Object Oriented Programming

Lecture No. 2

Ing. Lukáš Slánský

Faculty of Electical Engineering and Informatics, University of Pardubice

13. 10. 2007

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Contents

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Inheritance

- Most of classes in real life can be ordered in some type of hierarchy

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Inheritance

- Most of classes in real life can be ordered in some type of hierarchy
  - Fridge, cofee-maker, microwave and others are

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Inheritance

- Most of classes in real life can be ordered in some type of hierarchy
  - Fridge, cofee-maker, microwave and others are *kitchen appliances*

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Inheritance

- Most of classes in real life can be ordered in some type of hierarchy
  - Fridge, cofee-maker, microwave and others are *kitchen appliances*
  - Kitchen appliances, computers, audiotechnic and others are

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Inheritance

- Most of classes in real life can be ordered in some type of hierarchy
  - Fridge, cofee-maker, microwave and others are *kitchen appliances*
  - Kitchen appliances, computers, audiotechnic and others are *electircal appliances*
  - ...

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Inheritance

- Most of classes in real life can be ordered in some type of hierarchy
  - Fridge, cofee-maker, microwave and others are *kitchen appliances*
  - Kitchen appliances, computers, audiotechnic and others are *electircal appliances*
  - ...
- For this type of hierarchy are typical ISA relations

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Inheritance

- Most of classes in real life can be ordered in some type of hierarchy
  - Fridge, cofee-maker, microwave and others are *kitchen appliances*
  - Kitchen appliances, computers, audiotechnic and others are *electircal appliances*
  - ...
- For this type of hierarchy are typical ISA relations (ISA = is a)

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Inheritance

- Most of classes in real life can be ordered in some type of hierarchy
  - Fridge, cofee-maker, microwave and others are *kitchen appliances*
  - Kitchen appliances, computers, audiotechnic and others are *electircal appliances*
  - ...
- For this type of hierarchy are typical ISA relations (ISA = is a)
- Hierarchy can be written by graph of classes

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

ISA Relation

- ISA relation:

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

ISA Relation

- ISA relation:
  - All dogs are animals, all birds are animals, all reptiles are animals, ...

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

ISA Relation

- ISA relation:
  - All dogs are animals, all birds are animals, all reptiles are animals, ...
  - ISA relation is not *symetric*

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

ISA Relation

- ISA relation:
  - All dogs are animals, all birds are animals, all reptiles are animals, ...
  - ISA relation is not *symetric* – Not all animals are dogs (or birds, or reptiles, ...)

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

ISA Relation

- ISA relation:
  - All dogs are animals, all birds are animals, all reptiles are animals, ...
  - ISA relation is not *symetric* – Not all animals are dogs (or birds, or reptiles, ...); it is *antisymetric*

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

ISA Relation

- ISA relation:
  - All dogs are animals, all birds are animals, all reptiles are animals, ...
  - ISA relation is not *symetric* – Not all animals are dogs (or birds, or reptiles, ...); it is *antisymetric*
  - ISA relation is *transitive*

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

ISA Relation

- ISA relation:
  - All dogs are animals, all birds are animals, all reptiles are animals, ...
  - ISA relation is not *symetric* – Not all animals are dogs (or birds, or reptiles, ...); it is *antisymetric*
  - ISA relation is *transitive* – All mammals are animals, therefore all dogs are animals

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

ISA Relation

- ISA relation:
  - All dogs are animals, all birds are animals, all reptiles are animals, ...
  - ISA relation is not *symetric* – Not all animals are dogs (or birds, or reptiles, ...); it is *antisymetric*
  - ISA relation is *transitive* – All mammals are animals, therefore all dogs are animals
  - ISA relation is *reflexive*

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

ISA Relation

- ISA relation:
  - All dogs are animals, all birds are animals, all reptiles are animals, ...
  - ISA relation is not *symetric* – Not all animals are dogs (or birds, or reptiles, ...); it is *antisymetric*
  - ISA relation is *transitive* – All mammals are animals, therefore all dogs are animals
  - ISA relation is *reflexive* – All dogs are dogs

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

ISA Relation

- ISA relation:
  - All dogs are animals, all birds are animals, all reptiles are animals, ...
  - ISA relation is not *symetric* – Not all animals are dogs (or birds, or reptiles, ...); it is *antisymetric*
  - ISA relation is *transitive* – All mammals are animals, therefore all dogs are animals
  - ISA relation is *reflexive* – All dogs are dogs
  - Mathematically said - ISA reation is partial orderring

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Properties in Hierarchy

What relation is between classes "dog" and "animal"?

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Class Properties in Hierarchy

What relation is between classes "dog" and "animal"?

- Dog has all animals' attributes

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Properties in Hierarchy

What relation is between classes "dog" and "animal"?

- Dog has all animals' attributes
  - ...and probably some more attributes

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Properties in Hierarchy

What relation is between classes "dog" and "animal"?

- Dog has all animals' attributes
  - ...and probably some more attributes
- Dog has all animals' methods

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Properties in Hierarchy

What relation is between classes "dog" and "animal"?

- Dog has all animals' attributes
  - ...and probably some more attributes
- Dog has all animals' methods
  - ...and probably some more methods

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Properties in Hierarchy

What relation is between classes "dog" and "animal"?

- Dog has all animals' attributes
  - ...and probably some more attributes
- Dog has all animals' methods
  - ...and probably some more methods
  - Some methods can differ in implementation (more later)

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Terminilogy of OOP

- Relation between animal and dog is called *inheritance* (č. dědění)

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Terminilogy of OOP

- Relation between animal and dog is called *inheritance* (č. dědění)
  - Dog takes (inherits) all its' attributes and methods

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Terminilogy of OOP

- Relation between animal and dog is called *inheritance* (č. dědění)
  - Dog takes (inherits) all its' attributes and methods
- Dog is in relation to mammal called

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Terminilogy of OOP

- Relation between animal and dog is called *inheritance* (č. dědění)
  - Dog takes (inherits) all its' attributes and methods
- Dog is in relation to mammal called *descendant* (č. potomek)

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Terminilogy of OOP

- Relation between animal and dog is called *inheritance* (č. dědění)
  - Dog takes (inherits) all its' attributes and methods
- Dog is in relation to mammal called *descendant* (č. potomek)
- Mammal is in relation to dog calles

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Terminilogy of OOP

- Relation between animal and dog is called *inheritance* (č. dědění)
  - Dog takes (inherits) all its' attributes and methods
- Dog is in relation to mammal called *descendant* (č. potomek)
- Mammal is in relation to dog calles *ancestor* (č. předek)

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Using inheritance in Pascal

```
type CAnimal=object
               private
                 Weight:Real;
                 ...
               public
                 procedure feed;
                 procedure sleep(howLong:Real);
                 ...
             end;
```

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Using inheritance in Pascal

```
type CAnimal=object
               private
                 Weight:Real;
                 ...
               public
                 procedure feed;
                 procedure sleep(howLong:Real);
                 ...
             end;
     CDog=object
             private
               ...
             public
               procedure bark;
               ...
           end;
```

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Using inheritance in Pascal

```
type CAnimal=object
               private
                 Weight:Real;
                 ...
               public
                 procedure feed;
                 procedure sleep(howLong:Real);
                 ...
             end;
     CDog=object(CMammal)
             private
               ...
             public
               procedure bark;
               ...
           end;
```

Object Inheritance Limitations

- You can inherit only from one ancestor

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Object Inheritance Limitations

- You can inherit only from one ancestor
  - There are (rare) languages with multiple inheritance – e.g. C++

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Object Inheritance Limitations

- You can inherit only from one ancestor
  - There are (rare) languages with multiple inheritance – e.g. C++
- Neither attributes nor methods can be forgotten

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Object Inheritance Limitations

- You can inherit only from one ancestor
  - There are (rare) languages with multiple inheritance – e.g. C++
- Neither attributes nor methods can be forgotten
  - Methods' implementation can be altered

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Object Inheritance Limitations

- You can inherit only from one ancestor
  - There are (rare) languages with multiple inheritance – e.g. C++
- Neither attributes nor methods can be forgotten
  - Methods' implementation can be altered – through redefining or polymorphism (more later)

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Object Inheritance Limitations

- You can inherit only from one ancestor
  - There are (rare) languages with multiple inheritance – e.g. C++
- Neither attributes nor methods can be forgotten
  - Methods' implementation can be altered – through redefining or polymorphism (more later)
  - Some good object-oriented languages (Java, C++, not Pascal) can distinguish between methods (and functions) according to number and type of parameters or return type

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

When to Use Inheritance?

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

When to Use Inheritance?

- If there is hierarchical ISA relation between classes
  - One class is specialisation of other one

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

When to Use Inheritance?

- If there is hierarchical ISA relation between classes
  - One class is specialisation of other one

Right usage:

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

When to Use Inheritance?

- If there is hierarchical ISA relation between classes
  - One class is specialisation of other one

Right usage:

- Class *man* is ancestor for classes *employee* and *customer*

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

When to Use Inheritance?

- If there is hierarchical ISA relation between classes
  - One class is specialisation of other one

Right usage:

- Class *man* is ancestor for classes *employee* and *customer*
  - Because both employee and customer are mans...

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

When to Use Inheritance?

- If there is hierarchical ISA relation between classes
  - One class is specialisation of other one

Right usage:

- Class *man* is ancestor for classes *employee* and *customer*
  - Because both employee and customer are mans...

Wrong usage:

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

When to Use Inheritance?

- If there is hierarchical ISA relation between classes
  - One class is specialisation of other one

Right usage:

- Class *man* is ancestor for classes *employee* and *customer*
  - Because both employee and customer are mans...

Wrong usage:

- Class *point* is ancestor for classes *circle* or *vector*

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

When to Use Inheritance?

- If there is hierarchical ISA relation between classes
  - One class is specialisation of other one

Right usage:

- Class *man* is ancestor for classes *employee* and *customer*
  - Because both employee and customer are mans...

Wrong usage:

- Class *point* is ancestor for classes *circle* or *vector*
  - *Neither* circle *nor* vector is point

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

When to Use Inheritance?

- If there is hierarchical ISA relation between classes
  - One class is specialisation of other one

Right usage:

- Class *man* is ancestor for classes *employee* and *customer*
  - Because both employee and customer are mans...

Wrong usage:

- Class *point* is ancestor for classes *circle* or *vector*
  - *Neither* circle *nor* vector is point
  - Circle (and vector) can contain point

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

When to Use Inheritance?

- If there is hierarchical ISA relation between classes
  - One class is specialisation of other one

Right usage:

- Class *man* is ancestor for classes *employee* and *customer*
  - Because both employee and customer are mans...

Wrong usage:

- Class *point* is ancestor for classes *circle* or *vector*
  - *Neither* circle *nor* vector is point
  - Circle (and vector) can contain point – but there is other type of relation, not ISA

Univerzita Pardubice
Fakulta elektrotechniky a informatiky