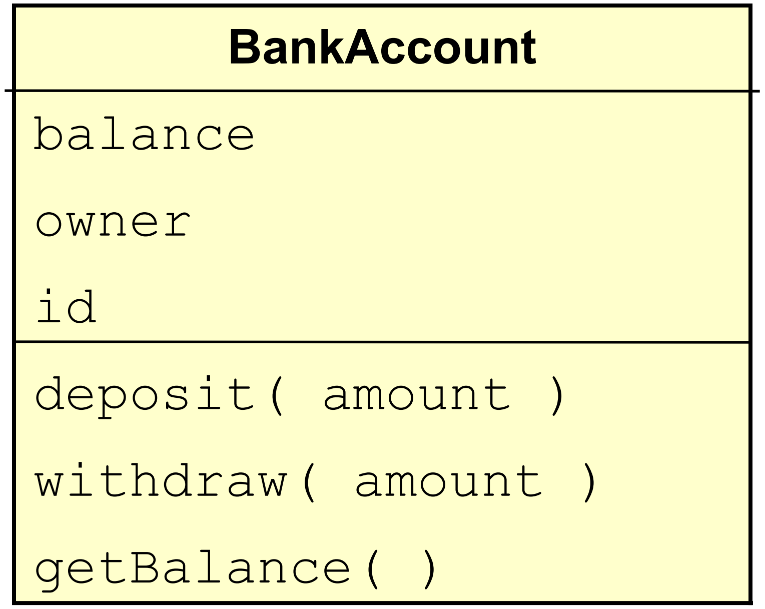
- A class diagram shows the structure of a class
- It can also show relationships between classes

Here is the *simplest possible class diagram*:



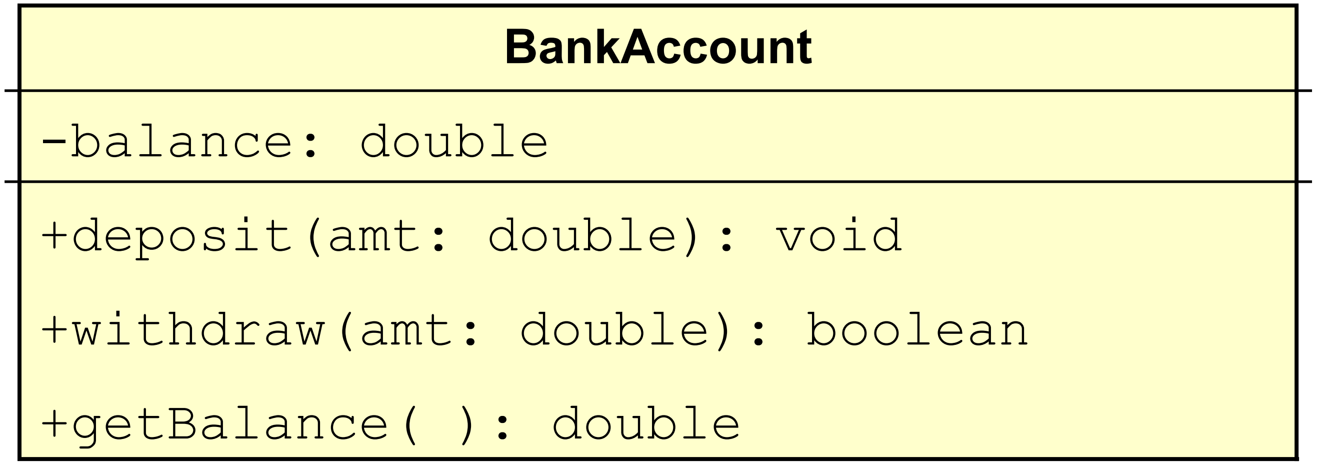
Class Diagrams methods & attributes



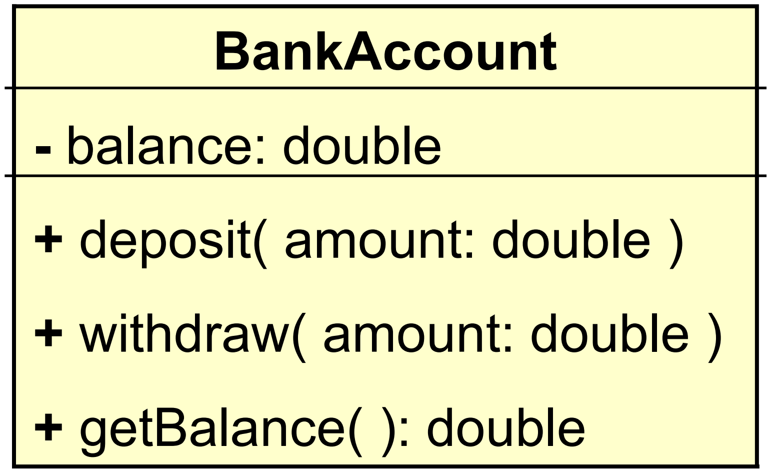


Class Diagram with data types

- Class diagram can show data types & visibility
- Not Java notation ("double balance")



Visibility of Members



- balance is private (visible only within **BankAccount** objects)
- deposit, withdraw, getBalance are public

Visibility Prefixes

+ means **public**

- Visible everywhere

– means **private**

- Visible only in the class in which it is defined

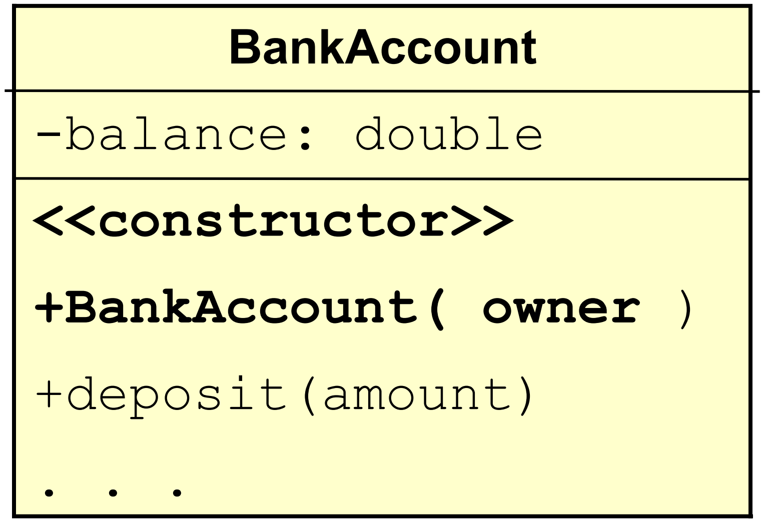
# means **protected**

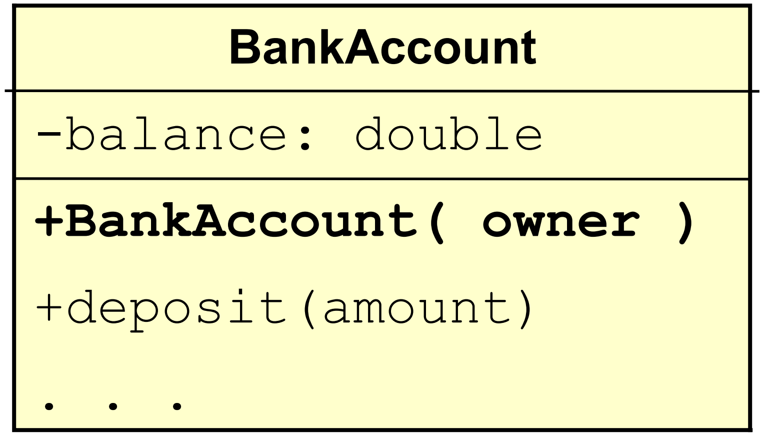
- Visible either within the class in which it is defined or within subclasses of that class

~ means **package** visibility

- visible to other classes in the same package

Constructors

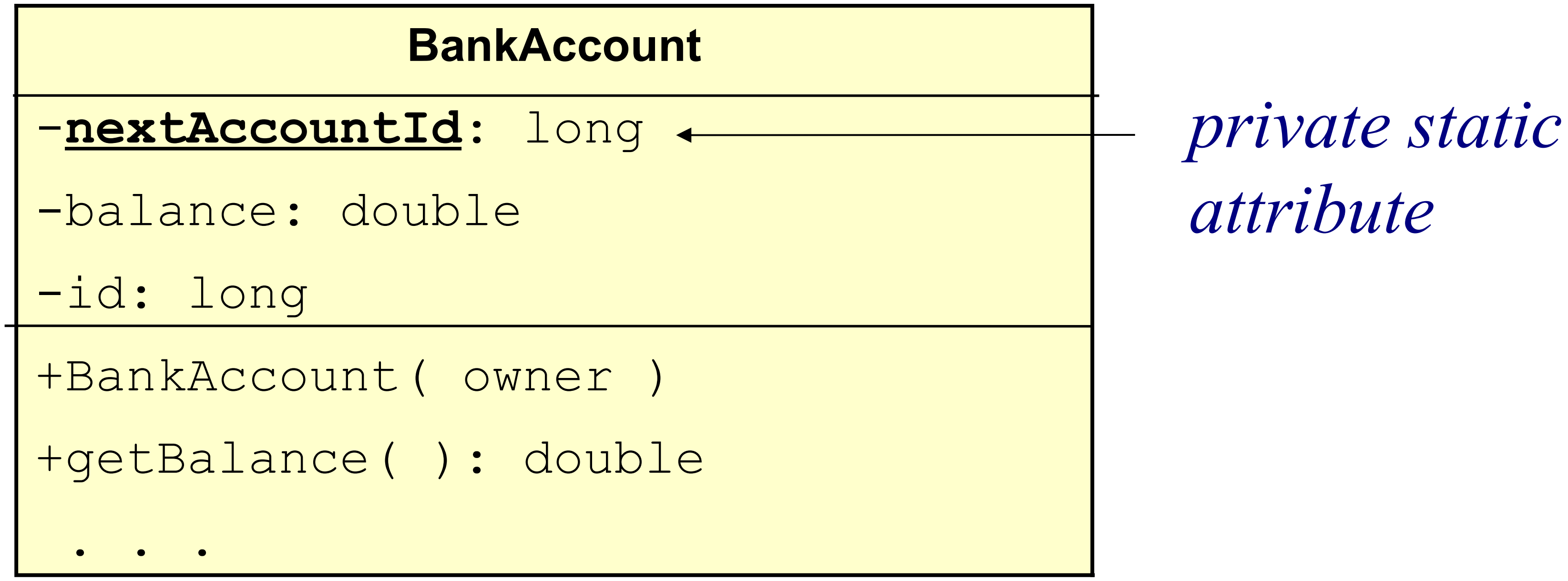




Static Members

- Use underscore to show static attributes or methods.

Example: BankAccount class has a static **`nextAccountId`** attribute.

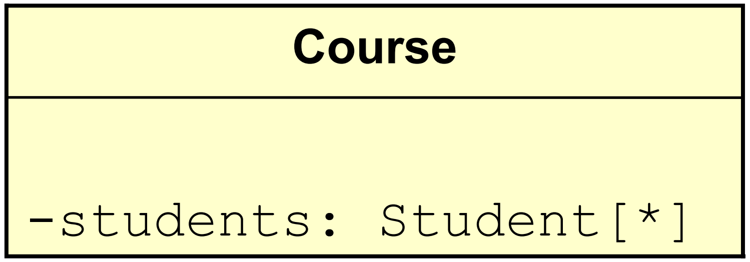


Practice: Draw UML of class

Showing Multiplicity in UML

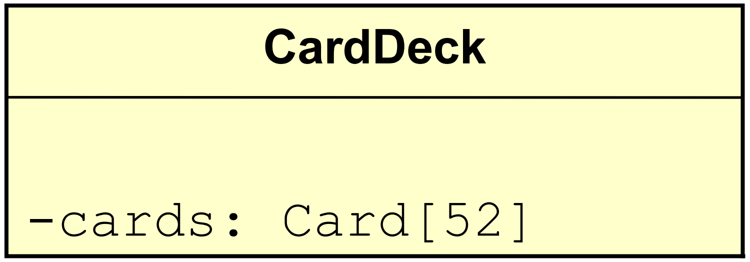
A Course has zero or more students.

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



A deck of cards has *exactly* 52 cards.

```
public class CardDeck {
   private Card[] cards =
         new Card[52];
```



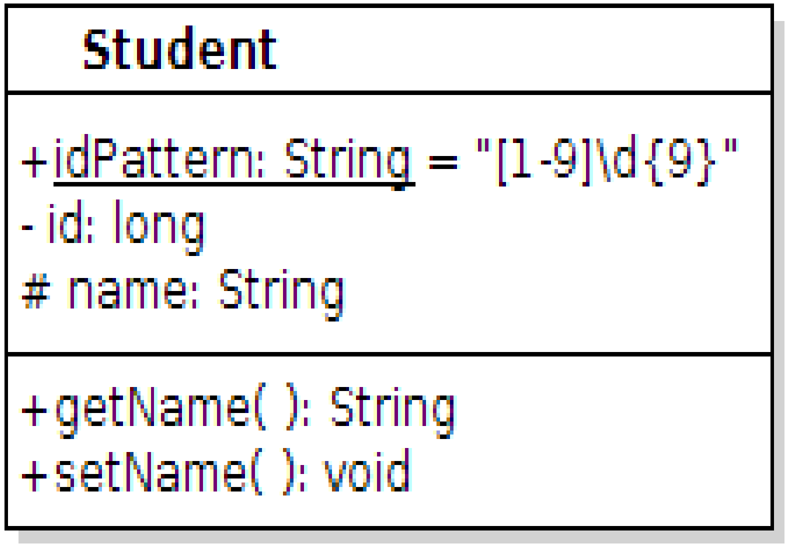
A Single Class

Draw a UML class diagram of this class.

```
public class Student {
    public static String idPattern = "[1-9]\\d{9}";
    private long id;
    protected String name;



    public String getName( ) { . . . }


    public void setName(String aname) { . . . }

```

A Single Class

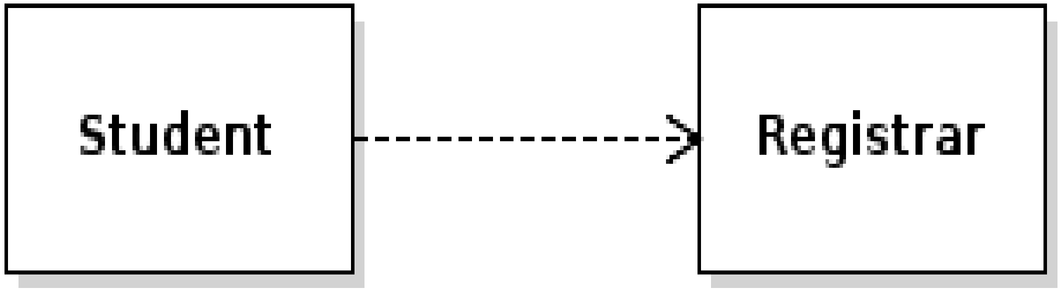
Draw a UML class diagram of this class.



Class with Dependency

A Student *uses* the Registrar to get his Courses, but he doesn't save a reference to it.

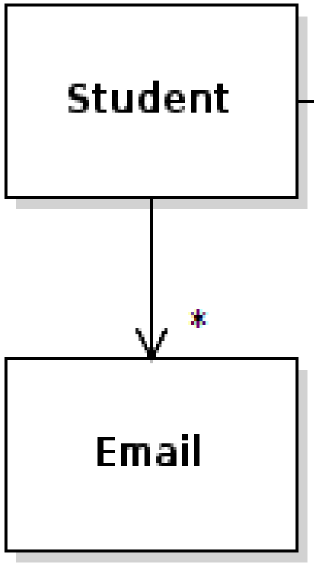
```
public class Student {
    private long id;
    //NO Registrar attribute!

    public void addCourse(Course course) {
        Registrar regis = Registrar.getInstance();
        regis.enroll(this, course);
```



Class with Association

A Student *has* an Address and 0 or more Emails.

```
public class Student {
    private Address homeAddress;
    /** his email addresses. He may have many. */
    private List<Email> emails;
```



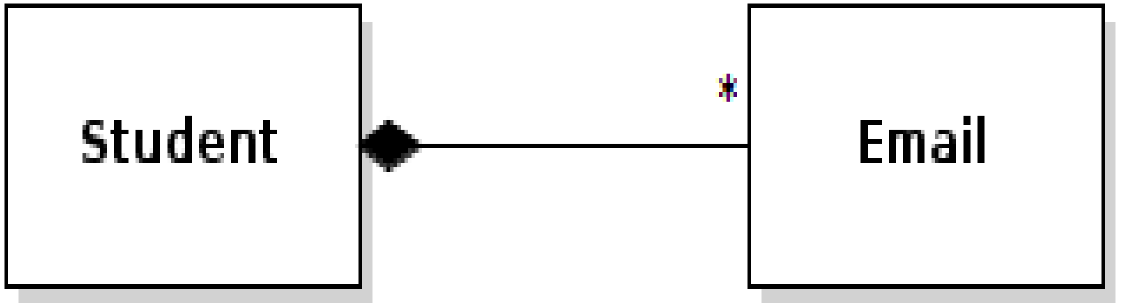
A Student owns his Email Addresses

Composition: A Student *owns* his Email addresses and when he is deleted we delete his addresses, too!

```
public class Student {
    /** student uniquely owns his email addresses*/
    private List<Email> emails;
```

Modeling:

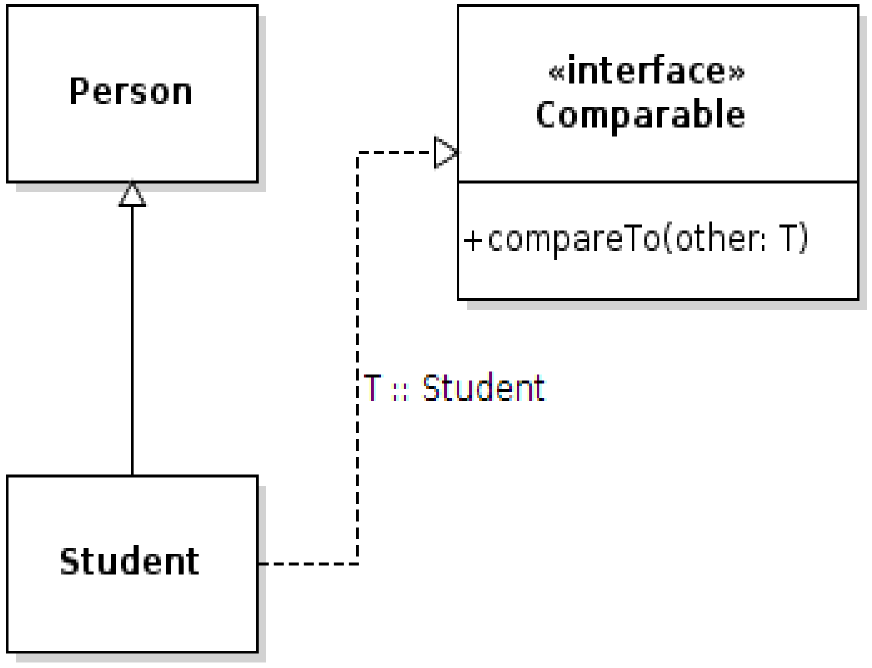
Composition shows "*ownership*" or "*is composed of*" (e.g.: a game board is composed of squares). Be *careful* about using it.



Inheritance & Implements

Student is a subclass of Person

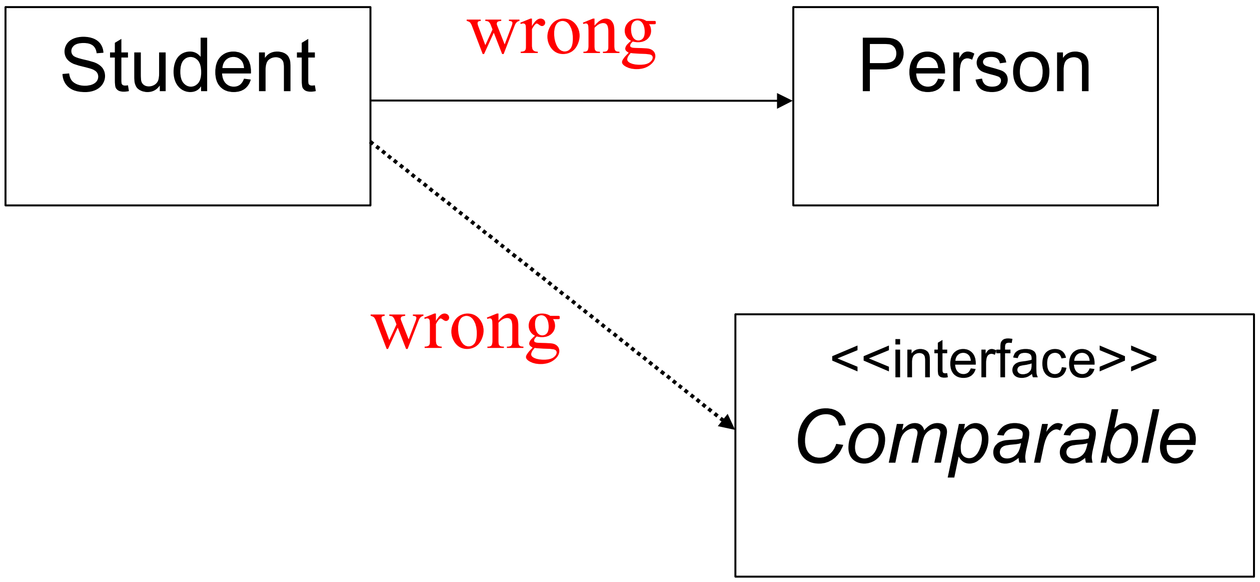
```
public class Student extends Person
        implements Comparable<Student> {
```



Errors

A UML diagram is for communication.

To communicate clearly, use the correct notation.



No partial credit for wrong relationships or bad notation.

Reference

UML Distilled, 3rd Edition. Chapter 3 & 5 cover UML class diagrams.