Metaprogramovanie

Aspektovo-orientované programovanie

Sergej Chodarev

9

Separation of Concerns

Objektovo-orientované programovanie

- Dekompozícia systému na objekty
- Zapuzdrenie (*encapsulation*)

Príklad

Záujem 1: Systém má udržiavať informáciu o cene produktov.

```
public abstract class Product {

    private BigDecimal price;

    ...

    public void setPrice(BigDecimal p) {
        price = p;
    }

    public BigDecimal getPrice() {
        return price;
    }
}
```

Viac záujmov

Záujem 2: Všetky zmeny ceny musia byť zaznamenávané.

```
public class Logger {

    private Writer writer;

    Logger() {
        //open log file
    }

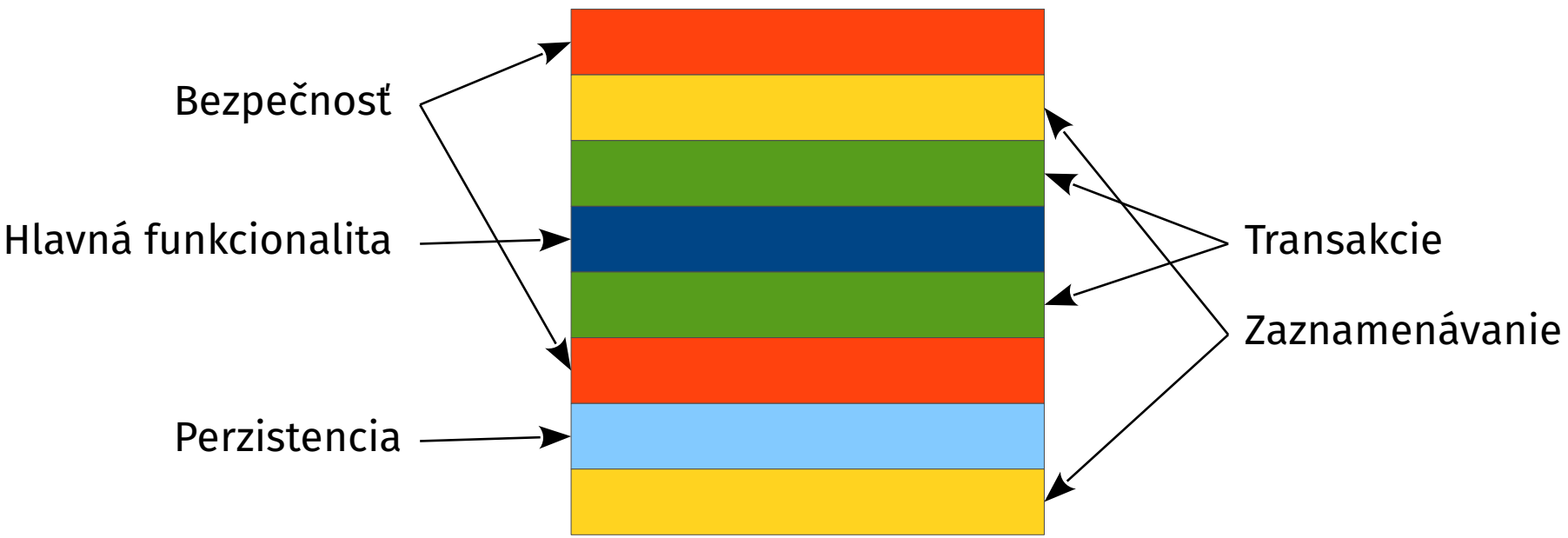
    void writeLog(String value) {
        //write value to log file
    }

}
```

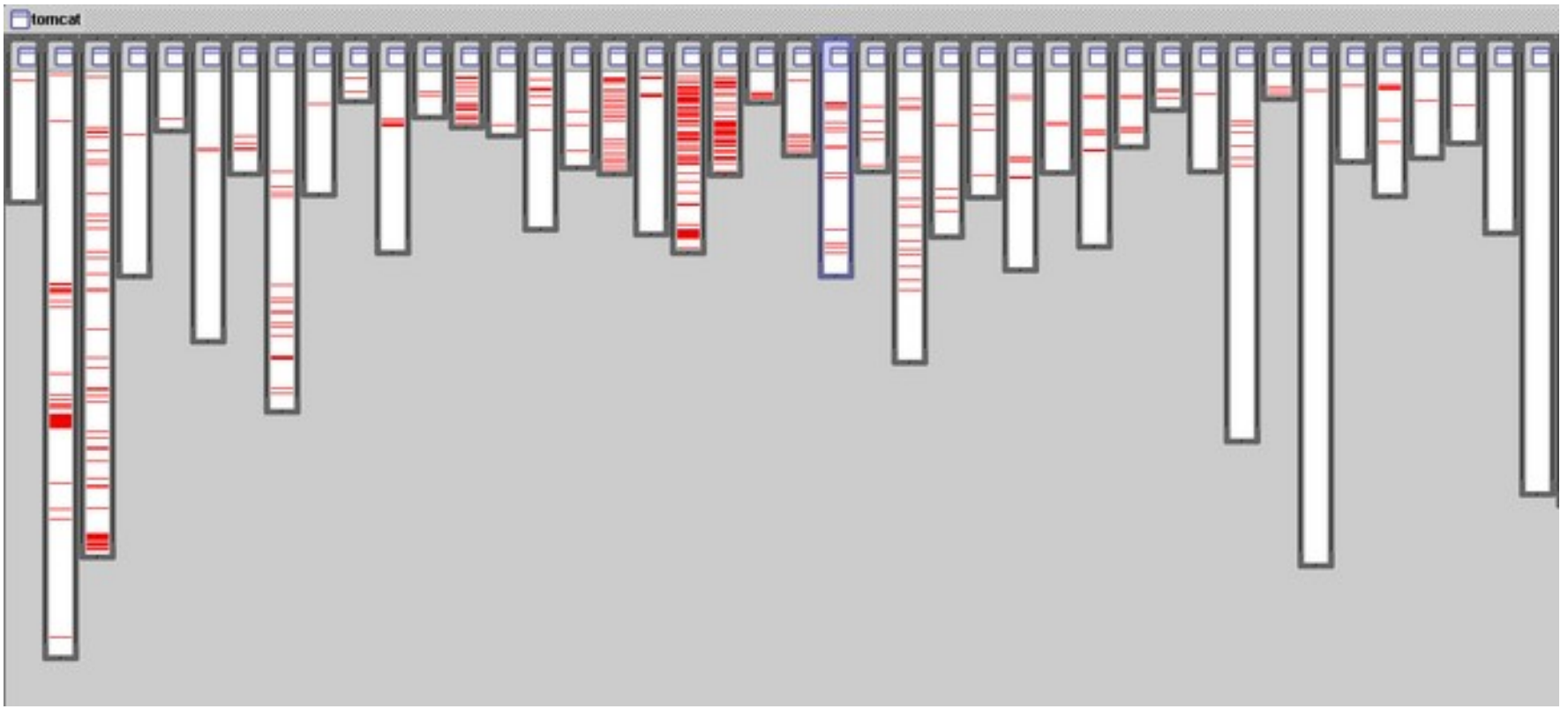
Prepletanie (*tangling*)

```
public abstract class Product {
    private BigDecimal price;
    private Logger logger;
    ...
    public void setPrice(BigDecimal p) {
        logger.writeLog("Changed " + price + " to " + p);
        price = p;
    }
    ...
}
```

Prepletanie (*tangling*)



Rozptýlenie (*scattering*)



Logging in org.apache.tomcat

Rozptýlenie (*scattering*)

- Duplikácia blokov kódu
- Rôzne časti implementácie záujmu v rôznych moduloch

Pretínajúce záujmy

- *Crosscutting concerns*
- Ortogonálne k dekompozícii na moduly

Tyrania dominantnej dekompozície

Riešenia

- Aplikačné rámce
- Reflexia
- Generovanie kódu
- Modifikácia kódu

- Návrhové vzory
  - observer
  - chain of responsibility
  - decorator
  - proxy

Aspektovo-orientované programovanie

Aspektovo-orientované programovanie

- Komponenty – základná dekompozícia podporovaná jazykom
- Aspekty – zapuzdrenie pretínajúcich záujmov
- 1997 — Gregor Kiczales (Xerox PARC)
- 2001 — AspectJ

Článok

- G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin, “**Aspect-oriented programming**,” ECOOP’97

Logger ako aspekt

```
public aspect LoggingAspect {

    private Logger logger;

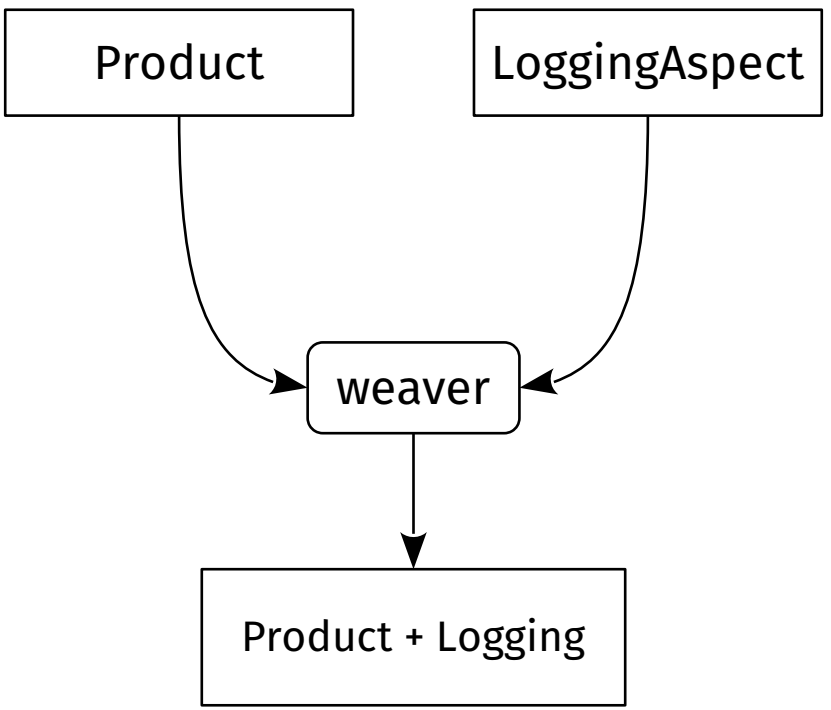
    public LoggingAspect() { logger = new Logger(); }

    before(Product product, BigDecimal price)
            : execution(* Product.setPrice(..))
              && this(product) && args(price) {

        logger.writeLog("Changed " + product.getPrice()
                        + " to " +price);

    }
}
```

Pretkávanie (*weaving*)



AspectJ

AspectJ

- https://eclipse.org/aspectj/
- AOP pre Java
- Jazyk pre definíciu pravidiel pretkávania
  - Rozšírenie triedy jazyka Java
  - **aspect** namiesto **class**
  - **.aj** namiesto **.java**
- Pretkávač (*weaver*)

AspectJ z rýchlika

Pravidlá pretkávania

- Bod spájania (*joinpoint*)
  - bod v procese vykonávania programu
  - volanie/vykonávanie metódy, prístup k členskej premennej, ...
- Bodový prierez (*pointcut*)
  - výraz pre výber množiny bodov spájania
- Odporučenie (*advice*)
  - funkcionalita vykonávaná vo vybraných bodoch spájania

Aspekt bezpečnosti

```
public aspect SecurityAspect {

    private Authenticator authenticator = new Authenticator();

    pointcut secureAccess()
        : execution(* MessageCommunicator.deliver(..));

    before() : secureAccess() {
        System.out.println("Checking and authenticating user");
        authenticator.authenticate();
    }
}
```

bodový prierez — názov

druh bodu spájania — vzor signatúry

druh odporučenia — bodový prierez

Aspekt profilovania

```
public aspect ProfilingAspect {

    pointcut publicOperation() : execution(public * *.*(..));

    Object around() : publicOperation() {
        long start = System.nanoTime();
        Object ret = proceed();
        long end = System.nanoTime();
        System.out.println(thisJoinPointStaticPart.getSignature()
                           + " took " + (end-start) + " nanoseconds");
        return ret;
    }

}
```

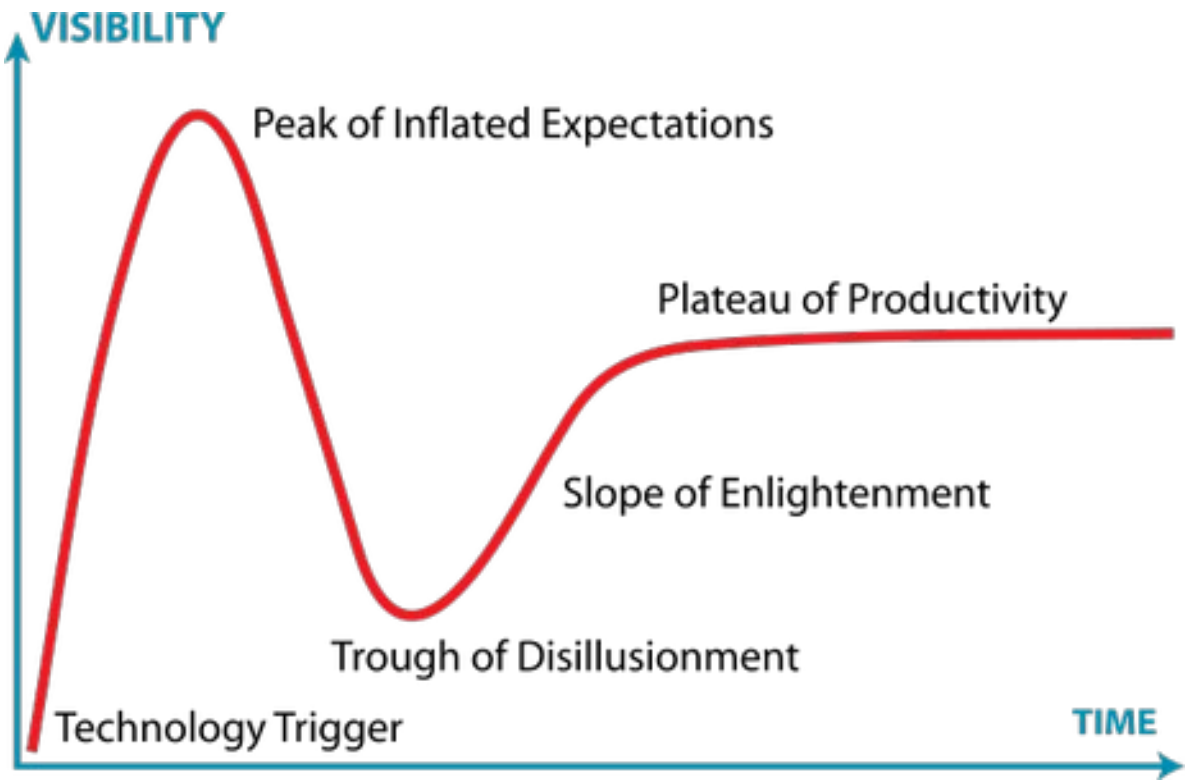
Statické pretínanie

- Doplnenie štruktúry triedy
- Príklad: zaznamenávanie posledného prístupu

Analógie

- CSS
  - vzhľad — pretínajúci záujem
  - selektor — bodový prierez
- Databázy
  - trigger — odporučenie

„Hype cycle“



by Jeremykemp at English Wikipedia

Typické použitie AOP

- Logging
- Tracing
- Profiling
- Pooling
- Caching
- Persistence
- Authentication
- Transaction
- Security
- Frameworks

Pretkávanie

Pretkávanie

- **ajc** – AspectJ Compiler
- Maven
- Vývojové prostredia (Eclipse, IntelliJ Idea)

Mechanizmy pretkávania (statické)

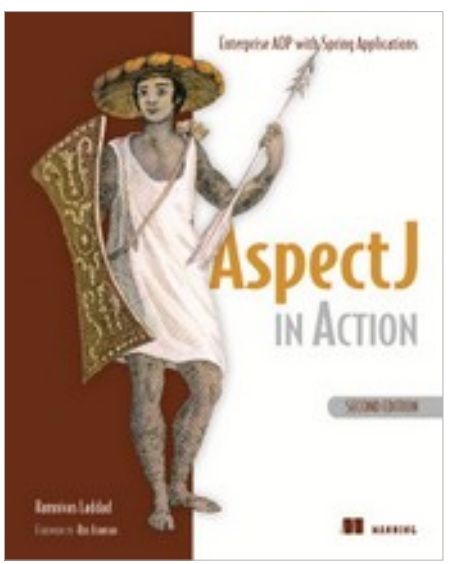
1. Pretkávanie zdrojových kódov
   - ajc namiesto javac
2. Pretkávanie binárnych súborov
   - samostatný preklad tried a aspektov
   - pretkávanie výsledného bajtkódu

Mechanizmy pretkávania (dynamické)

3. Pretkávanie pri načítaní (*load-time weaving*)
   - inštrumentačný agent
   - špeciálny ClassLoader
4. Pretkávanie počas behu
   - AspectJ nepodporuje
   - Spring AOP
   - dynamické proxy

Literatúra

- AspectJ Programming Guide
- AspectJ 5 Developes's Notebook – rozšírenia pridané vo verzii 5
- Ramnivas Laddad. AspectJ in Action, Second Edition (Manning 2009)



AspectJ po častiach

Body spájania

Metóda

- Vykonanie – **execution**
- Volanie – **call**

```
public void method() {
  // telo metódy
  …
}
```

execution

```
MyClass my;
…
my.method();
…
```

call

Konštruktor

- Vykonanie – **execution**
- Volanie – **call**

Členská premenná

- Priamy prístup k premennej
- Čítanie – **get**
- Zápis – **set**

Spracovanie výnimiek

- Zachytenie výnimky – **handler**

```
try {
  …
} catch(MyException e) {
  // kód spracovania výnimky
  …
}
```

handler

Inicializácia triedy

- Načítanie triedy a inicializácia statických premenných – **staticinitialization**

```
public class MyClass {
  static {
    // Inicializácia premenných
    …
  }
  …
```

staticinitialization

Inicializácia objektu

- Od vykonania rodičovského konštruktora – **initialization**
- Pred vykonaním rodičovského konštruktora – **preinitialization**

```
public MyClass(int i) {
  super(i, OtherClass.otherMethod(i));
  // Príkazy
  …
}
```

preinitialization

initialization

Odporučenie

- Vykonanie odporučenia – **adviceexecution**
- Bez signatúry

Vzor signatúry

Zástupné symboly

- `*`
  - postupnosť ľubovoľných znakov okrem bodky
- `..`
  - postupnosť ľubovoľných znakov vrátane bodky
  - ľubovoľný podbalík
  - ľubovoľný počet argumentov
- `+`
  - podtyp

Signatúra typu

- `Account`
- `*Account`
- `java.*.Date`
- `java..*`
- `Account+`

Anotácie a generické typy

- `@Secured Account`
- `@Business* Customer+`
- `Collection<@Sensitive *>`

Kombinovanie vzorov

- `!Collection`
- `Collection || Map`
- `java.util.RandomAccess+ && java.util.List+`
- `@Secured @Sensitive *`
- `@(Secured || Sensitive) *`

Signatúra metódy

- `public void Account.set*(*)`
- `public * Account.*(..)`
- `* Account.*(..)`
- `!public * Account.*(..)`
- `* *(..) throws SQLException`
- `* java.io.Reader.read(char[],..)`
- `@Secured * *(..)`
- `* (@BusinessEntity *).*(..)`

Signatúra konštruktora

- `Account+.new(..)`

Signatúra členskej premennej

- `private double Account.balance`
- `(@Sensitive *) *.*`

Bodový prierez

Bodové prierezy (*pointcuts*)

1. Prierezy podľa druhu bodu spájania (*kinded pointcuts*)
2. Ostatné bodové prierezy (*non-kinded pointcuts*)

Prierez podľa druhu bodu spájania

Iba jeden druh bodu spájania v bodovom priereze!

Prierez podľa typov v čase vykonávania

- **this**(«Typ»)
  - `this instanceof «Typ»`
- **target**(«Typ»)
  - objekt na ktorom je volaná metóda
- **args**(«Typy»)

Zachytenie kontextu

- Premenná namiesto typu
- Parameter bodového prierezu

```
before(Product product, BigDecimal price)
        : execution(* Product.setPrice(..))
          && this(product) && args(price) { … }
```

Deklarácia bodového prierezu

- Pomenovaný bodový prierez
  - `pointcut deliverMessage() : call(* Messanger.deliver(..));`
  - ```
    pointcut priceChange(Product product, BigDecimal price)
        : execution(* Product.setPrice(..))
          && this(product) && args(price)
    ```
- Anonymný bodový prierez
  - súčasť odporučenia
    ```

Odporučenie

before

```
before(Product prod, BigDecimal price)
            : priceChange(prod, price) {

    logger.writeLog("Changed " + prod.getPrice()
                  + " to " + price);

}
```

after

1) after()
2) after() returning
3) after() throwing

Príklady

```
after() returning : loggedOperation() {
    // zaznamenanie úspešnej operácie
}
after() throwing(Exception ex) : loggedOperation() {
    // zaznamenanie chyby
}
```

around

- Obklopuje bod spájania
- Má návratovú hodnotu
- Vykonanie obklopeného kódu – **proceed()**

Reflexia

- thisJoinPoint
- thisJoinPointStaticPart
- thisEnclosingJoinPointStaticPart

Príklad

Riešenie nespoľahlivej komunikácie

- Nespoľahlivý komunikačný kanál
- Opakovanie požiadaviek
- Zdroj: AspectJ in Action, Second Edition

Jacques-Louis David. Smrť Sokrata (1787)