Výjimky

Obsah

Výjimky

- Výjimky - proč a jak, co to vlastně je výjimka
- Syntaxe bloku s ošetřením (zachycením) výjimky
- „Únik“ výjimky z metody - deklarace metody propouštějící výjimku
- Reakce na výjimku
- Kaskády výjimek
- Kategorizace výjimek (hlídané, běhové, vážné chyby)
- Vlastní typy výjimek, objektová hierarchie výjimek
- Klauzule finally

Co a k čemu jsou výjimky

- podobně jako v C/C++, Delphi
- výjimky jsou mechanizmem, jak psát robustní, spolehlivé programy odolné proti chybám "okolí" - uživatele, systému...
- v Javě jsou výjimky ještě lépe implementovány než v C++ (navíc klauzule `finally`WIKIPEDIA

[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=finally])

- není dobré výjimkami "pokrývat" chyby programu samotného - to je hrubé zneužití

Výjimky technicky

- *Výjimka* (*Exception*) je objekt třídy `java.lang.Exception` WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java.lang.Exception]
- Objekty -*výjimky*- jsou vytvářeny (vyvolávány) buďto
  - automaticky běhovým systémem Javy, nastane-li nějaká běhová chyba, např. dělení nulou, nebo
  - jsou vytvořeny samotným programem, zdetekuje-li nějaký chybový stav, na nějž je třeba reagovat - např. do metody je předán špatný argument
- Vzniklý objekt výjimky je předán buďto:
  1. v rámci metody, kde výjimka vznikla - do bloku `catch` WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=catch] -> výjimka je v bloku `catch` WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=catch] tzv. **zachycena**
  2. výjimka "propadne" do nadřazené (volající) metody, kde je buďto v bloku `catch` WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=catch] zachycena nebo opět propadne atd.
- Výjimka tedy "putuje programem" tak dlouho, než je zachycena
- -> pokud není, běh JVM skončí s hlášením o výjimce

Syntaxe kódu s ošetřením výjimek

Základní syntaxe:

```
try {
   //zde může vzniknout výjimka

} catch (TypVýjimky proměnnáVýjimky) {
   // zde je výjimka ošetřena
   // je možné zde přistupovat k proměnnéVýjimky
}
```

Příklad - Otevření souboru může vyvolat výjimku [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/vyjimky/OtevreniSouboru.java]

Bloku `try` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=try] se říká *hlídaný blok*, protože výjimky (příslušného hlídaného typu) zde vzniklé jsou zachyceny.

Syntaxe metody propouštějící výjimku

Pokud výjimka nikde v těle numůže vzniknout, překladač to zdetekuje a vypíše:

`... Exception XXX is never thrown in YYY ...` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=... Exception XXX is never thrown in YYY ...]

Příklad s propouštěnou výjimkou -Otevření souboru s propouštěnou výjimkou [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/vyjimky/OtevreniSouboru2.java]

```
modifikatory návratovýTyp nazevMetody(argumenty) throws TypPropouštěnéVýjimky {
   ... tělo metody, kde může výjimka vzniknout ...
}
```

Reakce na výjimku

Jak můžeme reagovat?

1. Napravit příčiny vzniku chybového stavu - např. znovu nechat načíst vstup
2. Poskytnout za chybný vstup náhradu - např. implicitní hodnotu
3. Operaci neprovést („vzdát“) a sdělit chybu výše tím, že výjimku „propustíme“ z metody

Výjimková pravidla:

1. Vždy nějak reagujme! Neignorujme, nepotlačujme, tj.
2. blok `catch` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=catch] *nenechávejme prázdný*, přinejmenším vypišme `e.printStackTrace()` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=e.printStackTrace()]
3. Nelze-li reagovat na místě, propusťme výjimku výše (a popišme to v dokumentaci...) - Příklad komplexní reakce na výjimku [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/vyjimky/VyjimkyZachytZopakuj.java]

Kaskády výjimek

V některých blocích `try`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=try] mohou vzniknout *výjimky více typů*:

- pak můžeme bloky `catch`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=catch] řetězit, viz přechozí příklad: Příklad komplexní reakce na výjimku [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/vyjimky/VyjimkyZachytZopakuj.java]
- Pokud `catch`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=catch] řetězíme, musíme respektovat, že výjimka je zachycena nejbližším příhodným `catch`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=catch]
- Pozor na řetězení `catch`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=catch] s výjimkami typů z jedné hierarchie tříd: pak musí být výjimka z podtřídy (tj. speciálnější) uvedena - zachycována - dříve než výjimka obecnější - Takto ne! [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/vyjimky/KaskadaVyjimekSpatne.java]

Kategorizace výjimek a dalších chybových objektů

- Všechny objekty výjimek a chybových stavů implementují rozhraní java.lang.Throwable - „vyhoditelný“
- Nejčastěji se používají tzv. *hlídané výjimky* (checked exceptions) - to jsou potomci/instance třídy java.lang.Exception
- Tzv. *běhové (runtime, nebo též nehlídané, unchecked) výjimky* jsou typu/typu potomka --> java.lang.RuntimeException - takové výjimky nemusejí být zachytávány
- *Vážné chyby JVM* (potomci/instance java.lang.Error) - obvykle signalizují těžce napravitelné chyby v JVM - např. *Out Of Memory*, *Stack Overflow*..., ale též např. chybu programátora: AssertionError

Vlastní hierarchie výjimek

- Typy (=třídy) výjimek si můžeme definovat sami, např. viz - Výjimky ve světě chovatelství [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/svet.html]
- bývá zvykem končit názvy tříd - výjimek - na *Exception*

Klauzule `finally`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=finally]

Klauzule (blok) *finally*:

- Může následovat ihned po bloku `try`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=try] nebo až po blocích `catch`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=catch]
- Slouží k "úklidu v každém případě", tj.
  - když je výjimka *zachycena* blokem `catch`WIKIPEDIA [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=catch]
  - i když je výjimka *propuštěna* do volající metody
- Používá se typicky pro uvolnění systémových zdrojů - uzavření souborů, soketů...

Příklad: Úklid se musí provést v každém případě... [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/vyjimky/VyjimkyFinally.java]

Odkazy

- Sun Java Tutorial - Lesson: Handling Errors with Exceptions [http://java.sun.com/docs/books/tutorial/essential/exceptions/index.html]
- demo programy z učebnice - Výjimky [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/vyjimky]