Programming in JAVA

lecture 6

Object oriented programming

Object oriented programming

1. In object oriented programming we create data structures that integrate data with associated functions.
2. These data structures are called **classes**.
3. When we want to use classes, we create **objects**.
4. An object reflects what is defined by its design (in its class).

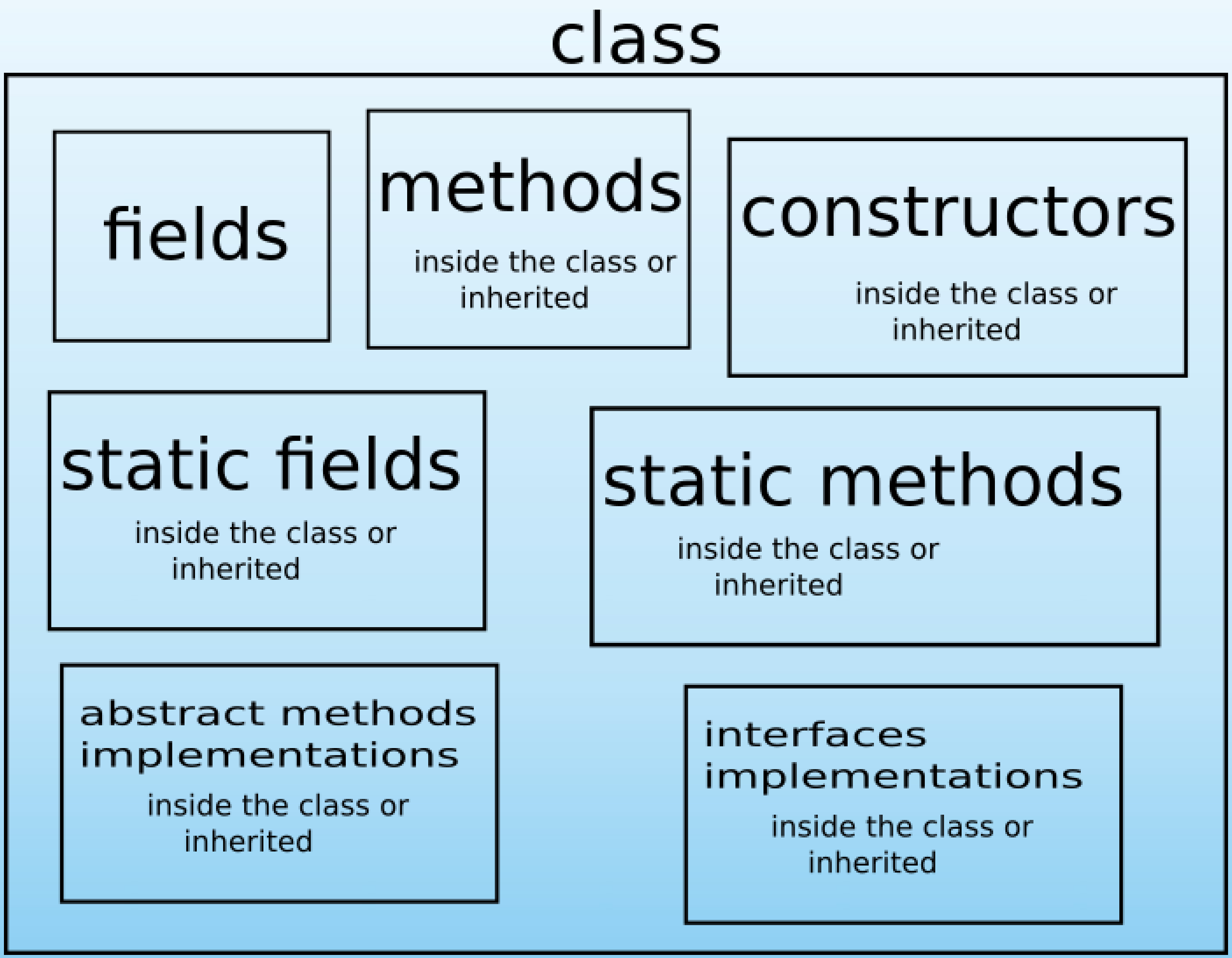
Object oriented programming

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods).

source: *wikipedia.org*

What is inside a class ?

What can be inside a class ?


class
fields
methods
inside the class or inherited
constructors
inside the class or inherited
static fields
inside the class or inherited
static methods
inside the class or inherited
abstract methods implementations
inside the class or inherited
interfaces implementations
inside the class or inherited

Fields in class

1. Fields are literally variables inside a class.

2. They are supposed to be preceded by a declaration of access.

3. Declarations of access to choose are: `public`, `protected`, `private`.

`public`, - a field can be used from outside of the class

`private` – no permition to access fields from outside of the class

`protected` – similar to private but fields can be used in descendant class

Example of fields declaration

```
public class Test
{

    protected Integer number;
    public    String str;
}
```

Methods

1. Methods are literally functions that are defined inside a class.
2. They are supposed to be preceded by a declaration of access.
3. Declarations of access to choose are: `public`, `protected`, `private`.
4. Methods should be linked to the functionality of the class and should operate on some of the class fields.

Example of methods declarations

```
public class Test
{
    protected Integer number;
    public String str;

    public boolean equals(Integer _number,
                          String _str)
    {
      return number.equals(_number)&&str.equals(_str);
    }

    public boolean equals(Test t)
    {
      return t.number.equals(liczba)&&t.str.equals(str);
    }
}
```

Constructors

1. Constructor is a special function, that is called when an object of the class is created (when the "**new**" operator is used).
2. They are often used to fill fields with some starting data or perform some starting operations.
3. Constructors are usually public.
4. A class can contain several constructors.
5. In the source code, a constructor can be easly identified as it has the same name as the name of the class and do not have the type of returned value.
6. If not defined, a default constructor is created automatically.

Constructors - example

```
public class Test
{
    public Integer number;
    public String str;

    public Test(Integer _number, String _str)
    {
        number = _number;
        str = str;
    }
    public Test(Test t)
    {
        number = t.number;
        str = t.str;
    }
}
```

The pointer **this**

this – is a pointer that can be used as a

reference to the object the code is tunning in.

Example of **this** pointer in constructors

```
public class Test
{

  public Integer number;
  public String str;

  public Test(Integer number, String str)
  {
    this.number = number;
    this.str = str;
  }

  public boolean equals(Integer number, String str)
  {
   return this.number.equals(number)&&
                                    this.str.equals(str);
  }
}
```

Static methods

- We can use static method even before an object is initializated
- A static method is called by providing the name of the class, not the name of the object.
- Static methods can be used to program in procedural manner.

Static methods - example

```
public class Math_Functions
{
  public static double pow(double number ,
                                int power)
  {
        double result = number;
        for(int i=0; i<(power-1) ; i++)
            result = result*number;

        return result;
  }
}
```

Inheritance

1. A class can inherit functionality from another class.
2. The keyword **extends** is used inside the code to indicate inheritance.
3. In Java a class can inherit from only one class.

Example of inheritance specification

```
public class NewJFrame extends javax.swing.JFrame
{
                    .....
```

The super keyword

super – is a reference to the constructor from parent class