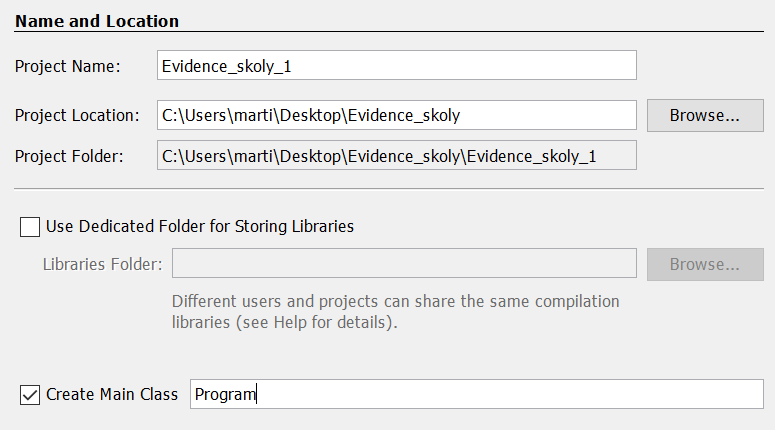
# 2. Třída Žák

Nyní se pustíme do samotného programování. Veškeré aspekty OOP v Javě si osvojíme na jediném příkladu, který budeme postupně programovat a upravovat. Vytvoříme evidenci školy.

Program budeme tvořit takovým způsobem, aby co nejlépe vyhovoval nasazení ve školním prostředí  
a zároveň abychom se z něj co nejvíce naučili. Nebudeme tedy programovat komplexní školní systém vhodný pro plošné nasazení v základním a středním školství, nicméně, jak uvidíte na závěr, program bude funkční i použitelný.

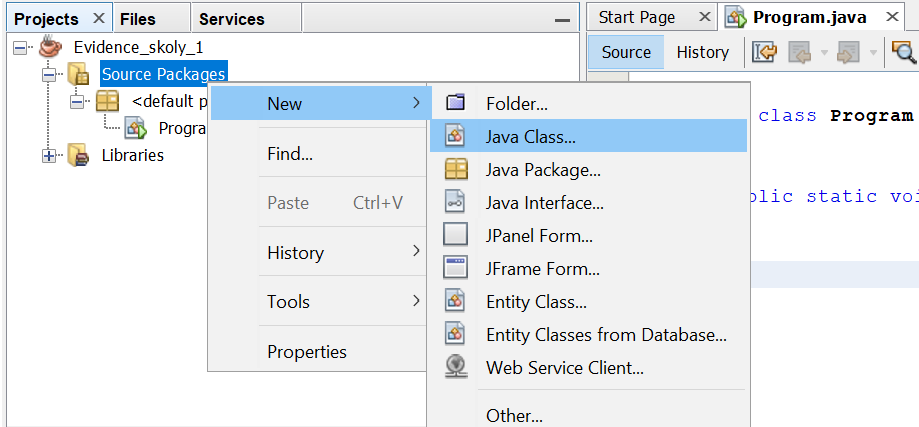
Veškeré screeny pocházejí z programu Apache NetBeans. Pokud chcete, můžete k programování použít právě toto prostředí. Nezapomeňte na instalaci JDK (Java Development Kid).

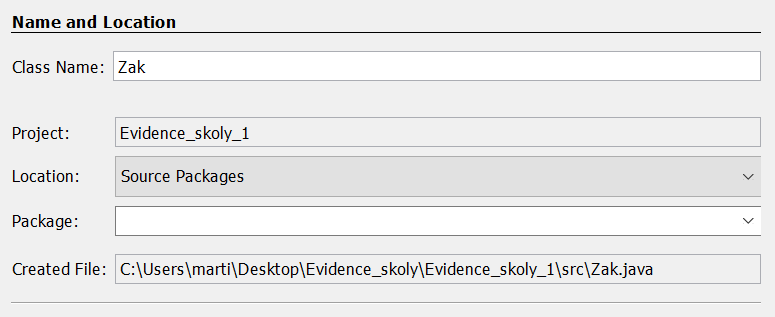


Vytvořili jsme projekt Evidence\_skoly\_1, kde zatím jedinou třídou je hlavní třída programu - Program.java. Třída pak obsahuje jedinou metodu – main. Kód uvnitř této metody se vykoná právě při spuštění programu. Pokud se vyznáte v tvorbě internetových stránek, je to něco jako soubor index.html.

Není automaticky dané, že hlavní třída či metoda se automaticky vygeneruje. Každé IDE (vývojové prostředí) se chová jinak. Pokud tomu tak není tvoříme kód ručně.

Pokračujeme a tvoříme naši první vlastní entitu – třídu pro ukládání žáků. Nepsaným pravidlem pro názvy tříd je první velké písmeno v názvu a rovněž vynechání diakritiky. Naši první třídu tedy pojmenujeme Zak.java.

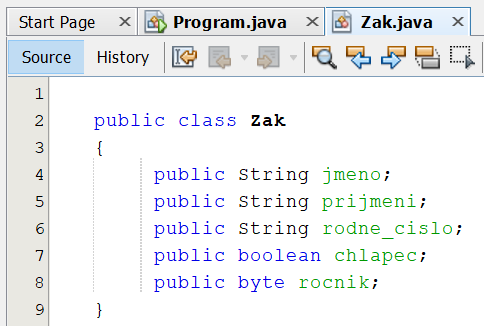




Rozmyslíme si atributy, které by měl náš žák mít uloženy. Tvorbu classy vždy přirovnávám k tvoření vstupního papírového formuláře pro nového žáka přijímaného do školského zařízení. Které informace od něj požadujeme, abychom jej mohli plně začlenit do systému školy?

Zdůrazňuji, že zde tvoříme **třídu** – předpis. Teprve až žák náš pomyslný vstupní formulář vyplní a odevzdá, získáváme tzv. **objekt – instanci dané třídy**.

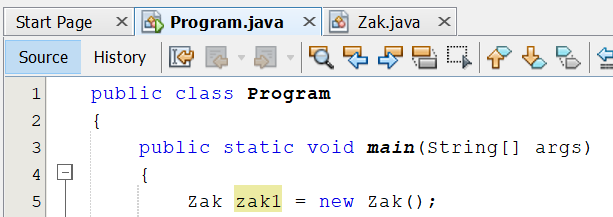
Náš žák bude prozatím definován následujícími atributy:

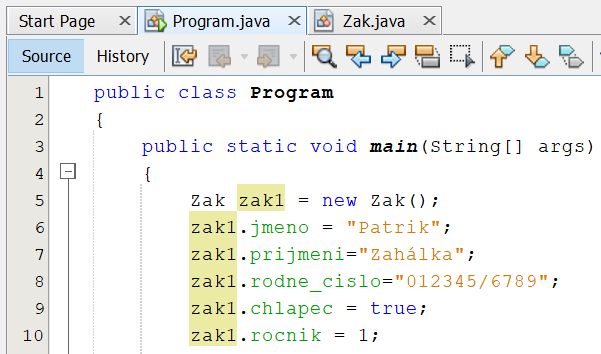


## Komentář

* String není jednoduchý datový typ, nýbrž třída (řetězec znaků). Proto je oproti datovým typům název psán velkým písmenem. S takovými třídami se sem tam během studia potkáme.
* Rodné číslo nemá formát čísla a ani není použito k výpočtům, proto pro jeho ukládání nevolíme int.
* Co se týká pohlaví, náš školský systém stále používá dělení na chlapce a dívky, tedy na 2 pohlaví. Pro jednoduchost jsem tedy zvolil logický datový typ boolean, kdy false = dívka a true = chlapec. Šlo by samozřejmě použít i String, ale v tomto případě by se jednalo a zbytečně složitou práci s pamětí. Lze použít i výčtový datový typ.
* Pro uložení ročníku jsem se z důvodu úspornosti rozhodl použít datový typ byte. Neukládáme velká ani záporná čísla, tedy jeho použití nic nebrání.
* Před každým atributem je psáno klíčové slovo public. To nám v podstatě říká, že daný atribut můžeme odkudkoli v programu libovolně vypisovat a měnit. Z jakékoli třídy. Program nám to značně zjednodušuje, nicméně použití není zcela vhodné. Riskujeme nechtěný či záměrný přepis atributu v paměti. Program pak může padat, vykazovat chyby či nesoulad v uložených informacích.  
  Pro jednoduchost necháme aktuálně atributy jako public, korekcí se budeme zabývat posléze.

## Náš první objekt

Nyní už můžeme zkonstruovat našeho prvního žáka. Vytvoříme instanci třídy – objekt. Jak? Přesuneme se do metody main třídy Program.java a zavoláme tzv. konstruktor. Jedná se o metodu, která právě z třídy vytváří objekty. Kde jsme konstruktor vzali? Vznikl automaticky s naší novou třídou Zak, aniž bychom museli cokoli programovat. Přesvědčte se, program nenapíše chybu:

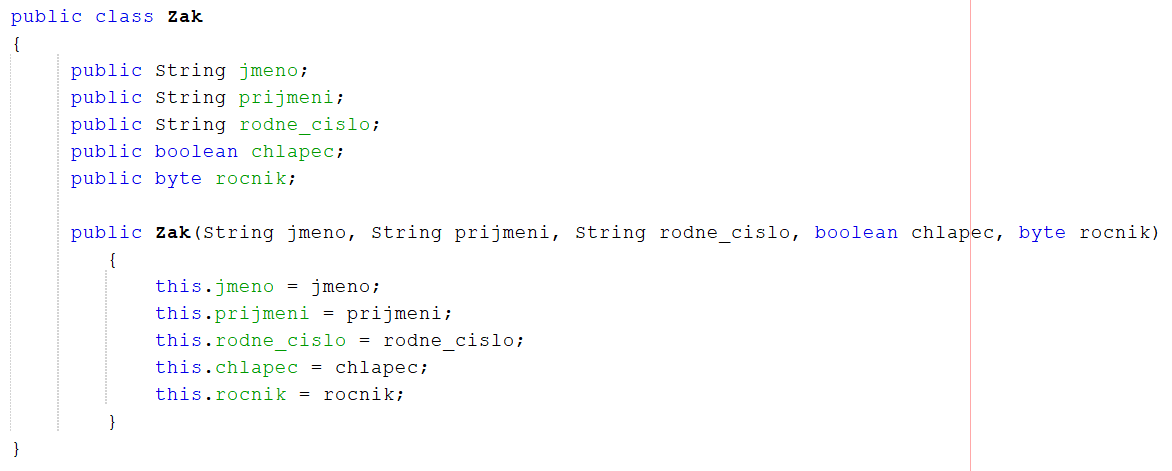
Co je ale v objektu zak1 aktuálně uloženo? Co je obsahem jeho atributů? Jsou to prostě defaultní hodnoty jednotlivých datových typů. Nic, co by bylo použitelné v praxi. Objekt zak1 už ale existuje a jeho atributy měnit lze. Umožňuje nám to klíčové slovo public uvedené u každého z atributů. Nejedná se  
o nejvhodnější postup tvorby objektů, ale OOP nám tento postup umožňuje:

V příští kapitole si ukážeme lepší – systematičtější postup, jak zkonstruovat nový objekt z naší třídy.

# 3. Kapitola - konstruktory

V minulé kapitole jsme konečně vytvořili první objekt třídy Zak. Učinili jsme tak pomocí defaultního konstruktoru, který každá třída obsahuje, aniž bychom jej museli definovat. Pro plnění atributu jsme pak použili tečkovou konvenci, např. zak1.rocnik = 1.

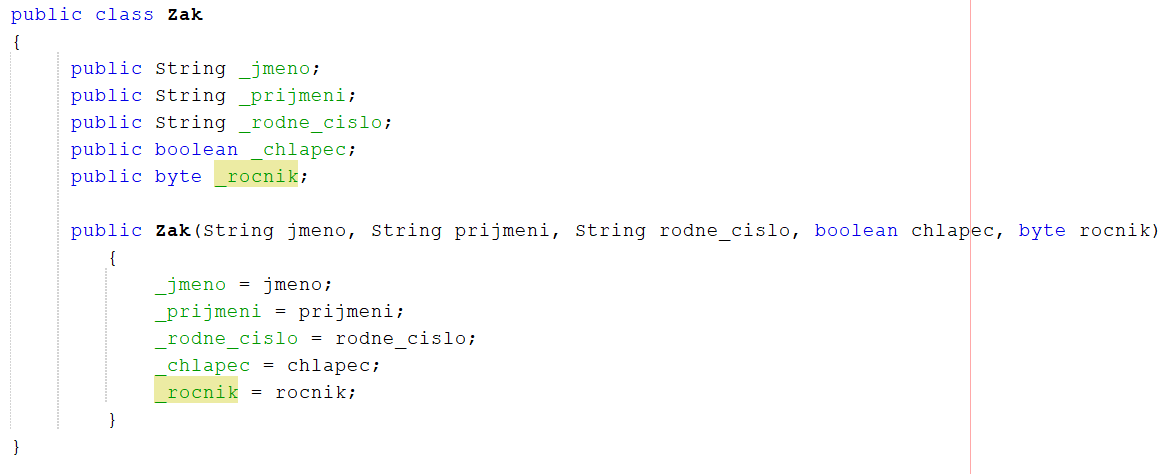
V této kapitole se podíváme na vhodnější postup při zakládání objektů – definování vlastních konstruktorů.

Rovnou si napíšeme, jak by takový konstruktor ve třídě Zak v našem případě vypadal:

## Pravidla pro tvorbu konstruktoru:

* Píšeme jej dovnitř dané třídy (jako každou jinou metodu).
* Měl by být public (veřejný), aby bylo možno konstruktor zavolat a objekt tak zkonstruovat i z jiné třídy, především z třídy Program.java.
* Má vždy stejný název, jako samotná třída (včetně malých a velkých písmen), tedy Zak.
* Do závorky v hlavičce konstruktoru píšeme seznam proměnných, pomocí kterých chceme objekt definovat. Mohou to být všechny atributy (jako v případě našeho příkladu) ale také nemusí.
* Ve složených závorkách je zpravidla umístěn kód, kdy hodnoty formálních proměnných (z hlavičky konstruktoru) ukládáme do skutečných proměnných daného objektu (označených slovem **this**)

Poznámka: Pokud bychom se chtěli vyhnout použití klíčového slova this, je možné názvy formálních a skutečných proměnných od sebe odlišit jiným způsobem, například takto za pomoci podtržítka:

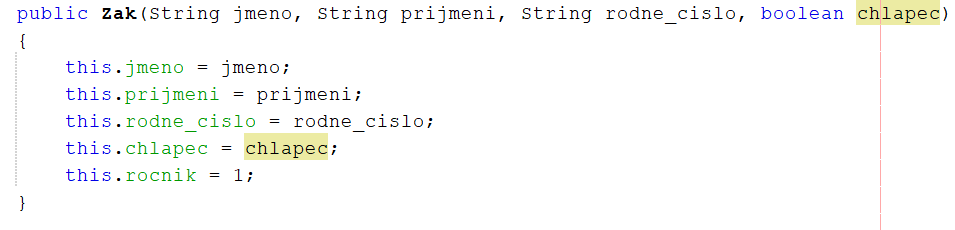


Konstruktor máme hotov a lze jej tedy zavolat z Program.java a vytvořit tak nového žáka.



## Přetížený konstruktor

Je pravdou, že ve škole drtivé množství nově přidaných žáků nastupuje právě do 1. ročníku. Je tedy zbytečné, aby tuto hodnotu uživatel byl nucen na vstupu neustále dokola opisovat.

Existuje zajímavé řešení, kdy vytvoříme ještě jeden náš konstruktor, ale na vstupu vynecháme právě ročník, takže uživatel jej nebude muset zadávat. V těle metody pak automaticky přiřadíme do proměnné hodnotu 1:

Získali jsme druhý konstruktory, které se stejně jmenuje, ale liší se ve vstupních parametrech. Tuto metodu nazýváme přetíženým konstruktorem.

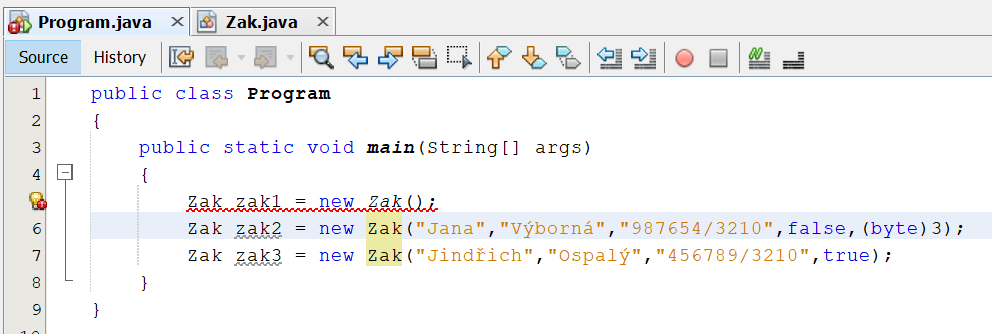
Nyní můžeme konstruktor jednoduše zavolat z třídy Program.java.



Přetížených konstruktorů (a tedy možností, jak zkonstruovat objekt) může být velké množství. Představte si situaci, kdy například chcete žáka vytvořit, ale nemáte k němu veškeré informace. Například neznáme mailovou adresu či telefonní kontakt na žáka. To sice není náš případ, nicméně často po nás může klient požadovat, aby bylo možné objekt vytvořit jen na základě omezené množiny údajů.

## Zrušení implicitního konstruktoru

Stojí za povšimnutí, že zatímco objekty zak2 a zak3 vytvořené pomocí nových konstruktorů jsme bez problémů vytvořili, zak1 nyní nově vykazuje chybu. Je to dáno skutečností, že v okamžiku vytvoření byť jen jediného vlastního konstruktoru ten implicitní přestane platit. Kompilátor ho již nevytvoří automaticky.



Pokud bychom chtěli z jakéhokoli důvodu tento „prázdný“ konstruktor zachovat, není problém jej do třídy Zak doprogramovat. Vytvoříme tím pouze další z přetížených konstruktorů. Konstrukce je snadná:



Nyní známe pojem konstruktor, víme jak se programuje a jak používá k vytvoření nového objektu. Známe pojmy implicitní a přetížený konstruktor.

V další kapitole se pokusíme přidat do naší nové třídy další, již existující třídu.