Encapsulation

Tobias Hanf, Maik Göken

December 19, 2022

Learn Programming with Java

Outline

Revision

Small Quiz

https://pingo.coactum.de/186364

4 Pillars of OOP

4 Pillars of OOP

- Generalisation
  - Class hierachie
  - Unit 07

4 Pillars of OOP

- Generalisation
    - Class hierachie
    - Unit 07
- Inheritance
    - Of attributes and methods
    - Unit 07

4 Pillars of OOP

- Generalisation
    - Class hierachie
    - Unit 07
- Inheritance
    - Of attributes and methods
    - Unit 07
- Encapsulation
    - data hiding
    - today

4 Pillars of OOP

- Generalisation
    - Class hierachie
    - Unit 07
- Inheritance
    - Of attributes and methods
    - Unit 07
- Encapsulation
    - data hiding
    - today
- Polymorphisim
    - Single Interface, Multiple functionality
    - Unit 07

Encapsulation

Encapsulation

- Data hiding
    - Restrict access
    - Allow only certain opertations
    - Check input
    - Every attribute should be hidden
- Code hiding
    - Hide implementation
    - only interact with ”interface”

Example Student

- Entered marks shouldn’t change
    - only allow appending of marks

Example Student

- Entered marks shouldn’t change
    - only allow appending of marks
- Not everybody should know the date of birth
    - Only age is needed (most of the time)

Example Student

- Entered marks shouldn’t change
    - only allow appending of marks
- Not everybody should know the date of birth
    - Only age is needed (most of the time)
- Enrollment number should never change

Example Student

- Entered marks shouldn’t change
  - only allow appending of marks
- Not everybody should know the date of birth
  - Only age is needed (most of the time)
- Enrollment number should never change
- Degree should only take certain values
  - must be a degree currently offered

Example Student

- Entered marks shouldn’t change
    - only allow appending of marks
- Not everybody should know the date of birth
    - Only age is needed (most of the time)
- Enrollment number should never change
- Degree should only take certain values
    - must be a degree currently offered
- How a mark is added shouldn’t be of anybodies concern
    - except of the class

Encapsulation in Java

Access modifiers

Access modifieres define who can access a member (attribute or method) of a class.

There are three access modifiers in Java:

Access modifiers

Access modifieres define who can access a member (attribute or method) of a class.

There are three access modifiers in Java:

- `public`
  - Anyone can access the member
  - Default (if no other modifiers is specified)
  - Almost all methods are `public`

Access modifiers

Access modifieres define who can access a member (attribute or method) of a class.

There are three access modifiers in Java:

- `public`
  - Anyone can access the member
  - Default (if no other modifiers is specified)
  - Almost all methods are `public`
- `private`
  - Only the *own class*(object) can access the member
  - Every attribute should be `private`

Access modifiers

Access modifieres define who can access a member (attribute or method) of a class.

There are three access modifiers in Java:

- `public`
  - Anyone can access the member
  - Default (if no other modifiers is specified)
  - Almost all methods are `public`
- `private`
  - Only the *own class*(object) can access the member
  - Every attribute should be `private`
- `protected`
  - The own class
  - And all subclasses can access the member

Syntax

```
class Syntax {
    // Attribute
    <modifier> <type> <name>;

    // Method
    <modifier> <ret-type> <name> (...) {
    ...
    }
}
```

Student example

```java
class Student {
    private String name;
    private int yearOfBirth;
    ...

    public int getAge() {
    ...
    }

    public boolean addMark(float mark) {
    ...
    }
}
```

The problem with encapsulation

What is the problem?

What Problem arises when we declare all attributes `private`?

(You can use the class `Student` as an exmaple.)

The Problem

We cannot set and get the attributes (like `name`,...) anymore.

The solution

Introduce methods for getting and setting attributes.

Getter, Setter and Constructor

Contructor

The constructor is a special method of a class.

Like the name implies the constructor constructs an Object.

- Has the same name as the class
- Will get called if a new object is constructed
- Mostly used for initializing attributes

Constructor in Java

```
class  <class-name> {
    public <class-name>(...) {
        ...
    }
    ...
 }
```

For the **`Student`** class

```
class Student {
    public Student(String name, ...) {
    ...
    }
}
```

Getter and Setter

Normal methods, no special meaning in Java.

Can be as simple as just returning and assigning the attributes.

But allows implementation of complex logic whithout changing the interface.

Getter and Setter in Java

Function name convetion:

- Getter methods
  - Start with get
  - Then the name of the attribute
    - eg. `getName()`
  - No paramters (most of the time)
- Setter method
  - Start with set
  - Then the name of the attribute
    - eg. `setName(String name)`
  - Parameter should have the same name as attribute

The `this` keyword

Sometimes we want to have a function parameter with the same name as an attribute.

We couldn't access the attribute in this case.

To solve the problem Java has the `this` keyword.

It is a reference to the current object and can only be used inside the class.

```java
this.<attribute>;
this.<method>();
```

Exercise

Apply the concept of encapsulationg

Take the University Resource Planning program from the last unit and apply the concept of encapsulation to the `Student` class.