## 7.1 Další praktické metody

V této kapitole si zkusíme vytvořit pár jednoduchých metod pracujících s objekty typu Zak.

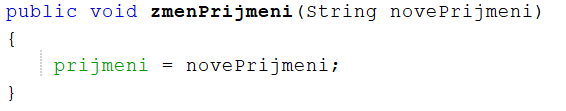
První metoda se jmenuje zmenPrijmeni. Jelikož vytváříme systém hodící se i pro střední školství, je možné, že žákyně by se již během studia mohla vdát a následně i změnit příjmení. Dává tedy smysl metodu naprogramovat.

Přemýšlejme, co do metody bude vstupovat. Proměnná prijmeni nikoli, neboť ji sice měníme, ale již ji máme k dispozici v objektu. Nemá tedy smysl, aby do metody vstupovala přes parametr. Budeme ale potřebovat nové příjmení, kterým chceme staré přepsat. Do metody tedy bude vstupovat textový řetězec String, který pojmenujeme novePrijmeni.

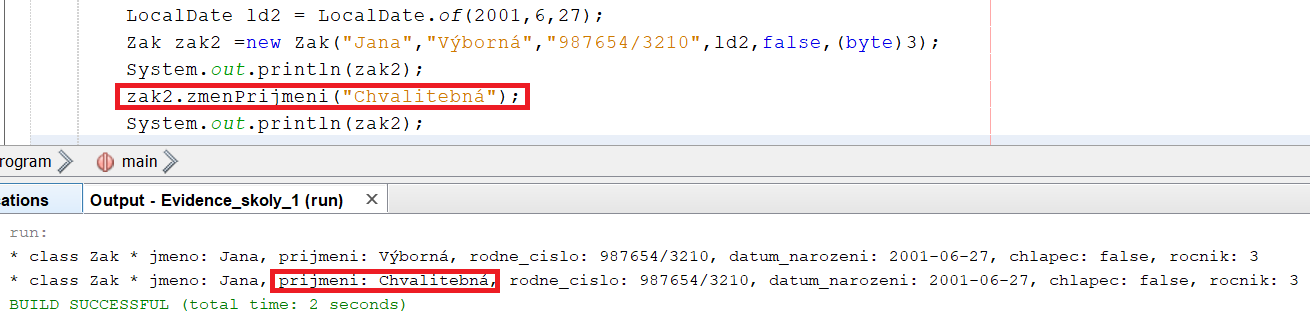
Výstup nebude mít metoda žádný. Dle zadání nepožadujeme, aby metoda něco vrátila či zobrazovala. Má jen změnit hodnotu proměnné prijmeni, toť vše. Výstup tedy nastavujeme jako void.

Metoda bude tak jako všechny dosud konstruované metody veřejná (public), abychom ji mohli zavolat z jiné třídy. Konkrétně ze třídy Program.java.

Zde vidíme hotovou metodu ve třídě Zak.java, vypadá velmi jednoduše:



Pro doplnění ještě volání metody ze třídy Program.java a výstup na konzoli:



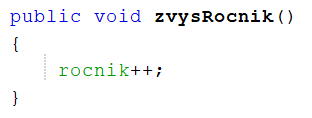
Další metoda zvysRocnik bude mít za úkol postoupit žáka do následujícího ročníku.

Ročník dle zadání zvyšujeme vždy o číslo 1, nemá tedy smysl, aby číslo ročníku bylo vstupem. Postačuje nám původní číslo ročníku již uložené v samotném objektu. Do metody prostě nebude vstupovat nic. Nesmíme ale zapomenout napsat za název metody **prázdné závorky**.

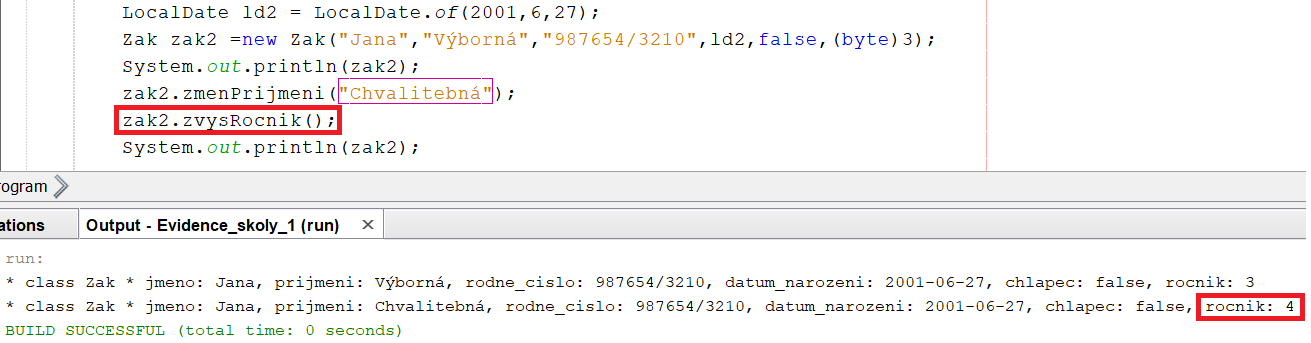
Výstupem nebude rovněž nic. Metoda proměnnou pouze modifikuje. V zadání nebylo o zpětném výpisu ani slovo. Výstup tedy v hlavičce označíme jako **void**.

Nezapomeneme opět připsat modifikátor přístupu **public**.

Vytvořili jsme tedy ve třídě Zak metodu, která nemá vstupy ani výstup:



Zde je zobrazeno volání metody zvysRocnik a rovněž důkaz správného fungování v podobě volání metody toString:



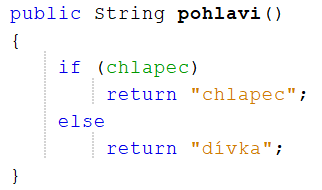
Další veřejná metoda nazvaná **pohlavi** nám bude vracet pohlaví žáka. Na základě stavu proměnné chlapec vracíme za true řetězec „chlapec“ a za false řetězec „dívka“.

Do metody tedy nebude vstupovat žádná hodnota, neboť proměnnou chlapec už má objekt k dispozici a žádnou jinou pro vyhodnocení pohlaví nepotřebuje.

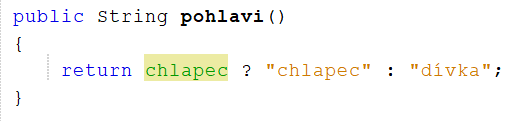
Z metody bude vystupovat text určující pohlaví žáka, tedy objekt typu String.

Mějme na paměti, že pokud v hlavičce diktujeme výstup, v těle metody musí tato výstupní hodnota:

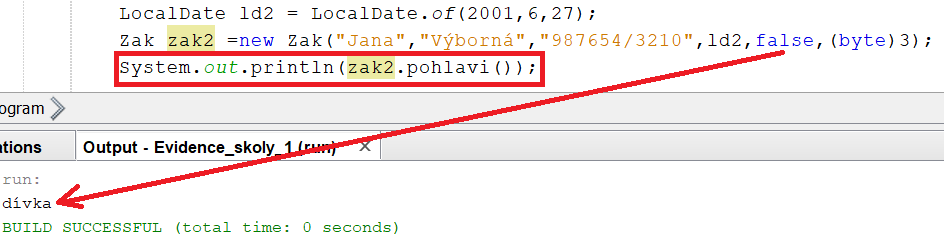
* být označena klíčovým slovem return
* existovat za každých okolností – není možné vynechat větev else. Za určitých okolností by totiž výstup neexistoval a kompilátor by Vám dal jasně najevo, že se mu tento stav nelíbí.



Pokud máme znalosti, lze tělo metody zapsat i pomocí jednoduchého ternárního operátoru:



Zde vidíte volání metody. Jelikož samotná metoda neřeší výpis textu na konzoli (což považuji za optimální), nesmíme opomenout výstup zobrazit ve třídě Progam.java pomocí příkazu System.out.println

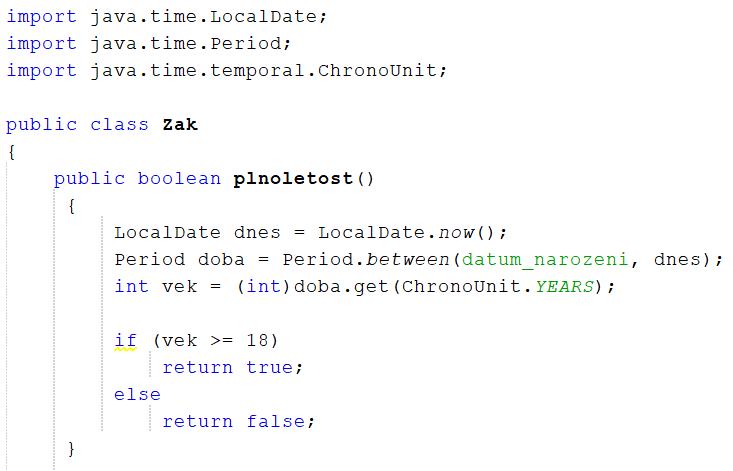


Poslední metoda bude nejobtížnější. K realizaci potřebujeme znalost práce s třídou LocalDate. Metoda plnoletost nám vrátí pravdivostní hodnotu true/false dle proměnné datum\_narozeni.

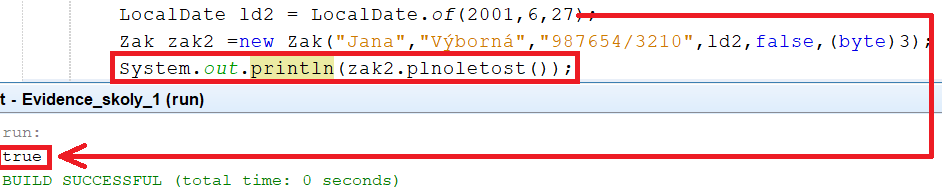
Datum narození žáka má objekt k dispozici, nebude tedy vstupem. Potom už potřebujeme k výpočtu pouze aktuální datum. To lze ovšem jednoduše zjistit v samotném těle metody. Vstupy tedy nemusíme řešit.

Výstupem rozumíme true, pokud je žák plnoletý a false, pokud plnoletý není. Na místo výstupu tedy uvedeme datový typ boolean.

Tělo metody není složité, ale žádá si určité znalosti týkající se práce s časovými intervaly. Pro správné fungování je nezbytné naimportovat v horní části třídy Zak příslušné balíčky:



Volání metody z Program.java je již jednoduché. Vypisujeme na konzoli výstup, který nám metoda vrací. V našem případě je žákyně již plnoletá:



V příští kapitole si ukážeme metody typické pro programy konstruované v Javě, metody get a set