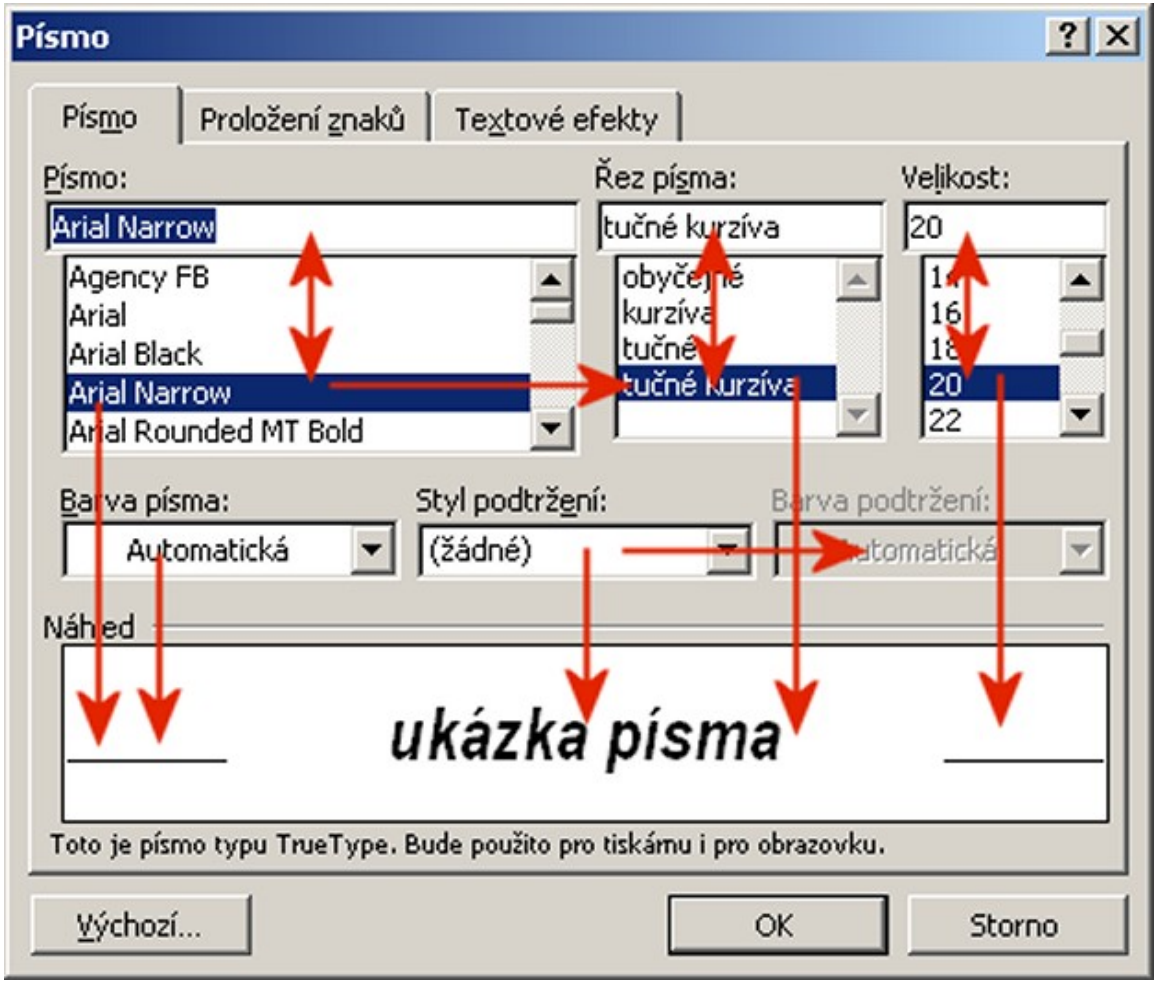
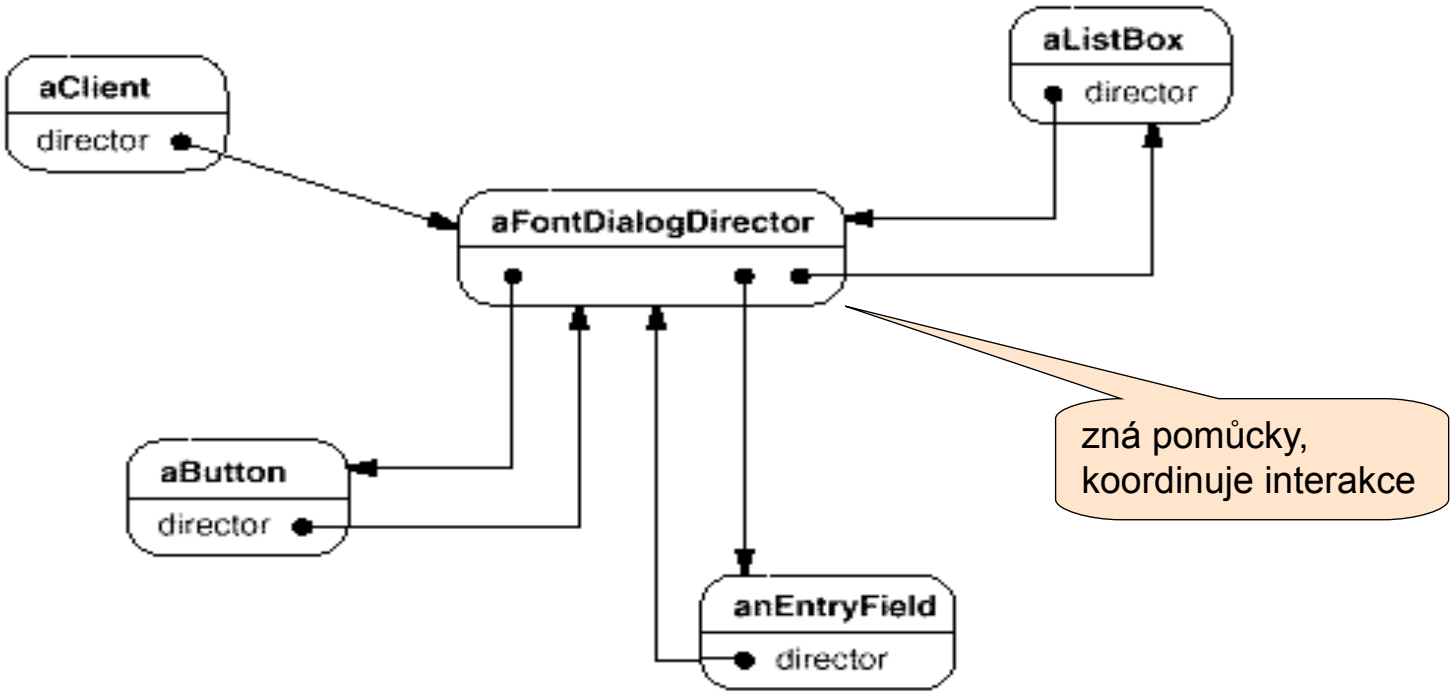
Mediator

Mediator – motivace

- FontDialog
  - závislosti mezi jednotlivými ovládacími prvky jsou netriviální

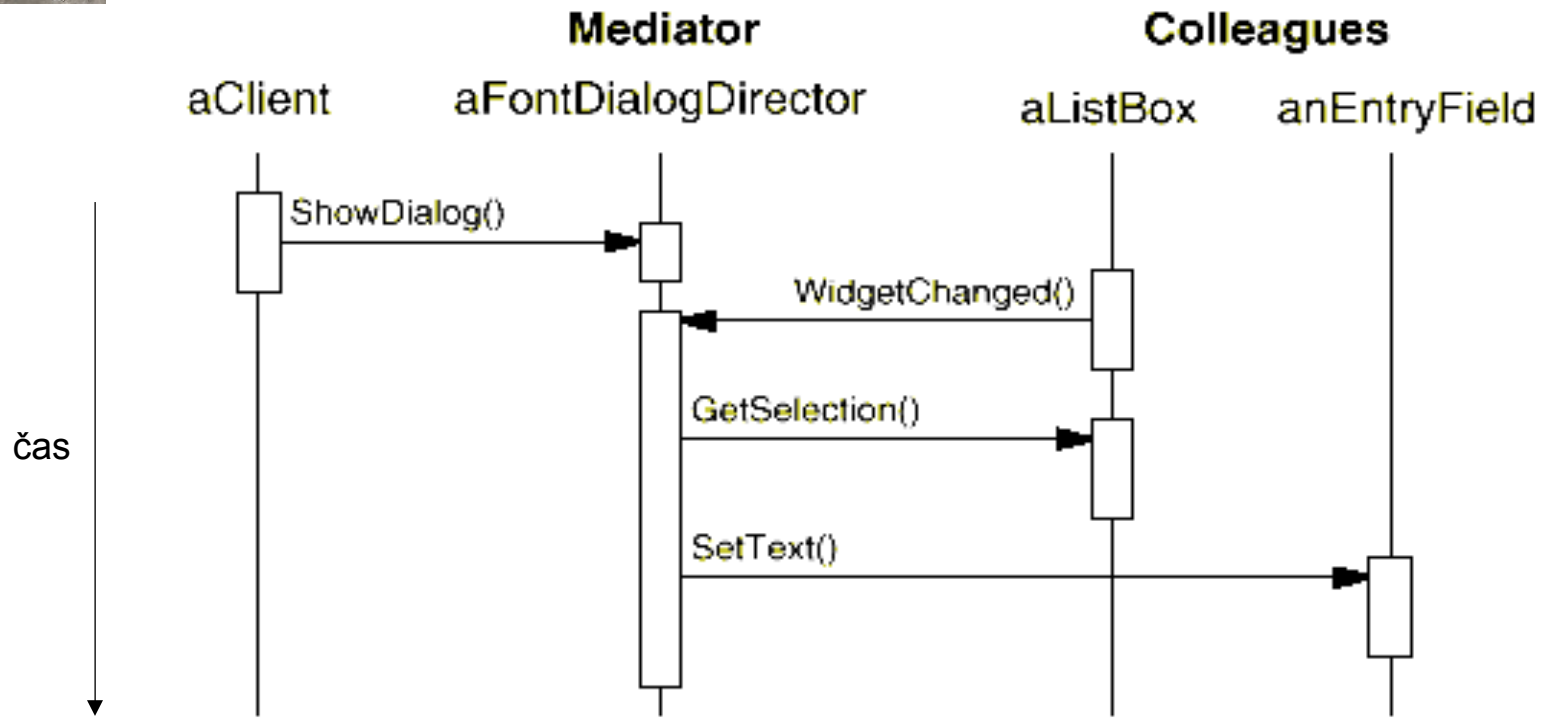


Mediator - motivace



- místo distribuce chování do jednotlivých pomůcek (widgetů) ho soustředíme do jednoho objektu - mediátora

Mediator - chování

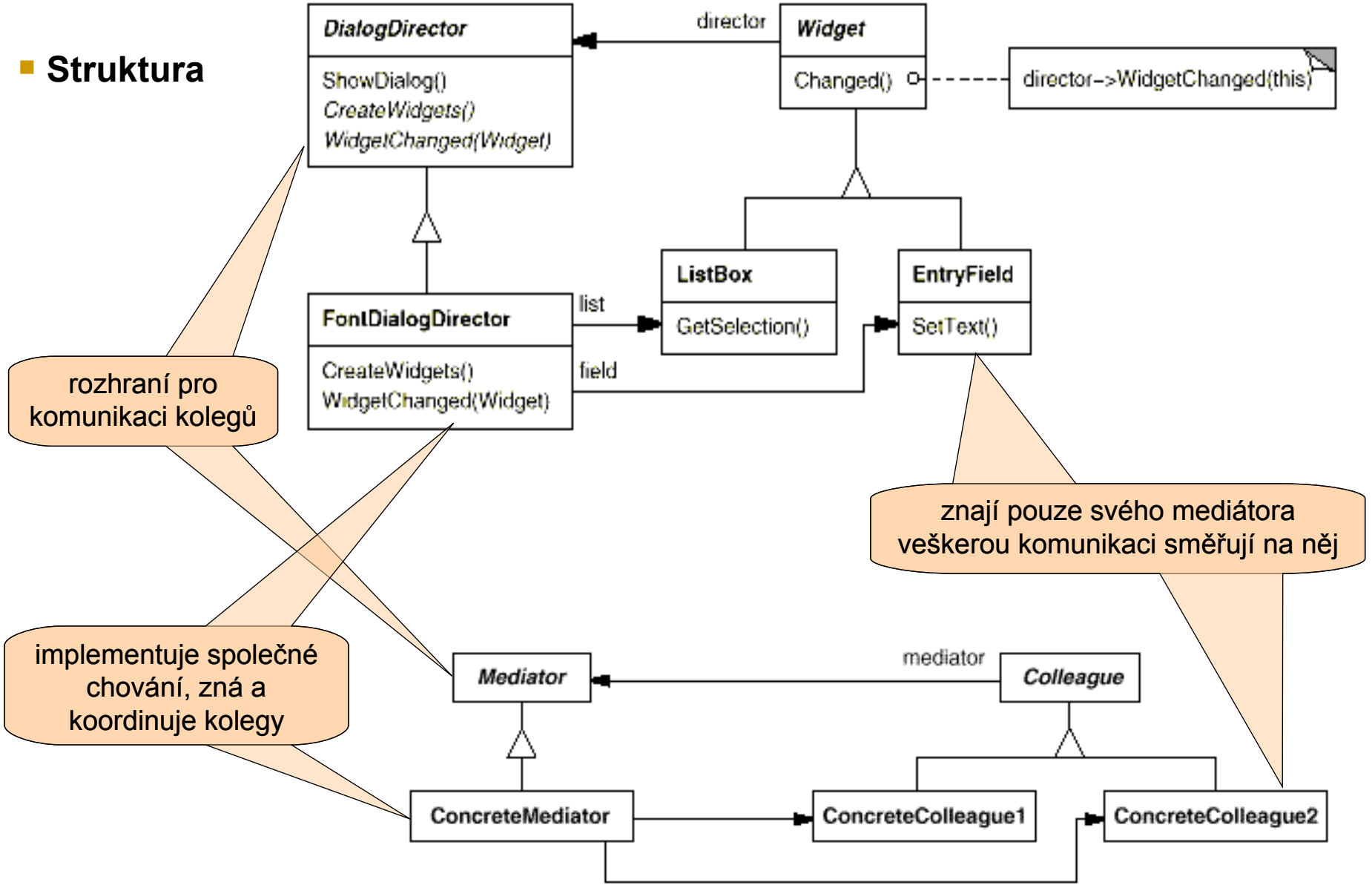


- uživatel po zobrazení dialogu změní výběr listboxu
- po ohlášení změny mediátor zjistí aktuální položku a předá jí do vstupního políčka

- **listbox a entry field o sobě vůbec neví**
  - o tom, že se má po změně někam něco vyplnit, se rozhoduje v mediátoru
- **interakce zprostředkovaně přes jeden objekt**
  - jednodušší změna chování

Mediator - struktura

- **Struktura**



Mediator

- **Účel**
  - prostředník
  - zapouzdřuje chování systému objektů
    - uvolňuje (ruší) přímou vazbu mezi komponentami v rámci systému
    - umožňuje měnit interakci objektů nezávisle na nich
- **Motivace**
  - Obecně
    - OOP - distribuce chování → velká propojenost jednotlivých objektů
    - propojenost → objekt většinou nemůže pracovat bez ostatních
      - i systém rozdělený na mnoho částí se chová jako monolit
    - změnit chování distribuované do mnoha (stovek) objektů obtížné
  - Konkrétně – GUI – dialogy
    - dialog je složen z mnoha objektíků - *tlačítka, menu, listboxy …*
    - chování součástí spolu úzce souvisí
      - tlačítko OK začne fungovat až vyplnění povinné položky
      - vybereme-li položku listboxu, nějaké vstupní políčko se předvyplní atp.
    - rozdílné dialogy - typicky rozdílné závislosti
      - jiné položky jsou povinné, listbox nemusí nic vyplnit…
      - nemůžeme jednoduše používat součásti pro víc dialogů
      - museli bychom např. podědit a měnit chování – složité

Mediator - použitelnost

- **Příklady typického použití**
  - skupina objektů má komunikovat definovaným, ale složitým způsobem
    - závislosti mezi objekty jsou složitě strukturované nebo těžko pochopitelné
  - znovupoužití objektu je obtížné jelikož spolupracuje s mnoha jinými objekty
  - chování distribuované mezi mnoho objektů bylo mělo být upravitelné bez nutnosti vytváření mnoha podtříd

Mediator - důsledky

- **Výhody a omezení**
  - omezuje odvozování mnoha podtříd
    - původně distribuované chování soustřeďuje na jedno místo
    - ostatní objekty je možné použít bez nutnosti úprav
  - ruší vazby 1:1 mezi jednotlivými kolegy
    - kolegové o sobě navzájem nevědí
    - každý může být použitý sám o sobě
  - zjednodušuje ‚protokol‘
    - n:m interakce nahrazuje 1:m, které jsou jednodušší k pochopení i rozšiřování
  - zabstraktňuje způsob spolupráce objektů
    - odděluje kooperaci mezi objekty od jejich samostatného chování
  - centralizuje řízení
    - za jednoduchost komunikace jsme zaplatili jedním extrémně složitým objektem
    - rozbili jsme monoliticky se tvářící systém, vytvořili jsme monoliticky se tvářící monolit

Mediator - implementace

- **Implementace**
  - varianty
    - abstraktní mediátor
      - kolegové pracují s několika funkčně různými mediátory
      - umožňuje měnit chování systému definicí nového potomka
    - přímo 1 konkrétní mediátor
      - kolegové pracují jen s ním – abstraktní zbytečný
      - přehledná komunikace, jasný protokol
  - komunikace kolega-mediátor
    - při ‚zajímavé' události
    - mediátor propaguje událost k dalším kolegům
  - implementace komunikace
    - speciální notifikační interface mediátora – viz dále
    - pomocí patternu Observer
      - kolegové vystupují jako subjekty, kdykoliv změní stav, upozorní observera-mediátora

Mediator - implementace

interface pro oznámení ‚zajímavých' událostí

```
class DialogDirector {
public:
    virtual ~DialogDirector();
    virtual void ShowDialog();
    virtual void WidgetChanged(Widget*) = 0;

protected:
    DialogDirector();
    virtual void CreateWidgets() = 0;
};
```

```
class Widget {
public:
    Widget(DialogDirector*);
    virtual void Changed();

    virtual void HandleMouse(MouseEvent& event);
    // ...
private:
    DialogDirector* _director;
};
```

```
void Widget::Changed() {
  _director->WidgetChanged(this);
}
```

každý kolega zná svého mediátora

Mediator - implementace

```
class EntryField : public Widget {
public:
   EntryField(DialogDirector*);

   virtual void SetText(const string text);
   virtual string GetText();
   virtual void HandleMouse(MouseEvent& event);
   // ...
};
```

jednotlivé widgety pro použití v dialozích

```
class ListBox : public Widget {
public:
   ListBox(DialogDirector*);

   virtual string GetSelection();
   virtual void SetList(List<string>& listItems);
   virtual void HandleMouse(MouseEvent& event);
   // ...
};
```

```
class Button : public Widget {
public:
   Button(DialogDirector*);

   virtual void SetText(string text);
   virtual void HandleMouse(MouseEvent& event);
   // ...
};
```

```
void Button::HandleMouse (...)
{
   // ...
   Changed();
}
```

Mediator - implementace

- **konkrétní chování vznikne jako potomek abstraktního mediátora**
- **může definovat**
  - podobu dialogu
  - chování celého systému
  - kooperaci mezi jednotlivými pomůckami

```
class FontDialogDirector : public DialogDirector {
public:
  FontDialogDirector();
  virtual ~FontDialogDirector();
  virtual void WidgetChanged(Widget*);

protected:
  virtual void CreateWidgets();

private:
  Button* _ok;
  Button* _cancel;
  ListBox* _fontList;
  EntryField* _fontName;
};
```

vytvoří svůj typ dialogu

mediátor si drží všechny kolegy

Mediator - implementace

- metoda CreateWidgets vytváří dialog, pro který je konkrétní mediátor napsán

```
void FontDialogDirector::CreateWidgets() {
  _ok = new Button(this);
  _cancel = new Button(this);
  _fontList = new ListBox(this);
  _fontName = new EntryField(this);

  // fill the listBox with font names
  // assemble the widgets in the dialog
}
```

každý kolega musí znát mediátora

- **metoda WidgetChanged definuje chování systému**
  - upozorní i ostatní související pomůcky, aby se systém choval korektně

```
void FontDialogDirector::WidgetChanged( Widget* changedWidget)
{
  if (changedWidget == _fontList) {
    _fontName->SetText(_fontList->GetSelection());
  } else if (changedWidget == _ok) {
      // apply font change and dismiss dialog
      // ...
  } else if (changedWidget == _cancel) {
      // dismiss dialog
  }
}
```

kolegové při volání zadají sami sebe jako parametr aby je mediátor poznal

Mediator – použití

- **Možná použití – počítačový svět**
  - chatovací server
    - konkrétní mediátor – chatovací room
    - kolegové – jednotliví lidé (resp. jejich klienti)
  - dům budoucnosti – ‚smart home‘
    - vše řízeno počítačem - mediátor
    - mnoho různorodých zařízení
      - lednice, která ví, co koupit
      - klimatizace, topení, různá teplotní či vlhkostní čidla
      - alarm, zamykání dveří, ...
    - potřeba koordinace
      - když spustí alarm, zavolá se policie, pošle se sms majiteli
      - dojde-li mléko, je potřeba ho objednat v e-shopu
  - ze života: letištní věž

Mediator – související NV

- **Související NV**
  - Facade
    - fasáda jen abstrahuje skupinu objektů do jednoho rozhraní
      - jednotlivé objekty o ní neví
    - součásti abstrahované skupiny typicky samostatné, žádná složitá kooperace
  - Observer
    - kolegové mohou komunikovat s mediátorem pomocí NV Observer
    - dynamický