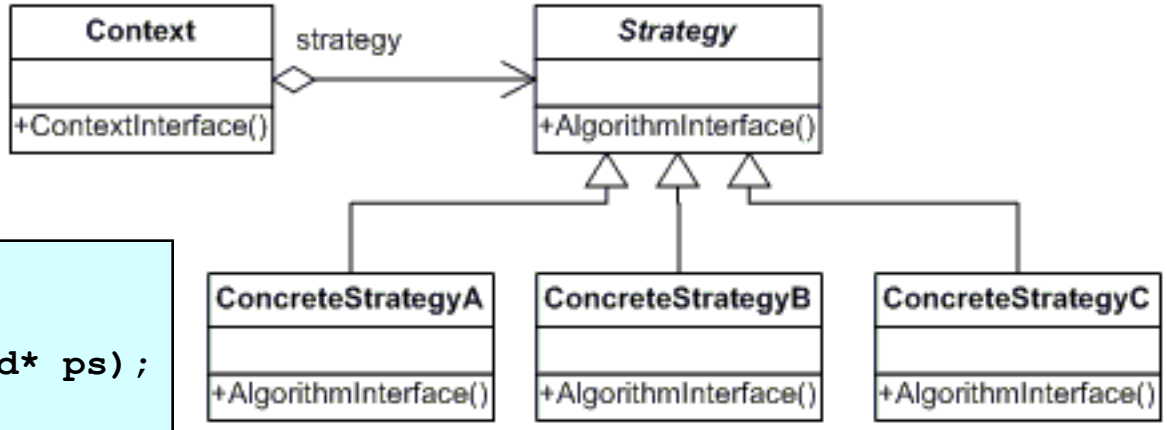
Strategy

Strategy – „All-in-1“ na začátek




```
class Context {
public:
  Context(AStrategy* s, void* ps);
  void Algorithm();
private:
  AStrategy* strategy;
  void* ParamStruct;
};
```

```
class AStrategy {
public:
  virtual void Algorithm(void* ParamStruct)=0;
protected:
  AStrategy();
};
```

```
class SpecificStrategy: public AStrategy {
public:
  virtual void Algorithm(void* ParamStruct);
  SpecificStrategy();
};
```

Příklad: Indentace kódu v textovém editoru

Code Indenter

<- Indent

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Collections;
using System.Text;
using System.Text.RegularExpressions;
using System.Windows.Forms;
using System.ComponentModel;
using System.Runtime.InteropServices;

namespace SampleApplication
{
    public class token
    {
        private Color tokenColor;
        private Font tokenFont;
        public token(Color tColor, Font tFont)
        {
            this.tokenColor = tColor;
            this.tokenFont = tFont;
        }
        public String getValue()
        {
            return this.tokenValue;
        }
        public Color getColor()
        {
            return this.tokenColor;
        }
        public Font getFont()
        {
            return this.tokenFont;
```

Code Indenter

<- Indent

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transiti
<html>
  <head>
    <meta http-equiv="content-type" content="text/htm
    <meta name="generator" content="PSPad editor, www
    <title>Tradiční layouty</title>
    <style type="text/css">
      @import url("./style.css");
    </style>
  </head>
  <body>
    <div class="stranka">
      <div class="hlavicka">
        <h1>Nadpis stranky</h1>
      </div>
      <div class="levy_sloupec">
        <div class="menu">
          Menu
          <ul>
            <li>polozka menu 1</li>
            <li>polozka menu 2</li>
            <li>polozka menu 3</li>
          </ul>
        </div>
      </div>
      <div class="pravy_sloupec">
        <div class="content">
          Lorem ipsum dolor sit amet consectetuer
          nunc diam consectetuer sed sed. Aliquam
          tincidunt ut elit sed Curabitur id pellente
          a Aliquam tempus. Curabitur lorem quis ac
          sodales ac sed pede quis enim eros. Porta i
```

Příklad: Layout managers – AWT (Border Layout)

NetBeans IDE 5.5 - JavaApplication1

File Edit View Navigate Source Refactor Build Run CVS Tools Window Help

Inspector

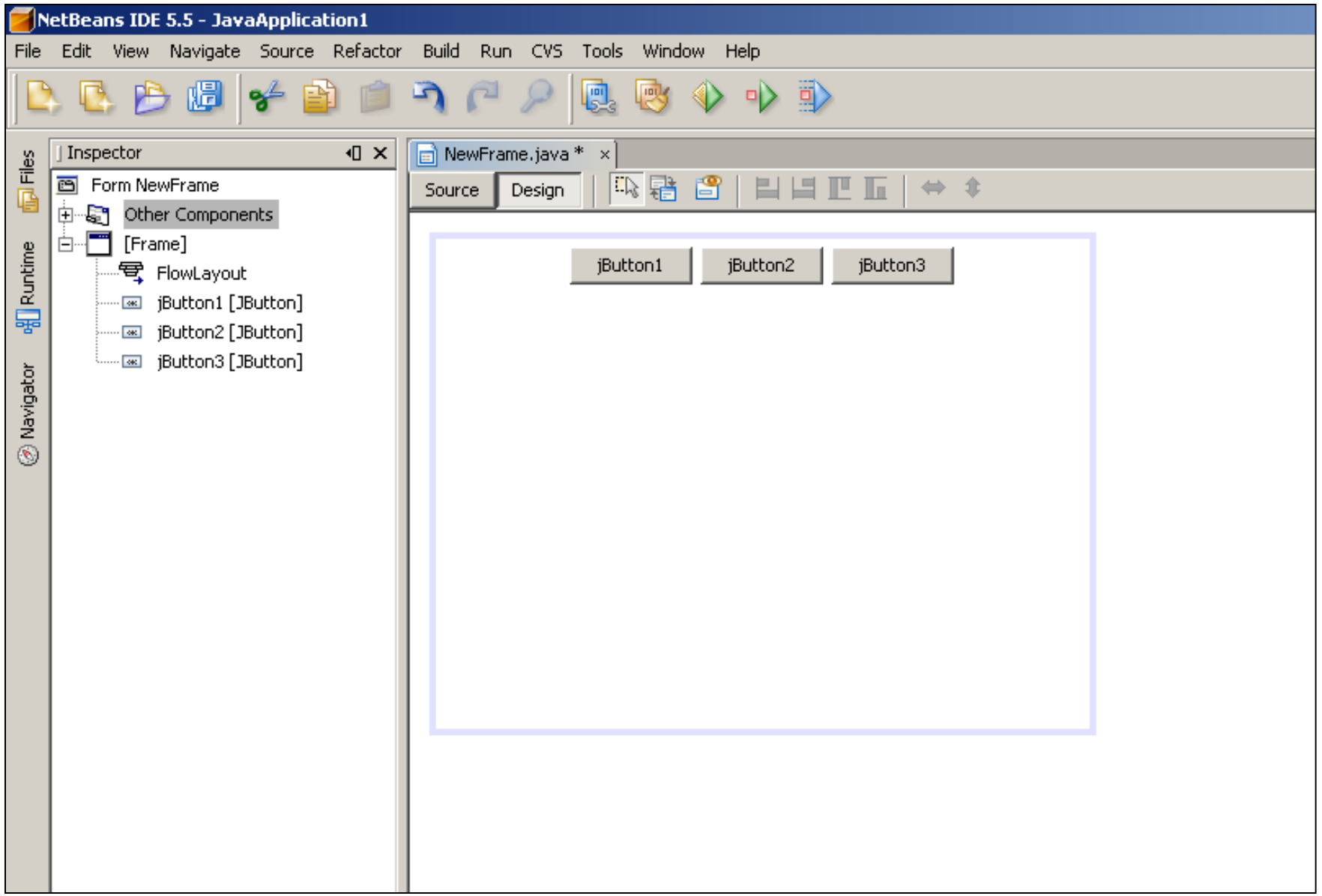
Form NewFrame
Other Components
[Frame]
BorderLayout
jButton1 [JButton]
jButton2 [JButton]
jButton3 [JButton]

NewFrame.java *

Source Design

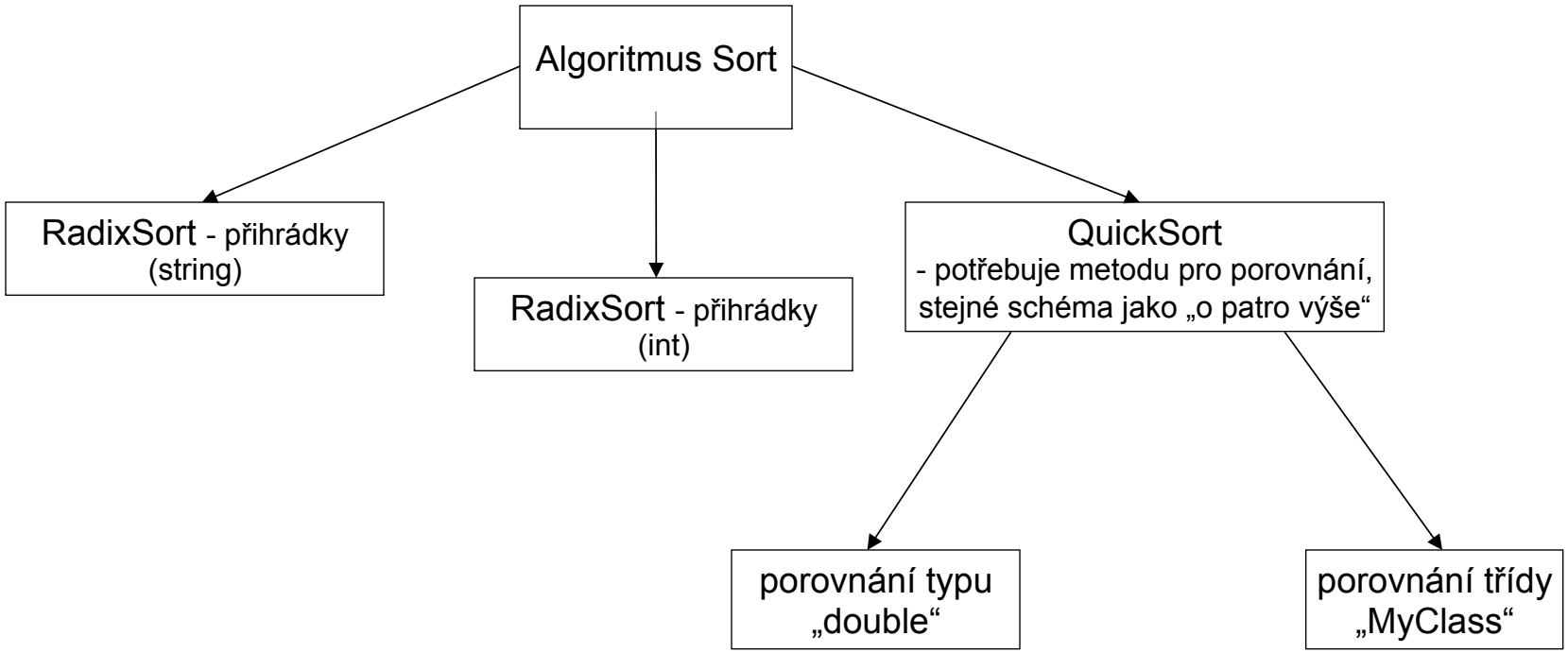
jButton2
jButton1
jButton3

Příklad: Layout managers – AWT (Flow Layout)



Příklad: Třídění záznamů

- Různé algoritmy třídění záznamů dle typu klíče
  - Celočíselné hodnoty a řetězce je možno třídit rychle pomocí přihrádek, ostatní typy je nutno třídit algoritmem založeným na porovnávání



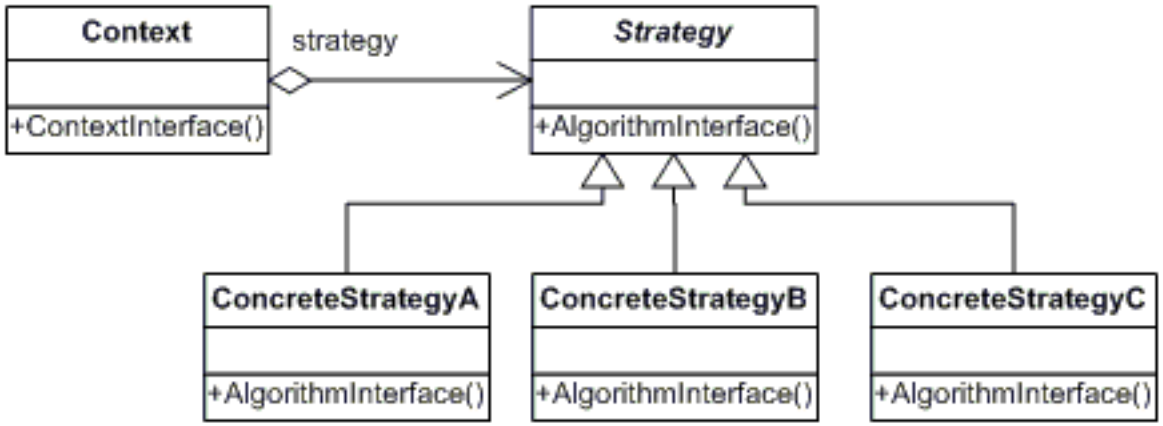
Soutěž

- **Co mají uvedené příklady společného?**

Strategy - použitelnost

- **Použitelnost**
  - pro více souvisejících tříd lišících se pouze chováním
    - v klientské třídě můžeme volit jedno z mnoha chování
  - pokud je výhodné volit z více variant algoritmů
    - např. umožní reflektovat časovou a prostorovou náročnost
  - pro algoritmy používající data, o kterých by klient neměl vědět
    - skrývá datové struktury specifické pro algoritmus

Strategy – struktura



- **Účastníci**
  - Context
    - obsahuje ukazatel na abstraktní strategii
    - chování je dáno konkrétní strategií
  - Strategy
    - abstraktní třída, definuje rozhraní společné všem algoritmům
    - virtuální metoda
  - ConcreteStrategy
    - specifická implementace virtuální metody, čili konkrétního algoritmu

Strategy – předávání parametrů strategii

- **implementace**
  - sdílení dat mezi kontextem a strategií:
    - předání kontextových dat v parametrech metody
      - mohou se předávat data, které daná strategie nepotřebuje
    - předání reference na kontext strategii
      - strategie přistupuje jen k datům, které potřebuje, kontext musí se strategií více spolupracovat
    - obecný kontext – v metodě kontextu se předá ukazatel na strukturu s parametry
      - konkrétní strategie musí ukazatel přetypovat, aby uměla s daty pracovat – náročnější na pozornost programátora

Strategy – předání strategie přes šablonu

- **strategie jako parametr šablony**
  - použitelné, pokud je strategie známa již v době kompilace a není ji potřeba měnit za běhu programu
  - není třeba definovat abstraktní třídu pro strategie
  - strategie je spojena s kontextem staticky – vyšší efektivita

```
template <class AStrategy>
class Context {
  void Operation() {
    theStrategy.DoAlgorithm();
  }
private:
  AStrategy theStrategy;
};
```

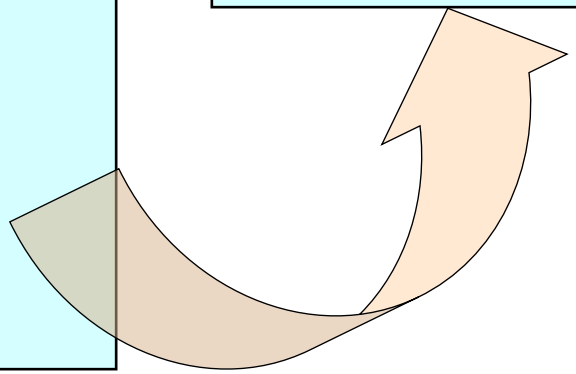
```
class MyStrategy {
public:
  void DoAlgorithm();
};

Context<MyStrategy> aContext;
```

Strategy – souvislosti závěrem

- Zapouzdřuje rodiny souvisejicích algoritmů
- Eliminuje nutnost podmiňovacích výrazů

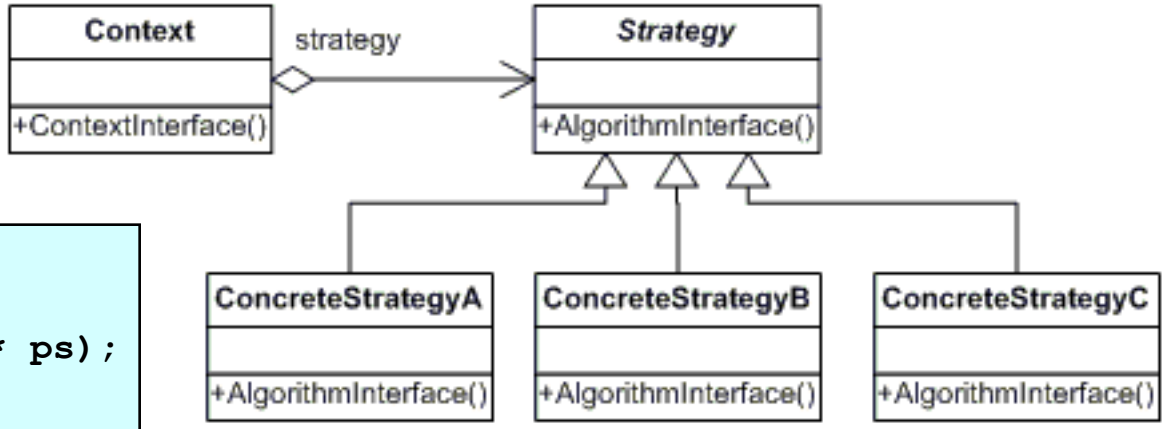
```
void Cotext::Algorithm () {
  switch (_strategy) {
   case STRICT:
     executeStrictAlgorithm();
     break;
   case RELAXED:
     executeRelaxedAlgorithm();
     break;
   // ...
  }
}
```

```
void Context::Algorithm () {
  _aStrategy->Algorithm();
}
```



- Výběr z implementací
- Klient musí být obeznámen s druhy Strategií
- Komunikační overhead mezi Strategií a Contextem
- Zvýšený počet objektů

Strategy – „All-in-1“ na konec




```
class Context {
public:
  Context(AStrategy* s, void* ps);
  void Algorithm();
private:
  AStrategy* strategy;
  void* ParamStruct;
};
```

```
class AStrategy {
public:
  virtual void Algorithm(void* ParamStruct)=0;
protected:
  AStrategy();
};
```

```
class SpecificStrategy: public AStrategy {
public:
  virtual void Algorithm(void* ParamStruct);
  SpecificStrategy();
};
```