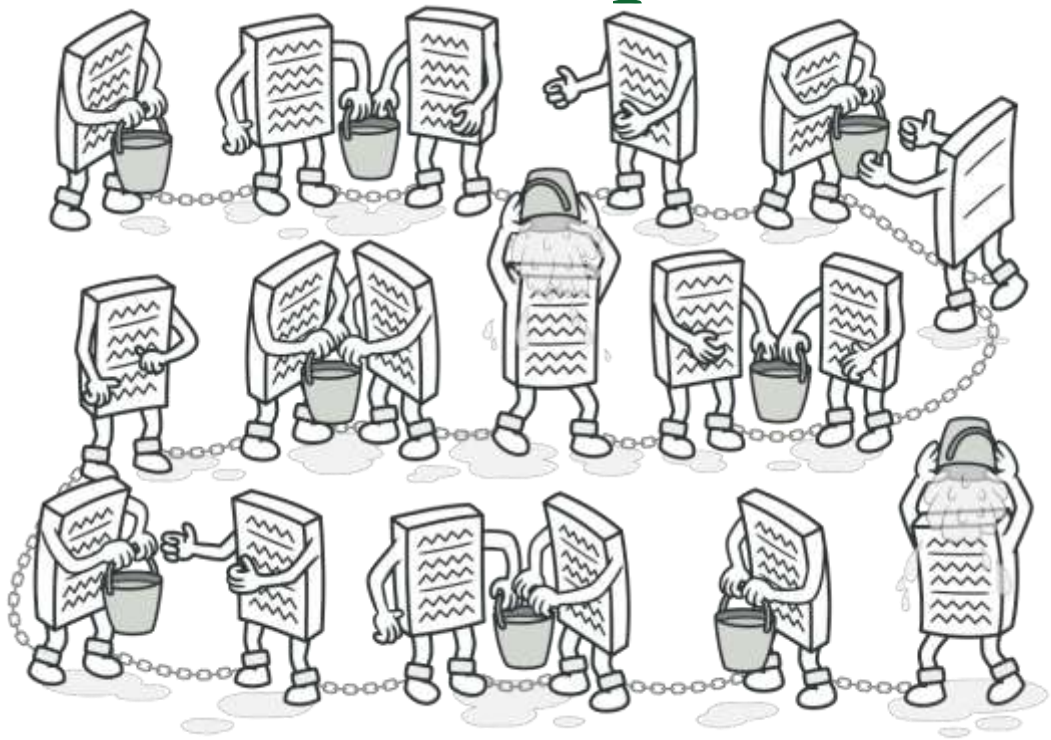
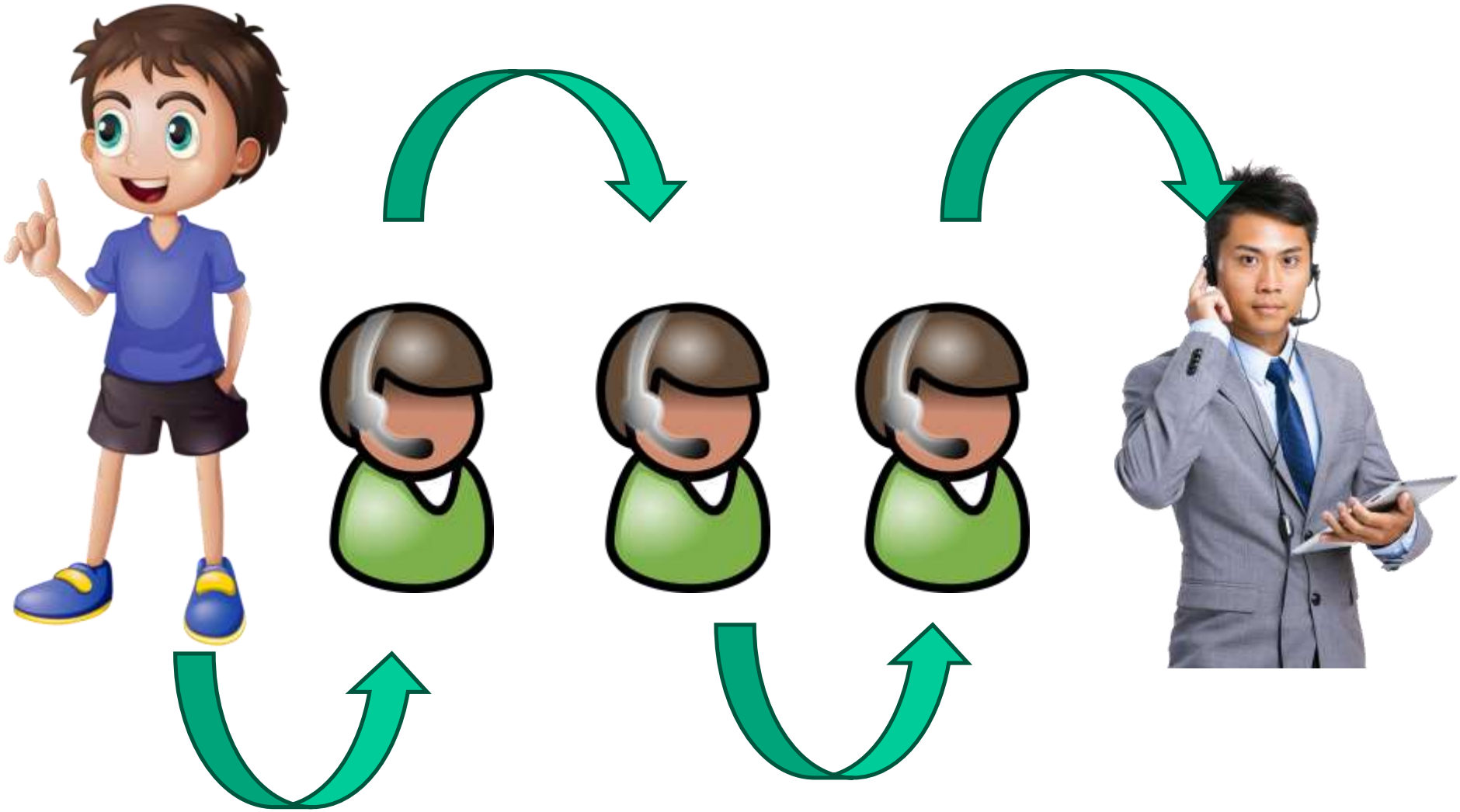
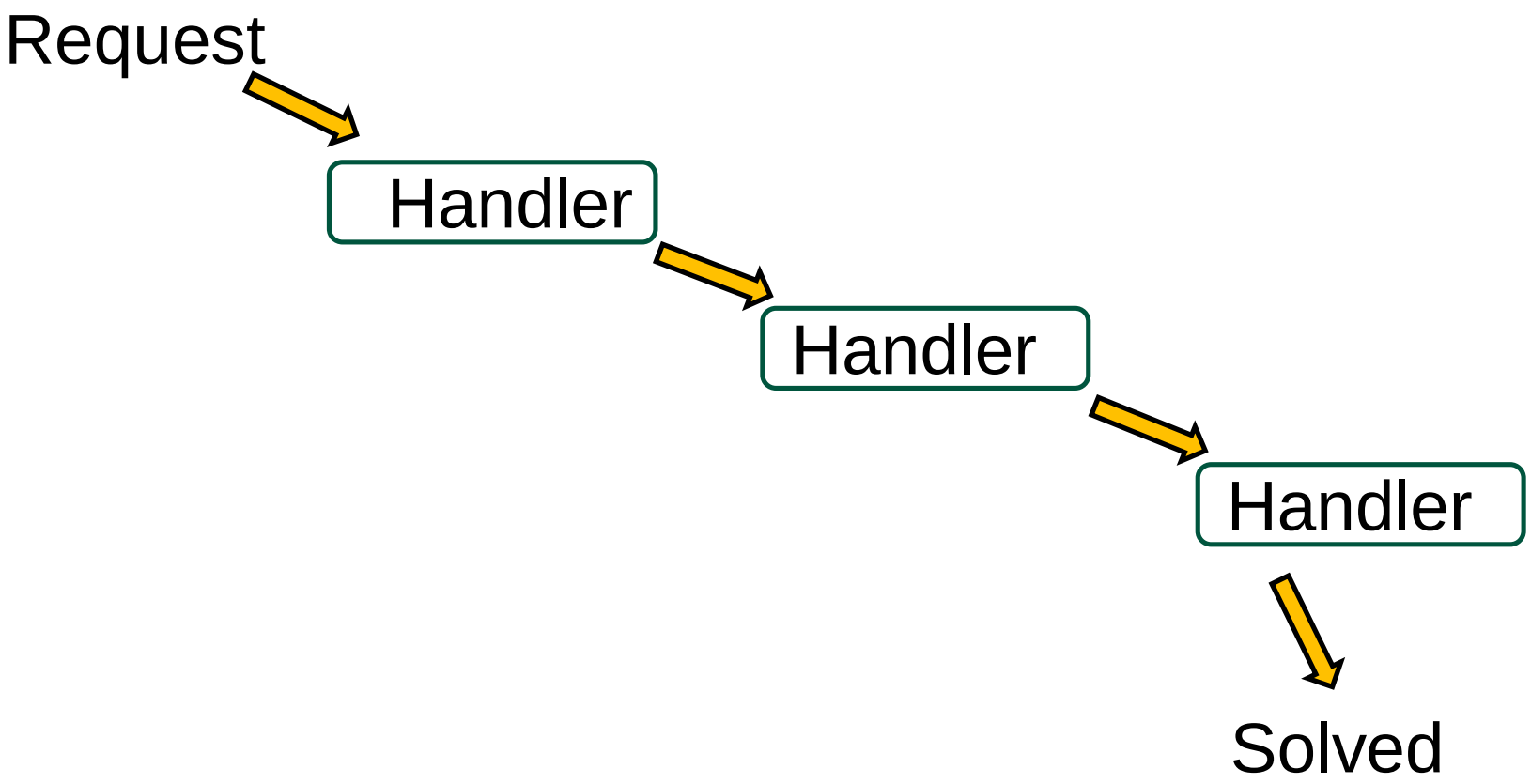
Chain of Responsibility



Martin Lejko

Příklad z života




Request
Handler
Handler
Handler
Solved

Chain of responsibility – motivační příklad

```
void Main() {
    // select code in editor tab
    var ide = new IDE(null);
    var editor = new CodeEditor(ide);
    var codeSelection = new CodeSelection(editor);

    codeSelection.HandleKey("Ctrl+F");
    codeSelection.HandleKey("Alt+F4");
}

interface IKeyHandler {
    void HandleKey(string key);
}

class IDE : IKeyHandler {
    IKeyHandler _handler;

    public IDE(IKeyHandler handler) => _handler = handler;

    public void HandleKey(string key) {
        if (key == "Ctrl+F") {
            "Full Search".Dump();
        }
        else if (key == "Alt+F4") {
            "Close Application?".Dump();
        } else {
            _handler?.HandleKey(key);
        }
    }
}
```

Chain of responsibility – motivační příklad

```
class CodeEditor : IKeyHandler {
    IKeyHandler _handler;

    public CodeEditor(IKeyHandler handler) => _handler = handler;

    public void HandleKey(string key) {
        if (key == "Ctrl+F")  {
            "Local Search".Dump();
        } else {
            _handler?.HandleKey(key);
        }
    }
}

class CodeSelection : IKeyHandler {
    IKeyHandler _handler;

    public CodeSelection(IKeyHandler handler) => _handler = handler;

    public void HandleKey(string key) {
        if (key == "Ctrl+F")  {
            "Selection Search".Dump();
        } else {
            _handler?.HandleKey(key);
        }
    }
}
```

Chain of responsibility – motivační příklad

- **Možné řešení**
  - jeden objekt zodpovědný za veškerou identifikaci mincí

```
class Coin {
public float Weight;
public float Diameter;

public Coin( float weight, float diameter) {
    Weight = weight; Diameter = diameter;
    }
}
```

```
class CoinHandler {
public void Handle(Coin coin) {
    if ( Math.Abs( coin.Weight - 3.6) < 0.02) &&
        Math.Abs( coin.Diameter - 20) < 0.1) {
        // Zpracuj 1 Kč
    }
    else if (...) {
    ...
}
}
```

Chain of responsibility – implementace

```
abstract class CoinHandlerBase
{
    public abstract bool HandleCoin( Coin coin);
}
```

```
class OneCrownHandler : CoinHandlerBase
{
    public override bool HandleCoin( Coin coin)
    {
        if( Abs( coin.Weight – 3.25) < 0.02) &&
            Abs( coin.Diameter – 18) < 0.1)
        {
            // Zpracování mince
            return true;
        }
        else
            return false;
    }
}
class TwoCrownsHandler : CoinHandlerBase { … }
class FiveCrownsHandler : CoinHandlerBase { … }
class TenCrownsHandler : CoinHandlerBase { … }
class TwentyCrownsHandler : CoinHandlerBase { … }
class FiftyCrownsHandler : CoinHandlerBase { … }
```

podmínka

zpracování

rozdíl
v podmínce
a zpracování

Chain of responsibility – implementace 2

```
CoinHandlerBase[] handlers = {
  new OneCrownHandler(),
  new TwoCrownHandler(),
  …
};

Coin one = new Coin(3.25, 18);
for (int j = 0; j < handlers.length; ++i) {
  if (handlers[j].HandleCoin(coin))
          break;
}
```

nutná znalost interní struktury

Chain of responsibility – implementace 3

```
abstract class CoinHandlerBase
{
    protected CoinHandlerBase successor_;

    public abstract void HandleCoin( Coin coin);

    public void SetSuccessor( CoinHandlerBase successor){
        successor_ = successor;
    }
}
```

```
class OneCrownHandler : CoinHandlerBase
{
    public override void HandleCoin( Coin coin)
    {
        if( podmínka )
            // Zpracování mince
        else if( successor_ != null )
            successor_.HandleCoin( coin);
    }
}
class TwoCrownsHandler : CoinHandlerBase { … }
class FiveCrownsHandler : CoinHandlerBase { … }
class TenCrownsHandler : CoinHandlerBase { … }
class TwentyCrownsHandler : CoinHandlerBase { … }
class FiftyCrownsHandler : CoinHandlerBase { … }
```

zřetězení

rozdíl
v podmínce
a zpracování

Chain of responsibility – implementace 3

```
CoinHandlerBase c1 = new OneCrownHandler();
CoinHandlerBase c2 = new TwoCrownsHandler();
CoinHandlerBase c5 = new FiveCrownsHandler();
CoinHandlerBase c10 = new TenCrownsHandler();
CoinHandlerBase c20 = new TwentyCrownsHandler();
CoinHandlerBase c50 = new FiftyCrownsHandler();
c1.SetSuccessor(c2);
c2.SetSuccessor(c5);
c5.SetSuccessor(c10);
c10.SetSuccessor(c20);
c20.SetSuccessor(c50);

Coin one = new Coin( diameter = 24.47, weight = 6.5 );
Coin ten = new Coin( diameter = 27.24, weight = 8.1 );
Coin unknown = new Coin( diameter = 42.89, weight = 1.0 );

CoinHandlerBase handler = c1;

handler.HandleCoin(one);
handler.HandleCoin(ten);
```

Vytvoření handlerů pro mince

Nastavení pořadí zpracování

Mince k použití

Vstupní bod

Chain of responsibility – unknown

- **řešení - default handler**

```
class UnknownHandler : CoinHandlerBase
{
    public override void HandleCoin( Coin coin)
    {
        // Zpracování neznáme mince
    }
}
```

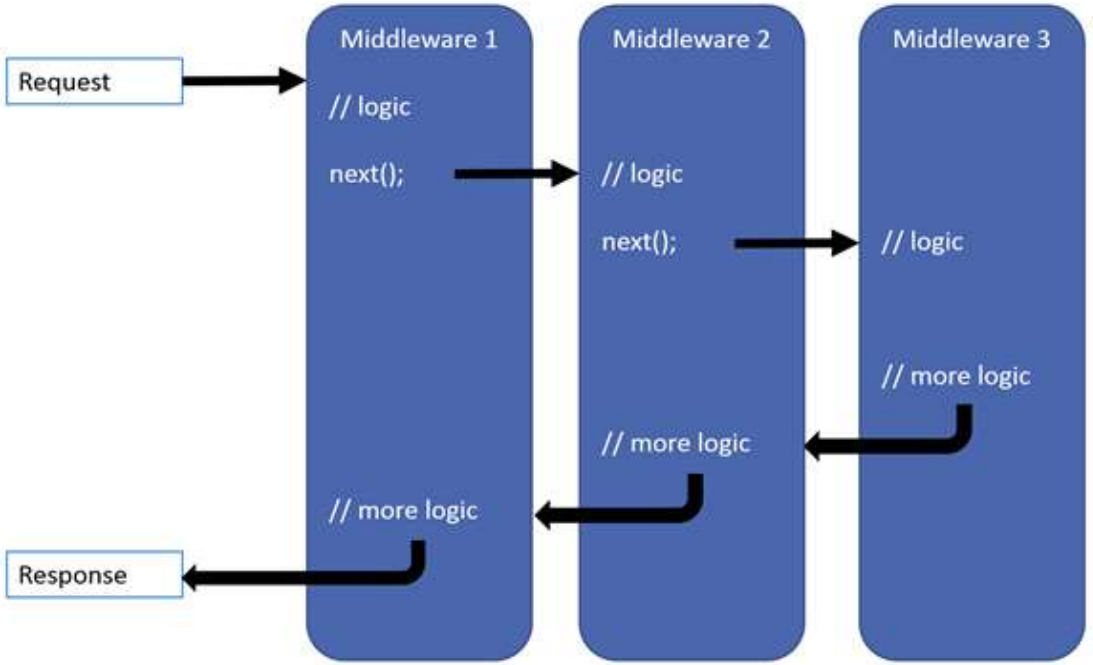
vytvoření handleru pro neznámé mince

```
...
CoinHandlerBase c50 = new FiftyCrownsHandler();
CoinHandlerBase unknownHandler = new UnknownHandler();
...
c50.SetSuccessor(unknownHandler);
...
Coin unknown = new Coin( diameter = 42.89, weight = 1.0 );
...
handler.HandleCoin(unknown);
```

neznámá mince UnknownHandler

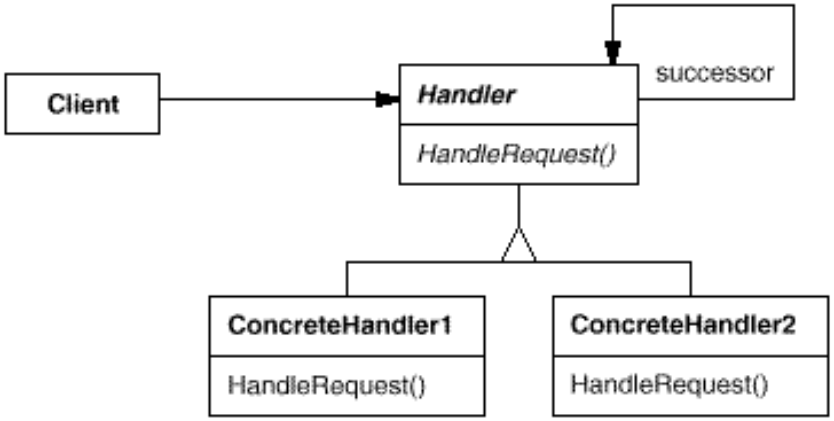
Chain of responsibility – ASP.NET

- **Framework pro vytváření webových aplikací a služeb v C#**
  - defaultní server přijímá HTTP requesty
  - předává předdefinovanému middleware (článek řetězce)
  - Ten ho:
    - 1. zpracuje a nebo ne
    - 2. pošle dál a nebo ne
    - 3. zpracuje a nebo ne

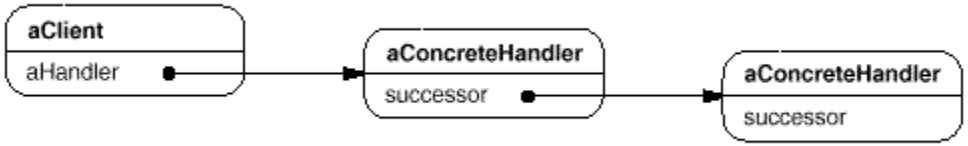


Chain of responsibility – struktura

- **Struktura:**



- **Účastníci:**
  - **Handler**
    - definuje rozhraní pro zpracování požadavků
    - volitelně implementuje ukazatel na následníka
  - **ConcreteHandler**
    - zachytává požadavek, který umí zpracovat
    - může přistupovat na svého následníka
  - **Client**
    - vyvolává požadavek předáním na objekt ConcreteHandler zapojený do řetězu
- **Implementace:**



Chain of responsibility – použitelnost

- **Použitelnost**
  - chceme zaslat požadavek a nevíme, kdo ho zpracuje, nebo nás to nezajímá
    - důležitý je výsledek
  - pokud více než jeden příjemce může přijmout požadavek a není *apriori* známo který
- **Výhody**
  - odděluje odesílatele od příjemců
  - zjednodušuje odesílatele (klienta)
    - nemusí se starat o strukturu příjemců
    - neobsahuje reference na všechny možné příjemce
  - možnost měnit řetěz dynamicky za běhu programu
- **Nevýhody**
  - není zaručeno, že nějaký příjemce zprávu přijme
    - počítání mincí bez UnknownHandler
    - implementace - default handler

Chain of responsibility – známé použití

- **Známé použití**
  - okenní systémy
    - běžně používáno pro zpracování událostí jako kliknutí myši, stisk klávesy
  - Frameworky pro tvorbu webových stránek
    - ASP.NET
  - síťové/distribuované systémy
    - v řetězu zapojena množina serverů nabízejících služby
    - klient předá požadavek libovolnému serveru
    - servery si mezi sebou posílají požadavek dokud jej některý nezpracuje

Chain of Resposibility – související NV

- **Související NV**
  - Composite
    - je-li řetěz objektů využívaný součástí rozsáhlejší stromové struktury, může být tato struktura tvořena pomocí Composite

MOTIVACE

- Programujeme systém evidence objednávek
- Kolik každá objednávka stojí?

2. Řešení pomocí Composite vzoru
- Interface Item
- Rekurzivní getPrice()
- Stačí rozbalit jen jednu krabici