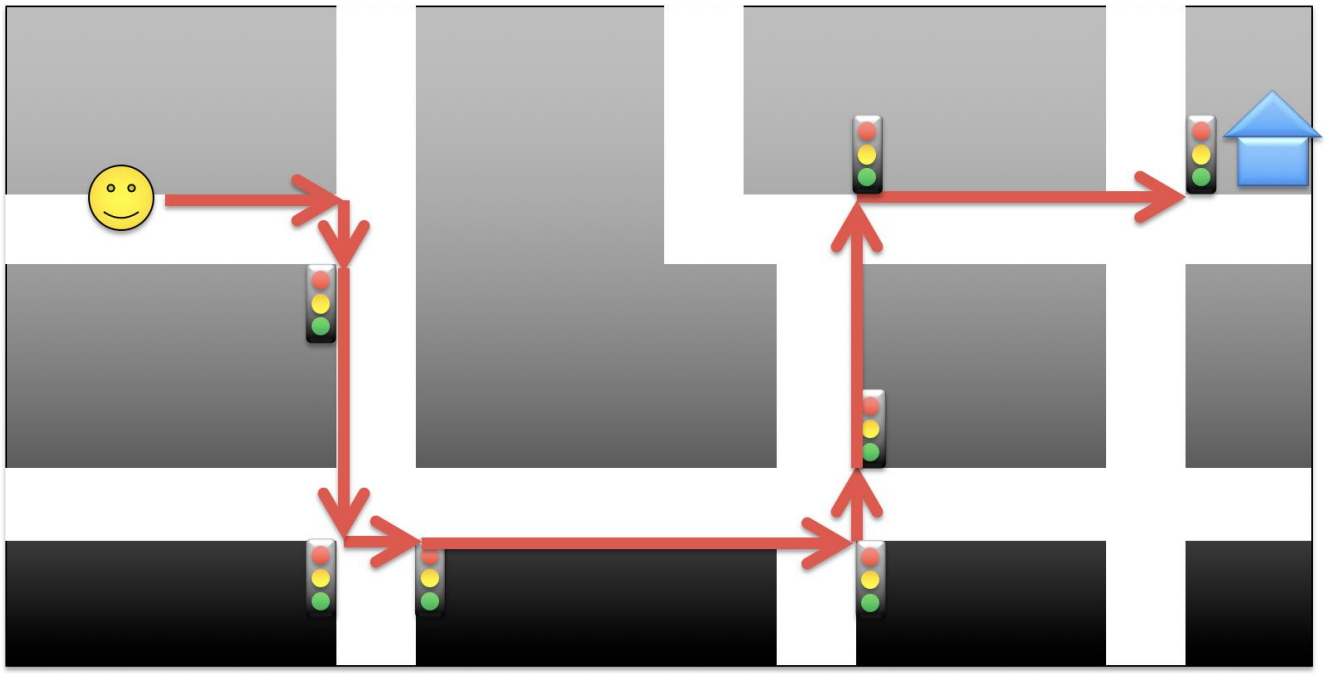
FUNKCIJAS, PROCEDŪRAS, METODES
33.nodarbība

APAKŠPROGRAMMAS

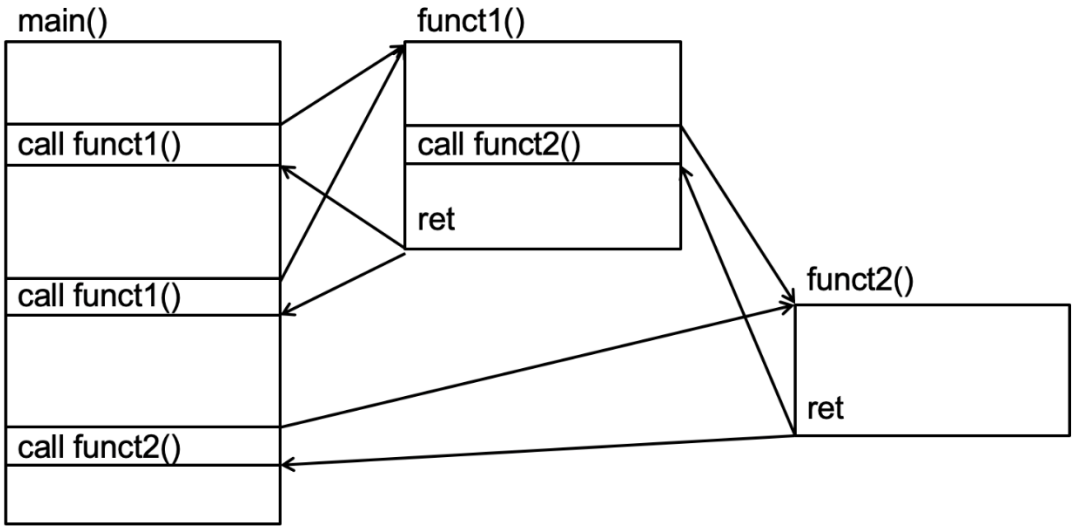


APAKŠPROGRAMMAS

- Vairāku komandu kopa, kas izpilda kādu uzdevumu (darbu).
- Var izsaukt no dažādām vietām.
- Apakšprogrammas var apvienot pakotnēs (bibliotēkās).
- Piemērs:

```
int taisnsturaLaukums( int platums, int augstums ) {
    return platums * augstums;
}
```

APAKŠPROGRAMMAS PIEMĒRS



FUNKCIJAS, PROCEDŪRAS, METODES

- **Funkcija** – apakšprogramma, kas atgriež kādu rezultātu.
  - Piem., int getSize();
- **Procedūra** – apakšprogramma, kas izpilda darbu, bet neko neatgriež.
  - Piem., void startTimer();
- **Metode** – apakšprogramma objektorientētā programmēšanā (OOP), kas saistīta ar klasi.

PIEMĒRI DAŽĀDĀS VALODĀS

PASCAL	C++	Java
Ir gan procedūras, gan funkcijas	Formāli ir tikai funkcijas (metodes)	Ir tikai metodes
Procedūras definē ar atslēgvārdu **procedure**, bet funkcijas – ar **function**	Ja nav nepieciešams atgriezt vērtību, tad izmanto tipu **void**	Ja nav nepieciešams atgriezt vērtību, tad izmanto tipu **void** (tukšs)

Piemērs: procedure print_num; function get_size(): real;	Piemērs: int getSize() { }; void printInt(int i) { };	Piemērs: int getSize() { }; void printInt(int i) { };
	Funkcija – ārpus klases Metode – klasē	

APAKŠPROGRAMMU IEDALĪJUMS

- Iebūvētās
  - parasti piedāvā:
    - ievadi un izvadi uz ekrāna (konsolē),
    - matemātiskās funkcijas,
    - simbolu virkņu apstrādi,
    - grafiku,
    - darbu ar failu sistēmu,
    - …
- Lietotāja veidotās

Java iebūvētās matemātiskās metodes (Math. …)

- abs – skaitļa modulis
- max,min – lielākais, mazākais skaitlis
- pow, sqrt – pakāpe (x^y), kvadrātsakne
- ceil, floor – atgriež no augšas/lejas tuvāko veselo skaitli
- round – noapaļo
  - ja decimāldaļa <0.4 uz leju, ja >=0.5 uz augšu
- acos, asin, atan, atan2, cos, sin, tan … - trigonometrija

- exp – e^x
- log, log10 – logaritmi
- random – pseidogadījumskaitlis intervālā [0;1)
  - new java.util.Random();
- toDegrees, toRadians – pārvērš starp grādiem un radiāniem
- … http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html

JAVA IEBŪVĒTĀS MATEMĀTISKĀS METODES – PIEMĒRI

Kods	Rezultāts
double a=1.3, b=1.7, c=-1.2;	1.2
System.out.println(Math.abs(c));	1.7
System.out.println(Math.max(a, b));	1.3
System.out.println(Math.min(a, b));	16.0
System.out.println(Math.pow(2, 4));	2.0
System.out.println(Math.ceil(a));	1.0
System.out.println(Math.floor(a));	1
System.out.println(Math.round(a));	2
System.out.println(Math.round(b));	

JAVA IEBŪVĒTĀS MATEMĀTISKĀS METODES – MATH.RANDOM()

- Math.random() atgriež gadījumskaitli no 0 (ieskaitot) līdz 1 (neieskaitot).
- Automātiski izsauc:
  new java.util.Random();
- Lai iegūtu veselu skaitli intervālā [a;b], lieto formulu:
  a+(int)(Math.random()*(b-a+1))
- Piemēram, lai iegūtu skaitli no 5 līdz 7:
  5+(int)(Math.random()*(7-5+1))

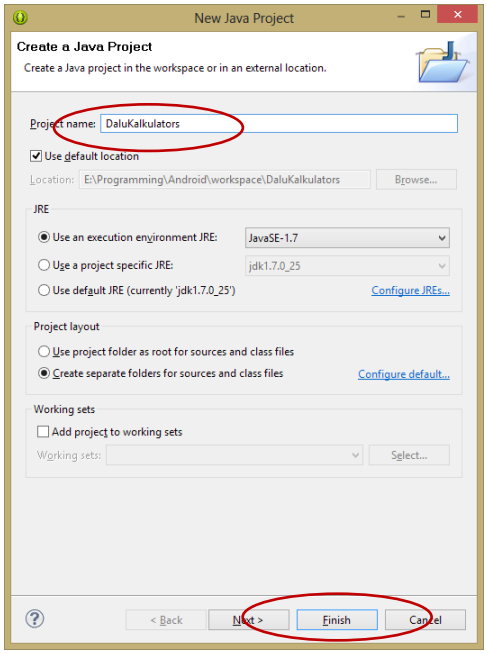
```
Kāpēc b-a+1? Ja a=5 un b=7
b-a = 7-5 = 2
[5;7] = {5,6,7}
random()*3-> [0;3)
(int)(random()*3) = {0,1,2}
5+(int)(random()*3) = {5,6,7}
```

PROGRAMMA – DAĻU KALKULATORS

- Uzrakstīsim programmu, kas veic darbības ar skaitli:
  - Programmā ievada skaitli,
  - Programma izvada:
    - skaitļa veselo daļu,
    - decimāldaļu,
    - noapaļotu skaitli.

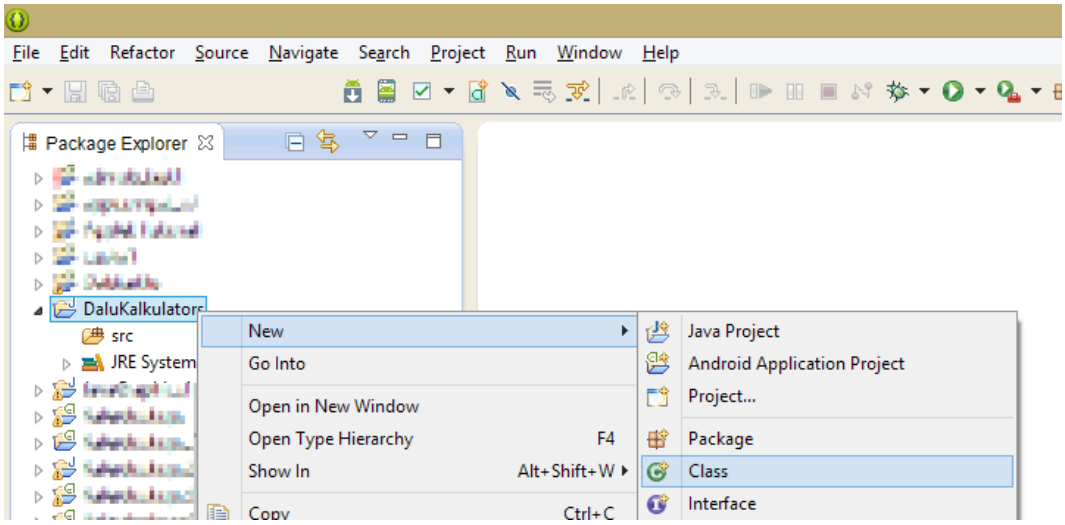
PROJEKTS – DAĻU KALKULATORS

- File/New/Java project
- Projekta nosaukums: **DaluKalkulators**



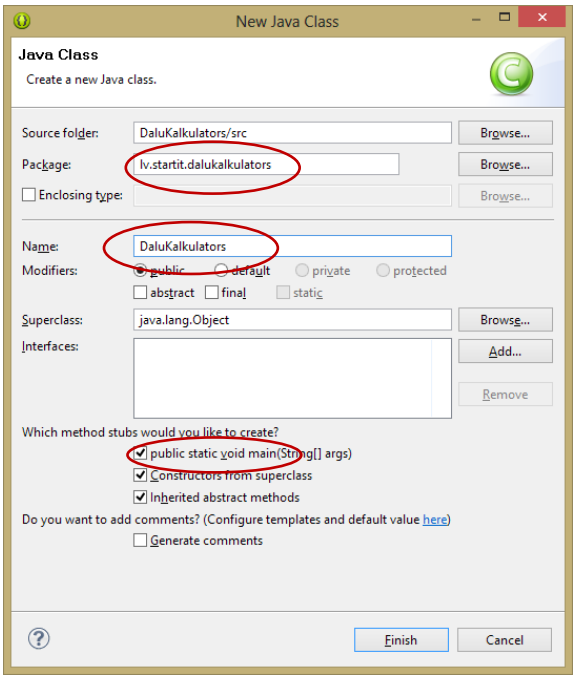
KLASE – DaluKalkulators I

- Spiežot labo peles pogu uz projekta nosaukuma, izvēlas New/Class



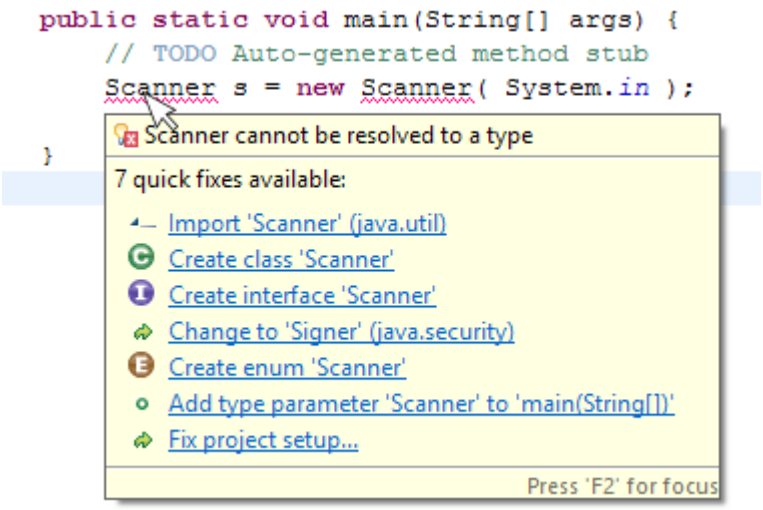
KLASE – DaluKalkulators II

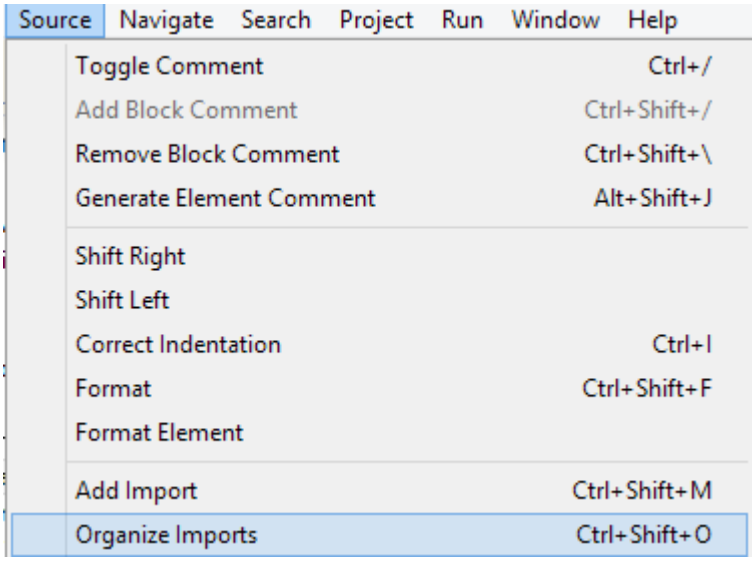
- Logā ievada:
  - Pakotne lv.startit.dalukalkulators
  - Nosaukums DaluKalkulators
  - Ieslēdz izvēli public static void main



KLASES DaluKalkulators KODS I

```
public static void main(String[] args) {
  // izveido skeneri
  Scanner s = new Scanner( System.in );
}
```





Ctrl+Shift+O

KLASES DaluKalkulators KODS II

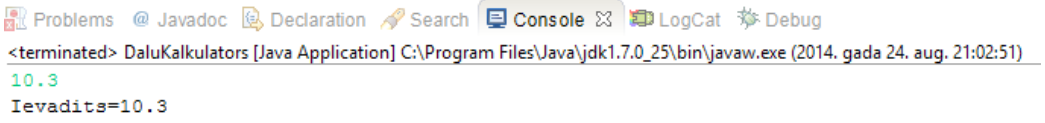
```
public static void main(String[] args) {
  // izveido skeneri
  Scanner s = new Scanner( System.in );
}
```

KLASES DaluKalkulators KODS III

```
public static void main(String[] args) {
  // izveido skeneri
  Scanner s = new Scanner( System.in );
  // izdrukā uz ekrāna lietotāja instrukciju
  System.out.println(
      "Dalu kalkulators. Ievadi decimalskaitli.");
  // ieslēdz ASV lokālizāciju
  //   (lai decimāldaļu atdalītu ar punktu)
  ievade.useLocale(Locale.US);
  // nolasa decimālskaitli
  double x = ievade.nextDouble();
  // izdrukā ievadīto skaitli (pārbaudei)
  System.out.println("Ievaditais skaitlis=" + x);
  // ja skeneris ir atvērts, tad aizver to
  if (ievade != null)
    ievade.close();
}
```

TESTĒŠANA

- Programmu vēlams testēt bieži, lai pēc iespējas ātrāk pamanītu kļūdas.



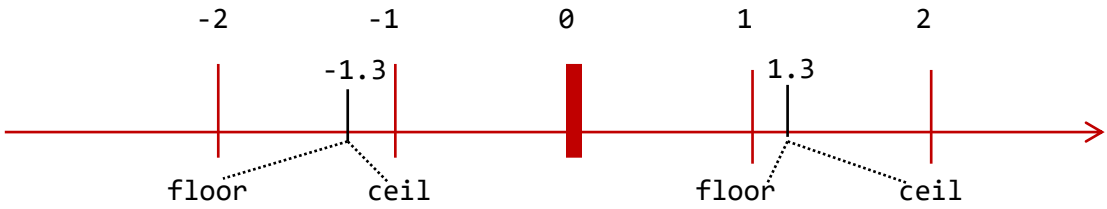
KLASES DaluKalkulators KODA TURPINĀJUMS

```
public static void main(String[] args) {
. . .
  System.out.println("Ievaditais skaitlis=" + x);
 // noapaļo skaitli
 System.out.println( "round=" + Math.round(x));
 // aprēķina tuvāko veselo, kas mazāks
 System.out.println( "floor=" + Math.floor(x));
 // aprēķina tuvāko veselo, kas lielāks
 System.out.println( "ceil=" + Math.ceil(x));
 // aprēķina skaitļa decimāldaļu
 System.out.println( "Decimal=" + (x-(int)x));
 // pārveido tipu par int
 System.out.println( "(int)=" + (int)x);  // ja
. . .
}
```

REZULTĀTI

Ievadits=1.3 **round=1** **floor=1.0** **ceil=2.0** **Decimal=0.300000...04** **(int)=1**	**Ievadits=1.5** **round=2** **floor=1.0** **ceil=2.0** **Decimal=0.5** **(int)=1**	**Ievadits=1.7** **round=2** **floor=1.0** **ceil=2.0** **Decimal=0.7** **(int)=1**
Ievadits=-1.3 **round=-1** **floor=-2.0** **ceil=-1.0**	**Ievadits=-1.5** **round=-1** **floor=-2.0** **ceil=-1.0**	**Ievadits=-1.7** **round=-2** **floor=-2.0** **ceil=-1.0**

Decimal=0.300…04 (int)=-1	Decimal=-0.5 (int)=-1	Decimal=-0.7 (int)=-1



JAUNAS METODES

- Java nav iebūvētas gatavas metodes, kas, piemēram:
  - atgriež decimāldaļu,
  - aprēķina skaitļa kvadrātu,
  - noskaidro skaitļa zīmi (-, 0, +),
  - …
- Uzrakstīsim paši savas metodes!

METODES DEKLARĒŠANA JAVĀ

```
public double printMsg(int errNum, String errMsg) {
    // drukāšanas kods
}
```

- Piekļuves modifikators (public, private, …)
- Atgrieztās vērtības tips (void,int,String,…)
- Metodes nosaukums
- Parametru saraksts vai tukšas iekavas
- Metodes ķermenis (figūriekavās)

METODES PARAMETRU SARAKSTS UN ATGRIEZTĀ VĒRTĪBA

- Parametrus raksta iekavās, katram norādot mainīgā tipu un nosaukumu. Parametri var nebūt.
- Piemēri:

```
void test1() { … }
int  test2() { … return … }
void test3(int skaitlisA) { … }
int  test4(String s,int b,Point p) {
     … return … }
```

METODES ATGRIEZTĀS VĒRTĪBAS IZMANTOŠANA

- Atgriezto vērtību var:
  - piešķirt mainīgajam:
    - int min = Math.min(a, b);
    - int positive = Math.abs(variable);
  - izmantot izteiksmēs:
    - int s = a + Math.min(b, c) + d;
    - System.out.println(Math.ceil(a));

METODE getDecimal

```
public class DaluKalkulators {
  public static void main(String[] args) {
    . . .
    System.out.println( "Decimal=" + getDecimal(x));
    System.out.println( "AbsDec=" + getAbsDecimal(x));
    . . .
  }
. . .
  static double getDecimal( double a ) {
```

```
    return a - (int)a;
  }
  static double getAbsDecimal( double a ) {
    return Math.abs(a - (int)a);
  }
}
```

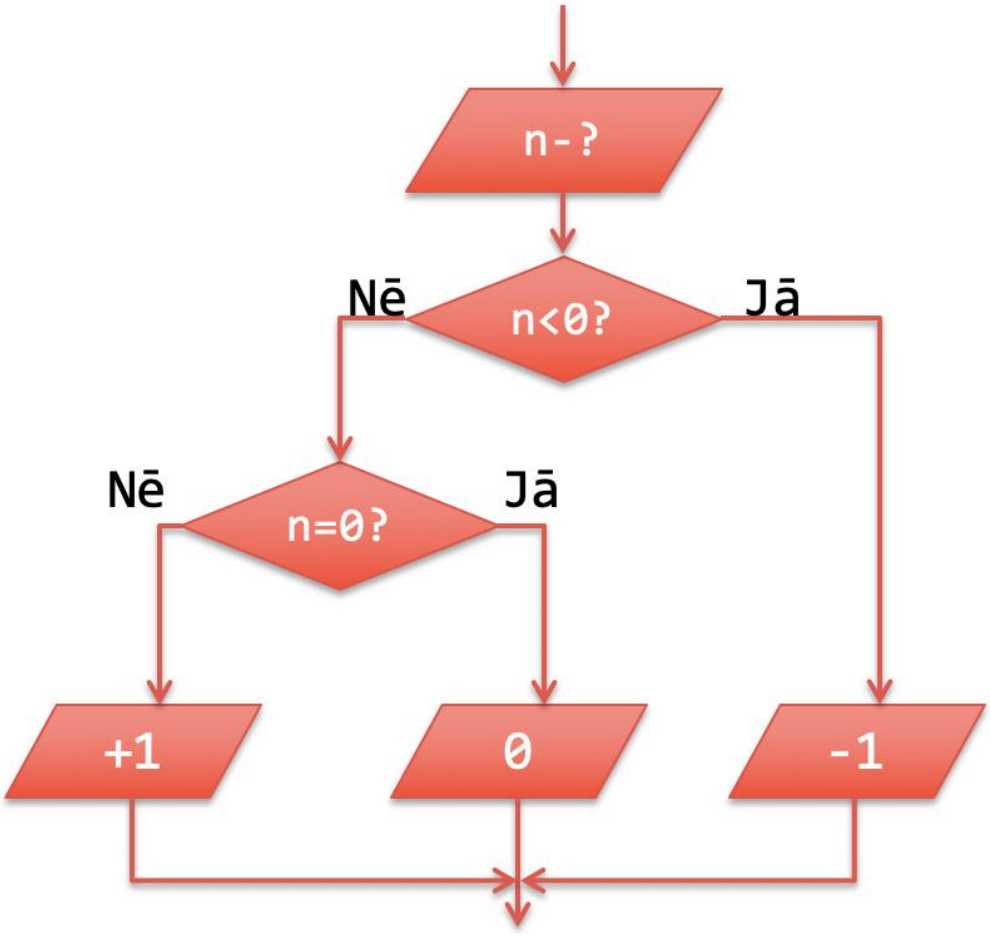
```
-1.2
Ievadits=-1.2
round=-1
floor=-2.0
ceil=-1.0
Decimal=-
0.19999999999999996
(int)=-1
Decimal=-
0.19999999999999996
AbsDec=0.199999999999999
96
```

METODES square, sign

```
public class DaluKalkulators {
. . .
  // metode aprēķina skaitļa kvadrātu
  static double square( double num ) {
    // var izmantot Math.pow(num, 2.0 );
    // bet reizināt ir efektīvāk nekā
    // izsaukt metodi pow
    return num * num;
  }

  // metode atgriež -1, ja skaitlis negatīvs,
  // 0, ja vienāds ar 0 un +1, ja pozitīvs
  static double sign( double num ) {
```

```
        if( num < 0 ) {
            return -1;
        } else {
            if( num == 0 ) {
                return 0;
            } else {
                return 1;
            }
        }
    }
}
```



METOŽU square, sign TESTS

```
public static void main(String[] args) {
  . . .
  System.out.println( "Kvadrats=" + square(x));
  System.out.println( "Ziime=" + sign(x));
  . . .
}
```

-2.3	0	3.2
Kvadrats=5.289…99 Ziime=-1.0	Kvadrats=0.0 Ziime=0.0	Kvadrats=10.240…02 Ziime=1.0

MAINĪGO REDZAMĪBA JAVĀ

- **Lokālie** mainīgie:
  - deklarē metodes ķermenī,
  - pieejami tikai metodes iekšienē, kamēr tā tiek izpildīta,
  - netiek inicializēti.
- **Lauki** (klases mainīgie, atribūti):
  - deklarē klasē, ārpus metodēm,
  - pieejami visām klases metodēm,
  - tiek inicializēti (0, false, null),
  - katrai klases instancei savi.

LOKĀLO MAINĪGO PIEMĒRS

```
public class TestClass1 {
  public void testPrint() {
     int i = 0; // vajag inicializēt
     String str = ""; // vajag inicializēt
```

```
            str = str + i;
            i = i + 1;
            str = str + i;
            i = i + 1;
            str = str + i;
            System.out.println( str );
        }
    }
    // pārbaude
    public static void main(String[] args) {
            TestClass1 t1 = new TestClass1();
            t1.testPrint();// izdrukā 012
    }
```

LAUKU (KLASES MAINĪGO) PIEMĒRS I

```
    public class TestClass2 {
       private int x; // automātiski inicializējas ar 0
       public int getX() {
          return x;
       }
       public void setX( int xValue) {
          x = xValue;
       }
       public void addX( int add ) {
          x += add;
       }
    }
    // pārbaude
    public static void main(String[] args) {
       TestClass2 t2 = new TestClass2();
```

```
        t2.setX( 5 );
        t2.addX( 10 );
        System.out.println( t2.getX() ); // izdrukā 15
    }
```

a += b	a = a + b
a -= b	a = a - b
a *= b	a = a * b
a /= b	a = a / b
…	

LAUKU (KLASES MAINĪGO) PIEMĒRS II

```
    public class TestClass3 {
        private int x, y; // lauki
        public void setX( int x ){ // x šeit ir lokāls
            this.x = x; // this.x piekļūst klases mainīgajam
        }
        public void setY( int y ){ // y šeit ir lokāls
            this.y = y;
        }
        public int getXY() {
            int z; // lokāls mainīgais
            z = x+y; // x un y ir lauki no klases
            return z;
        }
```

```
}
// pārbaude
public static void main(String[] args) {
   TestClass3 t3 = new TestClass3();
   System.out.println( t3.getXY() ); // atgriež 0
   t3.setX(3);
   t3.setY(5);
   System.out.println( t3.getXY() ); // atgriež 8
}
```

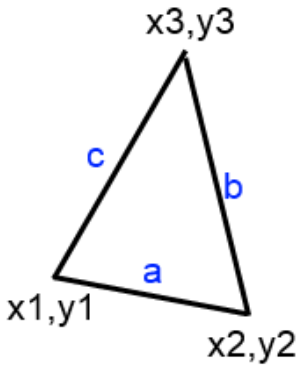
KĻŪDAS

```
public class TestClass4 {
  private String str = "";
  public void printNum( int i ) {
      str = str + i;
      System.out.println( str );
      int i=1; // i jau definēts iepriekš
      System.out.println( i );
}
  public void illegal() {
      int j;
      j = j + 3; // mainīgā «j» vērtība nav inicializēta
      i = i + 1; // mainīgais «i» nav definēts šajā metodē
  }
}
```

TRĪSSTŪRA LAUKUMS

- Uzrakstīsim metodi, kas aprēķina trīsstūra laukumu
- $S = \sqrt{p(p-a)(p-b)(p-c)}$, kur p=pusperimetrs
- Tīsstūri uzdosim ar koordinātēm x_1, y_1, x_2, y_2, x_3, y_3

- Jāaprēķina katras malas garums
- $d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$



METODES getLineDistance UN getTriangleArea

```
public double getLineDistance( int x1, int y1,
                               int x2, int y2 ) {
  return Math.sqrt( Math.pow(x2-x1, 2)+
                    Math.pow(y2-y1, 2) );
  // vai Math.sqrt( (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1))
}

public double getTriangleArea( int x1, int y1,
                               int x2, int y2,
                               int x3, int y3 ) {
  double a, b, c, p, S;
  a = getLineDistance(x1, y1, x2, y2);
  b = getLineDistance(x2, y2, x3, y3);
  c = getLineDistance(x3, y3, x1, y1);
  p = (a+b+c)/2; // pusperimetrs
  S = Math.sqrt(p*(p-a)*(p-b)*(p-c));
  return S;
}
```