Hlídané a vlastní výjimky, blok `finally`

Tomáš Pitner, upravil Marek Šabo

Výjimky a jejich hlídání překladačem

- Java je staticky (překladově) typovaný jazyk a jako takový zná místa potenciálního vyhození výjimky.
- V dosud uváděných příkladech se neprojevovalo, že by nás nějak hlídal.
- Bylo to proto, že jsme dosud používali tzv. *běhové (nehlídané) výjimky* (runtime-/unchecked exceptions), jejichž místa vzniku překladač nesleduje a nehlídá, jak na ně reagujeme.
- Nyní nastíníme úplnou kategorizaci výjimek vč. *hlídaných*.

Kategorizace výjimek a dalších chybových objektů

- hlídané výjimky, *checked exceptions*:
  - potomky/instancemi třídy `java.lang.Exception`
  - překladač sleduje místa jejich vzniku a povinnou reakci na ně
  - nebo jejich možné propuštění z metody ven v případě deklarace `throws`
- běhové (nehlídané) výjimky, *unchecked exceptions*:
  - jsou typu nebo potomky `java.lang.RuntimeException`
  - a nemusejí být *zachytávány*.

    vážné chyby JVM
    : signalizují těžce napravitelné chyby v JVM potomky

  - instance `java.lang.Error`
  - např. *Out Of Memory*, *Stack Overflow* …,
  - ale též např. chybu programátora: `AssertionError`.

Kdy nehlídanou výjimku

> Unchecked runtime exceptions represent conditions that, generally speaking, reflect errors in your program's logic and cannot be reasonably recovered from at run time.
>
> — Gosling Arnold and Holmes

Kdy použít hlídanou a nehlídanou výjimku

- *Unchecked exceptions*: represent defects in the program (bugs) — often invalid arguments passed to a non-private method.

- *Checked exceptions*: represent invalid conditions in areas outside the immediate control of the program (invalid user input, database problems, network outages, absent files).

Metody propouštějící výjimku

- Ne všechny hlídané výjimky se musejí v metodě vzniku zachytit pomocí `catch`.
- Některé mohou být z metody *propuštěny* (mohou "propadnout výše").
- Indikováno v hlavičce takové metody pomocí `throws`. Příklad:

```
public void thisMethodMayThrowException() throws TypeOfTheExceptionThrown {
   ... method body, here the exception can emerge ...
}
```

- Pokud daná hlídaná výjimka nikde v těle nemůže vzniknout, překladač to zdetekuje a vypíše:

```
Exception TypeOfTheExceptionThrown is never thrown in thisMethodMayThrowException
```

Příklad s propouštěnou výjimkou

```
private static void openFile(String filename) throws IOException {
   System.err.println("Trying to open file " + filename);
   FileReader r = new FileReader(filename);
   // success, now do further things
}
public static void main(String[] args) {
   try {
      openFile(args[0]);
      System.err.println("File opened");
   } catch (IOException ioe) {
      System.err.println("Cannot open file");
   }
}
```

Vlastní typy výjimek

- Typy (=třídy) výjimek si můžeme definovat sami.
- Bývá zvykem končit názvy tříd výjimek na `Exception` (např. `OverloadedException`).

Klauzule `finally`

- Klauzule (blok) `finally` může následovat ihned po bloku `try` nebo až po blocích `catch`.

- Slouží k "úklidu v každém případě", tj.
  - když je výjimka *zachycena* blokem `catch`,
  - i když je výjimka *propuštěna* do volající metody.
- Používá se typicky pro *uvolnění (systémových) zdrojů* — uzavření souborů, soketů.

Příklad `finally` (1)

```
public class NotEnoughParametersException extends Exception {
   private int countParam;
   public NotEnoughParametersException(int countParam) {
      this.countParam = countParam;
   }
   // ...
}
```

- Pozn. Všimněte si, že u výjimek stejně jako u jiných tříd můžeme mít atributy, konstruktory, atd.

Příklad `finally` (2)

```
try {
   if (countParam < 2) // we can throw exception directly
      throw new NotEnoughParametersException(countParam);
   // here we go unless an exception thrown
   System.out.println("Correct number of params: " + countParam);
} catch (NotEnoughParametersException mp) {
   // here we go in case an exception is thrown
   System.out.println("Less parameters than needed: " + mp.getCountParam());
} finally {
   // here we go always
   System.out.println("End");
}
```

Odkazy

- Oracle Java Tutorials: Lesson: Handling Errors with Exceptions