Objektovo orientované programovanie

(reťazce, polia, matematické funkcie, náhodné čísla, informácie o systéme)

2. prednáška (2.časť)

Vladislav Novák
FEI STU v Bratislave
25.9.2014

Obsah

Reťazce

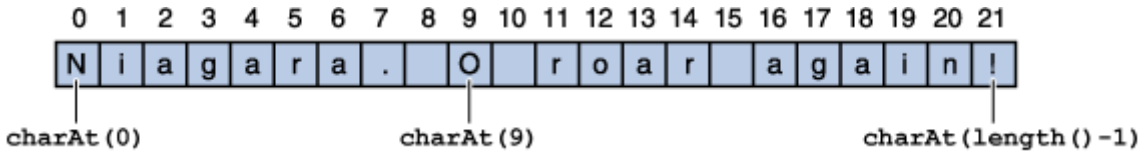
Inštancie (objekty) triedy `String` sú *nemenné* (*immutable*). To znamená, že po vytvorení objektu nemožno meniť jeho atribúty (nemožno meniť reprezentovaný textový reťazec).

Objekty typu `String` (aj všetky ostatné objekty) sa porovnávajú volaním metódy `equals`, nie operátorom porovnania == !

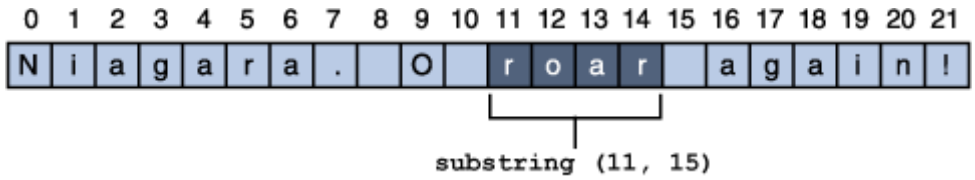
príklad:

```
String str1  = "abcd";
String str2  = "efgh";
String str3  = "efgh";
String str4a = "ef";
String str4b = "gh";
String str4  = str4a + str4b; //"efgh"

//správny spôsob porovnania (porovnanie podľa hodnoty objektu)
System.out.println(str1.equals(str2)); //false (rôzne hodnoty)
System.out.println(str2.equals(str3)); //true (rovnaké hodnoty)
System.out.println(str2.equals(str4)); //true (rovnaké hodnoty)

//nesprávny spôsob porovnania (porovnanie referencii na objekty)
System.out.println(str1 == str2); //false (rôzne referencie, iné objekty)

System.out.println(str2 == str3);  //môže vrátiť true, pretože vďaka optimalizácii
                                   //môžu str2 a str3 referencovať ten istý objekt

System.out.println(str2 == str4);  //false (hodnoty sú rovnaké, ale sú to rôzne objekty
                                   //str2 a str4 obsahujú rôzne referencie)
```

```
String anotherPalindrome = "Niagara. O roar again!";
char aChar = anotherPalindrome.charAt(9);
```




```
String anotherPalindrome = "Niagara. O roar again!";
String roar = anotherPalindrome.substring(11, 15);
```



Trieda `String` obsahuje statickú metódu `format()`, ktorá vytvára formátovaný reťazec rovnako ako rovnomenná metóda v triede `PrintStream` (str 5). Táto metóda je preťažená[1]:

```
static String  format(Locale l, String format, Object... args)
static String  format(String format, Object... args)
```

príklad:

```
public static void main(String[] args) {
    String ret;

    int i = 10;
    double d = 20;
    String s = "ahoj";

    ret = String.format("i = %d, d = %f, s = %s%n", i, d, s);

    System.out.println(ret);
}
```

výstup:

```
i = 10, d = 20,000000, s = ahoj
```

[1] Preťažená metóda znamená, že existuje viacero verzii jednej metódy, ktoré sa odlišujú parametrami. Inak povedané existuje viacero metód s rovnakým menom, ale rôznymi parametrami. Preťaženými metódami sa budeme zaoberať neskôr.

Niektoré ďalšie metódy:

`boolean` **`equals`**`(Object anObject)`
Porovná reťazec s iným objektom. Ak sú rovnaké vráti true, inak false.

`boolean` **`equalsIgnoreCase`**`(String anotherString)`
Porovná reťazec s iným reťazcom. Pri porovnávaní nezáleží na veľkosti písma.

`String` **`trim`**`()`
Vráti kópiu reťazca, v ktorej sú vynechané biele znaky zo začiatku a konca.

`int` **`compareTo`**`(String anotherString)`
Lexikograficky porovná reťazec s iným reťazcom.

`int` **`compareToIgnoreCase`**`(String str)`
Lexikograficky porovná reťazec s iným reťazcom, ale nezáleží na veľkosti písma.

`String[]` **`split`**`(String regex, int limit)`
Rozdelí reťazec podľa regulárneho výrazu. Vráti pole reťazcov.

`boolean` **`startsWith`**`(String prefix)`
Testuje či sa reťazec začína špecifikovaným prefixom.

`boolean` **`endsWith`**`(String suffix)`
Testuje či je reťazec ukončený špecifikovaným prefixom.

`int` **`indexOf`**`(int ch, int fromIndex)`
Vráti index prvého výskytu znaku `ch`. Hľadanie začína od znaku s indexom `fromIndex`.

`int` **`indexOf`**`(String str, int fromIndex)`
Vráti index začiatku prvého výskytu podreťazca `str`. Hľadanie začína od znaku s indexom `fromIndex`.

`char[]` **`toCharArray`**`()`
Vráti pole obsahujúce znaky reťazca.

`String` **`toLowerCase`**`()`
Vráti kópiu reťazca, v ktorej sú veľké písmená skonvertované na malé písmená.

`String` **`toUpperCase`**`()`
Vráti kópiu reťazca, v ktorej sú malé písmená skonvertované na veľké písmená.

`String` **`replace`**`(char oldChar, char newChar)`
Vráti kópiu reťazca v ktorej sú znaky `oldChar`, nahradené `newChar`.

`String` **`replaceAll`**`(String regex, String replacement)`
Vráti kópiu reťazca, v ktorej je nahradený každý podreťazec zodpovedajúci regulárnemu výrazu `regex`, reťazcom `replacement`.

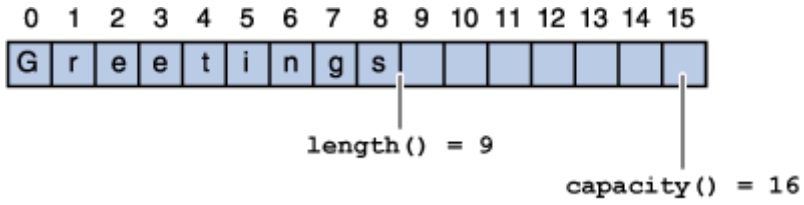
`String` **`replaceFirst`**`(String regex, String replacement)`
Vráti kópiu reťazca, v ktorej je nahradený prvý podreťazec zodpovedajúci regulárnemu výrazu `regex`, reťazcom `replacement`.

Trieda StringBuilder

Objekty typu `StringBuilder` sa podobajú objektom typu `String`, ale možno ich upravovať. `StringBuilder` reprezentuje postupnosť znakov, ktorej dĺžka a obsah sa môže meniť.

Dĺžka a kapacita

```
public static void main(String[] args) {
    StringBuilder sb = new StringBuilder();
    sb.append("Greetings");

    System.out.println(sb.length());   //9
    System.out.println(sb.capacity()); //16 (kapacita sa môže meniť)
}
```



Pridávanie znakov

Základnými operáciami ktoré nie sú dostupné v triede `String` sú metódy `append()` a `insert()`. Tieto metódu sú preťažené, aby mohli prijímať argumenty ľubovoľného typu. Metódy `append()` pridávajú znaky na koniec postupnosti, metódy `insert()` vkladajú znaky na určenú pozíciu.

príklad – manipulácia s reťazcom pomocou `StringBuilder`

```
public static void main(String[] args) {
    StringBuilder sb = new StringBuilder("prve druhe");
    System.out.println(sb); //prve druhe
    sb.append(" tretie");
    System.out.println(sb); //prve druhe tretie
    sb.insert(5, "AAAA");
    System.out.println(sb); //prve AAAAdruhe tretie
    sb.delete(2, 5);
    System.out.println(sb); //prAAAAdruhe tretie
    sb.setCharAt(4, 'B');
    System.out.println(sb); //prAABAdruhe tretie
    sb.replace(10, 14, "DD");
    System.out.println(sb); //prAABAdruhDDetie
    sb.reverse();
    System.out.println(sb); //eiteDDhurdABAArp
    System.out.println(sb.length());   //16
    System.out.println(sb.capacity()); //26
    sb.ensureCapacity(40);
    System.out.println(sb.capacity()); //54
}
```

Formátovaný výstup – trieda PrintStream

Objekt `System.out` reprezentujúci štandardný výstup je inštanciu triedy `PrintStream`. Táto trieda obsahuje dve formátovacie metódy `fromat()` a `printf()`, ktoré sú vzájomne ekvivalentné. Obidve metódy majú rovnakú syntax. Obidve metódy sú preťažené:

```
PrintStream format(Locale locale, String format, Object... args)
PrintStream format(String format, Object... args)
PrintStream printf(Locale locale, String format, Object... args)
PrintStream printf(String format, Object... args)
```

- argument `locale` určuje národné zvyklosti
- argument `format`, predstavuje formátovací reťazec
- argumenty `args` obsahujú hodnoty určené pre formátovací reťazec

Formátovací reťazec určuje ako budú formátované objekty `args`. Obsahuje neformátovaný text a špecifikátory formátu. Špecifikátory formátu začínajú znakom % a na konci majú konverziu. Znak konverzie určuje informácie o type argumentu, ktorý sa bude formátovať. Medzi percento (%) a konverziu možno vložiť príznaky a špecifikátory. Podrobná dokumentácie je v `java.util.Formatter`
http://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html

príklady konverzie:
`d` – desiatkové číslo
`f` – desatinne číslo
`s` – reťazec
`n` – znak nového riadku (lepšie použiť `%n` ako `\n`)

príklad:

```
//treba naimportovat:
import java.util.Locale;

public static void main(String[] args) {
  int i = 10;
  double d = 20.12345;
  String s = "ahoj";
  System.out.format("i = %d%n", i);  //"i = 10"
  System.out.format("i = %+d%n", i); //"i = +10"

  System.out.format("d = %f%n", d);     //"d = 20,123450"
  System.out.format("d = %.2f%n", d);   //"d = 20,12"
  System.out.format("d = %10.2f%n", d); //"d =       20,12"

  System.out.format(Locale.FRANCE, "d = %f%n", d);//"d = 20,123450"
  System.out.format(Locale.ENGLISH,"d = %f%n", d);//"d = 20.123450"

  System.out.format("s = %s%n", s); //"s = ahoj"

  System.out.format("i = %d, d = %f, s = %s%n", i, d, s);
  //"i = 10, d = 20,123450, s = ahoj"
}
```

Polia – trieda Arrays

celý názov: `java.util.Arrays`

```
import java.util.Arrays;

public class PoleTest1 {
    public static void main(String[] args) {
        String[] pole1 = {"prvy", "druhy", "treti", "stvrty"};
        String[] pole2;
        String[] pole3 = new String[5];

        pole2 = Arrays.copyOf(pole1, pole1.length);
        System.out.println(Arrays.toString(pole2));
        //[prvy, druhy, treti, stvrty]
        System.out.println(Arrays.equals(pole1, pole2));
        //true

        Arrays.sort(pole2);
        System.out.println(Arrays.toString(pole2));
        //[druhy, prvy, stvrty, treti]
        System.out.println(Arrays.equals(pole1, pole2));
        //false

        int index = Arrays.binarySearch(pole2, "stvrty");
        System.out.println("index="+index+",hodnota="+pole2[index]);
        //index=2,hodnota=stvrty
        System.out.println(Arrays.equals(pole1, pole2));
        //false

        Arrays.fill(pole3, "neznamy");
        System.out.println(Arrays.toString(pole3));
        //[neznamy, neznamy, neznamy, neznamy, neznamy]

        String[][] ps = { {"jeden", "dva" } , { null , "tri" } };
        System.out.println(Arrays.toString(ps));
        //[[Ljava.lang.String;@c17164, [Ljava.lang.String;@1fb8ee3]
        System.out.println(Arrays.deepToString(ps));
        //[[jeden, dva], [null, tri]]
    }
}
```

Pred volaním `binarySearch` musí byť pole usporiadané.

Podobne ako `deepToString` existujú aj metódy `deepEquals` a `deepHashCode`.

Trieda Math

Trieda `java.lang.Math` obsahuje konštanty a metódy pre matematické výpočty. Všetky metódy triedy `Math` sú statické.

Trieda obsahuje konštanty:
`Math.E` – základ prirodzeného logaritmu (2.718281828459045)
`Math.PI` – ludolfovo číslo (3.141592653589793)

Metódy:

Základné matematické metódy

Metóda	Popis

`double abs(double d)`
`float abs(float f)`
`int abs(int i)`
`long abs(long lng)`
	Vrátia absolútnu hodnotu argumentu.

`double ceil(double d)`	Vráti najmenšie celé číslo, ktoré je väčšie, alebo rovné argumentu. Návratový typ je double.
`double floor(double d)`	Vráti najväčšie celé číslo, ktoré je menšie, alebo rovné argumentu. Návratový typ je double.
`double rint(double d)`	Vráti celé číslo, ktoré zaokrúhlením argumentu. Návratový typ je double.
`long round(double d)` `int round(float f)`	Vráti celé číslo, ktoré zaokrúhlením argumentu. Návratový typ je long, alebo int.

`double min(double arg1, double arg2)`
`float min(float arg1, float arg2)`
`int min(int arg1, int arg2)`
`long min(long arg1, long arg2)`
	Vráti minimum dvoch argumentov.

`double max(double arg1, double arg2)`
`float max(float arg1, float arg2)`
`int max(int arg1, int arg2)`
`long max(long arg1, long arg2)`
	Vráti maximum dvoch argumentov.

Exponenciálne a logaritmické metódy

Metóda	Popis
`double exp(double d)`	Vráti mocninu základu prirodzeného logaritmu.
`double log(double d)`	Vráti prirodzený logaritmus argumentu.
`double pow(double base, double exponent)`	Vráti `base` umocnený na `exponent`.

`double sqrt(double d)`	Vráti druhú odmocninu argumentu.

Trigonometrické metódy

Metóda	Popis
`double sin(double d)`	Vráti sínus argumentu.
`double cos(double d)`	Vráti kosínus argumentu.
`double tan(double d)`	Vráti tangens argumentu.
`double asin(double d)`	Vráti arkus sínus argumentu.
`double acos(double d)`	Vráti arkus kosínus argumentu.
`double atan(double d)`	Vráti arkus tangens argumentu.
`double atan2(double y, double x)`	Konvertuje súradnice `x`, `y` z karteziánskej súradnicovej sústavy do polárnej súradnicovej sústavy `(r, theta)` a vráti `theta`.
`double toDegrees(double d)` `double toRadians(double d)`	Konverzia uhlov medzi stupňami a radiánmi.

príklad - zaokrúhľovanie:

```
public static void main(String[] args) {
   double a = 4.2;
   double b = 4.8;

   System.out.println(Math.ceil(a)); //5.0
   System.out.println(Math.ceil(b)); //5.0

   System.out.println(Math.floor(a)); //4.0
   System.out.println(Math.floor(b)); //4.0

   System.out.println(Math.rint(a)); //4.0
   System.out.println(Math.rint(b)); //5.0

   System.out.println(Math.round(a)); //4L
   System.out.println(Math.round(b)); //5L
}
```

Náhodné čísla

Jednoduchší spôsob:

Trieda `Math` obsahuje metódu `random()`, ktorá vracia pseudonáhodné číslo v rozmedzí 0.0 a 1.0. Rozsah zahrňuje číslo 0.0, ale nezahrňuje číslo 1.0. Pri prvom volaní táto metóda vytvorí inštanciu triedy `java.util.Random`, ktorá je potom použitá aj pri ďalších volaniach.

```
public static void main(String[] args) {
    for(int i=0; i<5; i++) {
        System.out.println(Math.random());
    }

    for(int i=0; i<5; i++) {
       System.out.println((int)(Math.random()*10));
    }
}
```

V prípade väčších nárokov:

Na generovanie náhodných čísiel slúži trieda `java.util.Random`.

```
//treba naimortovat:
import java.util.Random;

public static void main(String[] args) {
    //rovnomerne rozdelenie(nextInt existuje aj bez argumentu)
    Random rnd1 = new Random();
    for(int i=0; i<5; i++) {
        System.out.println(rnd1.nextInt(10));
    }
    System.out.println();

    //rovnomerne rozdelenie
    Random rnd2 = new Random();
    for(int i=0; i<5; i++) {
        System.out.println(rnd2.nextDouble());
    }
    System.out.println();

    //prirodzene rozdelenie(stred.hodn.: 0, štand.odchylka: 1)
    Random rnd3 = new Random();
    for(int i=0; i<5; i++) {
        System.out.println(rnd3.nextGaussian());
    }
    System.out.println();

    //nasledujuce dva cykly vygeneruju tu istu postupnost
    //(rovnaka inicializacia)
    final long INICIALIAZACIA= 100;
    Random rnd4 = new Random(INICIALIAZACIA);
    for(int i=0; i<5; i++) {
        System.out.println(rnd4.nextInt());
    }
    System.out.println();

    rnd4.setSeed(INICIALIAZACIA);
    for(int i=0; i<5; i++) {
        System.out.println(rnd4.nextInt());
    }
```

```
    System.out.println();
}
```

Zisťovanie informácii o systéme

Metódy triedy `System`

```
static Properties     getProperties()
static String         getProperty(String key)
static String         getProperty(String key, String def)
```

príklad:

```
public class MainSystemGetProperty {
    private static void printProperty(String property) {
        System.out.println(property + "= \"" +
                           System.getProperty(property) + "\"");
    }

    public static void main(String[] args) {
        printProperty("line.separator");
        printProperty("file.separator");
        printProperty("path.separator");
        printProperty("user.dir");
        printProperty("user.name");
        printProperty("java.version");
        printProperty("java.class.path");
        printProperty("java.class.path");
        printProperty("java.class.version");
        printProperty("java.specification.name");
        printProperty("java.ext.dirs");
        printProperty("os.name");
        printProperty("os.arch");
        printProperty("os.version");
        printProperty("java.ext.dirs");
    }
}
```

Viac informácii:
http://docs.oracle.com/javase/8/docs/api/java/lang/System.html#getProperties--