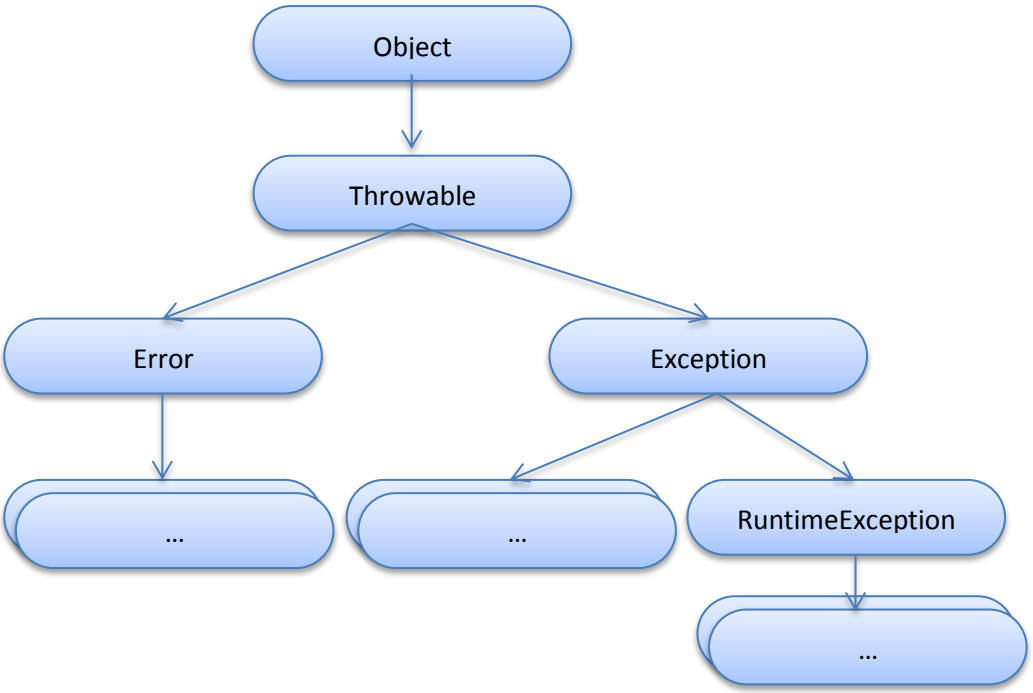
Téma 18: Chyby v programování a mechanismus výjimek v jazyce JAVA

Mechanismus výjimek je jednou z velmi silných bezpečnostních prvků Javy. Mnoho starších programovacích jazyků sice umožňuje testovat chybové kódy, ale mnoho programovacích jazyků nechává ošetření chyb a výjimek na programátorovy. Java je v tomto ohledu představitel moderních jazyků. Přímo na úrovni kompilátoru nutí programátora, aby ve svém kódu reagoval na možné chybové stavy. Pokud na ně nereaguje, program se nepřeloží.

Pod pojmem **výjimka** je míněn výjimečný stav nebo nepřesně chyba programu.

Třídy, jež 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 vyjímku jenž je během programování potřeba ošetřit.
- RuntimeException – Jsou vyjímky, jež nepotřebují nijak ošetřit, ale pak programpadá.

V Javě jsou dva způsoby ošetření vyjímek.

- Propuštění vyjíky výš
- Zachycení vyjímky

Pokud chci vyjímku propustit výš, jednoduše zadám za signaturu metody příkaz throws ExceptionsName. Propuštění vyjímky výš může sloužit k propuštění do vyšší metody, ale nikdy by to nemělo být skrz main, jelikož pak program padá z vyjímkou. Výjimka by se vždy měla zachytit.

```
public static void method() throws FileNotFoundException {
FileReader f = new FileReader("file");
}
```

K zachycení nám slouží tři bloky pro práci s výjimkami `try-catch(ExceptionName e)-finally`. Blok `try` slouží pro kód, v němž může nastat výjimka. V bloku `catch(VyjimkaName e)` se výjimka zachytí, blok `catch` má jako paramet vyjímku která může nastat v bloku `try`, bloků `catch` může být i více, ale je důležité zvolit pořadí vyjímek tak abychom nedali předka před potomka, jelikož by se k potomkovi nic nedostalo. V bloku `finally` je kód, který se provede v každém případě, ať už skončí blok `try` vyhozením vyjímky, nebo zda proběhne dobře.

Pokud chci vyjímku zachytit, tak musím použit vždy minimálně blok `try` a jeden z bloků `catch` nebo `finally`.

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

V programování může programátor udělat **tři druhy chyb**:

- Syntax error
- Semantic error
- Runtime error

Syntax error nebo-li syntatická chyba jak už název napovídá je to chyba v syntaxi programovacího jazyka, například místo for napíšeme fot nebo tak nějak, je to to že překladač neporozumí danému příkazu třeba proto, že je špatně napsán a tudíž neexistuje. Tyto chyby vždy odhalí compiler.

Někteří tvrdí že nejhorší je run-time error, já si myslím, že je to sematic error nebo-li, sémantická chyba, chyba v logice programu. Je to například prohození příkazu, nebo ještě hůř již špatný návrh algoritmu daného řešení. Chybu neodhalí překladač, je to jen a jen na programátorovi, i když mu může právi ulehčit debuger.

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