java.lang.String

šaltinis:
http://www.fatih.edu.tr/~moktay/2009/spring/ceng104/The.String.Class.ppt

Turinys:

- Tiesioginės eilutės
- Eilučių konstruktoriai
- Dažniausiai naudojami String klasės metodai
- Eilučių nekeičiamumas
- Skaičiaus vertimas eilute

Apie String klasę

- String tipo klasės objektas apibrėžia eilutės simbolių seką
- String klasė yra *java.lang* pakete, kuris nereikalauja jo importavimo sakinio.
- Kaip ir kitos klasės, String turi aibę konstruktorių ir metodų.
- Skirtingai nuo kitų klasių, String objektams galima taikyti dvi operacijas: + ir +=, kurios apjungia eilutes.

Tiesioginės (*literal*) eilutės

- tai bevardžiai String objektai.
- apibrėžiamos parašant tekstą kabutėse: “*Tai tiesioginė eilutė*”
- nekviečia konstruktoriaus.
- gali būti priskirtos String tipo kintamiesiems.
- gali būti perduotos konstruktorių ir eilučių parametrais.
- Turi aibę naudingų String klasės metodų.

Tiesioginių eilučių pavyzdžiai

```
//priskiriame tiesioginę eilutę kintamajam
String vardas = “Mindaugas”;

//kviečiame tiesioginės eilutės metodą
char pirmojiRaidė = “Mindaugas”.charAt(0);

//kviečiame String kintamojo metodą
char pirmasisSimbolis = vardas.charAt(0);
```

String tipo kintamuosius galima sudėti

```
int myInt=4;
 String anotherString=myString+"myInt yra "+myInt;
```

anotherString reikšmė bus "Sveikas! myInt yra 4". Kadangi anotherString yra objektas, jo elemntais galime manipuliuoti naudodami String clasės metodus.

Pavyzdžiui išskirsime pirmuosius aštuonis simbolius:

```
 String helloString = anotherString.substring(8);
```

String: Užduotis0

Turim:

String tekstas = “Java ”;

String prideti = “kalba”;

Sukonstruoti sakinį :

String sakinys1 ? ????; *// panaudoti* +

String sakinys2 ? ????; *// panaudoti* +=

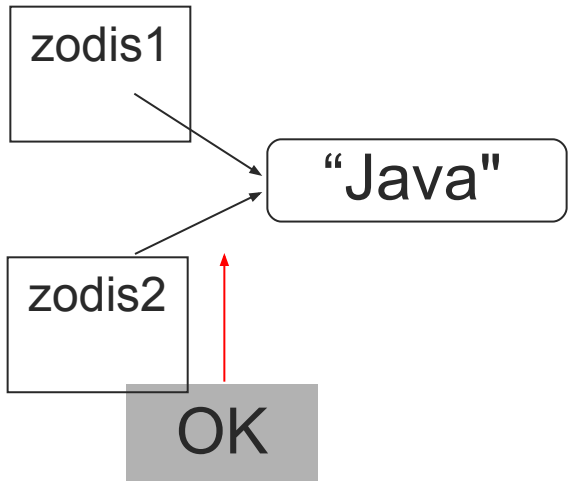
String nekintamumas

- Kartą sukurta, eilutė negali būti pakeista
- Objektai pasižymintys tokia savybe vadinami **nekintamais** (*immutable)*.
- Nekintami objektai patogūs tuo, kad visos nuorodos į juos yra saugios, nes nėra pavojaus, kad objekto turinys pakito ir skirtingos nuorodos žymi skirtingo turinio objektą.

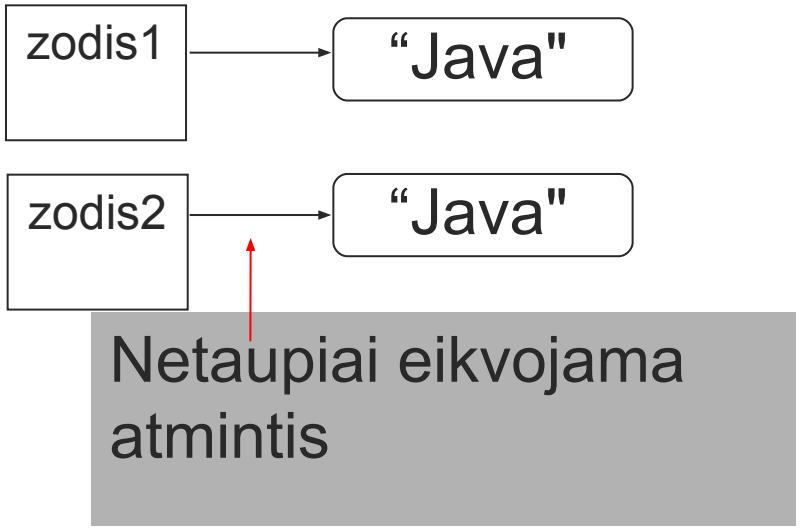
Nekintamų objektų privalumai

Naudoja mažiau atminties

```
String zodis1 = "Java";
String zodis2 = zodis1;
```



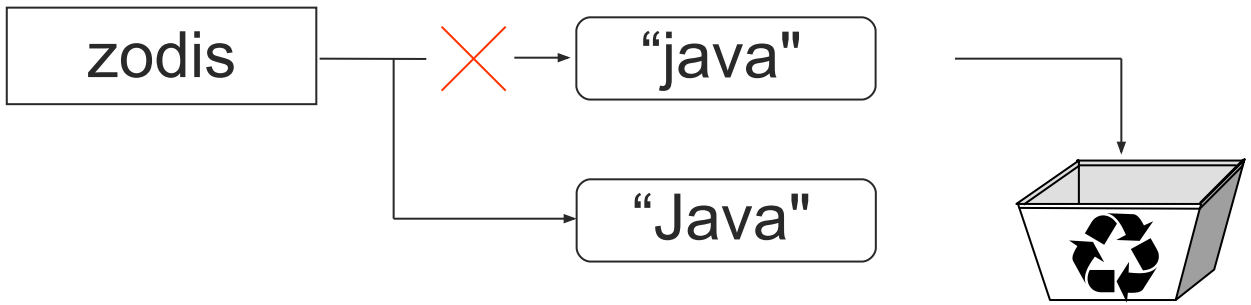

```
String zodis1 = “Java";
String zodis2 = new String(zodis1);
```



Nekintamumo trūkumai

Neefektyvu, kai jums reikia naujos panašaus turinio eilutės.

```
String zodis = “java";
char ch = Character.toUpperCase(zodis.charAt (0));
zodis =  ch + zodis.substring (1);
```



Tuščios eilutės

- Tuščia eilutė neturi nei vieno simbolio; jos ilgis lygus 0.

```
String word1 = "";
String word2 = new String();
```

Empty strings

- Tuščia eilutė skiriasi nuo neinicijuotos:.

```
private String errorMsg;
```

errorMsg yra **null**

Konstruktorius su tuščiu parametrų sąrašu ()

- Toks konstruktorius sukuria tuščią eilutę. Naudojamas retai.

```
String tuščia = new String();
```

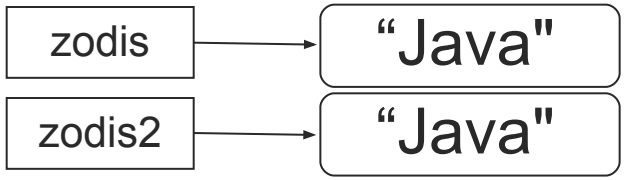
- Dažniau tokiu atveju naudojama tuščia tiesioginė eilutė.
- Tuščia eilutė dažnai yra analogas pradinės nulinės reikšmės kaupiant sumą.

```
String tuščia = “”;//tarp kabučių nieko nėra
```

Eilutės kopijavimas panaudojant konstruktorių

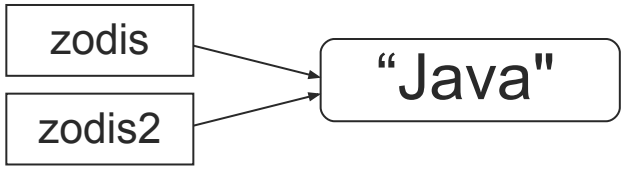
- Egzistuojančios eilutės kopija gaunama kreipiantis į konstruktorių perduodant parametru kopijuojamos eilutės objektą. Taip pat retai naudojama.
- Skiriasi nuo priskyrimo operatoriaus.

```
String zodis = new String(“Java”);
String zodis2 = new String(zodis);
```



Priskiriant: abu kintamieji tampa nuorodomis į tą pačią eilutę.

```
String zodis = “Java”;
String zodis2 = zodis;
```



String tipas

```
String myString = "Sveikas!";                    // ---------- 1 objektas

String myString = new String( "Sveikas!" );      // ---------- 2 objektai JDK7

                                1                2
```

Kiti konstruktoriai

Galima naudoti masyvą, kad sukurti eilutę.

```
char[] raides = {‘J’, ‘a’, ‘v’, ‘a’};
String zodis = new String(raides);//”Java”
```

```
byte[] baitai = {(byte)0x4a, (byte)0x61,
(byte)0x76, (byte)0x61};
String zodis = new String(baitai);//”Java”
```

Metodai — ***length***, ***charAt***

int length();

- Grąžina kiek eilutėje yra simbolių

char charAt(i);

- Grąžina simbolį esantį i-ojoje pozicijoje.

Simboliai, kaip ir masyvų atveju, numeruojami pradedant nuo nulinės pozicijos.

	Grąžina:
”Žodis".length();	5
”Žodis".charAt (2);	’d'

String: Užduotis2

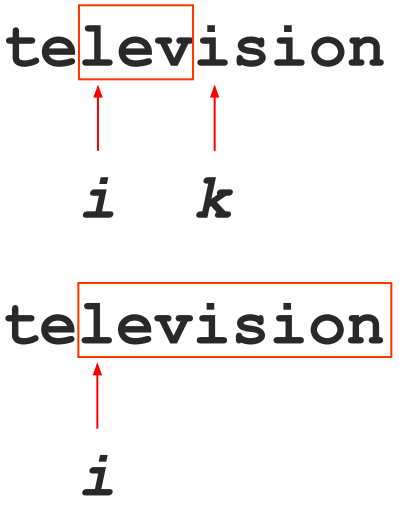
String tekstas = “Java kalba”;

length ?
charAt ?

substring metodas

Grąžina naują eilutę kopijuojant nurodytų simbolių seką.

- String subs = zodis.**substring** (i, k);
- grąžina eilutę sudarytą iš simbolių esančių nuo **i** iki **k-1** pozicijos
- String subs = zodis.**substring** (i);
  - grąžina eilutę pradedant nuo i-osios pozicijos iki galo.




```
”television".substring (2,5);
“immutable".substring (2);
“zodis".substring (9);
```

Grąžina:

→ “lev"

→ “mutable"

→ "" (tuščia eilutė)

String: Užduotis2

String s = “Mano Vardas Java!”

- **.substring** (i);
- **.substring** (i, k);

Sujungimo metodai

String zodis1 = “ap”, zodis2 = “gal”; zodis3 = “voti”;
int num = 2;

String result = zodis1 **+** zodis2;

//apjungiami zodis1 ir zodis2 “apgal“

String result = zodis1.**concat** (zodis2);

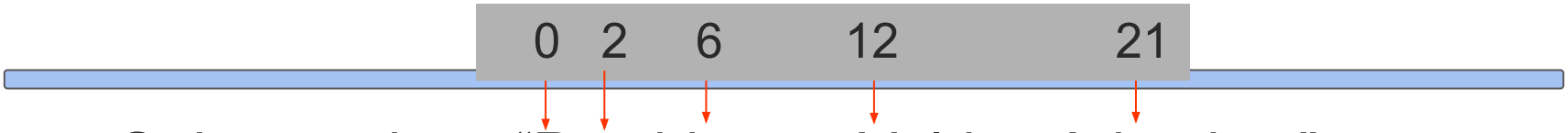
//tas pats kaip zodis1 + zodis2 “apgal“

result += zodis3;

//prijungia zodis3 prie result “apgalvoti”

result += num; //konvertuoja num į eilutę
//ir prijungia ją prie result “apgalvoti2”

indexOf metodas



String vardas =“Prezidentas Valdas Adamkus";

	Grąžina:
vardas.indexOf (‘P');	0
vardas.indexOf (‘e');	2
vardas.indexOf (“Valdas");	12
vardas.indexOf (‘e', 3);	6
vardas.indexOf (“Jonas");	–1
vardas.lastIndexOf (‘a');	21

(ieško pradedant 3-ia pozicija)

(nerasta)

Eilučių sutapimo (***equals***) metodas

boolean b = zodis1.**equals**(zodis2);

grąžina **true** jei zodis**1** sutampa su zodis**2**

boolean b = zodis1.**equalsIgnoreCase**(zodis2);

grąžina **true** jei zodis**1** sutampa su zodis**2** ignoruojant didžiųjų/mažųjų raidžių skirtumą

```
b = “Tiesa”.equals(“Tiesa”);//true
b = “Tiesa”.equals(“tiesa”);//false
b = “Tiesa”.equalsIgnoreCase(“tiesa”);//true
```

```
if(team.equalsIgnoreCase(“tiesa”))
   System.out.println(“Go You “ + team);
```

String: Užduotis3

1. `equals`
2. **equalsIgnoreCase**

Paprašyti vartotojo įvesti du sakinius, juos priskirti kintamiesiams.

Palyginkite ar įvestos frazės yra lygios, ar yra identiškos ignoruojant mažųjų/didžiųjų raidžių skirtumą

Eilučių palyginimo (***comapreTo***) metodas

int diff = zodis1.**compareTo**(zodis2);
grąžina “skirtumą” zodis**1** – zodis**2**
int diff = zodis1.**compareToIgnoreCase**(zodis2);
grąžina “skirtumą” zodis**1** – **zodis2**,
ignoruojant didžiųjų/mažųjų skirtumą

Dažnai konkreti “skirtumo” zodis**1 - zodis2** reikšmė nenaudojama, o tik skirtumo ženklas. Jei „skirtumas“ neigiamas, zodis1 eina prieš zodis2, lygus nuliui - zodis1 ir zodis2 sutampa, teigiamas - zodis1 eina po zodis2. Pvz.

```
if (zodis1.compareTo(zodis2) > 0){
      //zodis1 eina po zodis2…
}
```

Palyginimo pavyzdžiai

```
//neigiami skirtumai
diff = “apple”.compareTo(“berry”);//a prieš b
diff = “Zebra”.compareTo(“apple”);//Z prieš a
diff = “dig”.compareTo(“dug”);//i prieš u
diff = “dig”.compareTo(“digs”);//dig trumpesnis
```

```
//nulinis skirtumas
diff = “apple”.compareTo(“apple”);//sutampa
diff = “di”.compareToIgnoreCase(“DI”);//sutampa
```

```
//teigiami skirtumai
diff = “berry”.compareTo(“apple”);//b po a
diff = “apple”.compareTo(“Apple”);//a po A
diff = “BIT”.compareTo(“BIG”);//T po G
diff = “huge”.compareTo(“hug”);//huge ilgesnis
```

String: Užduotis4

compareTo
compareToIgnoreCase

trim metodas

String zodis2 = zodis**1.trim** ();
grąžina naują eilutę sudarytą iš zodis**1**
atmetus jos pradžioje ir gale esančius tarpus.
Viduriniai tarpai neatmetami.

```
String zodis1 = “  Sveikas,  Jonai !   “;
String zodis2 = zodis1.trim();
//zodis2 yra “Sveikas,   Jonai !” – be tarpų galuose
//zodis1 lieka “  Sveikas,  Jonai !   “ – su tarpais
```

replace metodas

String zodis2 = zodis1.**replace**(senaChar, nujaChar);
grąžina naują eilutę, kurioje zodis**1** eilutėje visi **senaChar** simboliai pakeisti **naujaChar** simboliu

```
String zodis1 = “mama“;
String zodis2 = zodis1.replace(‘m’, ‘p’);
//zodis2 yra “papa”, o zodis1 lieka “mama“
```

Didžiųjų/mažųjų raidžių keitimas

String zodis2 = zodis1.**toUpperCase**();
String zodis3 = zodis1.**toLowerCase**();

grąžina naują eilutę pakeičiant zodis**1** mažąsias (didžiąsias) raides didžiosiomis (mažosiomis)

```
String zodis1 = “SveiKas“;
String zodis2 = zodis1.toUpperCase();//”SVEIKAS”
String zodis3 = zodis1.toLowerCase();//”sveikas”
//zodis1 lieka “SveiKas“
```

Pakeitimai !!!

Pvz, kad pakeisti zodis1 jo didžiosiomis raidėmis, reikia parašyti.

```
zodis1 = zodis1.toUpperCase();
```

Tipinė klaida:

```
zodis1.toUpperCase();
```

zodis1 lieka nepakeistu.

String: Užduotis5

1. trim
2. replace*
3. toUpperCase
4. toLowerCase

Wrapper classes

Primitives & Wrappers

- Java has a *wrapper* class for each of the eight primitive data types:

Primitive Type	Wrapper Class	Primitive Type	Wrapper Class
boolean	Boolean	float	Float
byte	Byte	int	Integer
char	Character	long	Long
double	Double	short	Short

Use of the Wrapper Classes

- Java's *primitive* data types (boolean, int, etc.) are not classes.
- Wrapper classes are used in situations where objects are required, such as for elements of a Collection:

```
List<Integer> a = new ArrayList<Integer>();
methodRequiringListOfIntegers(a);
```

Object => Value

- Each wrapper class Type has a method typeValue to obtain the object's value:

```
Integer i1 = Integer.valueOf(42);
Boolean b1 = Boolean.valueOf("false");
System.out.println(i1.intValue());
System.out.println(b1.intValue());

=>
42
false
```

String => value

- The Wrapper class for each primitive *type* has a method parse*Type*() to parse a string representation & return the literal value.

```
Integer.parseInt("42")      => 42
Boolean.parseBoolean("true")  => true
Double.parseDouble("2.71") => 2.71
//…
```

- Common use: Parsing the arguments to a program:

Sample values:

```
boolObj = new Boolean(Boolean.TRUE);
charObj = new Character('a');
byteObj = new Byte("100");
shortObj = new Short("32000");
intObj = new Integer(2000000);
longObj = new Long(50000000000000000L);
floatObj = new Float(1.42);
doubleObj = new Double(1.42);

printWrapperInfo(); //method to print objects above
```

Each Number Wrapper has a MAX_VALUE constant:

```
byteObj = new Byte(Byte.MAX_VALUE);
shortObj = new Short(Short.MAX_VALUE);
intObj = new Integer(Integer.MAX_VALUE);
longObj = new Long(Long.MAX_VALUE);
floatObj = new Float(Float.MAX_VALUE);
doubleObj = new Double(Double.MAX_VALUE);

printNumValues("MAXIMUM NUMBER VALUES:");
```

Skaičių vertimas tekstu

Yra trys būdai tai padaryti:

1. String s = "" + skaicius;

```
s = “” + 123;//”123”
```

2. String s = Integer.toString (i);
   String s = Double.toString (d);

```
s = Integer.toString(123);//”123”
s = Double.toString(3.14); //”3.14”
```

3. String s = String.valueOf (num);

```
s = String.valueOf(123);//”123”
```

Integer ir **Double** yra **int** ir **double** pirminių klasių analogai, kurie skaičius išreiškia objektais. Jie turi aibę naudingų statinių metodų, vienu kurių čia ir pasinaudojome.

String: Užduotis6

1. main metode paprašyti įvesti vartotjo skaičių ir jį priskirti String tekstas kintamajam
    a. scanner.nextLine(); !!!
    b. Paversti (pakeisti tipą) į: **int sk;**
    c. * 1000 išvesti rez.

Klausimai pakartojimui:

1. Kokiam paketui priklauso String klasė?
2. Kuo skiriasi String klasės objektai nuo kitų klasių?
3. Kokia reikšmė bus grąžinta: “Laba diena”.length()?
4. Kokie objektai vadinami “immutable” ?

Klausimai pakartojimui:

1. Kokie nekintanmų String tipo objektų Javoje privalumai?
2. Kokie nekintanmų String tipo objektų Javoje trūkumai?
3. Kaip apibrėžti tuščią eilutę?

Klausimai pakartojimui:

1. String miestas = "Vilnius";
   Ką grąžins miestas.charAt (2)?
2. miestas.substring(2, 4)?
3. miestas.lastIndexOf(‘i’)?
4. Ką atlieka ***trim*** metodas?

Klausimai pakartojimui:

1. “sam”.equals(“Sam”) grąžins ?
2. Koks bus reikšmės “sam”.compareTo(“Sam”) ženklas?
3. Kokia bus ***s*** reikšmė?
   s = “dėdė”.replace(‘d’, ‘m’);
4. Kaip pakeičia **s** eilutę metodo s.toUpperCase() iškvietimas ?
5. Koks paprasčiausias būdas skaičių paversti eilute (tekstu) ?

String klasės

```
public static void main(String[] args) {
    String test = "Hello my friend!";
    test.substring(5);
    String hello = test.substring(0, 5);
    String friend = test.substring(8, 15);
    char m = test.charAt(6);
    String replacedE = test.replace('e', 'W');
    int myIndex = test.indexOf("my");

    System.out.println("test: " + test);
    System.out.println("hello: " + hello);
    System.out.println("friend: " + friend);
    System.out.println("m: " + m);
    System.out.println("replacedE: " + replacedE);
    System.out.println("myIndex: " + myIndex);}
```

- test: Hello my friend!
- hello: Hello
- friend: friend
- m: m
- replacedE:
  - HWllo my friWnd!
- myIndex: 6

Pakartoti: “String is different”

String atvejis išskirtinis, nes inicijavimas atliekamas nenaudojant **new** operatoriaus. Tai padaryta sąmoningai, nes eilutės yra vienas dažniausiai naudojamų bet kokioje programavimo kalboje elementų ir norisi, kad darbas su eilutėmis būtų kuo paprastesnis.

String objektai yra nekintantys: visi metodai atliekantys manipuliacijas su tekstu, grąžina naują String objektą.

String: Užduotis7

Parašyti programą, kuri paprašytu vartotojo įvesti tekstą ir išspausdintu įvesto teksto simbolių skaičių.

String: Užduotis7

Parašyti programą, kuri paprašytu vartotojo įvesti tekstą ir išspausdintu įvesto teksto simbolių skaičių.

```
String str = "example.com";   // žodžiai

int len = str.length();
System.out.println("The string length of '"+str+"' is: "+len);
```

String: Užduotis8

- Parašyti programą kuri apskaičiuotu kiek įvestame tekste yra žodžių:
  - įvestas tekstas negali prasidėti ir baigtis tarpu;
  - įvestas tekstas turi bent vieną žodį
  - visi žodžiai atskirti tarpu

String: Užduotis8

```
public static void main(String[] args) {
    String s = "Mano testas čia"; // vartotojas įveda
    int zodziai = 1;
    for (int i = 0; i < s.length(); i++) {
        if (s.charAt(i) == ' ') {
            zodziai++;
        }
    }
    System.out.println("Žodžiai = " + zodziai);
}
```

String: Užduotis9

String

1. Parašykite klasę, turinčią main metodą. Jame parašykite programos fragmentą, kuris suskaičiuotų ir atspausdintų, kiek tekstinėje eilutėje yra balsių. Tekstinė eilutė paduodama iš komandinės eilutės. Tarkime, kad tarpų joje būti negali. Balsių sąrašui saugoti panaudokite simbolių masyvą arba eilutę.

???????????????????????

???
???
????????????????

String: Užduotis10

1. Vartotojo paprašyti įvesti tekstinę eilutę: scanner.nextLine();
2. Išvesti kiek simbolių turi eilutę
3. Išspausdinti viską didžiosiomis raidėmis
4. Rasti ar eilutė turi ‘š’ raidę, parašyti ar raidė rasta ar ne, jei taip kokia jos pozicija (pirmoji)?
5. Išspausdinti kiekvieną raidę naujoje eilutėje;
6. *Paprašyti vartotojo įvesti dar viena tekstinę eilutę:*
   a. *patikrinti ar sutampa su prieš tai įvesta reikšme*
   b. *patikrinti ar sutampa su prieš tai įvesta reikšme, ignoruojant didžiųjų/mažųjų raidžių neatitikimą*
7. *Kiek eilutėje žodžių?*
8. *Kiek eilutėje ‘a’ raidžių ir kokios pozicijos?*