Software Engineering Essentials

Basics of Object Oriented Programming #1

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics

Learning goals

1) Use the Eclipse IDE to create new Java projects and run simple programs
2) Understand the object oriented programming principles encapsulation and inheritance
3) Apply these two principles

Assumptions about your existing knowledge

Experiences with an imperative or object-oriented programming language:

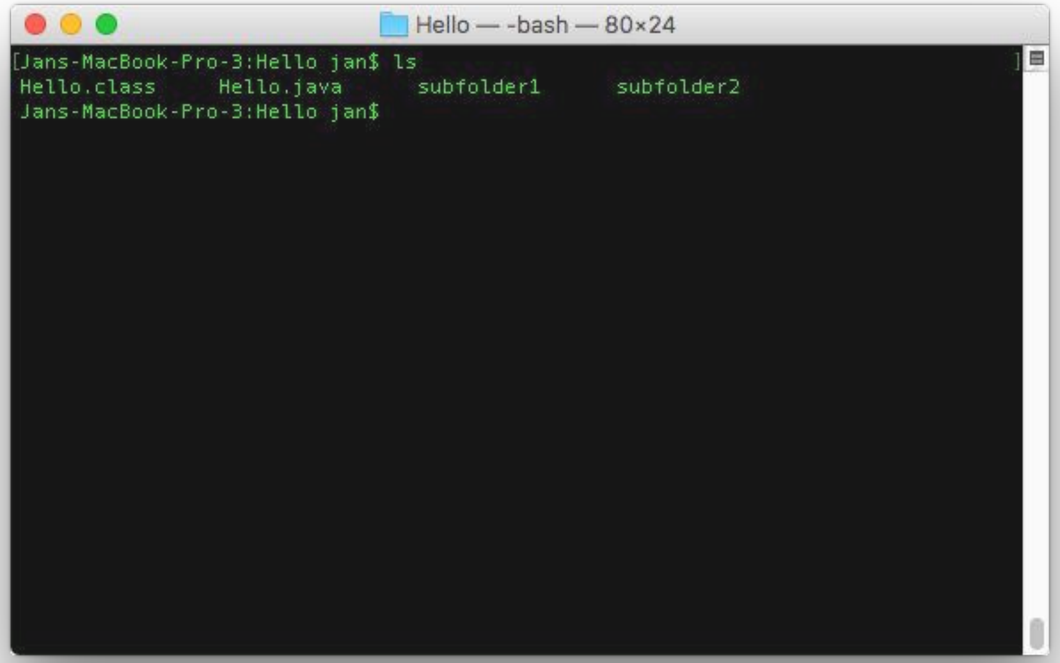
- Java
- C++
- C#
- Pascal
- C

Ability to use or adapt to the Java syntax

Why do we need an integrated development environment (IDE)?

Compile multiple source code files in different folders?

(with Maven/Ant)



Syntax highlighting?

(with Vim plugins)

Why do we need object oriented programming?

Divide and conquer (latin: dīvide et īmpera):

Described as a political instrument by Traiano Boccalini [TB1678]

Also derived from Sun Tzu in “The Art of War” (Sun Wu):

It is the rule in war, if ten times the enemy's strength, surround them; if five times, attack them; if double, be able to divide them; if equal, engage them; if fewer, defend against them; if weaker, be able to avoid them. [AOW]

[AOW] The Art of War, AD760 [TB1678] Traiano Boccalini. La Bilancia Politica Di Tutte Le Opere, 1678

Why do we need object oriented programming?

"During system design, developers define the design goals of the project and decompose the system into smaller subsystems that can be realized by individual teams." [BD09]

The object oriented programming paradigm supports four major principles:

- Encapsulation
- Inheritance

} OOP #1

- Polymorphism
- Abstraction

} OOP #2

[BD09] Bernd Bruegge and Allen Dutoit. Object Oriented Software Engineering Using UML, Patterns, and Java. Prentice Hall, 3rd edition, 2009

Object oriented programming - encapsulation

Encapsulation means creating classes to define:

- **Structure** by using attributes
- **Functionality** by providing methods/procedures

Java supports encapsulation by using ***classes*** with ***attributes*** for structuring and ***methods*** for describing functionality

Student
+majorSubject:String +minorSubject:String +courseList:List<Course>
+joinCourse(c:Course):void +dropCourse(c:Course):void

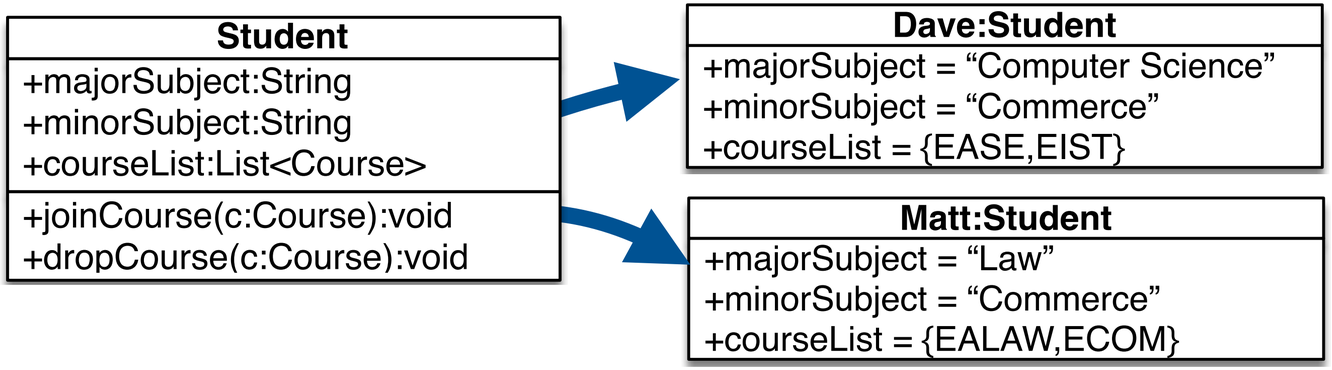
```
public class Student {

    public String majorSubject;
    public String minorSubject;
    public List<Course> courseList;

    public void joinCourse(Course c) {
    }

    public void dropCourse(Course c) {
    }
}
```

Object oriented programming - encapsulation

Java uses a **constructor** creates a concrete ***instance/object*** of a ***class***

A constructor is a special method which:

- Does not have a return type
- Carries the name of the class
- Instantiates specified attributes of an object



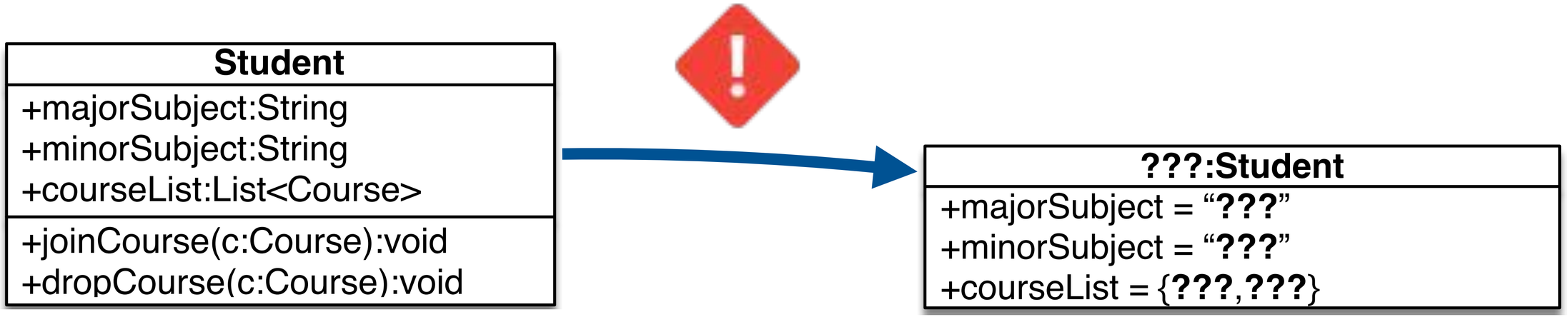

```
public Student(String majorSubject, String minorSubject) {
        this.majorSubject = majorSubject;
        this.minorSubject = minorSubject;
        this.courseList = new LinkedList<Course>();
    }
```

Object oriented programming - encapsulation

Each ***class*** has a ***default constructor*** (implicit)

```
public Student() {

}
```

Custom ***constructor***s can be used to initialize attributes properly



Object oriented programming - inheritance

Allows to reuse own data structures by using a inheritance hierarchy:

- Reuse of **structure** by accessing attributes of a ***super-class***
- Reuse of **functionality** by accessing methods/procedures of a ***super-class***

Real world example:

- *Each car is a vehicle*
- *Each motorbike is a vehicle*

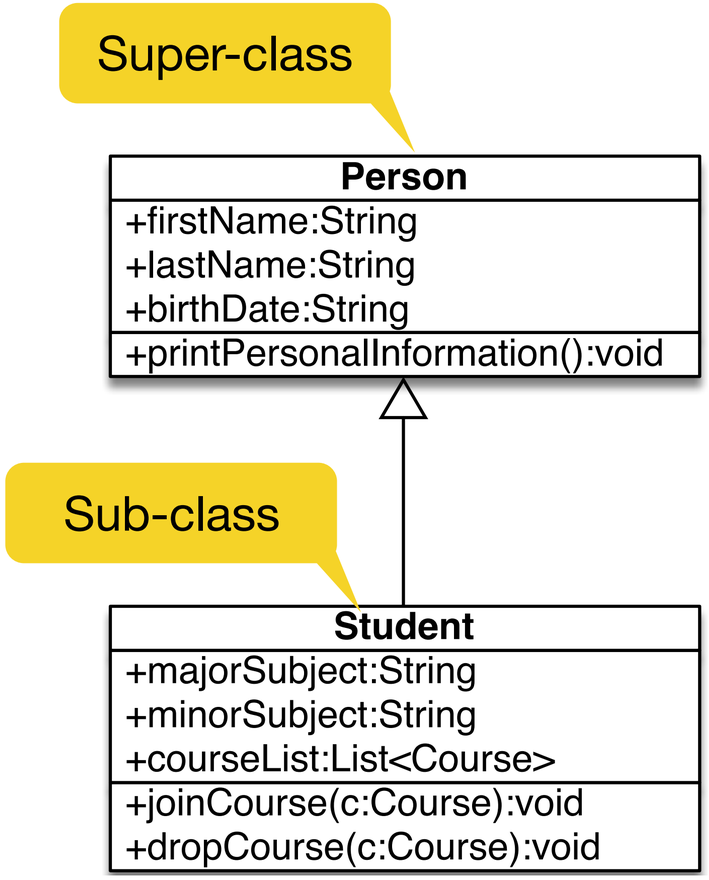
➡ *Both share **structural** information e.g. wheels*

➡ *Both share **functionality** e.g. move()*

➡ But both can have additional attributes and functionality

Object oriented programming - inheritance

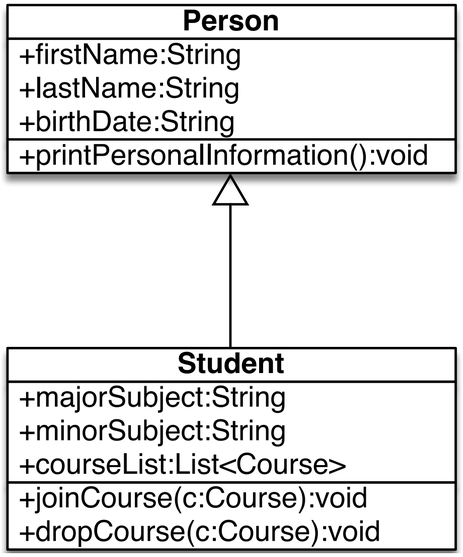
- Java supports inheritance by defining ***sub-classes*** and an association to a ***super-class***
- Use the keyword ***extends*** to establish an inheritance hierarchy




```java
public class Person {
      public String firstName;
      public String lastName;
      public String birthDate;

      public void printPersonalInformation() {
            System.out.println(firstName + " "+lastName +" " + birthDate);
      }
}
public class Student extends Person{

      public String majorSubject;
      public String minorSubject;
      public List<Course> courseList;
      //…
}
```

Object oriented programming - inheritance

- To reuse attributes from the **super-class**, a sub-class delegates the initialization to the ***super-class constructor***
- Call ***super()*** within the **sub-class** and pass values from the sub-class to the super-class




```
public class Person{
public Person(String firstName, String lastName, String birthDate) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.birthDate = birthDate;
}
}
```

set values

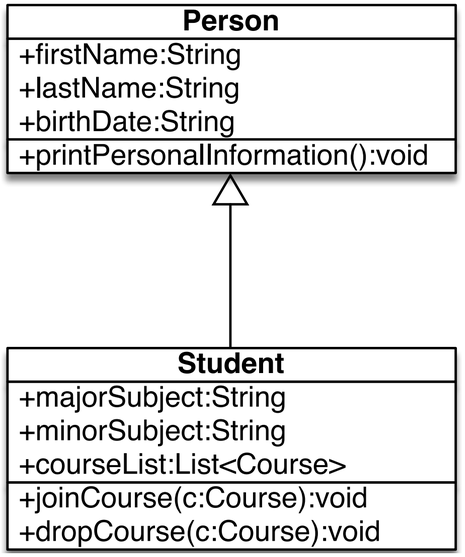
```
public class Student extends Person{
public Student(String firstName, String lastName, Date birthDate,
               String majorSubject, String minorSubject) {
    super(firstName, lastName, birthDate);
    this.majorSubject = majorSubject;
    this.minorSubject = minorSubject;
    this.courseList = new LinkedList<Course>();
}
}
```

set additional values

pass values to Person constructor

Object oriented programming - inheritance

- Sub-classes can access (non-private) attributes and methods of super classes
- If sub-classes do not declare a method, the Java compiler automatically searches in the super class hierarchy




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

    Student s = new Student("Bob", "Davis", "12/20/1990", "Computer Sc.", "Commerce");
     Person p = new Person("John", "Lewis", "1/23/1960");

    System.out.println(s.firstName); //prints the firstName of the Student instance
    System.out.println(p.firstName); //prints the firstName of the Person instance

    s.printPersonalInformation(); //inherits this method from Person to Student, so it can
                                  //be called even if not declared in Student itself

    p.printPersonalInformation(); //uses this method declared in Person

}
```

Object oriented programming - inheritance

Be aware that an inheritance relation only holds ***in one*** direction:

Every student is a person

But

Not ***every*** person is a student

Therefore only assignments from a more precise/specialized type to a more generic type are allowed

Summary

We compared command line interfaces vs IDE

We installed an IDE (Eclipse)

We covered two out of four major object oriented programming principles:

- Encapsulation
- Inheritance

} OOP #1 ☑

- Polymorphism
- Abstraction

} OOP #2 ☒

Software Engineering Essentials

Basics of Object Oriented Programming #1

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics