OBCHODNÍ AKADEMIE KROMĚŘÍŽ

Objektové programování a abstraktní datové typy

Radim Halfar

Seminář z programování

Obsah

Obsah

Obsah

Výraz průměrného studenta
při probírání datových struktur

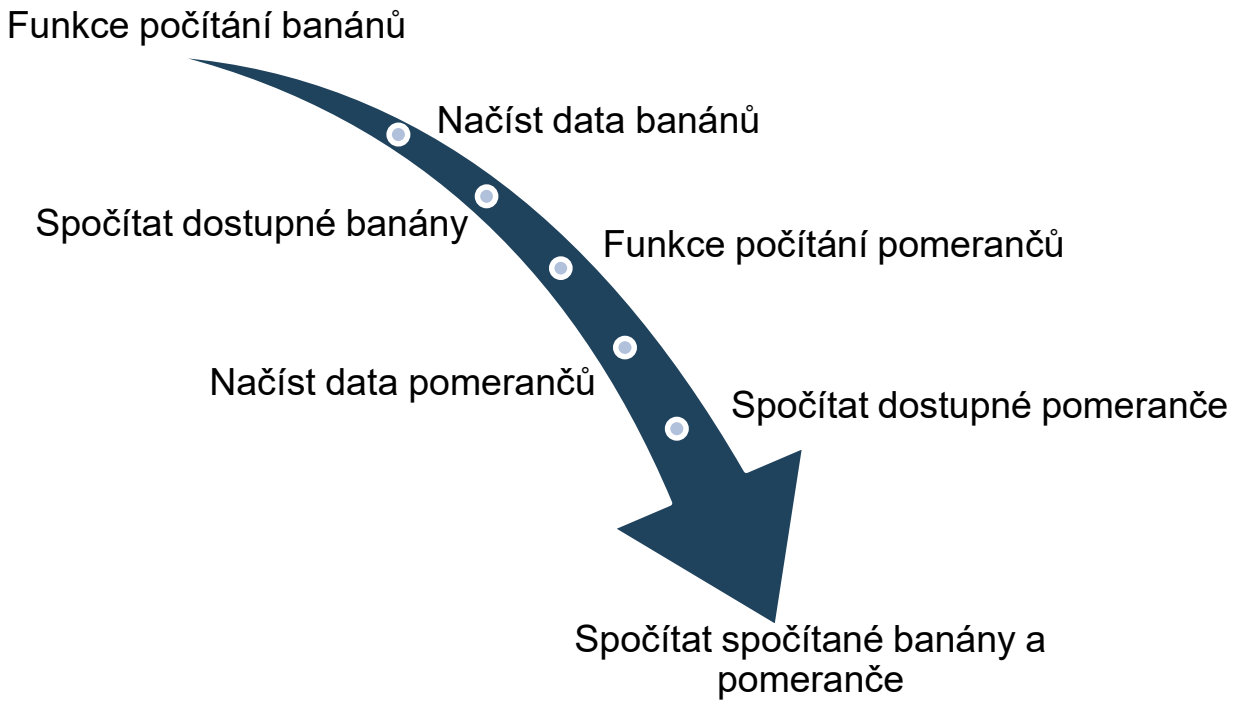
1 / Objektové programování

Programovací styl (vzorec myšlení, model), který představuje **způsob vývoje aplikací** v softwarovém inženýrství **pomocí přidružení dat k objektům.**

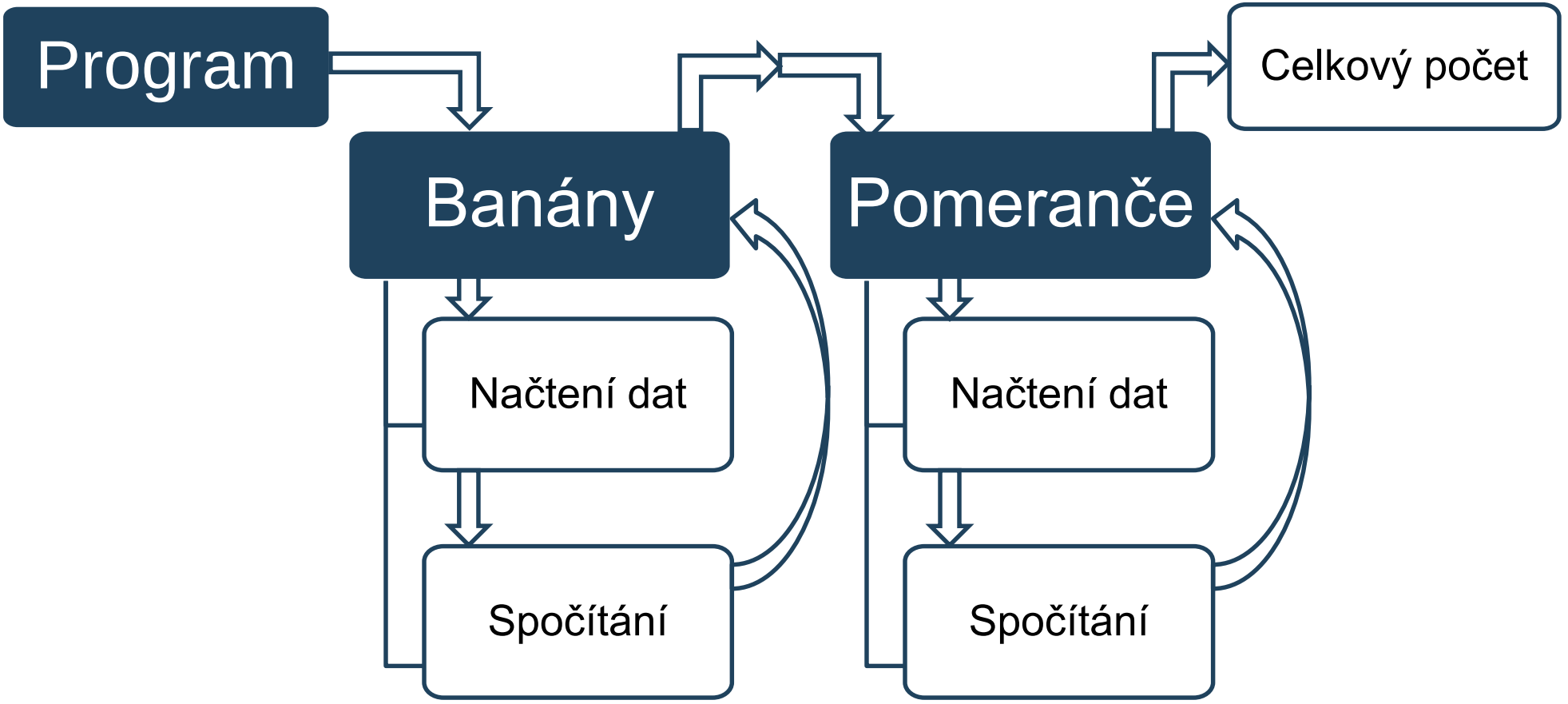
1 / Objektové programování

- Označováno jako **OOP** (Object-oriented programming)
- **Cíle**
  - Tvorba malých relativně samostatných znovupoužitelných jednotek (objektů)
  - **Řešit nevýhody procedurálního programování**
    - *Globálně viditelné funkce, znovu použitelnost, …*
- **Důvod použití**
  - Snazší a rychlejší vývoj aplikací
  - Snazší udržitelnost
  - Menší chybovost
  - Řeší nevýhody procedurálního programování

1 / Rozdíly – Strukturované



1 / Rozdíly – Objektové



1 / Základní pojmy OOP

Třída (Class)

Objekty (Object)

Vlastnosti (Property)

Metody (Method)

- Funkce (Function)
- Procedury (Procedure)

Viditelnost (Visibility)

Dále taky…

- Třídní vlastnosti (Static)
- Konstruktory (Constructor)
- Rozhraní (Interface)
- Dědičnost (Inheritance)

1 / Třída

- Množina objektů s určitými vlastnostmi
- Šablona objektu
- Definuje všechny možné stavy objektu
  - Pomocí vlastností

1 / Třída

Vykrajovátko na perníčky – Třída perníčků

1 / Třída

Zápis v programovacím jazyce C#:

```
namespace NazevJmenehoProstoru
{
    class NazevTridy
    {
        // Vlastnosti


        // Metody (Funkce a procedury)

    }
}
```

1 / Objekt

- **Instance třídy**
- Vzniká na základě (předpisu) třídy
- Může vzniknout tisíce objektů dle jedné třídy
- Uvnitř třídy lze odkázat na současný objekt pomocí klíčového slova **this**

1 / Objekt

Perníček – Objekt třídy perníčků

1 / Objekt

Zápis v programovacím jazyce C#:

```
static void Main(string[] args)
{
    // Založení nového objektu dle třídy
    NazevTridy nazevObjektu = new NazevTridy();

    // Volání metody založeného objektu
    nazevObjektu.nazevMetody(arg1, arg2);
}
```

1 / Vlastnosti (atributy)

- Konkrétní data či informace o objektu
- Jedná se o proměnné a jejich hodnoty

- Definují se pomocí třídy

1 / Vlastnosti

Vlastností na perníčku může být:

- Výraz tváře ➔ úsměv
- Barva těsta ➔ hnědá
- Dekorace ➔ cukr

```
class Pernicek
{
    string vyrazTvare = "úsměv";
    Color barvaTesta = Color.FromArgb(114, 85, 52);
    OzdobaPernicku ozdoba = new OzdobaPernicku("cukr");

    //místo pro metody
}
```

Perníček – Objekt třídy perníčků

1 / Metody objektu

- Funkce nebo procedura objektu definovaná v šabloně objektu (třídě)
- Mohou být i **statické** – nebude metodou objektu, ale celé třídy (volání přes třídu)
  - V takovém případě nelze přistupovat k vlastnostem objektu
  - Pouze v případě, je-li konkrétní objekt parametrem v metodě

1 / Metody objektu

Metody perníčku například mohou být:

- Pozdrav

```
internal class Pernicek
{
    //místo pro vlastnosti

    internal void sayHello()
    {
        Console.WriteLine("Perníček říká: Hello!");
    }
}
```

Perníčkův pozdrav
(Metoda objektu ze třídy perníčků)

1 / Viditelnost

- Označuje možný přístup ke třídě/metodě/vlastnosti uvnitř třídy z celého programu/knihovny
- Určuje se pomocí **přístupových modifikátorů**
- V rámci programovacího jazyka C# jsou 4:
  - Private
  - Protected
  - Public
  - Internal
- Pokud není zadán žádný, používá se *internal*

1 / Přístupové modifikátory

- **Private**
  - Metodu/vlastnost lze volat pouze z kontextu dané třídy (neboli uvnitř třídy)
- **Public**
  - Metoda/vlastnost volatelná i z ostatních částí programu nebo knihovny nebo dalších programů
- **Protected**
  - Nelze volat z ostatních částí programu, kromě případu dědění, kdy jsou vlastnosti/metody viditelné z třídy potomka
- **Internal**
  - Stejný jako *public*, pouze v rámci stejného programu

1 / Přístupové modifikátory

Umístění volajícího	public	protected internal	protected	internal	private protected	private
V rámci třídy	✔	✔	✔	✔	✔	✔
Odvozená třída (stejné sestavení)	✔	✔	✔	✔	✔	✘
Neodvozená třída (stejné sestavení)	✔	✔	✘	✔	✘	✘
Odvozená třída (jiné sestavení)	✔	✔	✔	✘	✘	✘
Neodvozená třída (jiné sestavení)	✔	✘	✘	✘	✘	✘

1 / Viditelnost

```
internal class Pernicek
{
    private int casPeceni = 8;

    internal void PectZridka()
    {
        this.casPeceni = 8;
        this.Pect();
    }

    internal void PectNormalne()
    {
        this.casPeceni = 10;
        this.Pect();
    }

    internal void PectDoCerna()
    {
        this.casPeceni = 50;
        this.Pect();
    }

    protected void Pect()
    {
        // Metoda na peceni
        // pracuje s casPeceni
    }
}
```

1 / Konstruktor

- **Speciální metoda třídy** (někdy tzv. magická funkce)
- Pro naplnění vlastností objektu počátečními hodnotami
- Zároveň spustí počáteční kód, při volání nového objektu pomocí *new*

1 / Konstruktor

Zápis v programovacím jazyce C#:

```
internal class NazevTridy
{
    //místo pro vlastnosti

    internal NazevTridy()
    {
        //Konstruktor bez parametrů
    }

    internal NazevTridy(datovyTyp parametrKonstuktoru)
    {
        //Konstruktor s parametrem
        vlastnostTridy = parametrKonstuktoru;
    }

    //místo pro metody
}
```

1 / Destruktor

- Další speciální metoda třídy
- Spustí se při odstraňování objektu z paměti programu (např. smazání z pole)
- Nespouští se v případě, kdy bývá celý program ukončen

1 / Destruktor

Zápis v programovacím jazyce C#:

```
internal class Pernicek
{
    //místo pro vlastnosti
    //místo pro metody

    ~Pernicek()
    {
        //Destruktor bez parametrů
    }
}
```

1 / Setter a getter

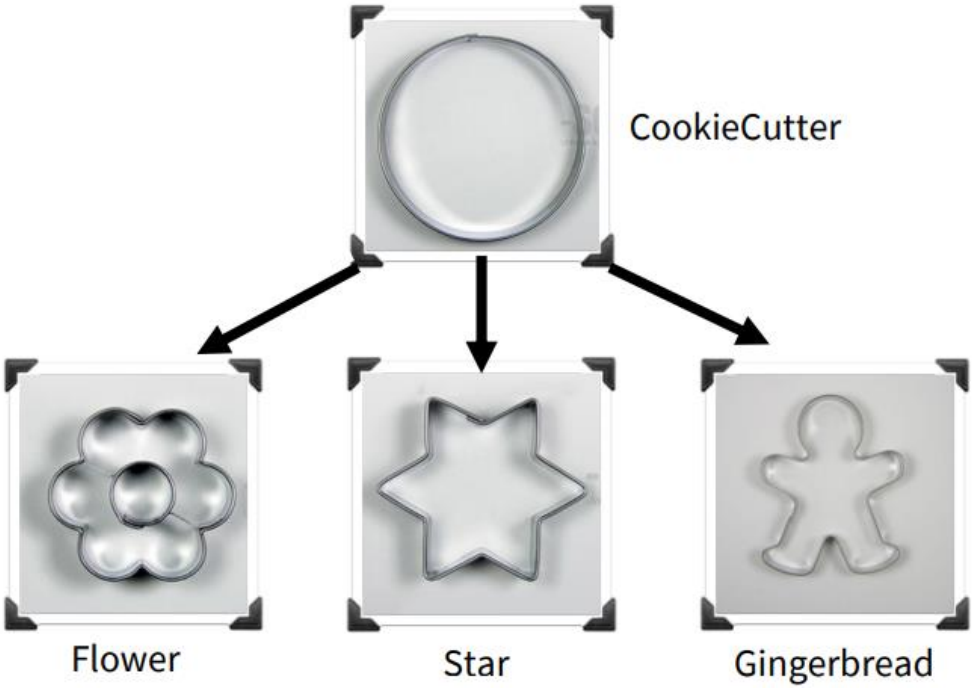
- **Getter**
  - Metoda, která se zavolá při přístupu k atributu (vlastnosti) v dané třídě.
- **Setter**
  - Metoda, která umožňuje nastavit nebo změnit hodnotu atributu (vlastnosti) v dané třídě.

1 / Setter a getter pro vlastnost v objektu

```
internal class Pernicek
{
    private int _casPeceni = 8;

    public int CasPeceni {
        get
        {
            // nějaký kód v rámci metody
            // pro získání vlastnosti
            return _casPeceni;
        }
        set
        {
            // nějaký kód v rámci metody
            // pro nastavení vlastnosti
            _casPeceni = value;
        }
    }
}
```

1 / Dědičnost objektů



2 / Abstraktní datové typy

- Informace níže

3 / Přístupy k datům v paměti

- Informace níže

3 / Předávání dat funkcím

- Informace níže

4 / Kolekce objektů

- Informace níže

5 / Generické funkce a třídy

- Informace níže

5 / Přetěžování operátorů

- Informace níže

6 / Výčtový typ (Enum)

- Informace níže

7 / Seznam (List)

- Informace níže

8 / Zásobník (Stack)

- Informace níže

8 / Fronta (Queue)

- Informace níže

9 / Strom (Tree)

- Informace níže

10 / Pole (Array)

- Informace níže

11 / Množina (Set)

- Informace níže

12 / Vektor (Vector)

- Informace níže

Dotazy?

Shrnutí

Děkuji za pozornost

Radim Halfar
Seminář z programování

halfar@oakm.cz
přízemí – dveře č. 156