Object Oriented Programming

Lecture No. 1

Ing. Lukáš Slánský

Faculty of Electical Engineering and Informatics, University of Pardubice

6. 10. 2007

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Contents

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Literature

1. Ilja Kraval – Základy objektově orientovaného programování (will be placed in the STAG)

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Literature

1. Ilja Kraval – Základy objektově orientovaného programování (will be placed in the STAG)
2. Arlow Jim – UML a unifikovaný proces vývoje aplikací

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Literature

1. Ilja Kraval – Základy objektově orientovaného programování (will be placed in the STAG)
2. Arlow Jim – UML a unifikovaný proces vývoje aplikací
3. Martin Fowler – Refaktoring – Zlepšení existujícího kódu

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

History of Programming Styles

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

History of Programming Styles

- Machine Code

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

History of Programming Styles

- Machine Code
  - only instructions and jumps written as processor commands

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

History of Programming Styles

- Machine Code
  - only instructions and jumps written as processor commands
- Structured Programming

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

History of Programming Styles

- Machine Code
  - only instructions and jumps written as processor commands
- Structured Programming
  - using procedures, loops, conditions etc.

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

History of Programming Styles

- Machine Code
  - only instructions and jumps written as processor commands
- Structured Programming
  - using procedures, loops, conditions etc.
  - better arranged source code

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

History of Programming Styles

- Machine Code
  - only instructions and jumps written as processor commands
- Structured Programming
  - using procedures, loops, conditions etc.
  - better arranged source code
- Modular Programming

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

History of Programming Styles

- Machine Code
  - only instructions and jumps written as processor commands
- Structured Programming
  - using procedures, loops, conditions etc.
  - better arranged source code
- Modular Programming
  - source code splitted into smaller parts – files

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

History of Programming Styles

- Machine Code
  - only instructions and jumps written as processor commands
- Structured Programming
  - using procedures, loops, conditions etc.
  - better arranged source code
- Modular Programming
  - source code splitted into smaller parts – files
  - more programmers are can work on the same project

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

History of Programming Styles

- Machine Code
  - only instructions and jumps written as processor commands
- Structured Programming
  - using procedures, loops, conditions etc.
  - better arranged source code
- Modular Programming
  - source code splitted into smaller parts – files
  - more programmers are can work on the same project
- Object Oriented Programming

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

History of Programming Styles

- Machine Code
  - only instructions and jumps written as processor commands
- Structured Programming
  - using procedures, loops, conditions etc.
  - better arranged source code
- Modular Programming
  - source code splitted into smaller parts – files
  - more programmers are can work on the same project
- Object Oriented Programming
  - merging data and operations into one package, reusability

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

History of Programming Styles

- Machine Code
  - only instructions and jumps written as processor commands
- Structured Programming
  - using procedures, loops, conditions etc.
  - better arranged source code
- Modular Programming
  - source code splitted into smaller parts – files
  - more programmers are can work on the same project
- Object Oriented Programming
  - merging data and operations into one package, reusability
- Component Programming

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

History of Programming Styles

- Machine Code
  - only instructions and jumps written as processor commands
- Structured Programming
  - using procedures, loops, conditions etc.
  - better arranged source code
- Modular Programming
  - source code splitted into smaller parts – files
  - more programmers are can work on the same project
- Object Oriented Programming
  - merging data and operations into one package, reusability
- Component Programming
- ???

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Core Terms of OOP

- Object (č. objekt)

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Core Terms of OOP

- Object (č. objekt)
- Encapsulation (č. zapouzdření)

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Core Terms of OOP

- Object (č. objekt)
- Encapsulation (č. zapouzdření)
- Message (č. zpráva)

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Core Terms of OOP

- Object (č. objekt)
- Encapsulation (č. zapouzdření)
- Message (č. zpráva)
- Class (č. třída)

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Object

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Object

- Has state and behaviour

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Object

- Has state and behaviour
- State i stored in variables – *attributes*

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Object

- Has state and behaviour
- State i stored in variables – *attributes*
- Behaviour is implemented by functions (or procedures) – *methods*

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Object

- Has state and behaviour
- State i stored in variables – *attributes*
- Behaviour is implemented by functions (or procedures) – *methods*
- *No attribute or method is visible from outside the object*

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Object

- Has state and behaviour
- State i stored in variables – *attributes*
- Behaviour is implemented by functions (or procedures) – *methods*
- *No attribute or method is visible from outside the object*
  - This is called *encapsulation*

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Message

- How to get across the "capsule" of encapsulation?

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Message

- How to get across the "capsule" of encapsulation?
- Is is possible to make I/O channel that can be used to send message and receive answer.

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Message

- How to get across the "capsule" of encapsulation?
- Is is possible to make I/O channel that can be used to send message and receive answer.
- This mechanism is implemented as calling the method (similar to calling function) in most of OO languages.

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class

- More objects has offten similar behaviour patterns (methods) and same attributes.

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class

- More objects has offten similar behaviour patterns (methods) and same attributes.
  - Alík, Punťa and Rex are all Dogs.

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class

- More objects has offten similar behaviour patterns (methods) and same attributes.
  - Alík, Punťa and Rex are all Dogs.
  - they have the same methods

Class

- More objects has offten similar behaviour patterns (methods) and same attributes.
  - Alík, Punťa and Rex are all Dogs.
  - they have the same methods – they can bark, run, eat, sleep, ...

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class

- More objects has offten similar behaviour patterns (methods) and same attributes.
  - Alík, Punťa and Rex are all Dogs.
  - they have the same methods – they can bark, run, eat, sleep, ...
  - they have same attributes

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class

- More objects has offten similar behaviour patterns (methods) and same attributes.
  - Alík, Punťa and Rex are all Dogs.
  - they have the same methods – they can bark, run, eat, sleep, ...
  - they have same attributes – name, coat, eyes, height, ...

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class

- More objects has offten similar behaviour patterns (methods) and same attributes.
  - Alík, Punťa and Rex are all Dogs.
  - they have the same methods – they can bark, run, eat, sleep, ...
  - they have same attributes – name, coat, eyes, height, ...
- We can make new (more general) abstraction

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class

- More objects has offten similar behaviour patterns (methods) and same attributes.
  - Alík, Punťa and Rex are all Dogs.
  - they have the same methods – they can bark, run, eat, sleep, ...
  - they have same attributes – name, coat, eyes, height, ...
- We can make new (more general) abstraction – class of objects Dog.

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class

- More objects has offten similar behaviour patterns (methods) and same attributes.
  - Alík, Punťa and Rex are all Dogs.
  - they have the same methods – they can bark, run, eat, sleep, ...
  - they have same attributes – name, coat, eyes, height, ...
- We can make new (more general) abstraction – class of objects Dog.
  - Alík, Punťa, Rex and all other dogs are object of the class Dog.

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Example in Pascal – I.

```
type TDog=object
  public
    procedure YourNameIs(newName:string);
    function WhatIsYourName:string;
    procedure Bark;
  private
    name: string;
end;
```

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Example in Pascal – II.

```
function TDog.WhatIsYourName:string;
begin
  WhatIsYourName:=name;
end;
```

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Example in Pascal – II.

```
function TDog.WhatIsYourName:string;
begin
  WhatIsYourName:=name;
end;

procedure TDog.YourNameIs(newName:string);
begin
  name:=newName;
end;
```

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Example in Pascal – II.

```
function TDog.WhatIsYourName:string;
begin
  WhatIsYourName:=name;
end;

procedure TDog.YourNameIs(newName:string);
begin
  name:=newName;
end;

procedure TDog.Bark;
begin
  WriteLn('Bow Bow');
end;
```

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Example in Pascal – III.

```
var Alik:TDog;
...
```

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Example in Pascal – III.

```
var Alik:TDog;
...
Alik.YourNameIs('Alicek');
```

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Example in Pascal – III.

```
var Alik:TDog;
...
Alik.YourNameIs('Alicek');
Alik.Bark;
```

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Class Example in Pascal – III.

```
var Alik:TDog;
...
Alik.YourNameIs('Alicek');
Alik.Bark;
WriteLn('My name is: ', Alik.WhatIsYourName);
```

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Encapsulation Pascal

- Object extension of Pascal is very (very!) simple

Univerzita
Pardubice
Fakulta elektrotechniky
a informatiky

Encapsulation Pascal

- Object extension of Pascal is very (very!) simple
- All attributes and methods are visible in the whole module in which they are declared (messages are module-wide accessible)

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Encapsulation Pascal

- Object extension of Pascal is very (very!) simple
- All attributes and methods are visible in the whole module in which they are declared (messages are module-wide accessible)
- Outside module (in the case of class in own module) is access driven by using keywords `public` and `private`

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Example of Encapsulation in Pascal

```
type TDog=object
  public
    procedure YourNameIs(newName:string);
    function WhatIsYourName:string;
    procedure Bark;
  private
    name: string;
end;
```

Univerzita Pardubice
Fakulta elektrotechniky a informatiky