



Primityvieji duomenų tipai

Primityvieji duomenų tipai: byte

- short – nedidelis sveikas skaičius
- nuo -128 iki 127
- Pavyzdžiai: 0, 1, 32, 101, ...
- Numatytoji reikšmė: 0
- Dydis atmintyje 1 baitas (8 bitai)
- Java:

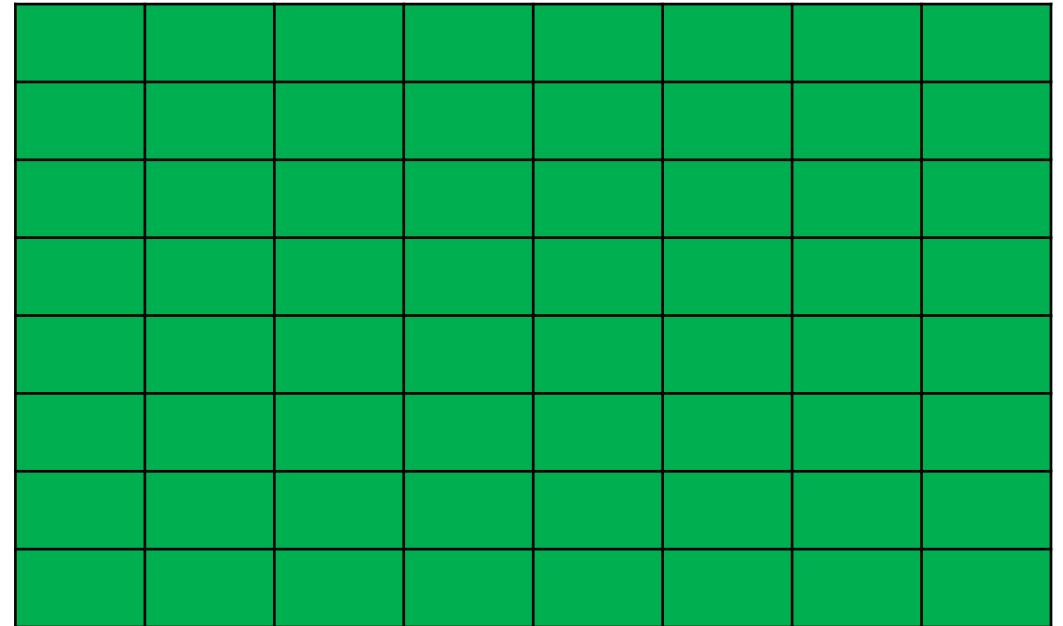
```
byte x = 32;
```

[illegible]

Primityvieji duomenų tipai: long

- long – didelis sveikas skaičius
- nuo -922372036854775808 iki 922372036854775807
- Pavyzdžiai: 0, 1, 5000000001, ...
- Numatytoji reikšmė: 0
- Dydis atmintyje 8 baitai (64 bitai)
- Java:

```
long x = 4;  
long x = 4l;  
long x = 4L;
```

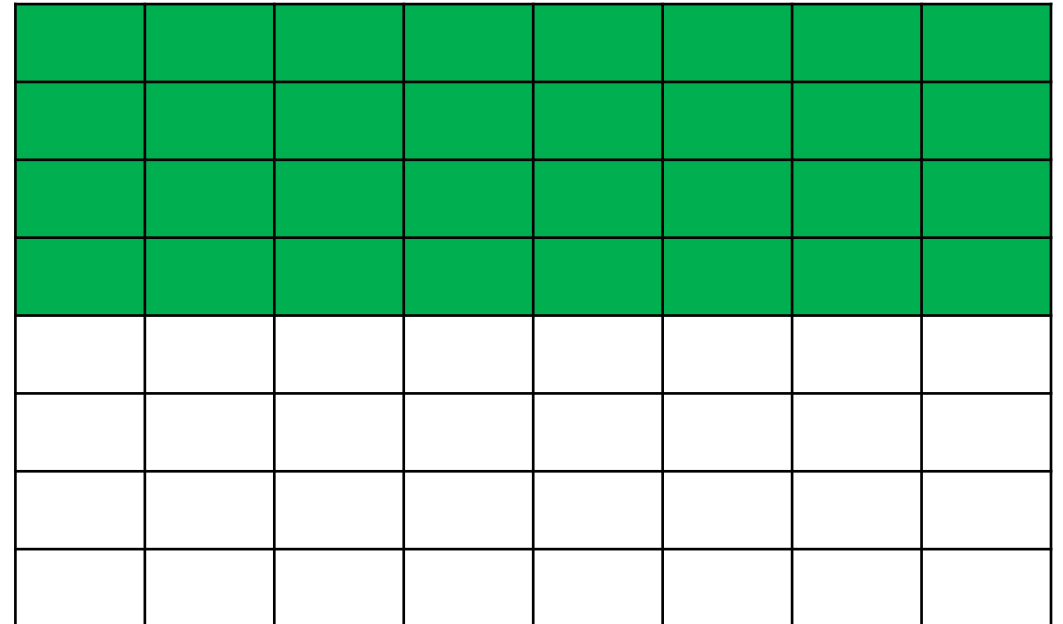


Primityvieji duomenų tipai: float

- float – slankaus kablelio skaičius
- nuo $-3.40282348 \times 10^{38}$ iki $3.40282347 \times 10^{38}$
- Pavyzdžiai: -0.35f, 3.141592654f, ...
- Numatytoji reikšmė: 0.0f
- Dydis atmintyje 4 baitai (32 bitai)
- Java:

```
float x = 3.14f;
```

```
float x = 3.14F;
```

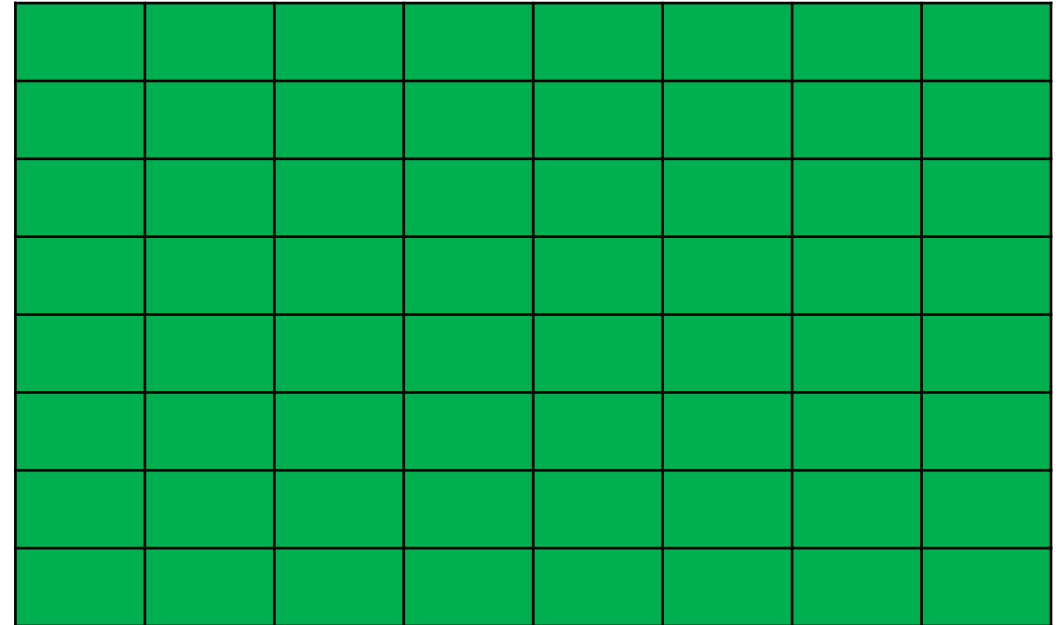


Primityvieji duomenų tipai: double

- double – didelis slankaus kablelio skaičius
- nuo $-1.79769313486231570 \times 10^{308}$ iki $1.79769313486231570 \times 10^{308}$
- Pavyzdžiai: -0.35d, 3.141592654d, ...
- Numatytoji reikšmė: 0.0d
- Dydis atmintyje 8 baitai (64 bitai)
- Java:

```
double x = 3.14d;
```

```
double x = 3.14D;
```



Primityvieji duomenų tipai: char

- [illegible]

[illegible]

Primityvieji duomenų tipai: char

Pavyzdys, kaip tą patį simbolį galima užrašyti keliais skirtingais būdais

```
4
5 public static void main(String[] args) {
6
7     char x = 'A';
8     char y = '\u0104';
9     char z = 104;
10
11     System.out.println(x);
12     System.out.println(y);
13     System.out.println(z);
14
15 }
16
```

Problems @ Javadoc Declaration Console

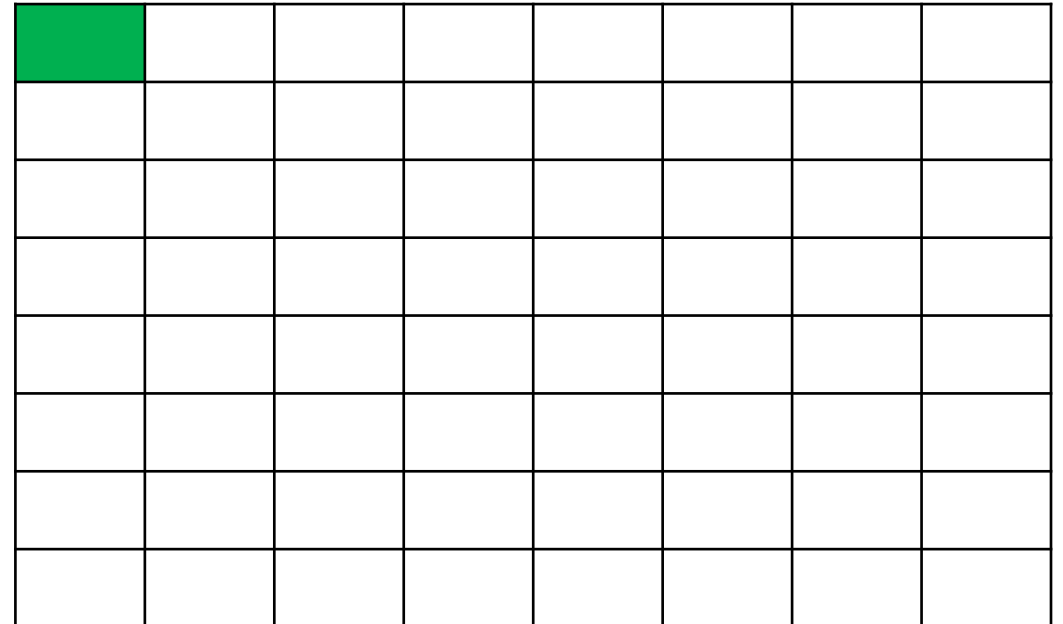
<terminated> DuomenuTipai [Java Application] C:\Program Files\Java\

A
A
A

Primityvieji duomenų tipai: boolean

- boolean – loginis duomenų tipas
- Galimos tik dvi reikšmės: true arba false
- Numatytoji reikšmė: false
- Dydis atmintyje 1 bitas
- Java:

```
boolean x = true;  
boolean x = false;
```



Aritmetiniai operatoriai

Operatorius	Paaškinimas
+	Sudėtis
-	Atimtis
*	Daugyba
/	Dalyba
%	Modulis
++	Reikšmės padidinimas vienetu
--	Reikšmės sumažinimas vienetu
+=	Reikšmės priskyrimas pridedant
-=	Reikšmės priskyrimas atimant
*=	Reikšmės priskyrimas padauginant
/=	Reikšmės priskyrimas padalijant
%=	Reikšmės priskyrimas moduliui

Aritmetinis operatorius +

```
4
5 public static void main(String[] args) {
6
7     int pirmasSkaicius = 2;
8     int antrasSkaicius = 5;
9     int treciasSkaicius = 30;
10
11     int suma = pirmasSkaicius + antrasSkaicius + treciasSkaicius;
12
13     System.out.println(suma);
14
15 }
16
```

Problems @ Javadoc Declaration Console

<terminated> DuomenuTipai [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (2018-11-30 15:37)

Aritmetinis operatorius -

```
4
5 public static void main(String[] args) {
6
7     int pirmasSkaicius = 100;
8     int antrasSkaicius = 15;
9     int treciasSkaicius = 10;
10
11     int skirtumas = pirmasSkaicius - antrasSkaicius - treciasSkaicius;
12
13     System.out.println(skirtumas);
14
15 }
16
```

Problems @ Javadoc Declaration Console

<terminated> DuomenuTipai [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (2018-06-75

Aritmetinis operatorius *

```
4
5 public static void main(String[] args) {
6
7     int pirmasSkaicius = 2;
8     int antrasSkaicius = 3;
9     int treciasSkaicius = 4;
10
11     int sandauga = pirmasSkaicius * antrasSkaicius;
12
13     System.out.println(sandauga);
14
15     sandauga = pirmasSkaicius + antrasSkaicius * treciasSkaicius;
16
17     System.out.println(sandauga);
18
19     sandauga = (pirmasSkaicius + antrasSkaicius) * treciasSkaicius;
20
21     System.out.println(sandauga);
22
23 }
```

Problems @ Javadoc Declaration Console

<terminated> DuomenuTipai [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (2018

6
14
20

Aritmetinis operatorius /

Jei dalinys ir daliklis abu yra **int** tipo kintamieji, tada rezultatas bus dalmens sveikoji dalis. Pvz. 9/4 rezultatas bus 2

Jei norime gauti tikslų rezultatą, turime naudoti **float** arba **double** tipo intamuosius.

```
4
5 public static void main(String[] args) {
6
7     float pirmasSkaicius = 9.99f;
8     float antrasSkaicius = 3f;
9
10    float dalmuo = pirmasSkaicius / antrasSkaicius;
11
12    System.out.println(dalmuo);
13
14 }
15
```

Problems @ Javadoc Declaration Console ✕

<terminated> DuomenuTipai [Java Application] C:\Program Files\Java\jre1.8.0_171\l
3.33

Aritmetinis operatorius %

% reiškia dalybos liekaną

```
5 public static void main(String[] args) {  
6  
7     float pirmasSkaicius = 10.0f;  
8     float antrasSkaicius = 3f;  
9  
10    float liekana = pirmasSkaicius % antrasSkaicius;  
11  
12    System.out.println(liekana);  
13 }  
14
```

Problems @ Javadoc Declaration Console

<terminated> DuomenuTipai [Java Application] C:\Program Files\Java\jre1.8.0_171\bin
1.0

Aritmetiniai operatoriai ++ ir --

```
4
5 public static void main(String[] args) {
6
7     int skaicius = 5;
8
9     System.out.println(skaicius);
10
11     // padidiname skaiciu vienetu
12     skaicius++;
13
14     System.out.println(skaicius);
15
16     // sumaziname skaiciu vienetu
17     skaicius--;
18
19     System.out.println(skaicius);
20
21 }
22
```

Problems @ Javadoc Declaration Console

<terminated> DuomenuTipai [Java Application] C:\Program Files\Java\jre1

5
6
5

```
5 public static void main(String[] args) {
6
7     int skaicius = 5;
8
9     System.out.println(skaicius);
10
11     // padidiname skaiciu vienetu
12     ++skaicius;
13
14     System.out.println(skaicius);
15
16 }
17
```

Problems @ Javadoc Declaration Console

<terminated> DuomenuTipai [Java Application] C:\Program Files\Java\jre1

5
6

skaicius++;

skaicius = skaicius + 1;

Aritmetiniai operatoriai ++ ir --

Nuo to, kurioje kintamojo pusėje yra parašytas aritmetinis operatorius, priklauso kada operacija bus įvykdyta

```
4
5 public static void main(String[] args) {
6
7     int skaicius = 5;
8     int kitasSkaicius = 5;
9
10    System.out.println(++skaicius);
11    System.out.println(kitasSkaicius++);
12
13    System.out.println("-----");
14
15    System.out.println(skaicius);
16    System.out.println(kitasSkaicius);
17
18 }
19
```

Problems @ Javadoc Declaration Console

<terminated> DuomenuTipai [Java Application] C:\Program Files\Java\jre

6
5

6
6

Aritmetiniai operatoriai +=, -=, *=, /= ir %=

Pavyzdys:

x yra 5, y yra 2

vykdomas toks Java sakiny:

`x += y;`

reiškia, kad prie x yra pridedama y reikšmė
ir gautas rezultatas (suma) priskiriama x kintamajam.
Tą patį rezultatą galima pasiekti tokiu sakiniu:

`x = x + y;`

```
4
5 public static void main(String[] args) {
6
7     int x = 5;
8     int y = 2;
9
10    // x yra 5, y yra 2
11    x += y;
12    System.out.println(x);
13
14    // x yra 7, y yra 2
15    x -= y;
16    System.out.println(x);
17
18    // x yra 5, y yra 2
19    x *= y;
20    System.out.println(x);
21
22    // x yra 10, y yra 2
23    x /= y;
24    System.out.println(x);
25
26    // x yra 5, y yra 2
27    x %= y;
28    System.out.println(x);
29 }
30
```

Problems @ Javadoc Declaration Console

<terminated> DuomenųTipai [Java Application] C:\Program Files\Java\jre

7
5
10
5
1

Nuorodos

- Java dokumentacija apie primityvius duomenų tipus:
<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>



java.lang.String

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.

```
String pirmaEilute = "venas";  
String antraEilute = "du";  
  
pirmaEilute = pirmaEilute + antraEilute;  
  
System.out.println(pirmaEilute);  
  
>> venasdu
```

```
String pirmaEilute = "venas";  
String antraEilute = "du";  
  
pirmaEilute += antraEilute;  
  
System.out.println(pirmaEilute);  
  
>> venasdu
```

Eilutės

- 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.

Eilučių pavyzdžiai

```
//priskiriame eilutę kintamajam  
String miestas = "Vilnius";  
  
//kviečiame eilutes metodą  
char pirmojiRaide = "Vilnius".charAt(0);  
  
//kviečiame String kintamojo metodą  
char pirmasisSimbolis = miestas.charAt(0);
```

Simbolis	V	i	l	n	i	u	s
Indeksas	0	1	2	3	4	5	6

Jei bandysime gauti simbolį su didesniu indeksu, tada gausime klaidą:

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: 8
```

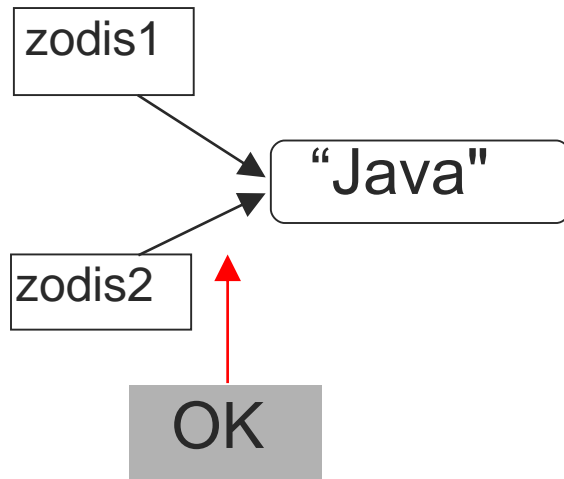

String nekintamumas

- 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.

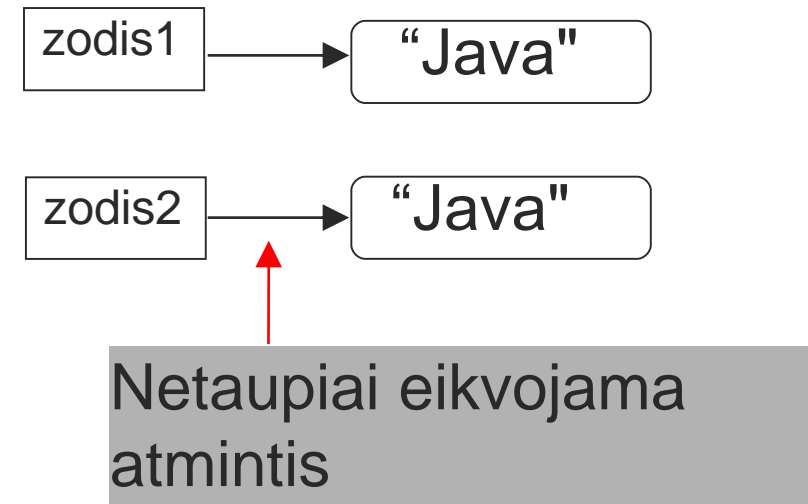
Nekintamų objektų privalumai

Naudoja mažiau atminties

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



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




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 parametų sąrašu

- Toks konstruktorius sukuria tuščią eilutę. Naudojamas retai

```
String tuščia = new String();
```

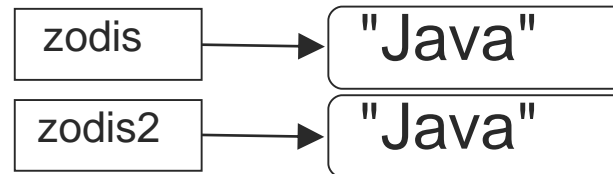
- Dažniau tokiu atveju naudojama tuščia eilutė
- Tuščia eilutė dažnai yra analogas pradinės nulinės reikšmės kaupiant sumą.

```
String tuščia = "";
```

Eilutės kopijavimas panaudojant konstruktorių

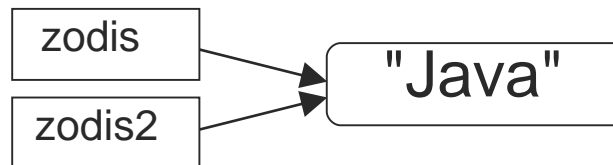
- Egzistuojančios eilutės kopija gaunama kreipiantis į konstruktorių perduodant parametru kopijuojamos eilutės objektą. Taip pat retai naudojama.
- Skiriasi nuo priskyrimo operatoriaus.

```
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;
```



Kiti konstruktoriai

Galima naudoti masyvą, kad sukurti eilutę

```
char[] raides = {'J', 'a', 'v', 'a'};  
String zodisIsRaidziu = new String(raides);
```

```
byte[] baitai = {0x4a, 0x61, 0x76, 0x61};  
String zodisIsBaitu = new String(baitai);
```

Metodas *replace*

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

```
String zodis1 = "mama";  
String zodis2 = zodis1.replace('m', 'p');
```

// zodis2 yra "papa", o zodis1 lieka "mama"



Metodai *charAt* ir *length*

- `int length();`
- `char charAt(i);`

Grąžina kiek eilutėje yra simbolių

Grąžina simbolį esantį i-ojoje pozicijoje.

Grąžina:

<code>"Žodis".length();</code>		5
<code>"Žodis".charAt (2);</code>		'd'

Užduotis: *charAt* ir *length*

- Sukurti eilutės tipo kintamąjį ir jam priskirti tekstą
- Rasti n-tąjį eilutės simbolį ir jį atspausdinti
- Rasti teksto ilgį

`char charAt(int)`

`int length()`

Metodas *substring*

- Grąžina naują eilutę kopijuojant nurodytų simbolių seką.

C	o	d	e	A	c	a	d	e	m	y
0	1	2	3	4	5	6	7	8	9	10

- 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 eilutės galo.
- "CodeAcademy".substring (2, 5); // rezultatas: "deA"
- "CodeAcademy".substring (2); // rezultatas: "deAcademy"

Užduotis: *substring*

- String s = "Mano vardas";
- Naudojant metodą **substring (i)** gauti tik vardą ir jį atspausdinti
- Naudojant metodą **substring (i, k)** gauti eilutę „Mano“ ir ją atspausdinti

Metodas *indexOf*

C	o	d	e	A	c	a	d	e	m	y
0	1	2	3	4	5	6	7	8	9	10

```
String vardas="CodeAcademy";
```

```
vardas.indexOf('C');           // 0
```

```
vardas.indexOf('d');           // 2
```

```
vardas.indexOf("Academy");      // 4
```

```
vardas.indexOf('d', 3);         // 7
```

```
vardas.indexOf("Pirmadienis");  // -1
```

```
vardas.lastIndexOf('e');        // 8
```

Eilučių palyginimo metodas *equals*

- `boolean b = zodis1.equals(zodis2);`
 - grąžina **true** jei **zodis1** sutampa su **zodis2**
- `boolean b = zodis1.equalsIgnoreCase(zodis2);`
 - grąžina **true** jei **zodis1** sutampa su **zodis2** ignoruojant didžiųjų/mažųjų raidžių skirtumą

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

Eilučių palyginimo metodas *compareTo*

- `int diff = zodis1.compareTo(zodis2);`
 - grąžina “skirtumą” `zodis1 - zodis2`
- `int diff = zodis1.compareToIgnoreCase(zodis2);`
 - grąžina “skirtumą” `zodis1 - zodis2`, ignoruojant didžiųjų/mažųjų skirtumą
- Dažnai konkreti “skirtumo” `zodis1 - 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.

Eilučių palyginimo metodas *compareTo*

```
int diff;

//neigiami skirtumai
diff = "apple".compareTo("berry");    // -1, nes a (97) prieš b (98)
diff = "Zebra".compareTo("apple");    // -7, nes Z (90) prieš a (97)
diff = "dig".compareTo("dug");         // -12, nes i (105) prieš u (117)
diff = "dig".compareTo("digs");        // -1, nes dig trumpesnis

//nulinis skirtumas
diff = "apple".compareTo("apple");     // 0, nes sutampa
diff = "di".compareToIgnoreCase("DI"); // 0, nes sutampa nepaisant
didžiuju/mažuju raidžiu

//teigiami skirtumai
diff = "berry".compareTo("apple");     // 1, nes b (98) po a (97)
diff = "apple".compareTo("Apple");     // 32, nes a (97) po A (65)
diff = "BIT".compareTo("BIG");         // 13, nes T (84) po G (71)
diff = "huge".compareTo("hug");        // 1, nes huge ilgesnis
```

<https://unicode-table.com/en/#control-character>

Metodas *trim*

- `String zodis2 = zodis1.trim ();`
 - grąžina naują eilutę sudarytą iš `zodis1` 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
```


Eilučių sujungimo metodai

```
String zodis1 = "ap", zodis2 = "gal", zodis3 = "voti";
```

```
String result = zodis1 + zodis2 + zodis3;
```

```
String result = zodis1.concat(zodis2).concat(zodis3);
```

```
String result = "";  
result += zodis1;  
result += zodis2;  
result += zodis3;
```

Metodai *toUpperCase* ir *toLowerCase*

- `String zodis2 = zodis1.toUpperCase();`
- `String zodis3 = zodis1.toLowerCase();`
- grąžina naują eilutę pakeičiant zodiac1 mažąsias (didžiąsias) raides didžiosiomis (mažosiomis)

```
String zodis1 = "Sveikas";  
String zodis2 = zodis1.toUpperCase(); // "SVEIKAS"  
String zodis3 = zodis1.toLowerCase(); // "sveikas"  
//zodis1 lieka "Sveikas"
```

Skaičių vertimas tekstu

- Yra trys būdai tai padaryti:

1. `String s = "" + skaičius;`

```
s = "" + 123; // "123"
```

2. `String s = Integer.toString(skaičius);`

```
String s = Double.toString(skaičiusSuKableliu);
```

```
s = Integer.toString(123); // "123"
```

```
s = Double.toString(3.14); // "3.14"
```

3. `String s = String.valueOf(skaičius);`

```
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.

Nepamiršti, kad *String* yra „kitokia“ klasė

- *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 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);           // test: Hello my friend!  
    System.out.println("hello: " + hello);         // hello: Hello  
    System.out.println("friend: " + friend);       // friend: friend  
    System.out.println("m: " + m);                 // m: m  
    System.out.println("replacedE: " + replacedE); // replacedE: HWllo my friWnd!  
    System.out.println("myIndex: " + myIndex);     // myIndex: 6  
}
```

Užduotis

- Išbandyti String klasės metodus:
 - `.substring()`
 - `.charAt()`
 - `.replace()`
 - `.indexOf()`
 - `.equals()`
 - `.compareTo()`
 - `.trim()`
 - `.toLowerCase()`
 - `.toUpperCase()`

Nuorodos

- Java dokumentacija apie *String*:
<https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>



`java.util.Scanner`

Duomenų įvedimas

```
import java.util.Scanner;

class InputTest {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Iveskite vardą:");
        String vardas = sc.next();

        System.out.println("Iveskite ūgį:");
        double ugis = sc.nextDouble();

        System.out.println("Vardas: " + vardas + " ūgis: " + ugis);

        sc.close();
    }
}
```

Duomenų įvedimas

Method	Description
<code>public String next()</code>	it returns the next token from the scanner.
<code>public String nextLine()</code>	it moves the scanner position to the next line and returns the value as a string.
<code>public byte nextByte()</code>	it scans the next token as a byte.
<code>public short nextShort()</code>	it scans the next token as a short value.
<code>public int nextInt()</code>	it scans the next token as an int value.
<code>public long nextLong()</code>	it scans the next token as a long value.
<code>public float nextFloat()</code>	it scans the next token as a float value.
<code>public double nextDouble()</code>	it scans the next token as a double value.

Nuorodos

- <https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html>



Konstantos ir Enum

Konstantos

- Tai klasės dalys, kurios negali būti keičiamos (priskiriama kita reikšmė)
- `public static final double PI = 3.141592653589793;`
- Aprašoma tik didžiosiomis raidėmis. Jei sudaro keli žodžiai, atskiriama apatiniu brūkšniu. `TIKSLUS_SKAICIUS_PI`
- Priskiriama programos paleidimo metu

Konstantos pavyzdys

```
public class ConstantTest {  
  
    public static final double PI = 3.141592653589793;  
    public static final String SKUBUS_PRANESIMAS = "Labai skubus pranesimas!";  
  
    public static void main(String[] args) {  
  
        System.out.println(PI); // 3.141592653589793  
        System.out.println(SKUBUS_PRANESIMAS); // Labai skubus pranesimas!  
  
        double skaicius = PI;  
        String pranesimas = SKUBUS_PRANESIMAS;  
  
        System.out.println(skaicius); // 3.141592653589793  
        System.out.println(pranesimas); // Labai skubus pranesimas!  
  
        PI = 3.4D; // The final field ConstantTest.PI cannot  
                // be assigned - taip daryti NEGALIMA  
  
    }  
  
}
```

Iš kitos klasės galima pasiekti
public konstantas užrašius taip



```
String message = ConstantTest.SKUBUS_PRANESIMAS;
```

Enum pavyzdys

```
public enum Diena {  
  
    PIRMADIENIS,  
    ANTRADIENIS,  
    TRECIADIENIS,  
    KETVIRTADIENIS,  
    PENKTADIENIS,  
    SESTADIENIS,  
    SEKMADIENIS  
}
```

Failas Diena.java

```
public class EnumTest {  
  
    public static void main(String[] args) {  
  
        // apsirasome savaites dienos kintamaji ir jam priskiriame enum reiksme  
        Diena savaitesDiena = Diena.TRECIADIENIS;  
  
        // enum reiksme galime atspausdinti  
        System.out.println(savaitesDiena);  
  
        // enum reiksmes galime palyginti  
        System.out.println(Diena.TRECIADIENIS.equals(Diena.TRECIADIENIS)); // true  
        System.out.println(Diena.TRECIADIENIS.equals(Diena.PENKTADIENIS)); // false  
  
    }  
}
```

Failas EnumTest.java

Enum pavyzdys

```
public enum Skaicius {  
    VIENAS(1),  
    DU(2),  
    TRY(3),  
    KETURI(4);  
  
    private final int numeris;  
  
    Skaicius (int num) {  
        numeris = num;  
    }  
  
    public int numeris() {  
        return numeris;  
    }  
}
```

Enum kintmasis, kuris saugo reikšmę

Enum konstruktorius priimantis skaičių

Metodas grąžinantis enum reikšmę

```
public class EnumTest {  
    public static void main(String[] args) {  
        Skaicius skaicius = Skaicius.KETURI;  
  
        System.out.println(skaicius);           // KETURI  
        System.out.println(skaicius.numeris()); // 4  
        System.out.println(Skaicius.KETURI.numeris()); // 4  
    }  
}
```



```
public class EnumTest {  
  
    public static void main(String[] args) {  
  
        public enum MatavimoVienetas {  
  
            MILIMITRAS(0.001f, "mm"),  
            CENTIMETRAS(0.01f, "cm"),  
            METRAS(1.0f, "m"),  
            KILOMETRAS(1000.0f, "km");  
  
            private float daugiklis;  
            private String trumpinys;  
  
            MatavimoVienetas(float x, String y) {  
                daugiklis = x;  
                trumpinys = y;  
            }  
  
            public float daugiklis() {  
                return daugiklis;  
            }  
  
            public String trumpinys() {  
                return trumpinys;  
            }  
        }  
    }  
}  
  
        float atstumas = 5.0f; // 5 metrai  
  
        float atstumasCM = atstumas / MatavimoVienetas.CENTIMETRAS.daugiklis();  
        float atstumasKM = atstumas / MatavimoVienetas.KILOMETRAS.daugiklis();  
  
        String cmTrumpinys = MatavimoVienetas.CENTIMETRAS.trumpinys();  
        String kmTrumpinys = MatavimoVienetas.KILOMETRAS.trumpinys();  
  
        System.out.println(atstumas + MatavimoVienetas.METRAS.trumpinys());  
        System.out.println(atstumasCM + cmTrumpinys );  
        System.out.println(atstumasKM + kmTrumpinys);  
  
        5.0m  
        500.0cm  
        0.005km
```

Enum URL

- <https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html>



Loginiai operatoriai ir sąlygos sakiniai

Loginiai operatoriai

```
boolean pirmas = true;  
boolean antras = false;
```

Operatorius	Pavadinimas	Pavyzdys	Rezultatas
!	Not	!pirmas	false, nes true keičiame priešingu
==	Equal to	pirmas == antras	false, nes pirmas nėra lygus antram
!=	Not equal to	pirmas != antras	true, nes pirmas nėra lygus antram
&& ir &	Conditional-AND	pirmas && antras	false, nes abu nėra true
ir	Conditional-OR	pirmas antras	true, nes bent vienas iš abiejų yra true
^	Exclusive OR	pirmas ^ antras	true, jei vienas ir TIK vienas iš dviejų yra true

Loginis operatorius !

!	true	false
	false	true

Loginis operatorius ==

==	true	false
true	true	false
false	false	true

Loginis operatorius !=

!=	true	false
true	false	true
false	true	false

Loginis operatorius &&

&&	true	false
true	true	false
false	false	false

false && true antros pusės netikrina, jei pirmoji yra false

false & true tikrina antrą pusę net, jei pirmoji yra false

Loginis operatorius ||

	true	false
true	true	true
false	true	false

false || true antros pusės netikrina, jei pirmoji yra true

false | true tikrina antrą pusę net, jei pirmoji yra true

Loginis operatorius \wedge

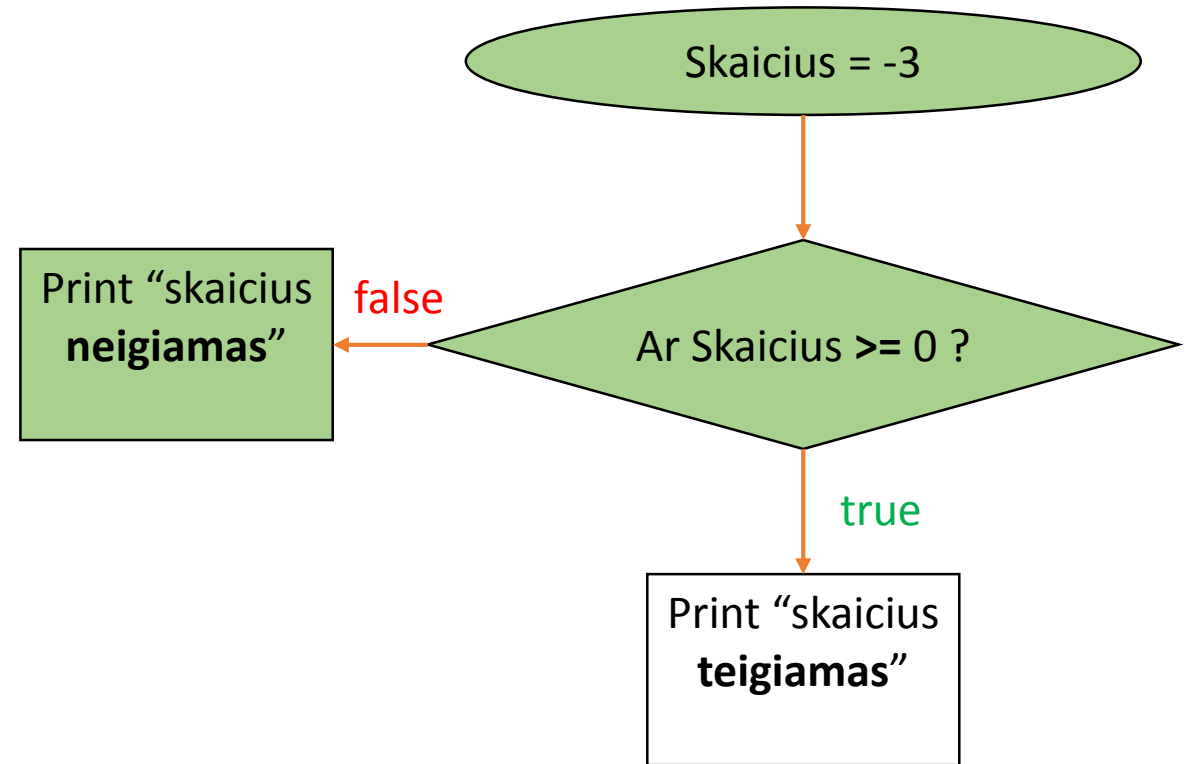
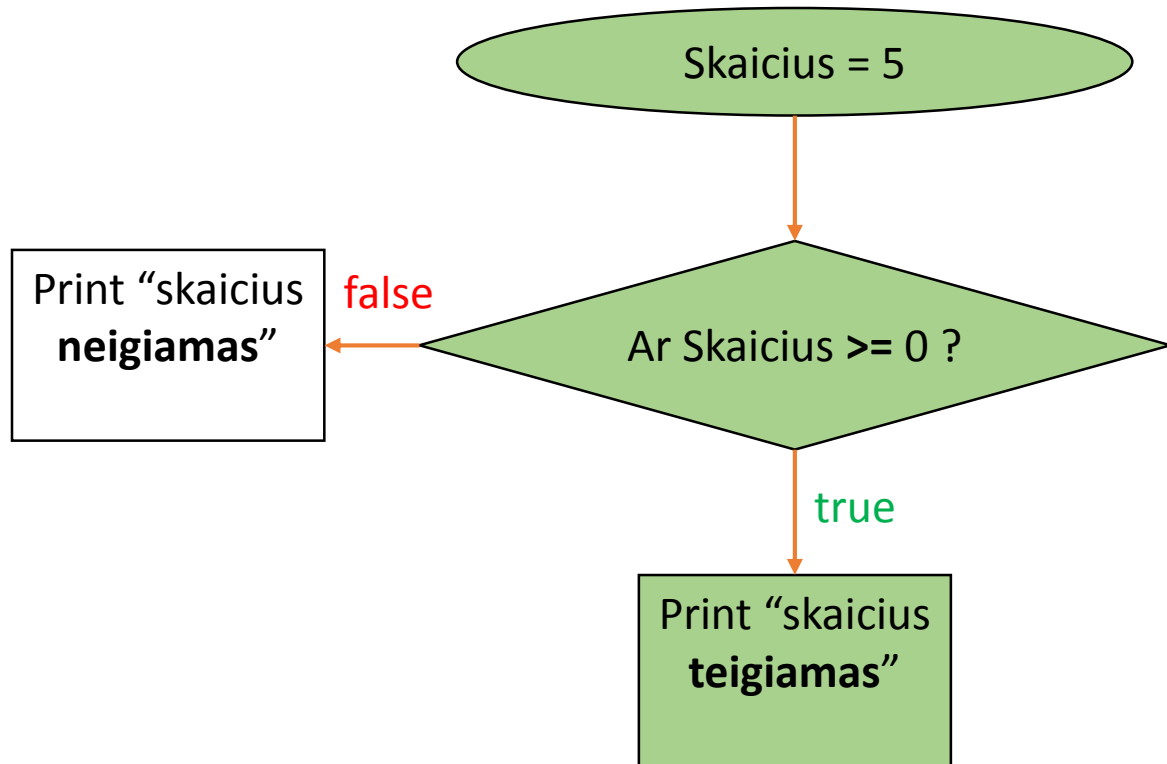
\wedge	true	false
true	false	true
false	true	false

Loginių operatorių pavyzdžiai Javoje

```
boolean pirmas = true;  
boolean antras = false;
```

```
System.out.println(!pirmas);           // false  
System.out.println(pirmas && antras);  // false  
System.out.println(false || true);    // true  
System.out.println(true ^ true);      // false  
System.out.println(pirmas && !antras | false); // true
```

Kas yra sąlygos?



IF

```
public class IfTest {  
    public static void main(String[] args) {  
        int skaicius = 5;  
  
        if (skaicius >= 0) {  
            System.out.println("Skaicius yra teigiamas");  
        }  
    }  
}
```

IF

```
if (sąlyga) {  
    // veiksmas, jei sąlyga tenkinama  
}
```

IF-ELSE

```
public class IfTest {  
    public static void main(String[] args) {  
        int skaicius = 5;  
  
        if (skaicius >= 0) {  
            System.out.println("Skaicius yra teigiamas");  
        } else {  
            System.out.println("Skaicius yra neigiamas");  
        }  
    }  
}
```

IF-ELSE

```
if (sąlyga) {  
    // veiksmas, jei sąlyga tenkinama  
} else {  
    // veiksmas, jei sąlyga netenkinama  
}
```


IF-ELSEIF-ELSE

```
public class IfTest {  
  
    public static void main(String[] args) {  
  
        int skaicius = 5;  
  
        if (skaicius > 0) {  
            System.out.println("Skaicius yra teigiamas");  
        } else if (skaicius == 0) {  
            System.out.println("Skaicius yra nulis");  
        } else {  
            System.out.println("Skaicius yra neigiamas");  
        }  
    }  
}
```

IF-ELSEIF-ELSE

```
if (sąlyga1) {  
    // veiksmas, jei sąlyga1 tenkinama  
} else if (sąlyga2) {  
    // veiksmas, jei sąlyga2 tenkinama  
} else if (sąlyga3) {  
    // veiksmas, jei sąlyga3 tenkinama  
} else {  
    // veiksmas, jei nei viena sąlyga netenkinama  
}
```

Skliausių { } naudojimas

```
public class IfTest {  
    public static void main(String[] args) {  
        int skaicius = 6;  
        if (skaicius >= 0)  
            System.out.println("skaicius yra teigiamas");  
        else  
            System.out.println("skaicius yra neigiamas");  
    }  
}
```

```
if (sąlyga)  
    // veiksmas  
else  
    // veiksmas
```

```
public class IfTest {  
    public static void main(String[] args) {  
        int skaicius = 6;  
        if (skaicius >= 0) {  
            System.out.println("skaicius yra teigiamas");  
            skaicius++;  
        } else {  
            System.out.println("skaicius yra neigiamas");  
            skaicius--;  
        }  
    }  
}
```

```
if (sąlyga) {  
    // veiksmas  
    // veiksmas  
} else {  
    // veiksmas  
    // veiksmas  
}
```

IF'ai IF'uose

```
public class IfTest {  
  
    public static void main(String[] args) {  
  
        int skaicius = 6;  
  
        if (skaicius >= 0) {  
            System.out.println("skaicius yra teigiamas");  
            if (skaicius % 2 == 0) {  
                System.out.println("skaicius yra lyginis");  
                if (skaicius < 10) {  
                    System.out.println("skaicius yra mazesnis uz 10");  
                }  
            } else {  
                System.out.println("skaicius yra nelyginis");  
            }  
        } else {  
            System.out.println("skaicius yra neigiamas");  
            if (skaicius % 2 == 0) {  
                System.out.println("skaicius yra lyginis");  
            } else {  
                System.out.println("skaicius yra nelyginis");  
            }  
        }  
    }  
}
```

Switch

```
public class IfTest {  
  
    public static void main(String[] args) {  
  
        int month = 8;  
        String monthString;  
        switch (month) {  
            case 1: monthString = "January";  
                    break;  
            case 2: monthString = "February";  
                    break;  
            case 3: monthString = "March";  
                    break;  
            case 4: monthString = "April";  
                    break;  
            case 5: monthString = "May";  
                    break;  
            case 6: monthString = "June";  
                    break;  
            case 7: monthString = "July";  
                    break;  
            case 8: monthString = "August";  
                    break;  
            case 9: monthString = "September";  
                    break;  
            case 10: monthString = "October";  
                    break;  
            case 11: monthString = "November";  
                    break;  
            case 12: monthString = "December";  
                    break;  
            default: monthString = "Invalid month";  
                    break;  
        }  
        System.out.println(monthString);    // August  
    }  
}
```

```
public static void main(String[] args) {  
  
    String month = "March";  
    int monthNumber;  
    switch (month) {  
        case "January" :    monthNumber = 1;  
                            break;  
        case "February" :  monthNumber = 2;  
                            break;  
        case "March" :     monthNumber = 3;  
                            break;  
        // ....  
        case "December" :  monthNumber = 12;  
                            break;  
        default:           monthNumber = -1;  
                            break;  
    }  
    System.out.println(monthNumber);    // 3  
}
```

Switch

```
switch(kintamasis) {  
    case KONSTANTA1: // veiksmas, jei kintamasis == KONSTANTA1  
                    break;  
    case KONSTANTA2: // veiksmas, jei kintamasis == KONSTANTA2  
                    break;  
    default:        // veiksmas, jei kintamasis nėra lygus nei  
                    // vienai konstantai  
                    break;  
}
```

IF-ELSE vienoje eilutėje

```
public class IfTest {  
    public static void main(String[] args) {  
        int skaicius = 6;  
        String rezultatas;  
        if (skaicius >= 0) {  
            rezultatas = "teigiamas";  
        } else {  
            rezultatas = "neteigiamas";  
        }  
        System.out.println(rezultatas);  
    }  
}
```

```
public class IfTest {  
    public static void main(String[] args) {  
        int skaicius = 6;  
        String rezultatas = skaicius >= 0 ? "teigiamas" : "neteigiamas";  
        System.out.println(rezultatas);  
    }  
}
```

kintamasis = sąlyga ? rezultatasJeiTaip : rezultatasJeiNe



Masyvai

Masyvas

- Tai duomenų tipas, susidedantis iš to paties tipo elementų
- Įtraukiant duomenis į masyvą turi būti žinomas masyvo dydis
- Masyvas indeksuojamas nuo 0

Masyvo kintamojo apibrėžimas

```
byte[]  anArrayOfBytes;  
short[] anArrayOfShorts;  
long[]  anArrayOfLongs;  
float[] anArrayOfFloats;  
double[] anArrayOfDoubles;  
boolean[] anArrayOfBooleans;  
char[]  anArrayOfChars;  
String[] anArrayOfStrings;
```

Masyvo kintamojo apibrėžimas

Galimi skirtingi būdai, bet antrojo varianto reikėtų vengti

```
float[] arrayOfFloats;  
float arrayOfFloats[];
```

Masyvo inicializavimas

```
anArray = new int[10];
```

sukuriamas masyvas su 10 vietų, skirtų int tipo skaičiams

Elementų į masyvą įdėjimas

```
anArray[0] = 100;
```

```
anArray[1] = 200;
```

```
anArray[2] = 300;
```

100	200	300							
0	1	2	3	4	5	6	7	8	9

Masyvo elementų panaudojimas

```
System.out.println("Element 1 at index 0: " + anArray[0]);  
System.out.println("Element 2 at index 1: " + anArray[1]);  
System.out.println("Element 3 at index 2: " + anArray[2]);
```

```
// Element 1 at index 0: 100  
// Element 2 at index 1: 200  
// Element 3 at index 2: 300
```

Masyvo inicializavimas iškart

```
int[] anArray = {  
    100, 200, 300,  
    400, 500, 600,  
    700, 800, 900, 1000  
};
```

Dvimačiai masyvai

```
String[][] names = {  
    {"Mr. ", "Mrs. ", "Ms. "},  
    {"Smith", "Jones"}  
};
```

```
System.out.println(names[0][0] + names[1][0]);    // Mr. Smith  
System.out.println(names[0][2] + names[1][1]);    // Ms. Jones
```

	0	1	2
0	Mr.	Mrs.	Ms.
1	Smith	Jones	

Masyvo ilgis

```
System.out.println(anArray.length);
```

Masyvo elementų kopijavimas

```
class ArrayCopyDemo {  
    public static void main(String[] args) {  
        char[] copyFrom = { 'd', 'e', 'c', 'a', 'f', 'f', 'e',  
                             'i', 'n', 'a', 't', 'e', 'd' };  
  
        char[] copyTo = new char[7];  
  
        System.arraycopy(copyFrom, 2, copyTo, 0, 7);  
        System.out.println(new String(copyTo));  
    }  
}
```

Masyvo elementų kopijavimas

```
class ArrayCopyDemo {  
    public static void main(String[] args) {  
        char[] copyFrom = { 'd', 'e', 'c', 'a', 'f', 'f', 'e',  
                             'i', 'n', 'a', 't', 'e', 'd' };  
        char[] copyTo = new char[7];  
  
        System.arraycopy(copyFrom, 2, copyTo, 0, 7);  
        System.out.println(new String(copyTo));  
    }  
}
```

Masyvas, iš kurio kopijuojame

Nuo kelinto elemento pradėdame kopijuoti

Masyvas, į kurį kopijuojame

Nuo kelinto elemento pradėti įrašinėti į naują masyvą

Kiek elementų kopijuoti

Užduotis

- Vartotojas įveda 10 vardų
- Tuos vardus sudeda į String tipo masyvą
- Atspausdina visus masyvo elementus tokiu formatu:
 - {vardas vien didžiosiomis raidėmis}-{vardo simbolių ilgis}-{elemento masyve numeris}
 - Pvz: "JURGIS-6-0"

Nuorodos

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>



Ciklai

Ciklai

- Tai **n** kartų arba kol tenkinama sąlyga kartojami kodo blokai

Ciklas FOR

```
for (initialization; termination; increment) {  
    statement(s)  
}
```


Ciklas FOR

arba <= 10

```
for (int i = 1; i < 11; i++) {  
    System.out.println("i yra: " + i);  
}
```

i yra: 1
i yra: 2
i yra: 3
i yra: 4
i yra: 5
i yra: 6
i yra: 7
i yra: 8
i yra: 9
i yra: 10

```
for ( ; ; ) {  
    // begalinis ciklas  
}
```

Ciklas FOR

```
int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  
  
for (int skaicius : numbers) {  
    System.out.println("Masyvo elementas yra: " + skaicius);  
}
```

Masyvo elementas yra: 1
Masyvo elementas yra: 2
Masyvo elementas yra: 3
Masyvo elementas yra: 4
Masyvo elementas yra: 5
Masyvo elementas yra: 6
Masyvo elementas yra: 7
Masyvo elementas yra: 8
Masyvo elementas yra: 9
Masyvo elementas yra: 10

Ciklas WHILE

```
while (expression) {  
    statement(s)  
}
```

Ciklas WHILE

```
int skaicius = 1;
while (skaicius < 11) {
    System.out.println("Skaicius yra: " + skaicius);
    skaicius++;
}
```

Skaicius yra: 1
Skaicius yra: 2
Skaicius yra: 3
Skaicius yra: 4
Skaicius yra: 5
Skaicius yra: 6
Skaicius yra: 7
Skaicius yra: 8
Skaicius yra: 9
Skaicius yra: 10

```
while (true) {
    // begalinis ciklas
}
```

Ciklas DO-WHILE

```
do {  
    statement(s)  
} while (expression);
```

Ciklas DO-WHILE

```
int skaicius = 1;  
do {  
    System.out.println("Skaicius yra: " + skaicius);  
    skaicius++;  
} while (skaicius < 11);
```

Skaicius yra: 1
Skaicius yra: 2
Skaicius yra: 3
Skaicius yra: 4
Skaicius yra: 5
Skaicius yra: 6
Skaicius yra: 7
Skaicius yra: 8
Skaicius yra: 9
Skaicius yra: 10

BREAK

```
String[] programavimoKalbos = { "Java", "Python", "Kotlin", "Pascal",  
                                "C++", "C#", "Ruby", "C", "R", "PHP" };
```

```
for (String zodis : programavimoKalbos) {  
    if (zodis.length() < 4) {  
        break;  
    }  
    System.out.println(zodis);  
}
```

Java
Python
Kotlin
Pascal

Nutraukia ciklą ir išeina iš jo

CONTINUE

```
String[] programavimoKalbos = { "Java", "Python", "Kotlin", "Pascal",  
                                  "C++", "C#", "Ruby", "C", "R", "PHP" };
```

```
for (String zodis : programavimoKalbos) {  
    if (zodis.length() < 4) {  
        continue;  
    }  
    System.out.println(zodis);  
}
```

Java
Python
Kotlin
Pascal
Ruby

Nebevykdo veksmų, kurie yra
žemiau *continue*,
bet ima kitą elementą ir tęsia ciklą toliau



Klasės ir metodai

Java programos struktūra

- Visos Java programos sudarytos iš keturių pagrindinių elementų:
 - Klasų
 - Metodų
 - Kintamųjų/atributų
 - Paketų

Klasės pavyzdys

```
public class Animal {  
  
}
```

```
[modifikatorius] class [vardas] {}
```

Klasės struktūra

- Java klasė susideda iš dviejų pagrindinių elementų:
 - **metodai**, kurie dažnai vadinami funkcijomis arba kitose kalbose – procedūromis
 - **kintamieji**, dar vadinami klasės atributais.

```
class Animal {  
    String name;  
    String getName() {  
        return name;  
    }  
    void setName(String newName) {  
        name = newName;  
    }  
}
```

← klasė

← kintamasis

← metodas

← metodas

Klasė vs Java byla (*.java)

- Java byla gali turėti keletą klasių
- public klasė *.java byloje turi būti tik viena
- public klasės pavadinimas turi sutapti su bylos pavadinimu

```
public class Animal {  
  
    String name;  
  
    String getName() {  
        return name;  
    }  
    void setName(String newName) {  
        name = newName;  
    }  
}  
  
class Address {  
  
    String city;  
  
    String getCity() {  
        return city;  
    }  
    void setCity(String newCity) {  
        city = newCity;  
    }  
}
```

Animal.java

Metodas *main*

Java aplikacijos startuoja ***main*** metodu

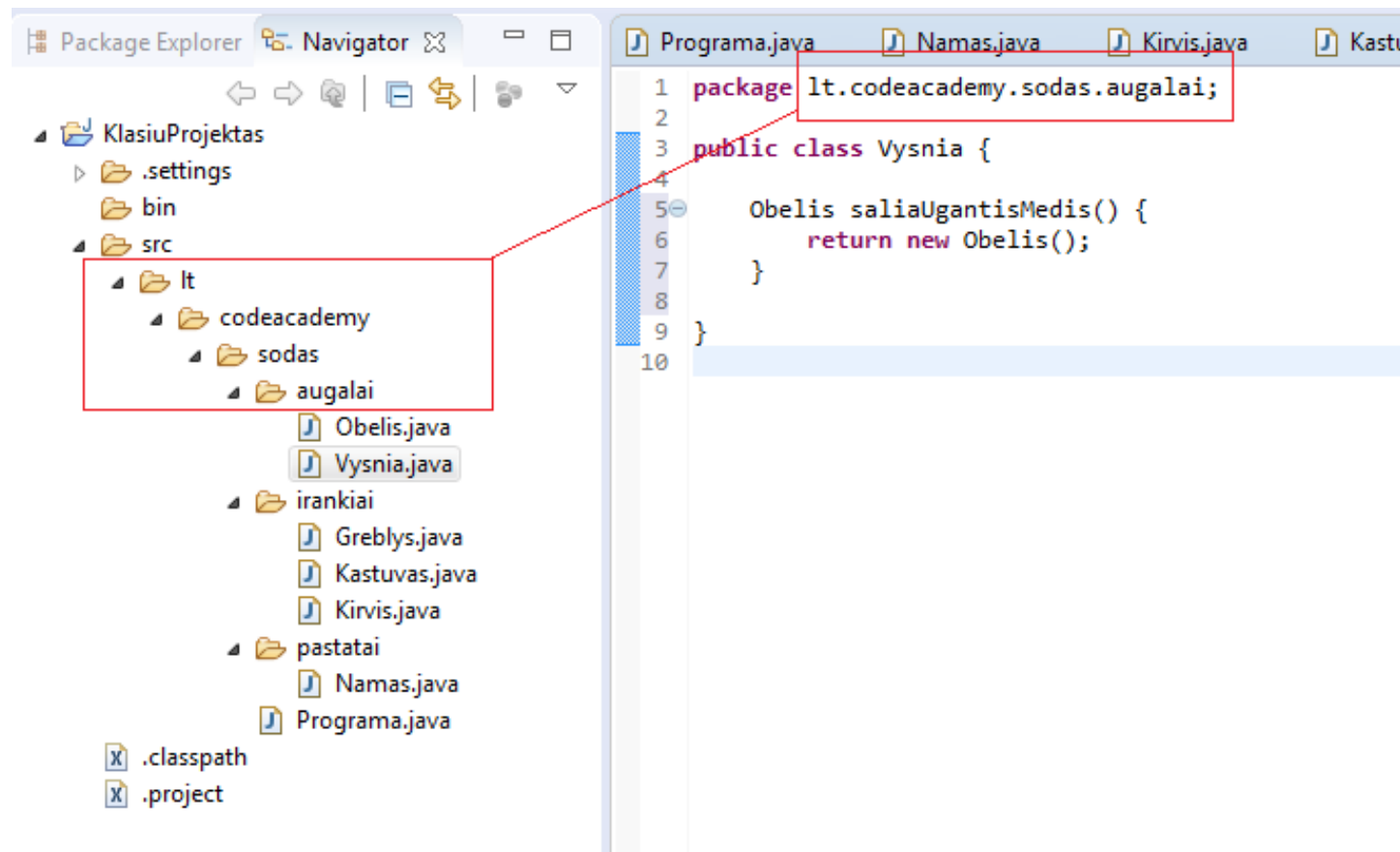
```
public class Pirmoji {  
    public final static void main(String S[]) {  
        System.out.println("Sveikas, Pasauli!");  
    }  
}
```

Paketai

- Java klases galima apjungti į vieną grupę, vadinamą paketu.
- Į vieną paketą tikslinga įtraukti giminingos paskirties programas.
- Mes ne kartą naudojome ***System.out.println*** metodą. **System** yra klasė patalpinta **java.lang** pakete.

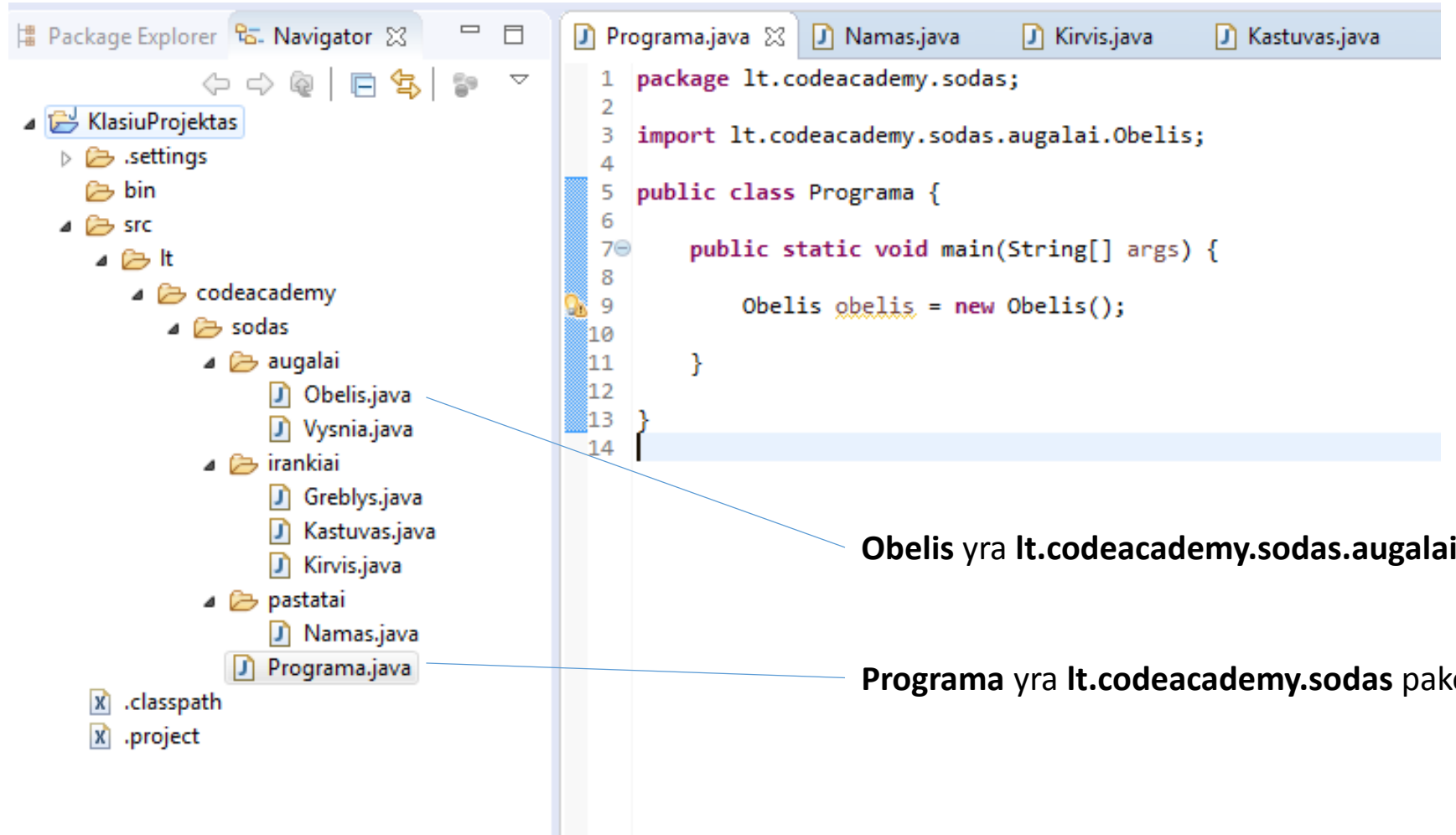
Paketų naudojimas

- Jei klasė yra pakete, tai klasėje reikia nurodyti paketo kelią naudojant **package**
- Jei viena klasė naudoja kitą klasę iš to paties paketo, tai naudojamos klasės importuoti nereikia



Paketų naudojimas

Kai naudojama klasė yra kitame pakete, tada reikia ją importuoti naudojant **import**



```
1 package lt.codeacademy.sodas;
2
3 import lt.codeacademy.sodas.augalai.Obelis;
4
5 public class Programa {
6
7     public static void main(String[] args) {
8
9         Obelis obelis = new Obelis();
10
11     }
12
13 }
14
```

Obelis yra lt.codeacademy.sodas.augalai pakete

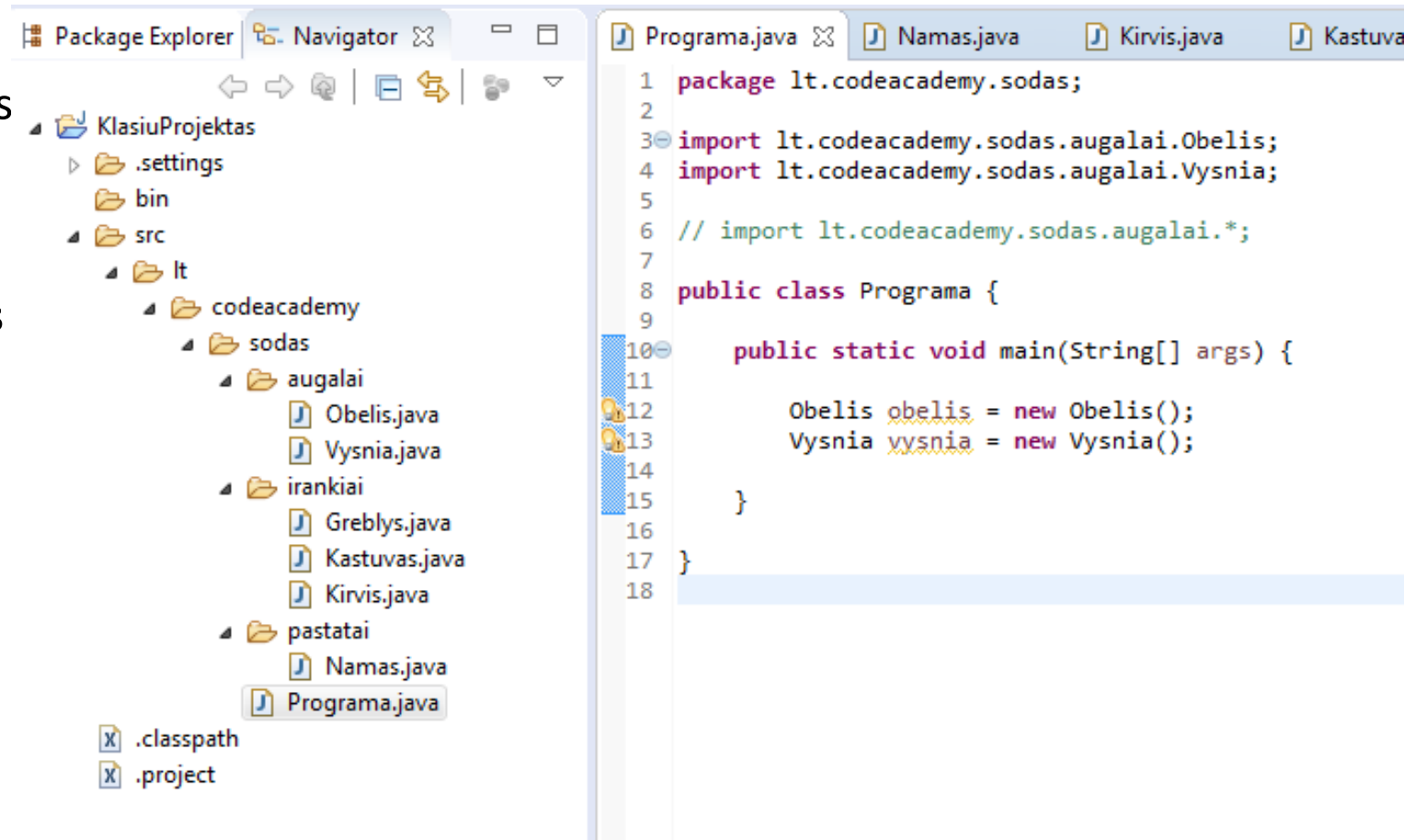
Programa yra lt.codeacademy.sodas pakete

Paketų naudojimas

Jei naudojame kelias klases iš to paties paketo, tai galime importuoti jas abi po vieną arba rašyti

import lt.codeacademy.augalai.sodas

kas reiškia, kad į klasę Programa bus importuotos visos klasės iš to paketo.



Paketų naudojimas

```
Programa.java
1 package lt.codeacademy.sodas;
2
3 import java.util.Scanner;
4
5 public class Programa {
6
7     public static void main(String[] args) {
8
9         Scanner sc = new Scanner(System.in);
10
11     }
12
13 }
14
```

```
Scanner.class
20 * Copyright (c) 2003, 2013, Oracle and/or its affiliates. All rights reserved.
25
26 package java.util;
27
28 import java.nio.file.Path;
29 import java.nio.file.Files;
30 import java.util.regex.*;
31 import java.io.*;
32 import java.math.*;
33 import java.nio.*;
34 import java.nio.channels.*;
35 import java.nio.charset.*;
36 import java.text.*;
37 import java.util.Locale;
38
39 import sun.misc.LRUCache;
40
42 * A simple text scanner which can parse primitive types and strings using
304 public final class Scanner implements Iterator<String>, Closeable {
305
306     // Internal buffer used to hold input
307     private CharBuffer buf;
308
309     // Size of internal character buffer
310     private static final int BUFFER_SIZE = 1024; // change to 1024;
311
```

Kintamieji

```
public class Automobilis {  
    String pavadinimas;  
    int duruSkaicius;  
    float variklioTuris;  
}
```

Klasė

Klasės kintamasis/atributas

Atminties išskyrimas

Pirminių kintamųjų atveju JVM žino, kiek tiksliai reikia išskirti kintamajam atminties, pvz. sveikajam skaičiui `myPrimitive` reikia 4 baitų. Tačiau sveikųjų skaičių masyvui iš anksto neaišku, kiek reikia išskirti atminties, ir todėl `myReference` kintamajam reikia papildomos komandos **`new`** `int[3]`, kuri nurodo, kad šiuo atveju mums reikės masyvo iš 3-ų sveikųjų skaičių. Taigi nuorodų kintamiesiems atmintis taip pat išskiriama programos metu, tik skirtumas tas, kad reikiamos atminties kiekis iš anksto nežinomas ir tam naudojamas **`new`** operatorius.

```
int myPrimitive = 5;
```

```
int[] myReference = new int[3];
```

Objektų nuorodos

Nuorodų kintamieji tik nurodo atminties vietą, kurioje talpinami kintamieji (pvz. masyvo elementai), o pirminiai kintamieji naudoja fiksuoto dydžio atminties fragmentus.

Palyginimas: `==` vs *equals()*

- Primityviųjų tipų kintamų palyginimui galima naudoti tik `==`
- Lyginama reikšmė

```
int a = 3;  
int b = 3;
```

```
System.out.println(a == b); // true
```

Palyginimas: `==` vs `equals()`

- Objektų palyginimui galima naudoti ir `==`, ir `equals()`
- Su `==` lyginamos nuorodos į objektus

[illegible]

Palyginimas: `==` vs `equals()`

- Su *equals()* lyginamos objektų reikšmės

```
String a = "tekstas";
String b = "tekstas";
String c = new String("tekstas");
String d = "kitoks tekstas";
```

[illegible][illegible]

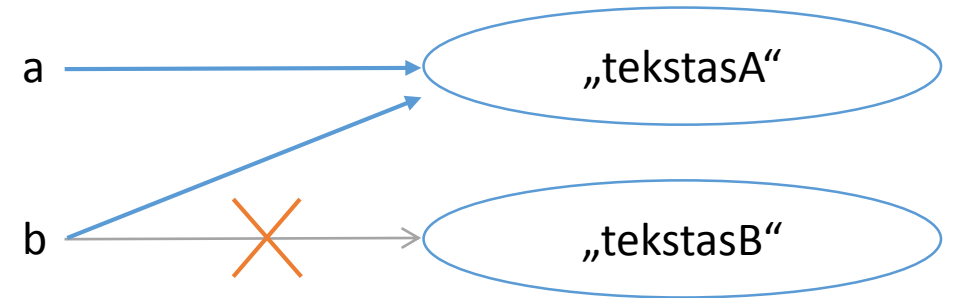
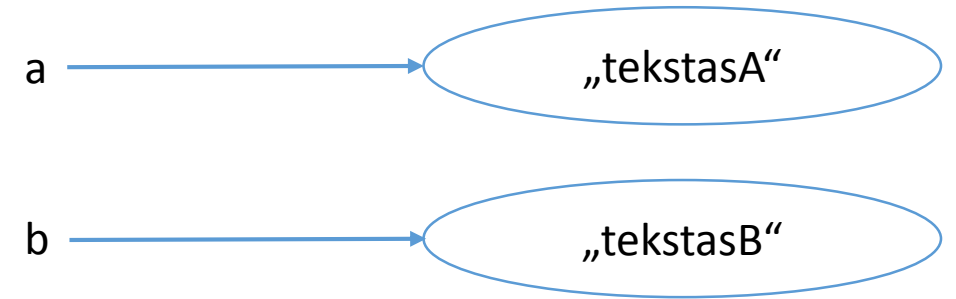
```
System.out.println(a.equals(d)); // false
                                // a ir d objektu reikšmės
                                // skirtingos
```

d = a;

[illegible]

Objektų priskyrimas

```
String a = "tekstasA";  
String b = "tekstasB";  
  
System.out.println(b); // tekstasB  
  
b = a;  
  
System.out.println(b); // tekstasA
```



Metodai

Tai kreipiniai į klase, kurie gali priimti parametrus ir gali grąžinti rezultatą.

Metodų pavyzdys

Metodas be parametų ir grąžinantis
klasės kintamojo rezultatą

Metodas su parametrai ir grąžinantis
paskaičiuotą rezultatą

```
public class Automobilis {
```

```
    private String pavadinimas;  
    private int pagaminimoMetai;  
    private float variklioTuris;
```

```
    public String getPavadinimas() {  
        return pavadinimas;  
    }
```

```
    public String automobilioDuomenys(float kaina) {  
        String rezultatas = "";  
        rezultatas += pavadinimas;  
        rezultatas += " | ";  
        rezultatas += pagaminimoMetai;  
        rezultatas += " | ";  
        rezultatas += variklioTuris;  
        rezultatas += " | ";  
        rezultatas += kaina;  
        return rezultatas;  
    }
```

```
}
```

Metodo struktūra

[modifikatorius] [grąžinamas tipas] [metodo pavadinimas] [parametrai] {}

- **Modifikatorius** – neprivalomas. Gali būti ***public, private, protected***
- **Grąžinamas tipas** – neprivalomas. Jei metodas nieko negrąžina, reikia nurodyti ***void***
- **Metodo pavadinimas** – privalomas. Prasideda mažąja raide
- **Parametrai** – neprivaloma. Jei metodas priima parametrus, jei visi išrašomi per kalblelį nurodant kiekvieno parametro tipą
- Jei metodas grąžina rezultatą, paskutinis metodo sakinys turi būti su žodžiu **return**

Metodų pavyzdžiai

```
public void setName(String name) {} // metodas priimantis 1  
parametra, bet nieko negrazinantis
```

```
String getName() {} // metodas nepriimantis jokiu parametru ir  
grazinantis String tipo rezultata
```

```
private void calculate(int x, int y, String text) {} // metodas  
priimantis 3 parametrus ir nieko negrazinantis
```

Metodų perkrovimas *overloading*

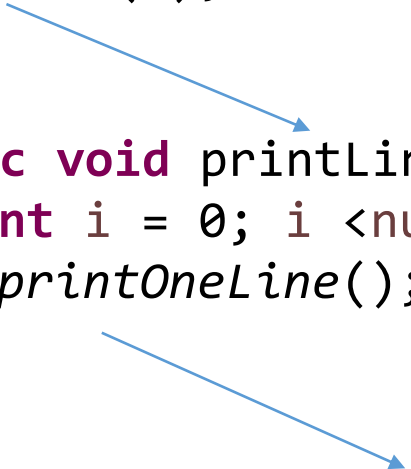
```
public class Programa {  
  
    public static void main(String[] args) {  
        metodos();  
        metodos(20);  
        metodos(1, 2);  
        metodos("tekstas");  
    }  
  
    public static void metodos() {  
        System.out.println("metodas be parametru");  
    }  
  
    public static void metodos(int skaicius) {  
        System.out.println("metodas su vienu int parametru");  
    }  
  
    public static void metodos(int skaicius1, int skaicius2) {  
        System.out.println("metodas su dviem int parametrais");  
    }  
  
    public static void metodos(String tekstas) {  
        System.out.println("metodas su vienu String parametru");  
    }  
  
}
```

Klasė gali turėti kelis metodus su tuo pačiu pavadinimu. Kuris metodas bus išviestas priklauso nuo paduodamų parametų skaičiaus ir tipo. Metodo grąžinamas tipas neturi įtakos.

metodas be parametru
metodas su vienu int parametru
metodas su dviem int parametrais
metodas su vienu String parametru

Metodų kvietimas

```
public class Programa {  
  
    public static void main(String[] args) {  
        printLines(5);  
    }  
  
    public static void printLines(int numberOfLines) {  
        for (int i = 0; i < numberOfLines; i++) {  
            printOneLine();  
        }  
    }  
  
    public static void printOneLine() {  
        System.out.println("*****");  
    }  
  
}
```



Konstruktoriai

- Konstruktorius yra skirtas sukonstruoti duotos klasės tipo objektą.
- Konstruktorius gali turėti parametrų
- Konstruktorius nieko negrąžina (net nerašomas **void**)
- Konstruktoriaus vardas privalo sutapti su klasės vardu
- Klasė gali turėti kelis konstruktorius
- Jei nėra aprašytas nei vienas konstruktorius, pagal nutylėjimą klasė turi konstruktorių be parametrų
- Konstruktoeriai įprastai būna **public**

Konstruktoriaus pavyzdys

```
public class Automobilis {  
  
    private String pavadinimas;  
  
    Automobilis() {  
    }  
  
    Automobilis(String pavadinimas) {  
        this.pavadinimas = pavadinimas;  
    }  
  
}
```

Konstruktorius *this()*

```
public class Automobilis {  
  
    private String pavadinimas;  
    private LocalDateTime sukurimoData;  
  
    Automobilis() {  
        sukurimoData = LocalDateTime.now();  
    }  
  
    Automobilis(String pavadinimas) {  
        this();  
        this.pavadinimas = pavadinimas;  
    }  
  
    public String getPavadinimas() {  
        return pavadinimas;  
    }  
  
    public void setPavadinimas(String pavadinimas) {  
        this.pavadinimas = pavadinimas;  
    }  
  
    public LocalDateTime getSukurimoData() {  
        return sukurimoData;  
    }  
  
    public void setSukurimoData(LocalDateTime sukurimoData) {  
        this.sukurimoData = sukurimoData;  
    }  
  
}
```

this() iškviečia kitą tos pačios klasės Konstruktorių. Kurį konstruktorių Kvieti priklauso nuo parametrų.

Konstruktorius *this()*

```
public class Programa {  
    public static void main(String[] args) {  
        Automobilis automobilis = new Automobilis("Tesla");  
        System.out.println(automobilis.getSukurimoData());  
    }  
}
```

Eiliškumas klasėje

1. Paketas

2. Import

3. Klasės aprašas

4. Konstantos

5. Klasės kintamieji

6. Konstruktoriai

7. Metodai

8. Set ir Get metodai

```
package lt.codeacademy;
```

```
import lt.codeacademy.info.Marke;
```

```
public class Automobilis {
```

```
    public static final String PAGAMINIMO_SALIS = "Italija";
```

```
    private String pavadinimas;
```

```
    private Marke marke;
```

```
    public Automobilis() {
```

```
    }
```

```
    public Automobilis(String pavadinimas, Marke x) {
```

```
        this.pavadinimas = pavadinimas;
```

```
        marke = x;
```

```
    }
```

```
    public float draudimoKaina(float iprastaKaina) {
```

```
        if (Marke.TESLA.equals(marke)) {
```

```
            return iprastaKaina * 10;
```

```
        } else if (Marke.DACIA.equals(marke)) {
```

```
            return iprastaKaina / 5;
```

```
        } else {
```

```
            return iprastaKaina;
```

```
        }
```

```
    }
```

```
    public void setPavadinimas(String pavadinimas) {
```

```
        this.pavadinimas = pavadinimas;
```

```
    }
```

```
    public String getPavadinimas() {
```

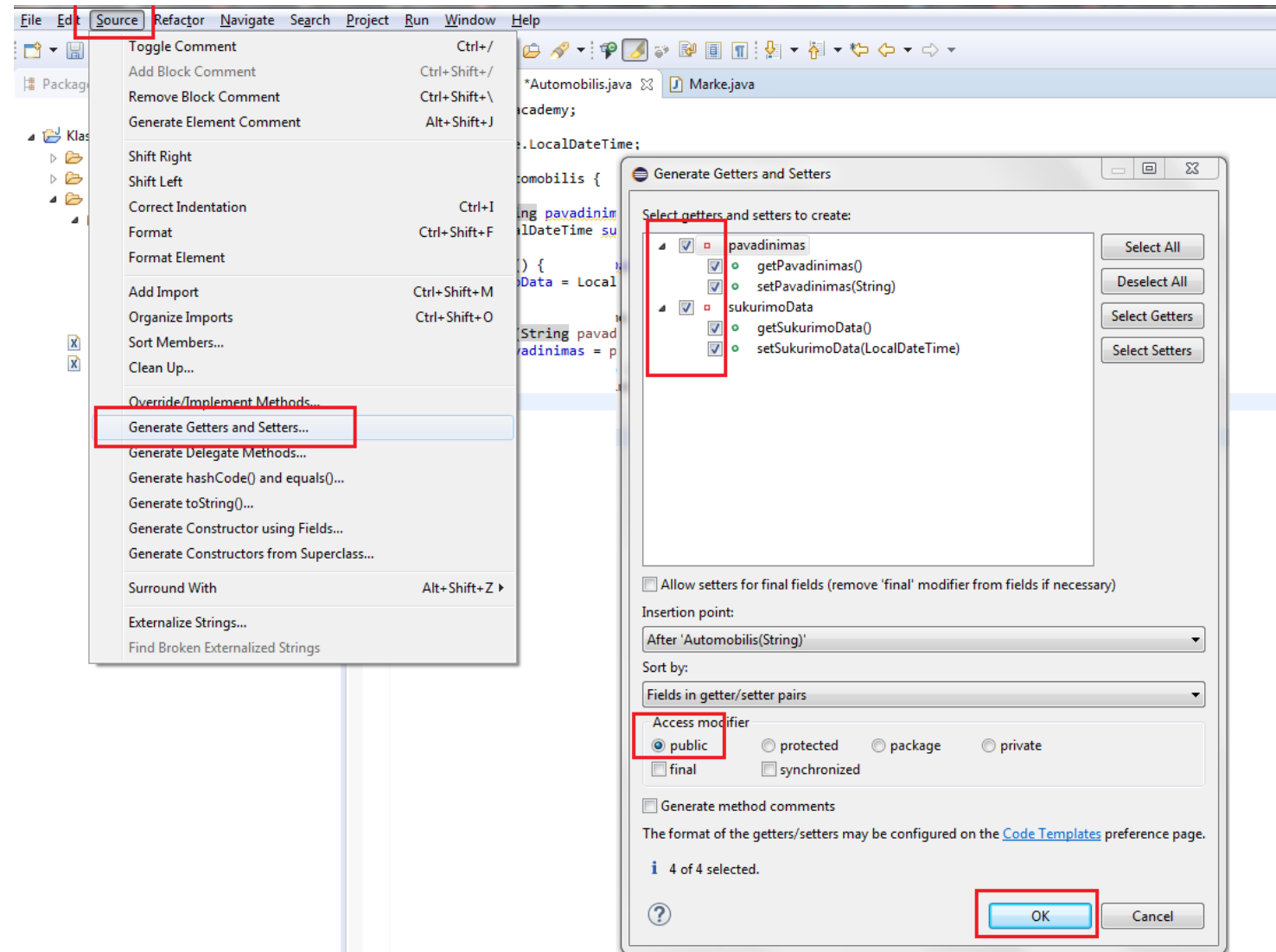
```
        return pavadinimas;
```

```
    }
```

```
}
```

get ir *set* metodu generavimas *eclipse*

Source -> Generate Getters and Setters...



Kintamųjų matomumas klasėje

```
public class Automobilis {
```

```
    private String pavadinimas;
```

```
    Automobilis(String pavadinimas) {  
        this.pavadinimas = pavadinimas;  
    }
```

```
    public String skaiciuok(int param) {
```

```
        int rezultatas = 0;
```

```
        rezultatas += param;
```

```
        String skirtukas = "-";
```

```
        return pavadinimas + skirtukas + rezultatas;
```

```
    }
```

pavadinimas – matomas visoje klasėje

param – metodo parametras matomas visame metode

rezultatas, skirtukas – metode apibrėžtas kintamasis matomas visame metode žemiau nei jis
Yra apibrėžtas

```
}
```

Kintamųjų matomumas klasėje

```
public class Automobilis {
```

```
    private String pavadinimas;
```

```
    Automobilis(String pavadinimas) {  
        this.pavadinimas = pavadinimas;  
    }
```

```
    public String skaiciuok(int param) {
```

```
        int rezultatas = 0;
```

```
        rezultatas += param;
```

```
        String skirtukas = "-";
```

```
        return pavadinimas + skirtukas + rezultatas;
```

```
    }
```

pavadinimas – matomas visoje klasėje

param – metodo parametras matomas visame metode

rezultatas, skirtukas – metode apibrėžtas kintamasis matomas visame metode žemiau nei jis
Yra apibrėžtas

```
}
```


Modifikatoriai

Modifikatorius	Klasė	Interfeisas	Konstruktorius	Metodas	Klasės kintamasis
Access modifiers					
<i>package-private</i>	+	+	+	+	+
private	-	-	+	+	+
protected	-	-	+	+	+
public	+	+	+	+	+
Other modifiers					
abstract	+	+	-	+	-
final	+	-	-	+	+
static	-	-	-	+	+

Modifikatoriai

- *package-private*
 - Klasė, metodas ir klasės kintamasis matomi TIK tame pačiame pakete esančioms klasėms. Joks modifikatorius nerašomas.
- **private**
 - Metodas matomas TIK toje klasėje.
 - Klasės kintamieji matomi TIK toje pačioje klasėje. Klasės kintamieji gali būti netiesiogiai matomi iš išorės naudjant *get* ir *set* metodus.
- **protected**
 - Metodas ir klasės intamasis matomi TIK klasėms, esančioms tame pačiame pakete arba paveldinčioms klasėms
- **public**
 - Klasė, metodas ir klasės kintamasis yra matomi iš visur

Modifikatoriai

- **abstract**
 - Klasė yra abstrakti (turi abstrakčių metodų)
 - Metodas yra abstraktus (neturi kūno)
- **final**
 - Klasė negali būti paveldėta
 - Metodas negali būti perrašytas
 - Kintamajam reikšmė gali būti priskirta tik vieną kartą. Jei nurodyta static final, tai konstanta ir reikšmė priskiriama dar kompiliavimo metu
- **static**
 - Metodas ir kintamasis gali būti prienami naudojant klasės pavadinimą, pvz Klasė.metodas()

Nuorodos

- Kur rasti Java klasių source code:
<https://www.mkyong.com/java/where-to-download-jdk-source-code/>
- Kaip source code prisidėti į *eclipse*:
<https://www.mkyong.com/eclipse/eclipse-how-to-attach-jdk-source-code/>
- Klasės ir objektai
<https://docs.oracle.com/javase/tutorial/java/javaOO/index.html>



static & final

static

- ***static*** metodus galima iškviešti nesukūrus objekto
- Metodai su ***static*** žyme yra laikomi algoritmais, kuriems nėra reikalingi objekto duomenys ir metodai. Statiniams metodams ir kintamiesiems paskiriama statinė atmintis

Statinio kintamojo panaudojimo pavyzdys

```
public class Account {  
  
    // klases atributai  
    private int amount;  
    private int number;  
    private static int numberOfAccounts;  
  
    // konstruktorius  
    public Account() {  
        this.number = numberOfAccounts;  
        numberOfAccounts++;  
    }  
  
    public void deposit(int amount) {  
        this.amount += amount;  
    }  
  
    public void withdraw(int amount) {  
        this.amount -= amount;  
    }  
  
    public int getAmount() {  
        return this.amount;  
    }  
  
    public int getNumber() {  
        return this.number;  
    }  
  
    public void setAmount(int a) {  
        this.amount = a;  
    }  
}
```

```
public class StaticVariable {  
  
    public static void main(String[] args) {  
        Account account = new Account();  
        System.out.println(account.getNumber()); // 0  
        new Account();  
        new Account();  
        new Account();  
        new Account();  
        account = new Account();  
        System.out.println(account.getNumber()); // 5  
        account = new Account();  
        System.out.println(account.getNumber()); // 6  
    }  
}
```

Statiniai atributai ir metodai

- Bendri visai klasei
- Bendri visiems klasės objektams
- Pasiekama ne per objektą, o per klasę
Account.createAccount();
- Naudojama globaliems duomenims bei veiksmams aprašyti

Statinių ir dinaminių narių palyginimas

- Daugeliu atvejų apibrėždami metodus mes nevartojome modifikatoriaus **static**. Šiuo atveju metodas interpretuojamas **dinaminiu** (pagal nutylėjimą)

Statinių ir dinaminių metodų kvietimas

- Statiniai metodai ir kintamieji kviečiami tiesiog rašant pastovų klasės pavadinimą
Account.createAccount();
- Dinaminis metodas ir kintamasis kviečiamas naudojant klasės egzemplioriaus pavadinimą
Account account = new Account();
int amount = account.getAmount();
- Kadangi ta pati klasė gali turėti daug egzempliorių skirtingais pavadinimais, tai ir to paties metodo ar kintamojo kvietimai atrodys skirtingai.
account1.getAmount(); ... account2.getAmount();

static privalumai

- Visi kintamieji ir metodai turi būti apibrėžti klasėje, statinis modifikatorius pažymi tuos metodus ir kintamuosius, kurie nepriklauso nuo egzemplioriaus.
- Kuriant naują egzempliorių (objektą) neišskiriama nauja vieta statiniams klasės metodams ir kintamiesiems ir jie visi prieinami pagal fiksuotą klasės pavadinimą.
- Statiniai kintamieji analogiškai kitose kalbose naudojamiems global kintamiesiems, skirtumas tik tas, kad jie prieinami tik žinant ir panaudojant klasės pavadinimą.

Statinio ir ne statinio metodų pavyzdys

```
class A {  
    void fun1() {  
        System.out.println("Hello I am Non-Static");  
    }  
    static void fun2() {  
        System.out.println("Hello I am Static");  
    }  
}  
  
class Person {  
    public static void main(String args[]) {  
        A oa = new A();  
        oa.fun1();  
        A.fun2();  
    }  
}
```

// non static method
// static method

final kintamieji

- Modifikatorius **final** prie kintamojo reiškia, kad kintamojo reikšmė nekis – ji galutinė
- Kintamajam reikšmę galima priskirti jį deklaruojant
- Jei reikšmę priskiria kompiliatorius, kintamojo vardas rašomas didžiosiomis raidėmis, t.y. Konstanta
final float MANO_PI=3.14;

final kintamieji

final modifikatorius nustato, kad kintamojo reikšmė negali būti pakeista. Šis kintamasis iš karto turi būti inicijuojamas ir bet kuris bandymas jį keisti iššauks kompiliavimo klaidą. **final** modifikatorius paprastai naudojamas apibrėžti konstantas.

final kintamieji

```
class F {  
    final int j = 10;  
  
    // kiti kintamieji <...>  
    F() {  
        // kodas ...  
    }  
  
    F(int i) {  
        // kodas .  
    }  
  
    F(Object o) {  
        // kodas ...  
    }  
    // kiti konstruktoriai ir metodai <...>  
}
```

Galima deklaruoti **final** kintamąjį ir jam iš karto priskirti reikšmę.

final kintamieji

```
class F {  
    final int j;  
  
    F() {  
        this.j = 1;  
    }  
  
    F(int i) {  
        this.j = i;  
    }  
  
    F(Object o) {  
        this.j = 1;  
    }  
}
```

Galima deklaruoti tuščią **final** kintamąjį, o jam reikšmę priskirti **kiekviename** konstruktoriuje

final kintamojo pavyzdys

```
class P {  
    int v = 10;  
}  
  
class D {  
    final P p1 = new P();  
    final P p2 = new P();  
}
```

```
public static void main(String[] args) {  
    D d = new D();  
    // negalima  
    // d.p1=new P(); d.p2=d.p1;  
  
    // galima  
    d.p1.v = 11;  
}
```

Jei kintamasis ne primitiviojo, o objektinio tipo, negalima jam priskirti kitą objektą, tačiau patį objektą modifikuoti galima

static final kintamasis

- Raktų pora **static final** rodo, kad tai konstanta visiems klasės objektams

```
static final float MANO_PI = 3.14f;
```

final metodas

- Jei metodas yra ***final***, jo negalima perrašyti paveldėtose klasėse
- Privatūs metodai ir taip yra ***final*** net nerašant raktinio žodžio
- Jei ***final*** parašysime prie klasės, tai iš jos negalima paveldėti

```
class A {  
    final int metodos() {  
        return 5;  
    }  
}  
class B extends A {  
    // perrašymas negalimas  
    int metodos() {  
        return 6;  
    }  
}
```



StringBuilder

String vs StringBuilder

- String – nemodifikuojami objektai
- StringBuilder – modifikuojami objektai

String vs StringBuilder

```
public static void main(String[] args) {  
  
    String eilute1 = "tekstas";  
    String eilute2 = new String("tekstas");  
  
    eilute1.toUpperCase();  
  
    System.out.println(eilute1);           // tekstas  
  
    StringBuilder sb = new StringBuilder("tekstas");  
    sb.reverse();  
  
    System.out.println(sb);               // satsket  
  
}
```

Objekto būseną/reikšmę
nekeičia.

Metodas *reverse*

```
StringBuilder sb = new StringBuilder("Labas vakaras");
```

```
sb.reverse();
```

Modifikuojamas pats
sb objektas.

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

```
// sarakav sabaL
```

Metodas *append*

```
StringBuilder sb = new StringBuilder("Labas vakaras");  
sb.append("VAKARAS");
```

```
System.out.println(sb);    // Labas vakarasVAKARAS
```

```
sb.append('Q');  
sb.append(9);  
sb.append(true);
```

Modifikuojamas pats
sb objektas.

```
System.out.println(sb);    // Labas vakarasVAKARASQ9true
```

```
sb.append('A').append('B').append('C');
```

```
System.out.println(sb);    // Labas vakarasVAKARASQ9trueABC
```

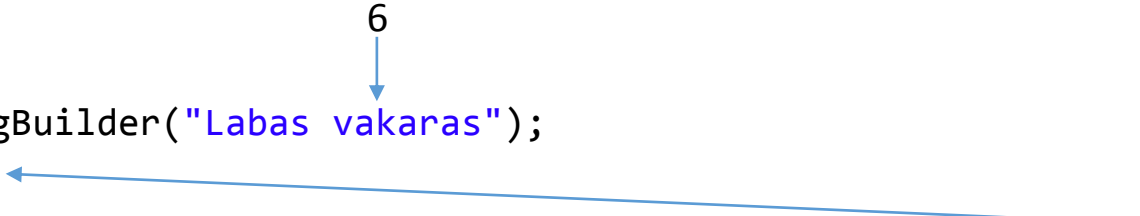

Metodas *length*

```
StringBuilder sb = new StringBuilder("Labas vakaras");  
int ilgis = sb.length();  
System.out.println(ilgis);           // 13
```

Objektas ***sb*** nėra modifikuojamas. Funkcija grąžina reikšmę, kurią reikia priskirti ***int*** tipo kintamajam.

Metodas *indexOf*

```
StringBuilder sb = new StringBuilder("Labas vakaras");  
int va = sb.indexOf("va");  
System.out.println(va);           // 6
```

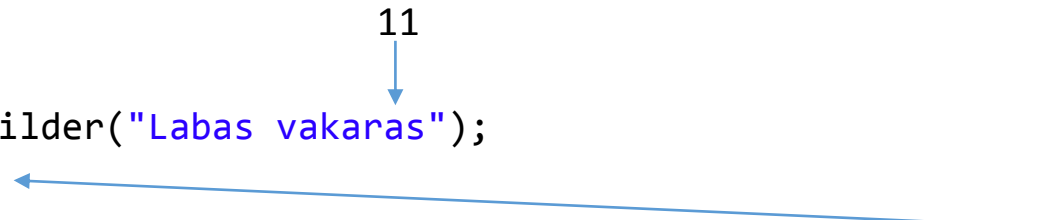


Objektas **sb** nėra modifikuojamas. Funkcija grąžina reikšmę, kurią reikia priskirti **int** tipo kintamajam.

```
StringBuilder sb = new StringBuilder("Labas vakaras");  
int va = sb.indexOf("va", 7);  
System.out.println(va);           // -1
```

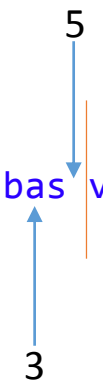
Metodas *lastIndexOf*

```
StringBuilder sb = new StringBuilder("Labas vakaras");  
int va = sb.lastIndexOf("a");  
  
System.out.println(va);    // 11
```



Objektas **sb** nėra modifikuojamas. Funkcija grąžina reikšmę, kurią reikia priskirti **int** tipo kintamajam.

```
StringBuilder sb = new StringBuilder("Labas vakaras");  
int va = sb.lastIndexOf("a", 5);  
  
System.out.println(va);    // 3
```



Metodas *toString*

```
String s = "Labas vakaras";  
StringBuilder sb = new StringBuilder(s);  
String eilute = sb.toString();  
System.out.println(eilute);
```

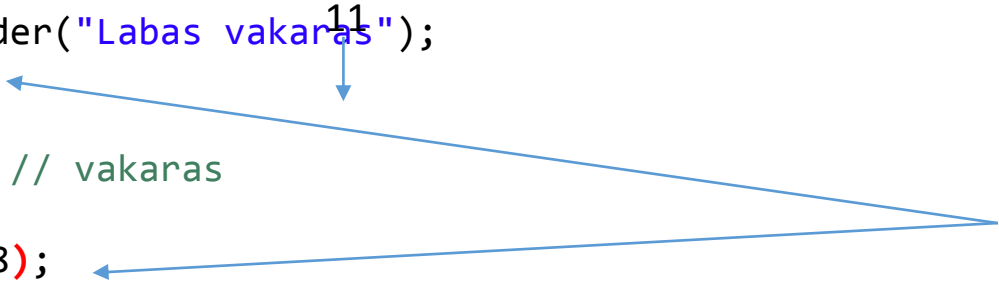
11
↓

// Labas vakaras

Objektas **sb** nėra modifikuojamas. Funkcija grąžina reikšmę, kurią reikia priskirti **String** tipo kintamajam.

Metodas *substring*

```
StringBuilder sb = new StringBuilder("Labas vakaras");  
String eilute = sb.substring(6);  
  
System.out.println(eilute);           // vakaras  
  
String eilute2 = sb.substring(6, 8);  
System.out.println(eilute2);         // va
```



Objektas **sb** nėra modifikuojamas. Funkcija grąžina reikšmę, kurią reikia priskirti **String** tipo kintamajam.

Metodas *delete*

```
StringBuilder sb = new StringBuilder("Labas vakaras");  
sb.delete(3, 6);
```

```
System.out.println(sb); // Labvakaras
```

Modifikuojamas pats
sb objektas.

Metodas *deleteCharAt*

```
StringBuilder sb = new StringBuilder("Labas vakaras");  
sb.deleteCharAt(3);
```

```
System.out.println(sb);           // Labs vakaras
```

Modifikuojamas pats
sb objektas.

Metodas *insert*

```
StringBuilder sb = new StringBuilder("Labas vakaras");  
sb.insert(6, "rytas");  
System.out.println(sb);           // Labas rytasvakaras
```

Modifikuojamas pats
sb objektas.

Metodas *replace*

```
StringBuilder sb = new StringBuilder("Labas vakaras");  
sb.replace(3, 6, "ai ");  
System.out.println(sb);           // Labai vakaras
```

Modifikuojamas pats
sb objektas.

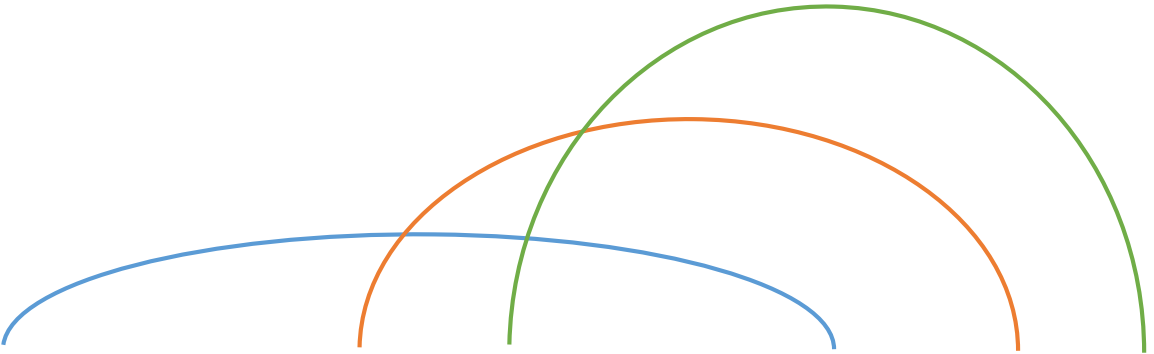


String.format

Metodas *String.format*

- Tai statinis *String* klasės metodas
- Kviečiamas nekuriant *String* tipo objekto (taip pat kaip visi statiniai metodai)
- Metodas skirtas formatuoti tekstą

String.format pavyzdys



```
String formatuotaEilute = String.format("Mano vardas %s. Dabar vyksta %d-oji %s pamoka", "Martynas", 8, "Java");  
System.out.println(formatuotaEilute); // Mano vardas Martynas. Dabar vyksta 8-oji Java pamoka
```

String.format specifikatoriai

Specifier	Applies to	Output
%b	Any type	“true” if non-null, “false” if null
%c	character	Unicode character
%d	integer (incl. byte, short, int, long)	Decimal Integer
%f	floating point	decimal number
%s	any type	String value

Skaičių formatavimas

```
String.format("%d", 93);           // 93
String.format("|%20d|", 93);       // |                               93|
String.format("|%-20d|", 93);     // |93
String.format("|%020d|", 93);     // |00000000000000000000000093|
String.format("%.2f", 93.12678f); // 93,13
```

Eilutės formatavimas

```
String.format("|%s|", "Hello World");    // Hello World
String.format("|%15s|", "Hello World");  // |      Hello World|
String.format("|%-15s|", "Hello World"); // |Hello World      |
```

Specialieji *String* simboliai

Escape Sequence	Description
\t	Insert a tab in the text at this point.
\n	Insert a newline in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

Specialusis simbolis `\t`

```
System.out.println("Labas\tVakaras");
```



Labas Vakaras

Specialusis simbolis `\n`

```
System.out.println("Labas\nVakaras");
```



Labas
Vakaras

Specialusis simbolis \'


```
System.out.println("Labas \'Vakaras\'");
```



Labas 'Vakaras'

Specialusis simbolis \"


```
System.out.println("Labas \"Vakaras\"");
```



Labas "Vakaras"

Specialusis simbolis \

```
System.out.println("Labas\\Vakaras\\")
```



Labas\Vakaras\

System.out.println vs. \n

```
System.out.println("Vienas");
```

```
System.out.println("Du");
```

```
System.out.println("Trys");
```

arba

```
System.out.println("Vienas\nDu\nTrys");
```

Vienas

Du

Trys

Nuorodos

- Formatter:

<https://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html>