Téma 13: Datový typ řetězce v jazyce Java

V Javě jsou **řetězce** prezentovány v objektu typu String, který v sobě uchovává řetězec charů. Pokud tedy chci vytvořit řetězec staří zadat `String můj_řetězec =“Hello word“;` jak je vidět řetězce se v Javě zapisují do uvozovek. Nový řetězec mohu vytvořit také pomocí příkazu `String můj_řetězec = new String(“Hello world“);`.

S objekty typu String lze **sčítat** pomocí operátoru plus `novy = “Hello word“ + “Dobry den“;`, nebo mohu využít metodu instancí typu String concat(String s) - `novy = “Hello word“.concat(“Dobry den“);`. Pokud řetězce sčítám, neměním původní instanci, ale pouze vytvářím novou instanci. Což je celkem náročná operace, a pokud se provádí častěji, je lepší využít StringBuffer, který má tuto nepříjemnost vychytanou.

Pokud chci dva řetězce **porovnat**, nemohu použít dvě rovnítka ==, ale musím použít metodu equals(String s), jelikož dvě rovnítka by vyzkoušela, zda se jedná o tentýž objekt `boolean stejny = “Hello word“.equals(“Dobry den“).`

Pokud chci z řetězce **vytáhnout jen jeho část**, musím použít metodu subString(int start, int end), kde start je počáteční znak odkud se má začít řetězec vybírat včetně indexu počátku, end je index, který značí, pokud se má řetězec vybírat, tentokrát znak s indexem endu není vybrán. `String ahoj = "Dobry den, pane uciteli Johanovsky".substring(0, 4);.`

Řetězec je vlastně pole znaků, a také pomocí metody `charAt(int i)` **můžeme přistupovat** k jednotlivým znakům řetězce. První znak má index 0, poslední `size() – 1`.

Jak jsem zmínil v minulém odstavci velikost řetězce zjistit pomocí metody `size()`.

Třída StringBuffer je třída pro řetězce, která uchovává větší pole a když dojde, tak ho zvětší, plus si pamatuje operace s řetězcem a pomocí speciální metody určí vždy, když je příliš velký nebo malý novou velikost pole.

Bezparametrická metoda toString() slouží k vytvoření řetězcové reprezentace objektu.

Je deklarována již ve třídě Object, pokud ji ve vlastní třídě nepřekryjeme, původní metoda vygeneruje takovýto řetězec: plné jméno třídy@adresa uložení instance v paměti

Příklad bez překrývání toString() ve třídě Clovek:

```
...
Clovek c = new Clovek("Jan Novák");
System.out.println(c.toString());
```