6. Knižnice / Logovanie / Výnimky

21. 10. 2013

ÚINF/PAZ1c

(Róbert Novotný)

„Od rutín bude očakávať, že sú vytvorené na racionálnych princípoch, vďaka ktorým zapadnú do seba ako stavebné kocky. [...] Komponenty bude môcť bez obáv považovať za čierne skrinky.“

-- M. D. McIlroy, Pásová výroba softvérových komponentov, 1968

HATERS GONNA HATE
„Nejsem si tak docela jistej, jestli to najít a použít nedělají spíš lepiči kódu, kteří bývají považováni za podřadné.“

„...nahliadne do katalógu
rutín s pestrými možnosťami
presnosti, robustnosti,
časovej či pamäťovej
zložitosti a všeobecnosti.“

Potrebujem zostaviť tento projekt

„Do jedného megaJARu potrebujem napchať logovanie SLF4j, Apache HTTP Client, ďalší modul pre spracovanie súborov. To všetko podpísať a zavesiť na web pod menom applet.jar"

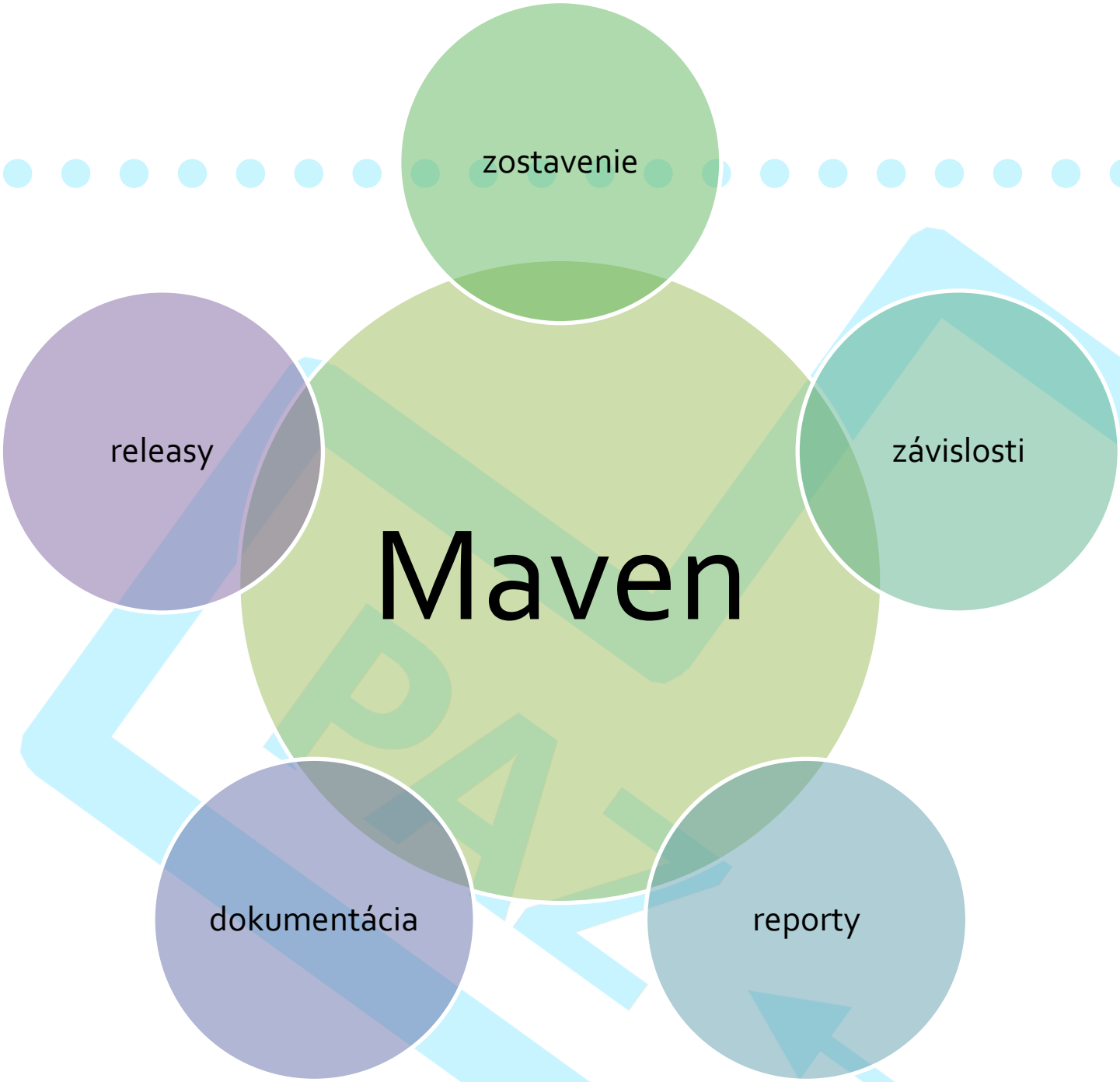
Maven

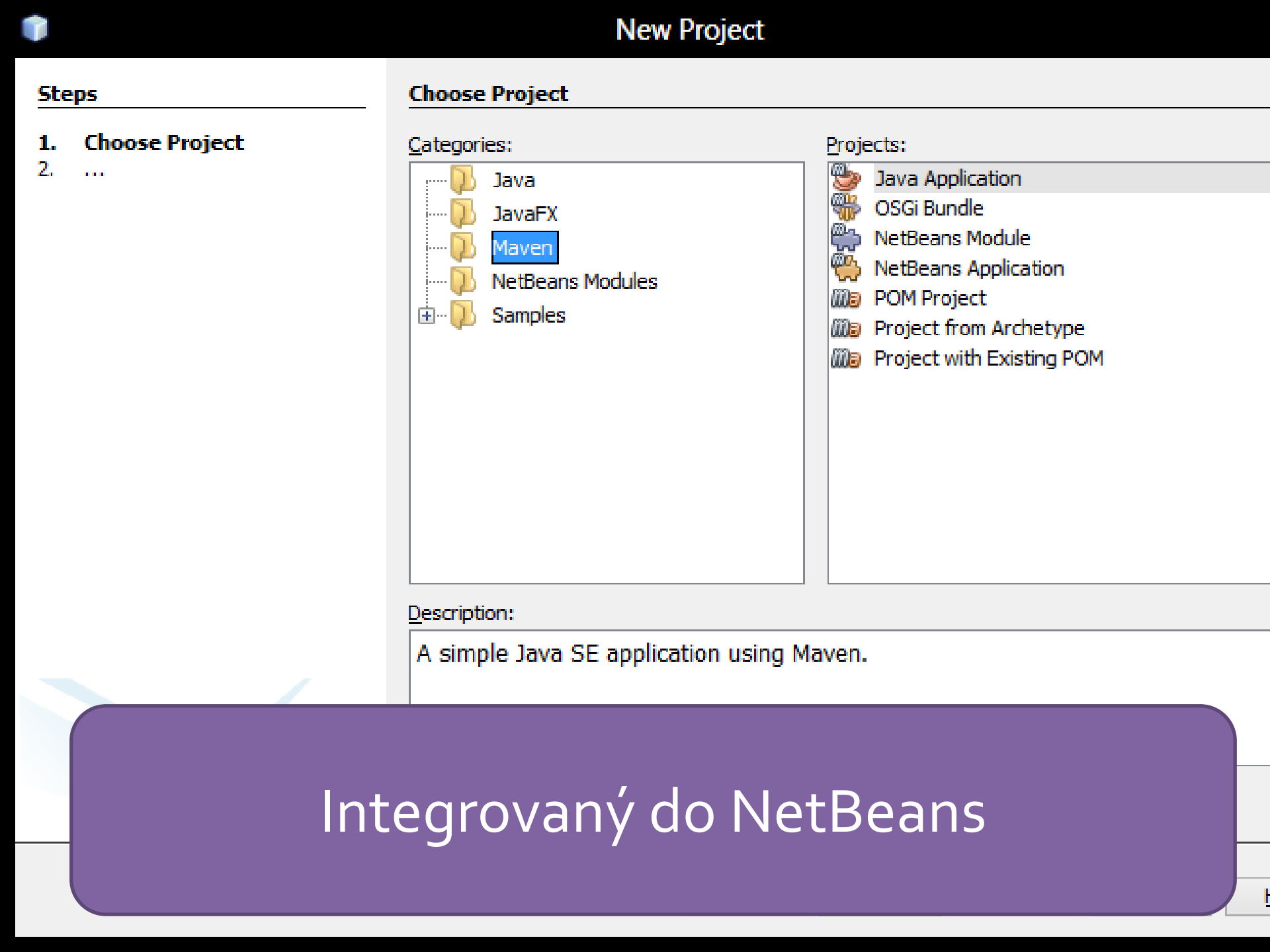
„znalec"

jidiš < hebr. mevin
(ten, kto chápe, rozumie)

Apache Maven

- build management tool
- konzistentné a jednotné zostavenie projektu
- postavený na konvenciách


zostavenie
releasy
závislosti
Maven
dokumentácia
reporty

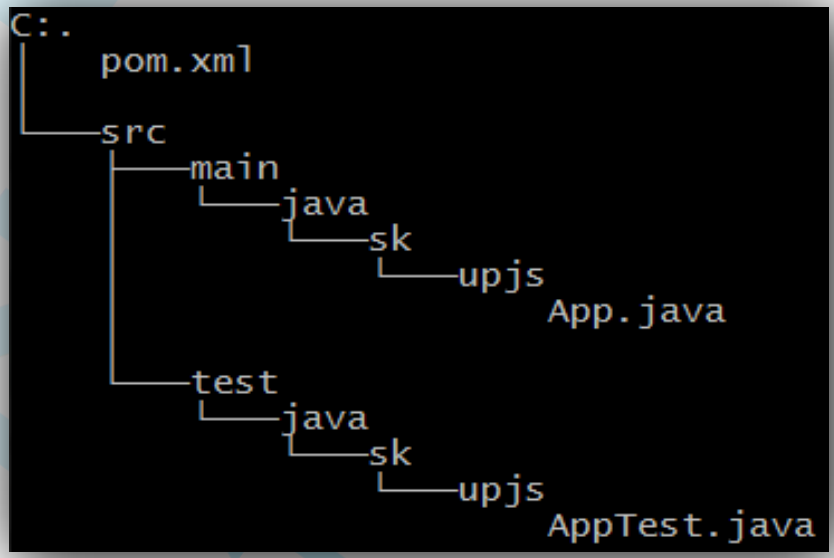


Integrovaný do NetBeans

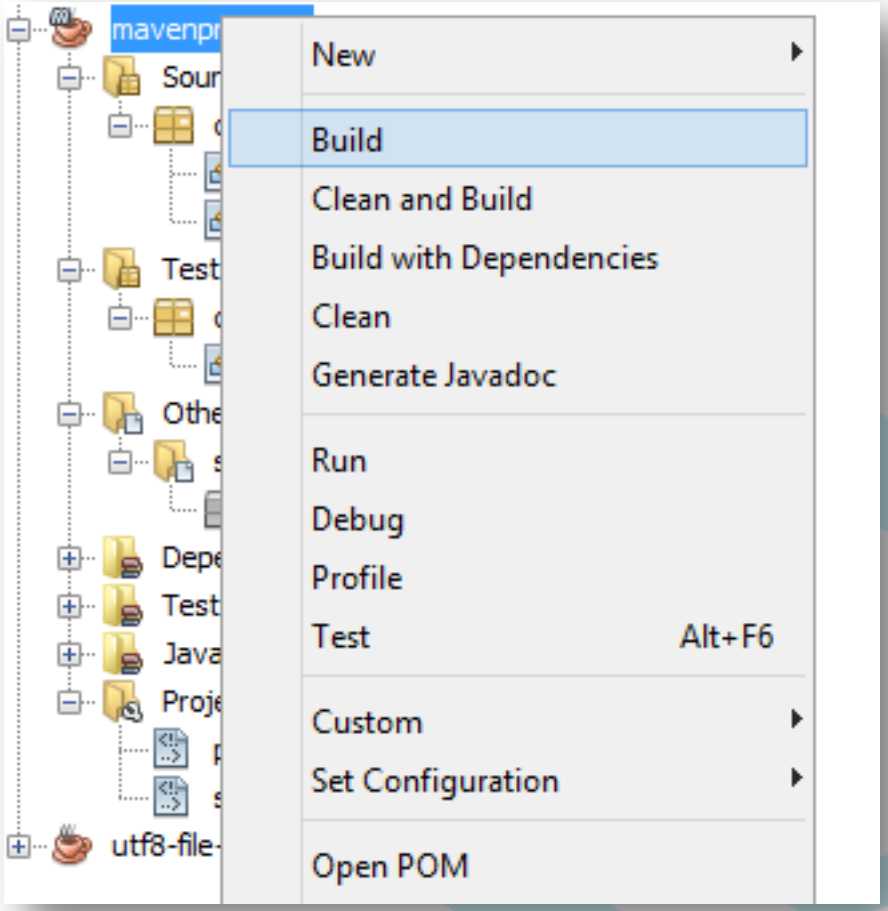
Konvencie v Mavene

- **src/main/java**: zdrojáky *.java
- **src/test/java**: zdrojáky *.java pre unit testy
- **src/main/resource:** konfiguráky, META-INF...

Nebojujte s konvenciami!



Vytvorenie JARu



- objaví sa v adresári **target**

mavenhello-1.0-SNAPSHOT.jar

http://rodcom1000.deviantart.com/art/Maven-126212535

Čo sa stalo?

- skompilovali sa zdrojáky
- zbehli unit testy
- vyrobil sa JAR
- JAR = hmatateľný výsledok

Vykonal sa **build lifecycle**

Spustenie aplikácie

```
java -cp target/mavenhello-1.0-SNAPSHOT.jar
sk.upjs.App
```

- z adresára v ktorom je projekt
  - C:/Projects/hellomaven
- výsledok:

```
Hello World!
```

Artefakty a identifikátory

- **artefakt**: hmatateľný výstup projektu
- jednoznačne identifikovaný
  - **groupId**: názov firmy
  - **názov artefaktu**: názov projektu/modulu
  - **verzia**

com.upjs.ics:hellomaven:1.0.0-SNAPSHOT

pom.xml – popisovač projektu

- jednoznačne identifikuje projekt
- + identifikátor
- + metadáta
- + závislosti
- + druh artefaktu
- + spôsob jeho buildovania

```
<project .... >
 <modelVersion>4.0.0</modelVersion>

<groupId>sk.upjs</groupId>
<artifactId>mavenhello</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>

<name>mavenhello</name>
<url>http://maven.apache.org</url>

<properties>
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<dependencies>
        <!-- -- >
</dependencies>
</project>
```

Popisovač artefaktu

Krajší názov projektu + webové sídlo

Kódovanie zdrojákov

Závislosti

Závislosti

- projekt má často závislosti
- artefakt závidí na iných artefaktoch
- databáza HSQL? dodajme do **pom.xml**

```
<dependency>
  <groupId>org.hsqldb</groupId>
  <artifactId>hsqldb</artifactId>
  <version>2.3.0</version>
</dependency>
```

The Central Repository

- funguje filozofia balíčkovacích správcov z Linuxu
- **Maven Central:** úložisko artefaktov projektov na webe
  - prítomný takmer každý známy projekt
- **Maven** preberá závislosti práve odtiaľ
- možnosť vyhľadávať na:

http://search.maven.org

Maven Local

- šetríme traffic: lokálna cache v profile

C:/Users/novotnyr/.m2

- závislosti sa hľadajú najprv v **lokálnej** cache
- ak sa nenájdu, ide sa do **Centralu**
  - odtiaľ sa stiahnu do lokálnej cache

Správa závislosti

- uvádzané do <dependency>
- podpora tranzitívnych závislosti

projekt závisí na Spring-JDBC

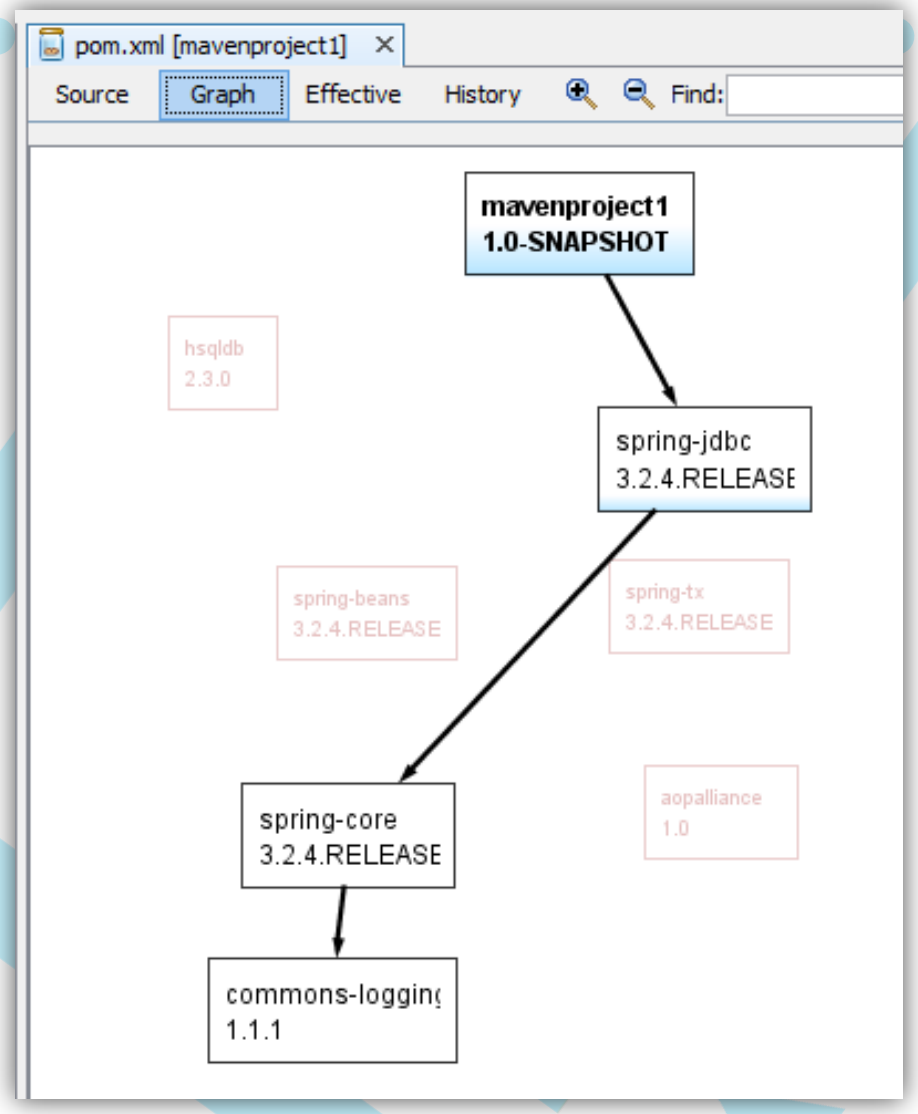
Spring-JDBC závisí na
- Jakarta Commons Logging
- Spring-Core
- ...

Tranzitívne závislosti

```
<project ... >
  ...
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>3.2.4.RELEASE</version>
    </dependency>
  </dependencies>
    ...
</project>
```

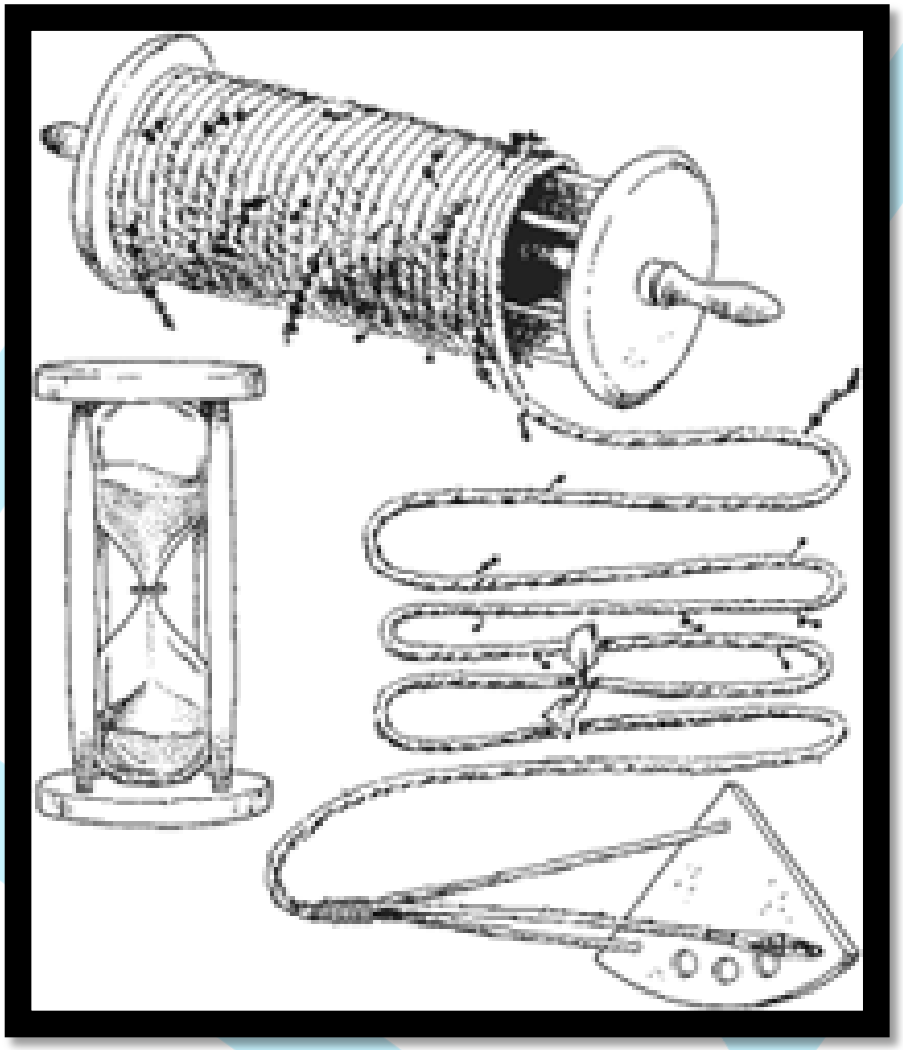
stačí uviesť koreňovú závislosiť

Graf tranzitívnych závislostí



LOGOVANIE

Log



Logging

- monitoring aplikácie
- typický pre systémy bez GUI

System.out / System.err

```
public void countElements(int... elements) {
  System.out.println("Spočítavam "
      + elements.length + " prvkov.");

  int sum = 0;

  for(int i = 0; i < elements.length; i++) {
    sum = sum + elements[i];

    System.out.println("Spracovávam " + i
        + ". prvok: " + elements[i]);
  }

  System.out.println("Výsledok: " + sum);
}
```

Problémy so System.out/err

- len dva kanály
  - *jeden pre oznamy, druhý pre chyby*
- nemožno zapínať a vypínať hlášky podľa priority
  - *čo ak chcem nepodstatné hlášky dočasne vypnúť?*
- neflexibilný výpis
  - *čo keď chcem vypísať dátum a zdroj hlášky?*

Logovacie frameworky

- k dispozícii **java.util.logging**
  - zabudovaný, ale veľmi nešťastný
- bývalý „štandard“ **log4j**
- dnešný štandard **slf4j**
  - stiahneme 2 JARy, pridáme do projektu

Logovanie

```
package sk.test.arrays;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Arrays {
  private static final Logger logger
      = LoggerFactory.getLogger(Arrays.class);

}
```

továreň!

Logger je objekt pre logovanie

Logovanie

```
public void countElements(int... elements) {
  logger.info("Spočítava sa {} prvkov.",
                  elements.length);
  int sum = 0;

  for(int i = 0; i < elements.length; i++) {
    sum = sum + elements[i];

    logger.debug("{}. prvok: {}", i,
       elements[i]);
  }

  logger.info("Výsledok: {}", sum);
}
```

Progres!

```
18:41:25.021 [main] INFO sk.test.Arrays -
Spočítavam 4 prvkov.
18:41:25.030 [main] DEBUG sk.test.Arrays –
0. prvok: 2
18:41:25.030 [main] DEBUG sk.test.Arrays –
1. prvok: 3
18:41:25.030 [main] DEBUG sk.test.Arrays –
2. prvok: 5
18:41:25.031 [main] DEBUG sk.test.Arrays –
3. prvok: 2
18:41:25.031 [main] INFO sk.test.Arrays -
Výsledok: 12
```

Logovanie podporuje výnimky

```
try {
   ...
} catch (PripojenieKDatabazeZlyhaloException e) {
   logger.error("Databaza nie je dostupna.", e);
}
```

Zaloguje sa hláška a **stack trace** výnimky.

VÝNIMKY

Výnimky pre výnimočné stavy

programátorské chyby

- wtf momenty

narušenie kontraktu

- nedodržané preconditions

nemožnosť naplniť kontrakt

- narušené postconditions, invarianty

zlyhanie zdrojov

- spadnuté servery, nenájdené súbory

Výnimka je súčasť kontraktu

- pri návrhu kontraktu uvažujte nad výnimkami
- hádzané výnimky nech zodpovedajú zapúzdreniu

java.lang.Integer

```
public static int parseInt(String s)
                    throws NumberFormatException
```

if the string does not contain a parsable integer.

Overenie *conditions

Deja vu slide!

- narušenie preconditions a postconditions = **výnimka**
- konvencia v Jave:
  - hodíme **runtime** výnimku
  - vlastnú alebo niektorú z preddefinovaných

IllegalArgumentException	Nesprávna alebo nepovolená hodnota parametra = narušenie precondition
NullPointerException	Niekde sa objavilo null (na vstupe?)
IllegalStateException	Trieda sa dostala do stavu, ktorý narúša invariant

[Ne]kontrolované výnimky

- nekontrolované (**unchecked**)
  - dedia od **RuntimeException**
- kontrolované (**checked**)
  - dedia od **Exception**
  - metóda ich musí uviesť do **throws** v hlavičke

Klient nie je schopný zotaviť sa / spamätať sa / učiniť nápravu.

Zopakovať pokus?, Náhradné riešenie?

Obmedzenia kontrolovaných výnimiek

- dilema **checked-unchecked** je náročná
- vždy ich treba **odchytiť** alebo **prehodiť** vyššie

```
URL adresa = new URL("http://www.upjs.sk");
```

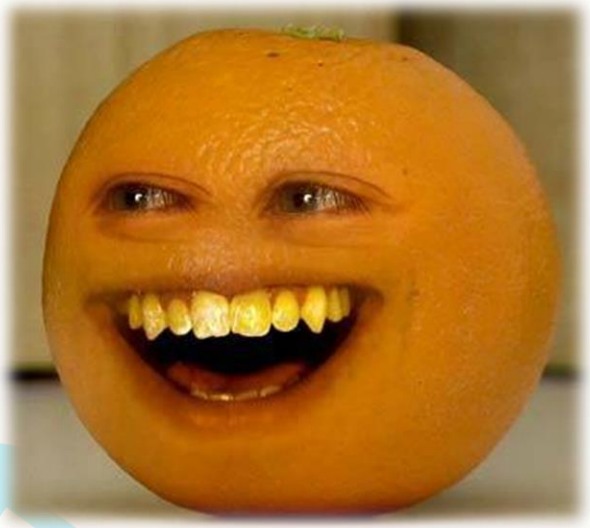
java.net.URL

konštruktor hádže **MalformedUrlException!**

Obmedzenia kontrolovaných výnimiek

```
try {
   URL adresa = new URL("http://www.upjs.sk");
} catch (MalformedUrlException e) {
   // format je vzdy korektny
}
```

- situácia typu „**nedodržali sme kontrakt**“
- ale kontrolovaná **otravná** výnimka



Obmedzenia kontrolovaných výnimiek

- metóda udáva v **throws** kontrolované výnimky
- prekrytá trieda **nemôže** tento zoznam **rozšíriť** o ďalšie kontrolované výnimky

```
class Hacker {
  void hackuj();
}
```

```
class UltraRychlyHacker extends Hacker {
  void hackuj()
      throws NedostatokDrogException
}
```

Obmedzenia kontrolovaných výnimiek

- metóda však môže hádzať nekontrolované výnimky aj bez uvedenia v **throws**

```
class UltraRychlyHacker extends Hacker {
  void hackuj()
}
```

```
class NedostatokDrogException
   extends RuntimeException
```

```
class UltraRychlyHacker extends Hacker {
  void hackuj()
      throws NedostatokDrogException
}
```

Kontrolované výnimky

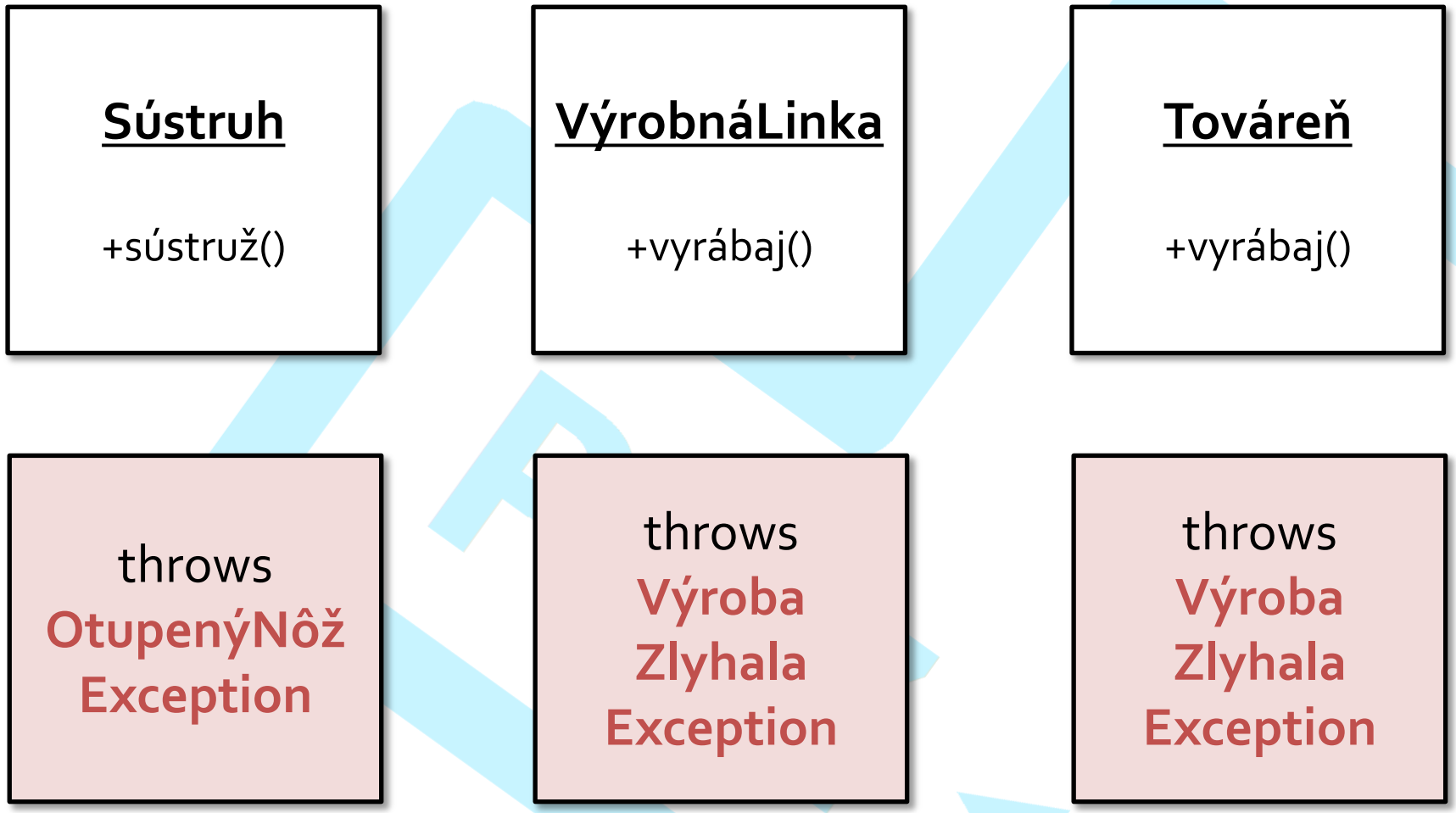
- žiadny iný bežný jazyk ich nemá
- s odstupom času kontroverzný krok

Všetky výnimky nech sú nekontrolované.

Aj tak ich uvádzame do **throws**.

\+ zadokumentujeme

Výnimky majú dodržiavať zapúzdrenosť



Systém padajúceho!

- lenivec prehadzuje výnimky o úroveň vyššie

```
public class Sustruh {
  void sústruž() throws Exception()
}
```

Narúša mieru abstrakcie + vyzerá strašne

Nebojte sa vytvárať vlastné výnimky!

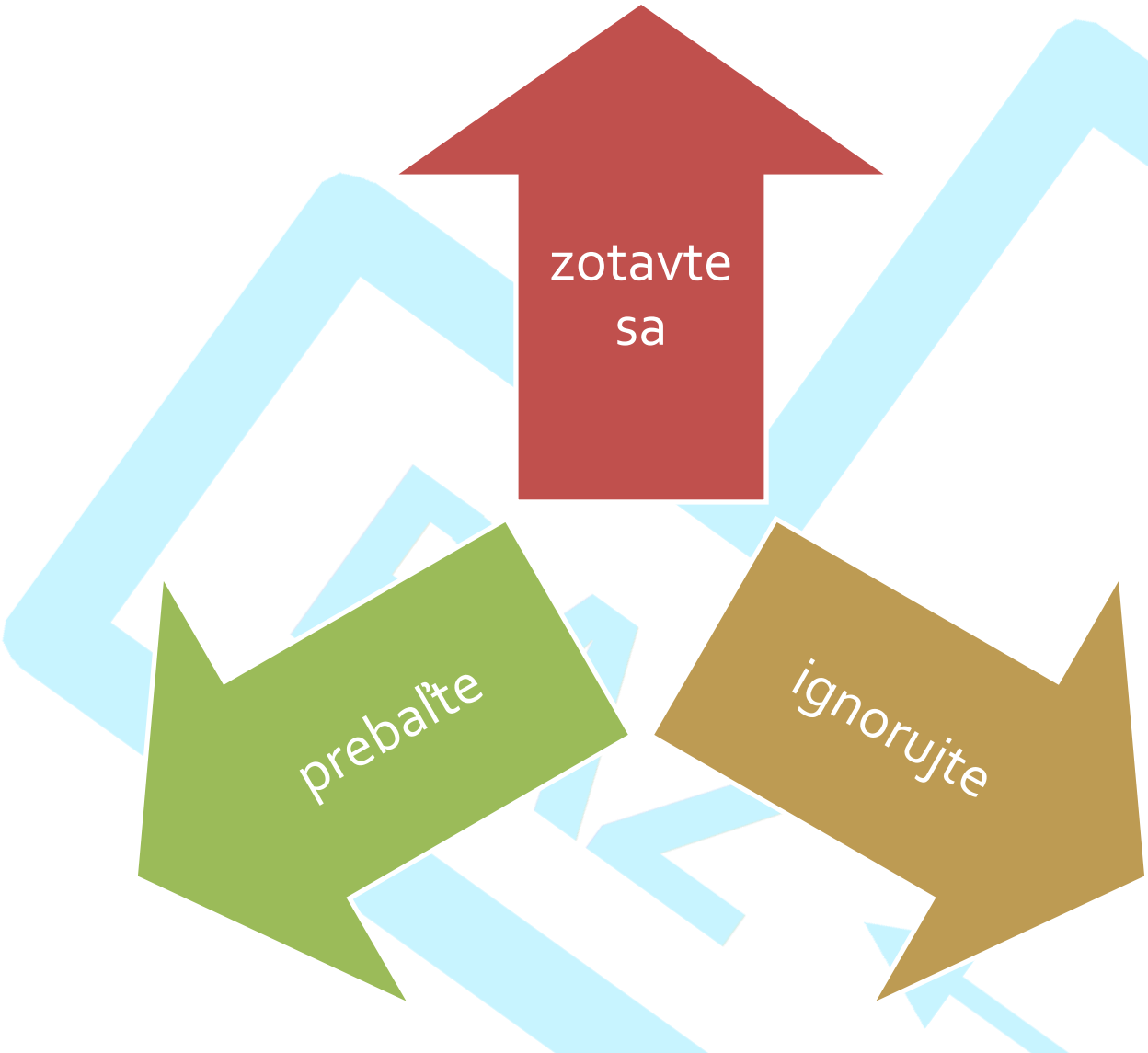
```
public class OtupenýNôžException
  extends RuntimeException
{
  private String kódNoža;

  public OtupenýNôžException(String kódNoža) {
    this.kódNoža = kódNoža;
  }

  public String getKódNoža() {
    return this.kódNoža;
  }
}
```

Výnimka je objekt

- **.getMessage()**
  - text hlášky
- **.getCause()**
  - príčina výnimky: v prípade prebaľovaných výnimiek
- **stack trace**
  - postupnosť vnáraní volaní metód

```
13:46:50.711 [martin] [http-bio-8889-exec-24] DEBUG n.g.c.w.CatalogueXmlRespo
net.[redacted].[redacted].UserNotFoundException: User [[redacted]] does not exist.
        at net.[redacted].[redacted].web.controllers.ContactController.setIntern
        at net.[redacted].[redacted].web.controllers.ContactController.addContac
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:1
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorIm
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAc
        at java.lang.reflect.Method.invoke(Method.java:601) ~[na:1.7.0_06]
        at org.springframework.web.method.support.InvocableHandlerMethod.invo
        at org.springframework.web.method.support.InvocableHandlerMethod.invo
        at org.springframework.web.servlet.mvc.method.annotation.ServletInvoc
        at org.springframework.web.servlet.mvc.method.annotation.RequestMappi
        at org.springframework.web.servlet.mvc.method.annotation.RequestMappi
        at org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAd
        at org.springframework.web.servlet.DispatcherServlet.doDispatch(Dispa
        at org.springframework.web.servlet.DispatcherServlet.doService(Dispat
        at org.springframework.web.servlet.FrameworkServlet.processRequest(Fr
        at org.springframework.web.servlet.FrameworkServlet.doPost(FrameworkS
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:641) [serv
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:722) [serv
        at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(A
        at org.apache.catalina.core.ApplicationFilterChain.doFilter(Applicati
        at com.avlesh.web.filter.responseheaderfilter.ResponseHeaderManagerFi
        at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(A
        at org.apache.catalina.core.ApplicationFilterChain.doFilter(Applicati
        at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapp
        at org.apache.catalina.core.StandardContextValve.invoke(StandardConte
        at org.apache.catalina.authenticator.AuthenticatorBase.invoke(Authent
```

Odchytávanie výnimiek



Zotavenie sa z výnimiek

- skúsiť inú možnosť výpočtu
- po čase opakovať
- opraviť a skúsiť znova
- ...

v praxi: situácií, keď sa dáz zotaviť, nie je veľa

```
try {
  účet.vyplaťKartou(cena);
} catch (PlatbaTerminálomException e) {
  účet.vyplaťVHotovosti(cena);
}
```

Prebalenie výnimiek

```
try {
   sústruh.sústruž()
} catch (OtupenýNôžException e) {
   throw new VýrobaZlyhalaException(
        "Zlyhanie kvôli chybe sústruhu", e);
}
```

VýrobnáLinka

VýrobaZlyhalaException

message: "Zlyhanie kvôli..."

cause:

Otupený nôž

message: "Otupil sa nôž..."

Spracovávanie má byť naozajstné

- skúsiť inú možnosť výpočtu
- opakovať pokus
- ...

```
try {
    ...
} catch (Exception e) {
  e.printStackTrace();
}
```

Nie!

Odchytávanie výnimiek

- výnimku **odchyťte a spracujte**
- alebo ju prehoďte **o úroveň vyššie**
- alebo **prebaľte** do popisnejšej výnimky a vyhoďte

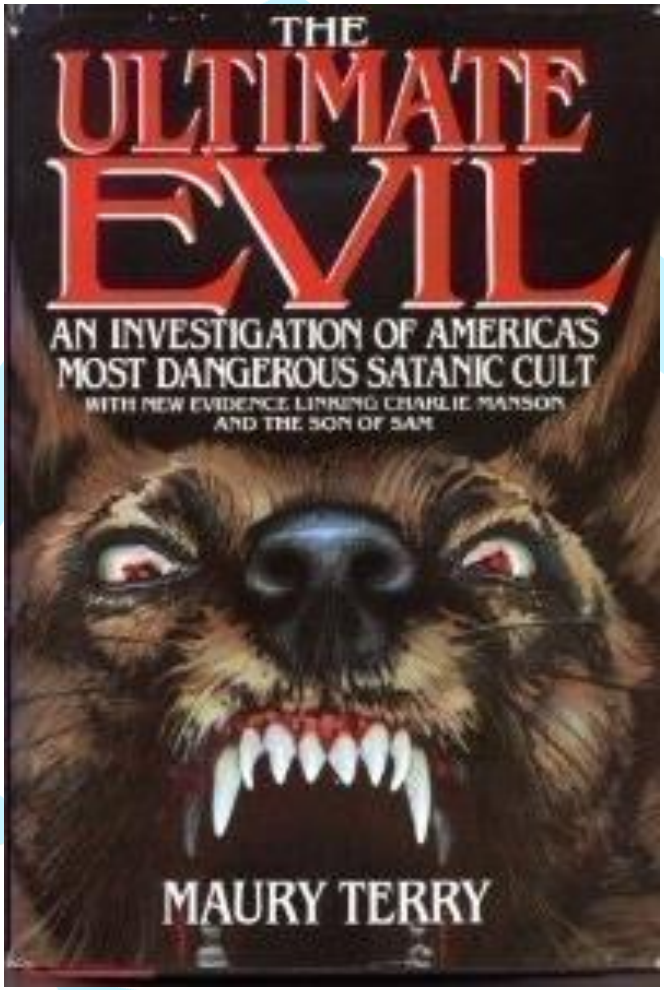
VýrobnáLinka

```
try {
   sústruh.sústruž()
} catch (OtupenýNôžException e) {
   throw new VýrobaZlyhalaException(e);
}
```

Spracovávanie výnimiek

- hltanie výnimiek je ultimátne zlo

```
try {
  //..
} catch (Exception e) {}
```



Vyhoďte alebo zalogujte!

- ak sa výnimka nedá spracovať, **zalogujte** ju
- inak ju [prebaľte a] **vyhoďte**

Nekombinujte!

```
try {
   ...
} catch (PripojenieKDatabazeZlyhaloException e) {
   logger.error("Databaza nie je dostupna.", e);
   throw new DuplicitnyObjektException(...);
}
```

tá istá chyba bude v logoch zbytočne viackrát

Sumár

- výnimky a logovania sú užitočná vec pri **odhaľovaní** chýb
- neraz **jediný** mechanizmus zistenia, čo je v programe zlé
- dodržiavajte konzistentne zásady a potešia sa klienti vašich tried!

Otázky?

Projekty!

- v duchu rozsiahlejších projektov z cvičení
- dvojčlenné tímy
- požiadavky na webe
- Míľnik 1 - koniec semestra:
  - prezentácia betaverzie
  - pridelenie cudzieho projektu
- Míľnik 2 – cca ½ januára:
  - prezentácia finálnej verzie
  - prezentácia cudzieho projektu

Projekty!

- 80% hodnotenia
- predbežné požiadavky:
  - 4 komplexné dialógové okná
  - 4 entity
  - DAO + továrne + singletony
  - relačná databáza
  - pokrytie unit testami

Úlohy

- do 28. 10. poslať na **robert.novotny@upjs.sk** a **michal.vasko@upjs.sk**
  - mená členov tímu
  - názov projektu
  - stručné vízie / predpokladané vlastnosti