DeepL
Subscribe to DeepL Pro to translate larger documents.
Visit www.DeepL.com/pro for more information.

Object-oriented programming

1. Lecture

LS 2025/2026

Author: Juraj Petrík

What will we (hopefully) be doing here?

- OOP and Java
- Software development in general and in the context of OOP
- Testing
- Dependency management
- Debugging
- Collections
- Generics
- Patterns
- GUI
- Annotations
- Documentation
- Logging
- Parallelism
- IO
- Persistence
- Reflection
- Metrics
- Other OO languages
- Invited lectures

Why another paradigm?

- To bring it closer to human thinking in the real world

- E.g. Java, Python, Scala, C++

- Beware of paradigm shift (training and practice required)

Procedural vs. object-oriented

```c
#include <stdio.h>
#include <string.h>

struct Person { char
    name[50]; int
    age;
};

void greet(struct Person p) {
    printf("Hello, %s! You are %d years old.\n",
        p.name, p.age);
}

int main() {
    struct Person person1; strcpy(person1.name,
    "Alice"); person1.age = 25;

    greet(person1); return
    0;
}
```

```java
class Person {
    private String name; private
    int age;

    public Person(String name, int age) {
        this.name = name;
        this.age  =  age;
    }

    public void greet() {
        System.out.println("Hello, " + name + "! You are " +
                age + " years old.");
    }
}

public class Main {
    public static void main(String[] args) { Person
        person1 = new Person("Alice", 25);
        person1.greet();
    }
}
```

What we will need

- IDE (Intellij IDEA, Eclipse for JAVA, Netbeans)
  - Visual Studio Code
  - Notepad
- JDK 21 (LTS)
- Pre-installed in classrooms/labs

Why object-oriented?

- Parallel to the real world
  - We perceive "things" as objects
  - The same type of object (class) can occur multiple times in the world (object).
- Objects
  - They have certain properties (attributes)
  - We can do something with them (methods)

Java

- Lots of materials
- Ecosystem
- Strongly typed language
- “Write once, run everywhere” – Google “Saying that Java is nice because it works on all OS's is like saying”

- In the end, it doesn't really matter ☺

How does Java work?

- JVM can optimize code during runtime
- Yes, Java applications can allocate a lot of memory
- Speed (very generalized): C++ > Java > Python

What are the minimum requirements for a programming language to be usable?

- "Memory"
- Branching
- Looping

Hello world

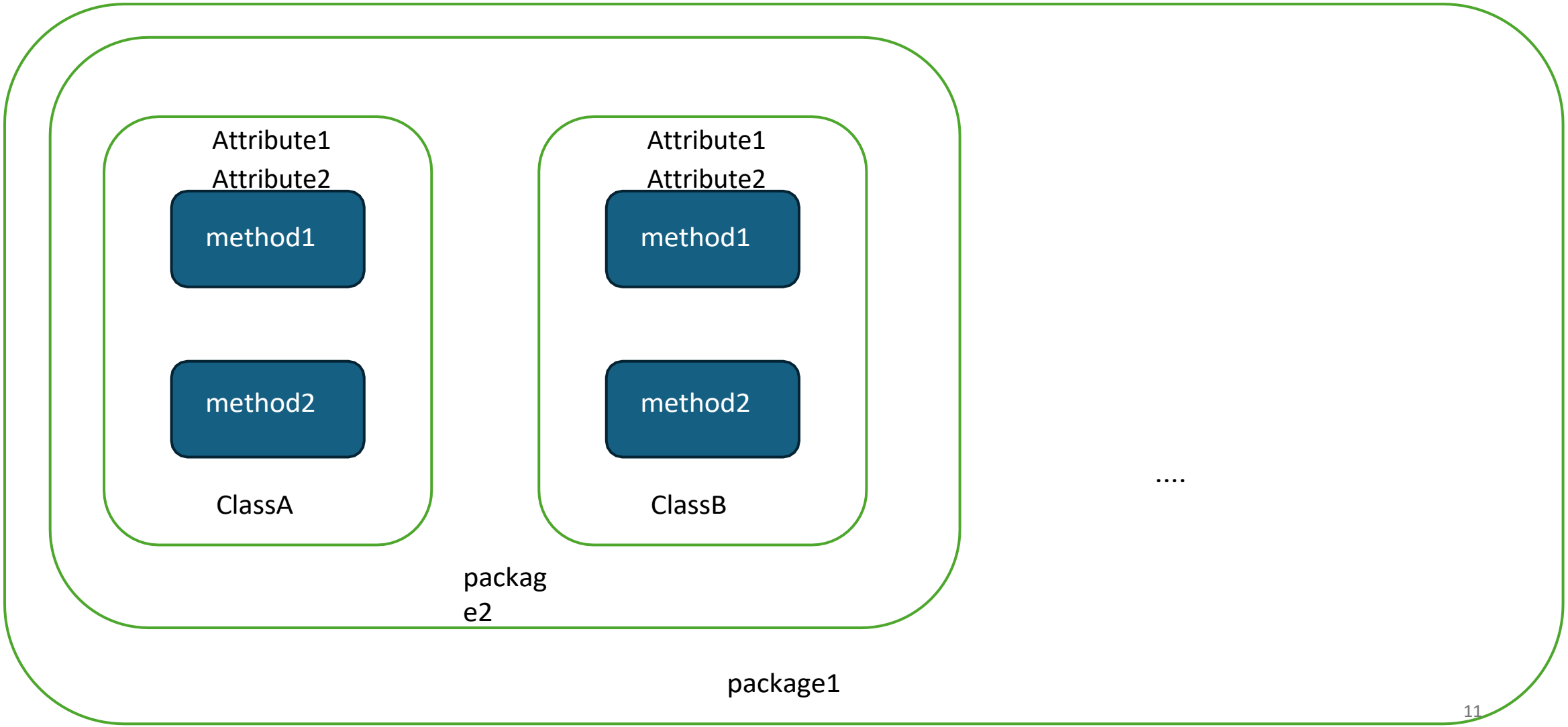
```
package sk.stuba.fiit;

public class Main {
    public static void main(String[] args) {
        System.out.printf("Hello and welcome!");

        for (int i = 1; i <= 5; i++) { System.out.println("i = "
            + i);
        }
    }
}
```

package (sk -> stuba -> fiit)

class

method (main is special)

Methods, classes, packages



Naming conventions

- Packages: *edu.cmu.cs.bovik.cheese*
- Classes: *class ImageSprite;*
- Interfaces: *interface RasterDelegate;*
- Methods: *getBackground();*
- Variables: *float myWidth;*
- Constants: *static final int MIN_WIDTH = 4;*

- https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html

Code formatting

- Yes
- How?
  - Consistently
    - In a team
    - Over time

Variables

- Primitive vs Object reference (https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html)
- Garbage Collector

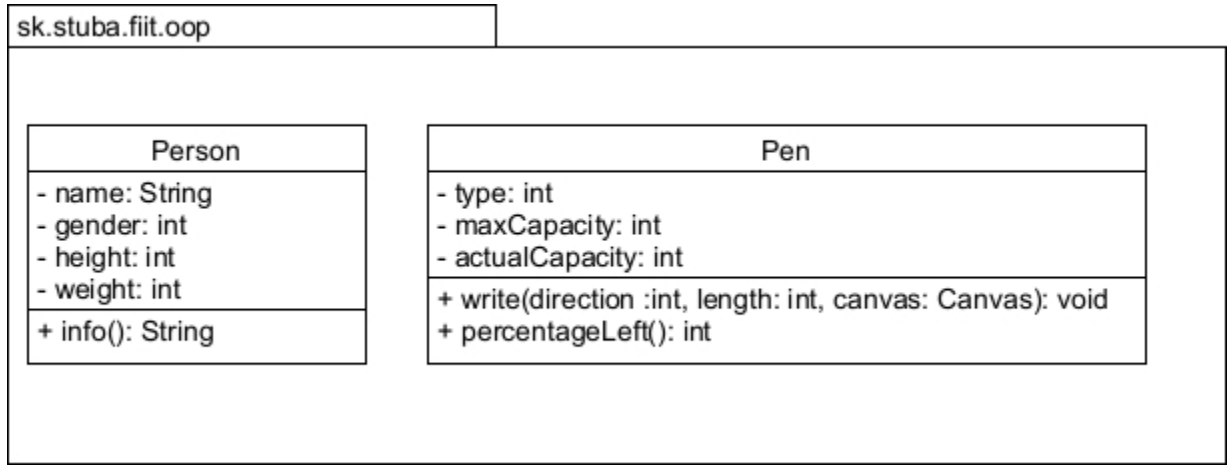
Class

- Blueprint for objects
- Attributes
- Methods

- Nested classes

(UML)

- Unified Modeling Language



Class vs Object

- Class = Template for creating objects
- Object = class instance, keyword new

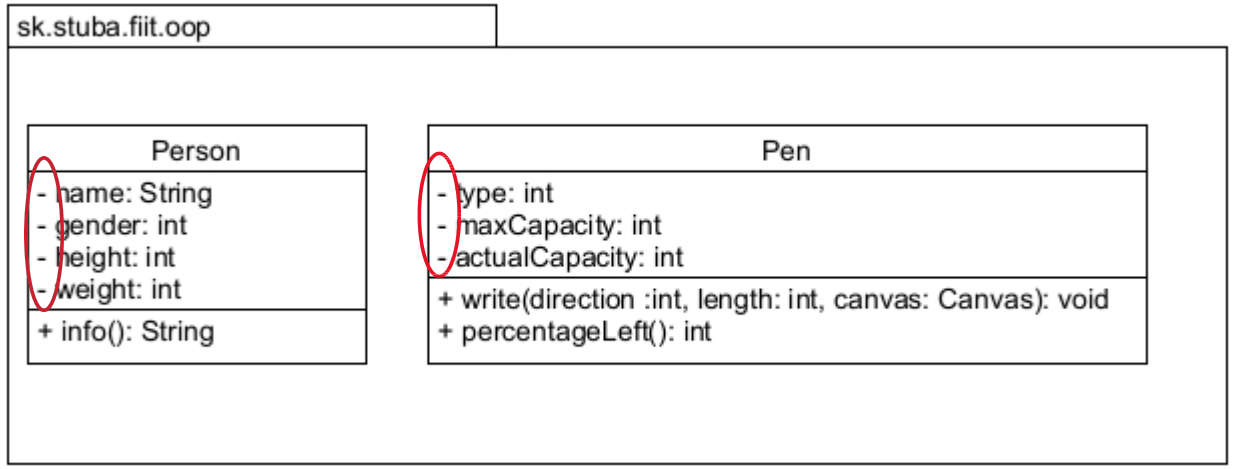
- Apple (class), my apple (object), your apple (object)

Basic elements of OOP

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction

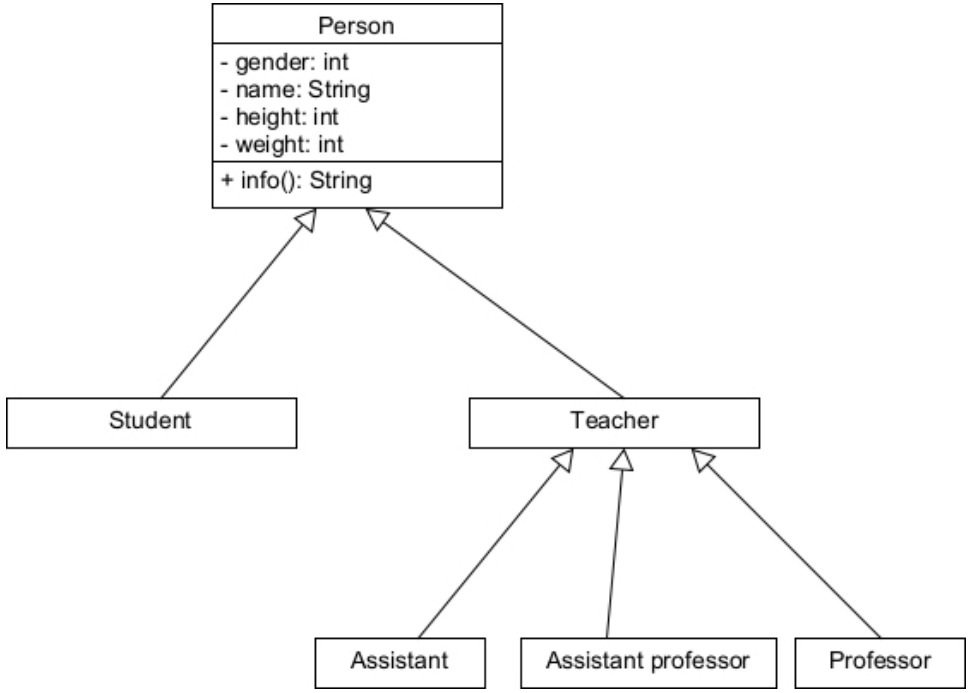
Encapsulation

- We do not manipulate data directly, but using methods
- In Java, for example, the keyword private
- Mutators, but not only that



Inheritance

- Allows classes to be "derived" from other classes
- A tree is created
- In Java, keyword extends



Polymorphism

- Compile time
  - Method overloading – same method name, different parameters
- Runtime
  - Method overriding – in inheritance
  - Virtual functions

Abstraction

- “Hiding” internal implementation
- Abstraction is at the design level, encapsulation at the implementation level
- In Java, for example, abstract class and inter