18. Chyby v programování a mechanismus výjimek v jazyce JAVA (zachycení a ošetření)

Chyby v programování

V programování může programátor udělat tři druhy chyb.

Syntax Error

Syntaktická chyba je chyba v syntaxi programovacího jazyka, například když se programátor přepíše → překladač neporozumí danému příkazu, protože je blbě napsán a tudíž neexistuje. Tyto chyby vždy odhalí **compiler**.

Semantic Error

Sémantická chyba je chyba v logice programu. Například prohození příkazu, nebo ještě hůř, špatný návrh algoritmu daného řešení. Chybu neodhalí překladač, je to jen a jen na programátorovi. V těchto případech pomůže **Debugger**.

Runtime Error

Run-time error je chyba za běhu programu, run-time chyba může nastat jen za nějakých podmínek. Run-time error pomůže odhalit **Java Virtual Machine**, jelikož vyhazuje výjimky, které říkají, co se stalo špatně a na jakém řádku kódu program spadl.

Výjimky v Javě

Java má velice propracovaný systém výjimek a chyb, které mohou nastat při běhu programu. Třídy, jenž vyznačují nějakou chybu nebo výjimku musí dědit od třech základních tříd nebo od jejich potomků.

Error – Vyznačují chybu, nelze zachytit a program vždy končí.

Exception – Vyznačují výjimku jenž je během programování potřeba ošetřit.

RuntimeException – Jsou výjimky, jenž nepotřebují nijak ošetřit, ale pak program padá.

Ošetření výjimek

V Javě jsou dva způsoby ošetření výjimek.

Propuštění výjimky výš

Při propouštění výjimek výš, se jednoduše za signaturu metody napíše příkaz **throws**. Propuštění (**delegování**) výjimky výš může sloužit k propuštění do vyšší metody, ale nikdy by to nemělo být skrz metodu **main**, jelikož pak program padá s výjimkou. Výjimka by se vždy měla zachytit.

```
public static void metoda() throws FileNotFoundException {
     FileReader fr = new FileReader("file.txt");
}
```

Zachycení výjimky

K zachycení slouží 3 bloky pro práci s výjimkami:

Try-catch: Blok try slouží pro kód, v němž může nastat výjimka. V bloku catch(**ExceptionName ex**) se výjimka zachytí. Blok catch má jako parametr výjimku, která může nastat v bloku try. Bloků catch může být i více, ale je důležité zvolit pořadí výjimek, tak aby byli v kaskádě (od potomka k dědicovi).

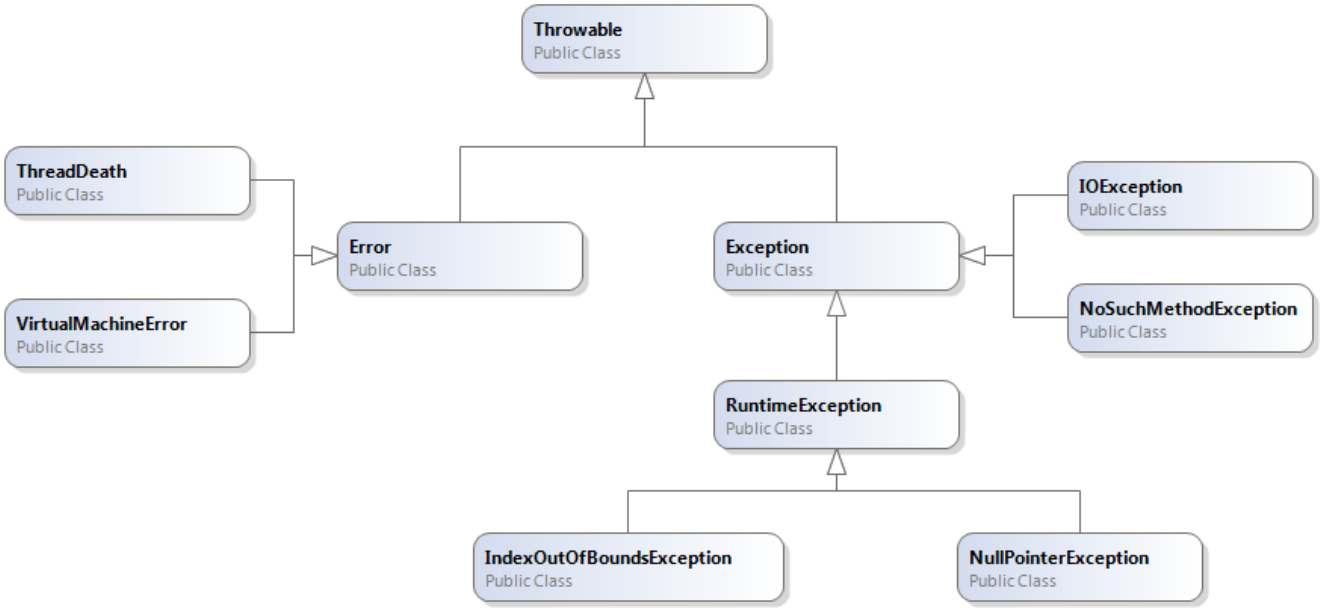
Try-finally: V bloku finally je kód, který se provede v každém případě, ať už skončí blok try vyhozením výjimky, nebo zda proběhne dobře.

Pokud se má výjimka zachytit, tak se musí použit vždy minimálně blok try a jeden z bloků catch nebo finally.

Try-with resources: blok dat, který dovoluje deklarovat jeden či více zdrojů pro daný blok. Zdroj musí být třída, která implementuje **AutoCloseable**. Výhodou tohoto zápisu je to, že se zdroj sám uzavře a uvolní, takže se nemusí použít finally blok. Při klasickém try-catchi s finally uzavřením zdroje je problém s **Výjimkou**, kterou může vyhodit metoda **close()** uvnitř finally metody.

Takže z try-catche vyleze tato Výjimka, i když uvnitř try byla jiná. Try-with resources řeší tento problém, pokud nastane chyba při uzavření zdroje, tak se ven tato chyba nepromítne, pokud byla před ni vytvořena jiná výjimka.

```
public static void metoda() {
	FileReader f = null;
	try {
		f = new FileReader("file.txt");
	} catch (FileNotFoundException ex) {
		System.err.println(ex.getMessage());
	} finally {
		if(f != null) {
			try {
				f.close();
			} catch (IOException ex) {
				System.err.println(ex.getMessage());
			}
		}
	}
}
```


Throwable
Public Class
ThreadDeath
Public Class
VirtualMachineError
Public Class
Error
Public Class
Exception
Public Class
IOException
Public Class
NoSuchMethodException
Public Class
RuntimeException
Public Class
IndexOutOfBoundsException
Public Class
NullPointerException
Public Class