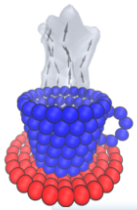
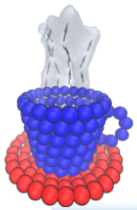


Object-oriented Programming

Axel Arnold Olaf Lenz

Institut für Computerphysik
Universität Stuttgart

March 17 - 21, 2014



Outline

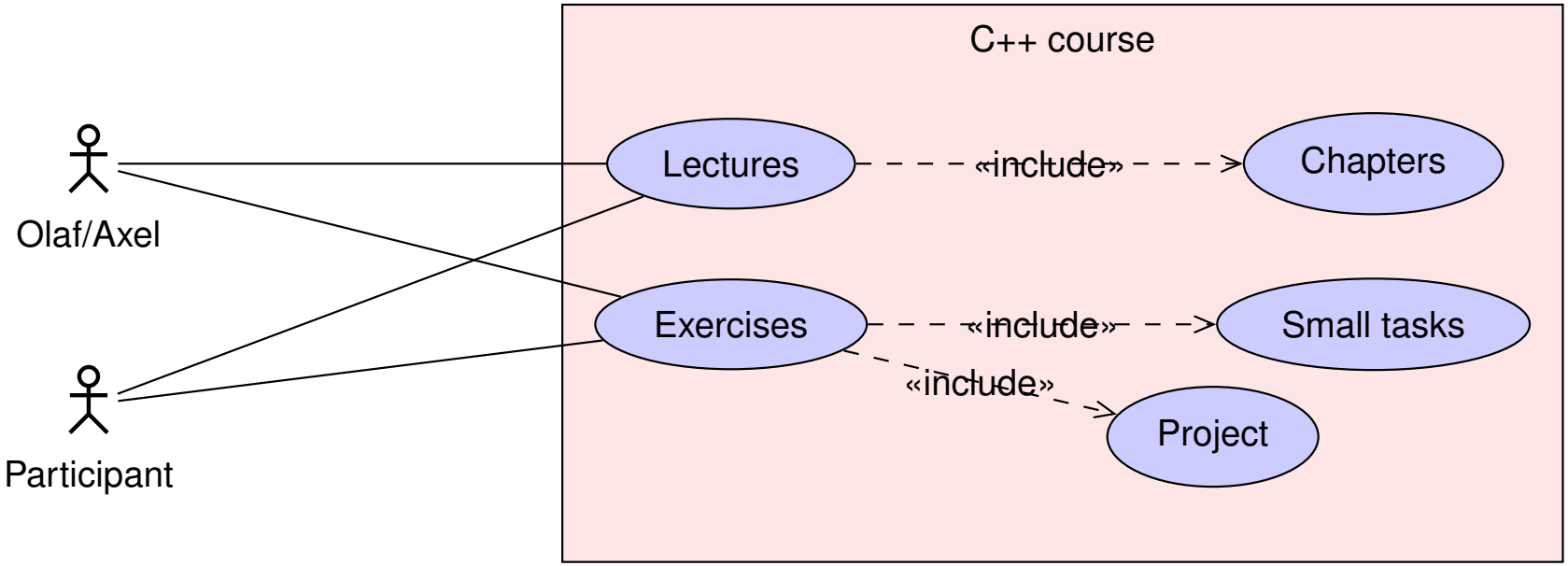
- Basic ideas
- UML – Universal Markup Language
- Classes, Interfaces and Methods
- Objects/Instances
- Composition/Fields
- Inheritance
- Polymorphism

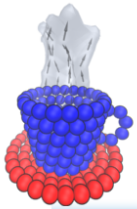
Literature

- Bruce Eckels, “Thinking in C++”, 2nd edition, Prentice-Hall
- Martin Fowler, “UML Distilled”, 3rd edition, Addison-Wesley

Object-oriented Programming (OOP)

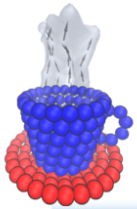
- OO is a language-indedependent concept
- in principle not limited to programming (OO design)
  $\Longrightarrow$ databases, business plans
- improves reusability and exchangability of code
- separation of partial problems
- “real world” modelling
- representation in **Universal Markup Language** (UML)





Basic ideas

- **Everything is an object**
  Axel, students, lecture
  project, world, spy, information
- **Objects interact by sending/receiving messages**
  Axel → students: object orientation is a concept
  world → map: what is the object at position X?
- **An object consists of objects**
  a course consists of lectures
  a world consists of land or water fields
- **Every object has a type**
  Axel is a teacher
  the map is a rectangle of land / water fields
- **All objects of the same type understand the same messages**
  all students hear the lecture
  all spies can retrieve information

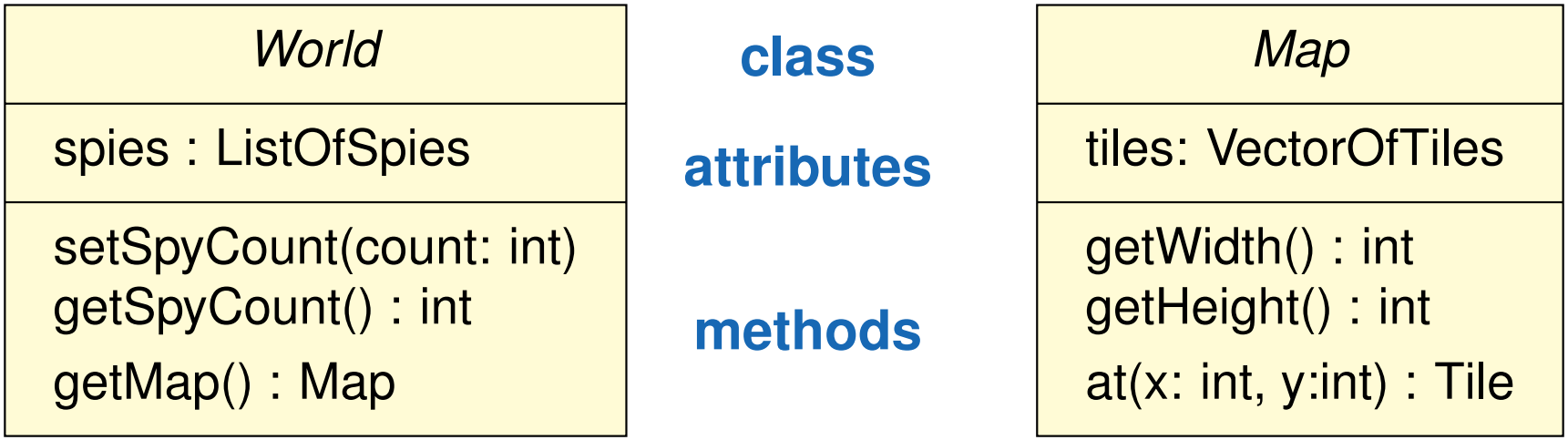


Basic ideas – in C++

- **Everything is an object**
  objects are variables

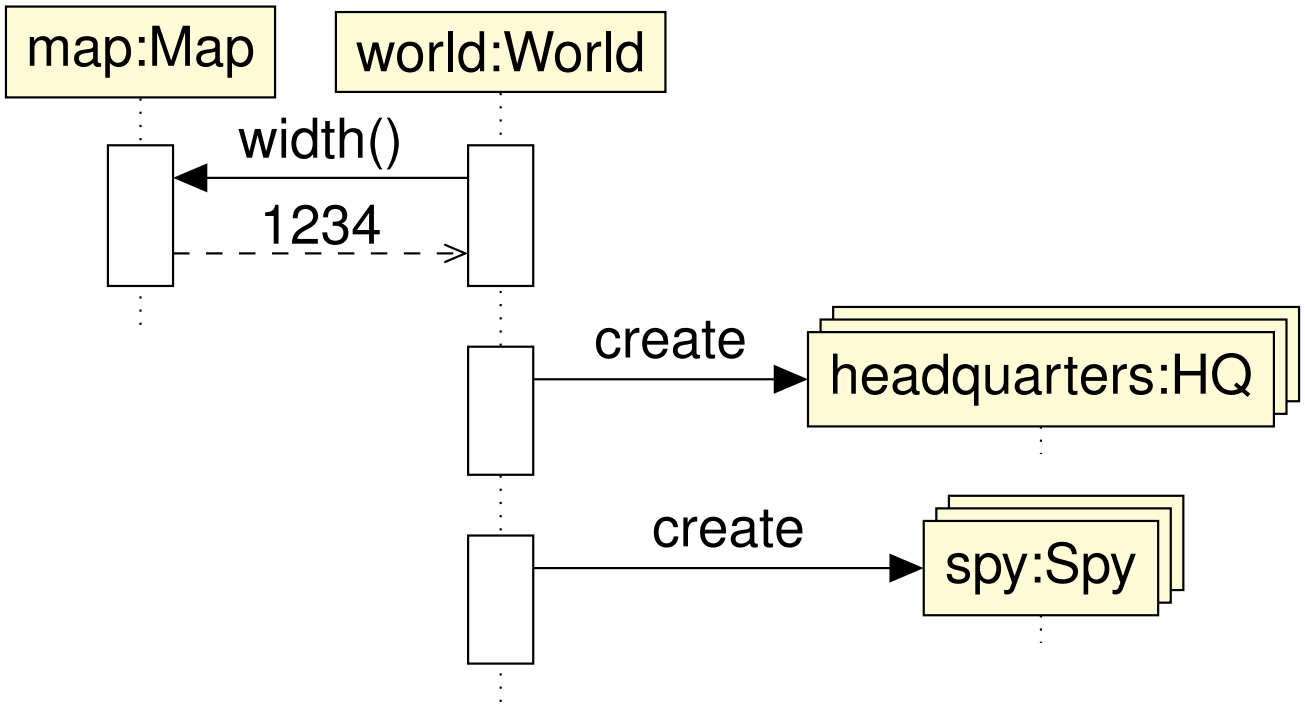
- **Objects interact by sending/receiving messages**
  methods (member functions) acting on an object (variable)
  exchanged information in arguments and return values
- **An object consists of objects**
  structured objects/records
  smallest units: C datatypes, int, char, ...
- **Every object has a type**
  classes or plain old data types in C++
  types can be derived from others
- **All objects of the same type understand the same messages**
  class declaration lists member functions (messages)
  member functions can be inherited

Classes, Interfaces and Methods



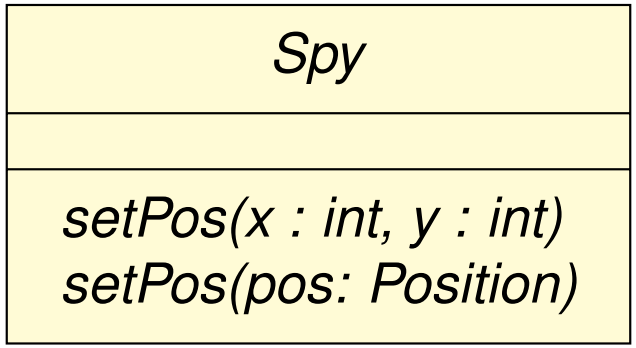
- **classes** describe the type of objects (define their **interface**)
- the interface consists of **methods** and **attributes** / properties
- methods
  - are functions that operate on objects of this class
  - can take extra arguments of arbitrary types
  - can return values of arbitrary types
- attributes are objects of arbitrary other types

Objects/Instances



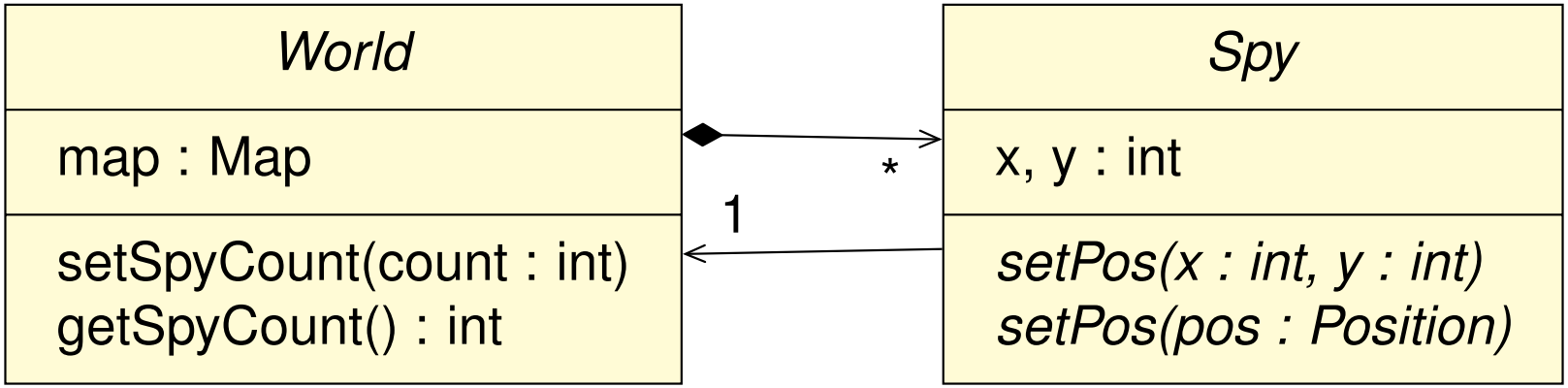
- every object has an immutable class assigned when it is created
- compare: declaration of a variable in C/C++
- objects communicate via their class interfaces
- classes can communicate via static member functions

Overloading and signature



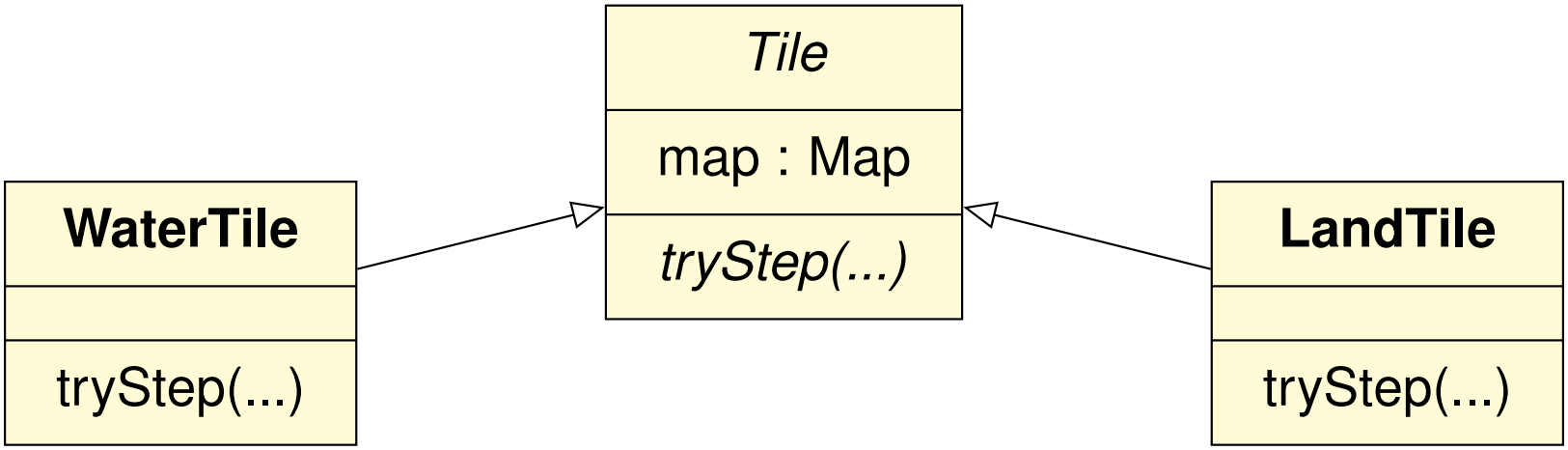
- a method is described by name and **signature**
- signature is formed by the types of all taken arguments
  - setPos(*x : int, y : int*)
  - setPos(*pos : Position*)
- methods with identical names but different arguments can exist in one class – **overloading**
- return type is *not* part of the signature – cannot always resolve overload at compile time
- C++: constness (**`const`**) is part of the signature

Composing classes



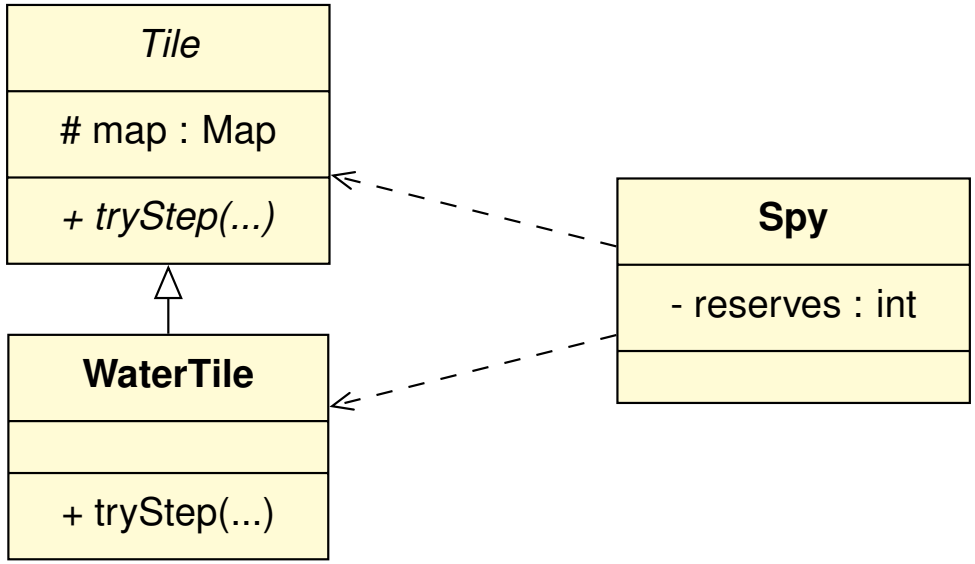
- objects are made of objects (**attributes**) — classes declare the types of these objects
- simple attributes appear below class name
- complex classes shown as **composition**
- "has-a" or "has-many" relations:
  - *a world hosts many spies*,
  - *a spy belongs to one world*

Inheritance and class hierarchy



- subclasses of classes – **class hierarchy**
- subclasses inherit methods and attributes of all superclasses
- no need to duplicate code
- .. but methods might behave differently (**polymorphism**)
  - in C++: explicitely declared by keyword `[style=inline]`virtual!
- **abstract classes** implement only parts of the interface
  - in C++: a class with purely virtual functions is abstract
  - explicitely declared by `virtual function() = 0;`

Implementation hiding



- **public** ('+') elements are visible to all
- **protected** ('#') elements are only visible to derived classes
- **private** ('-') attributes or methods are *not* visible to other objects
- `map` is protected $\Longrightarrow$ visible to `WaterTile`, not to `Spy`
- `Spy` can access `tryStep` of all tiles
- `reserves` is not visible to anybody