16. Výjimky

Význam chyby syntaxe a význam logické chyby

- Chyba syntaxe vzniká, když kód v jazyce Java neodpovídá **pravidlům syntaxe jazyka**. Syntaxe jazyka Java stanoví, jakým způsobem musí být kód napsán, aby byl platný a aby ho bylo možné přeložit do strojového kódu. Pokud je v kódu nějaká chyba syntaxe, překladač Javy nebude schopen kód přeložit do strojového kódu a vyhodí chybu, která přesně určuje, kde se v kódu nachází chyba
- Logická chyba je chyba v programu, která nevznikla v důsledku chybné syntaxe, ale v důsledku **špatného algoritmu nebo chybného myšlení programátora**. Program s logickou chybou může být syntakticky správný a může být i přeložen, ale jeho výstup může být nekorektní. Logické chyby jsou často obtížné odhalit a opravit, protože mohou vznikat v jakékoli části programu a mohou být způsobeny různými faktory, jako jsou chybné výpočty, špatná logika nebo nevhodná implementace algoritmu

Rozdíl Error × Exception

- Error
  - Obecný název pro výjimky, které vznikají v důsledku závažných chyb, které obvykle nelze předvídat ani ovlivnit
  - Tyto chyby obvykle ovlivňují celý běh programu a nelze je obvykle řešit v rámci běhu programu.
  - Některé příklady errorů jsou **OutOfMemoryError** nebo **StackOverflowError**
- Exception
  - Obecný název pro výjimky, které vznikají v důsledku chyb, které mohou být předvídatelné a ovlivnitelné v rámci běhu programu
  - Tyto výjimky obvykle vznikají v reakci na neplatný vstup, nesprávnou konfiguraci nebo neočekávané chování programu
  - Některé příklady výjimek jsou **IOException** nebo **FileNotFoundException**

Významné výjimky a významné chyby

- **NullPointerException**
  - Vzniká, když kód snaží použít referenci na objekt, která ukazuje na null hodnotu
- **IOException**
  - Vzniká, když dojde k nějaké chybě během operace vstupu nebo výstupu, např. soubor je chráněn proti zápisu či čtení
- **ArrayIndexOutOfBoundsException**
  - Vzniká, když kód snaží přistoupit k položce pole mimo jeho rozsah
- OutOfMemoryError
  - Vzniká, když kód spotřebuje veškerou dostupnou paměť
- StackOverflowError
  - Vzniká, když kód vytvoří příliš mnoho rekurzivních volání a zásobník volání se přeplní
- NoClassDefFoundError
  - Vzniká, když kód nemůže nalézt třídu, kterou potřebuje pro běh

Rozdíl mezi throws x throw

- "**throw**" se používá k vyvolání výjimky v kódu. Používá se v situacích, kdy chceme vyvolat výjimku v rámci metody nebo bloku kódu

```
throw new Exception("Toto je zpráva výjimky.");
```

  - Tento kód vyvolá novou instanci výjimky a způsobí, že program spadne, pokud není výjimka zachycena a zpracována
- "**throws**" se používá k označení výjimek, které může metoda vyvolat. Používá se v definici metody a označuje seznam výjimek, které by mohly být vyvolány uvnitř metody

```
public void metoda() throws Exception1, Exception2 {
    // tělo metody
}
```

  - Tento kód definuje metodu, která může vyvolat výjimky Exception1 a Exception2. Pokud je tato metoda volána z jiné metody, musí být výjimky Exception1 a Exception2 ošetřeny pomocí bloku try-catch nebo označeny pro předání výjimky pomocí klíčového slova "throws"

Ošetřování výjimek

- V Javě se ošetřování výjimek provádí pomocí bloku **try-catch**
- Blok try obsahuje kód, který může vyvolat výjimku, a blok catch obsahuje kód, který zachytí a zpracuje výjimku, pokud se objeví

```
try {
    // kód, který může vyvolat výjimku
} catch (TypVýjimky1 e1) {
    // kód pro zpracování výjimky TypVýjimky1
} catch (TypVýjimky2 e2) {
    // kód pro zpracování výjimky TypVýjimky2
} catch (Exception e) {
    // kód pro zpracování všech ostatních výjimek
}
```

Prakticky:

- Kód – zahození výjimky, odchycení výjimky, vztah mezi významnými výjimkami