Objektorienteeritud programmeerimine

4. märts, 4. loeng

Eno Tõnisson

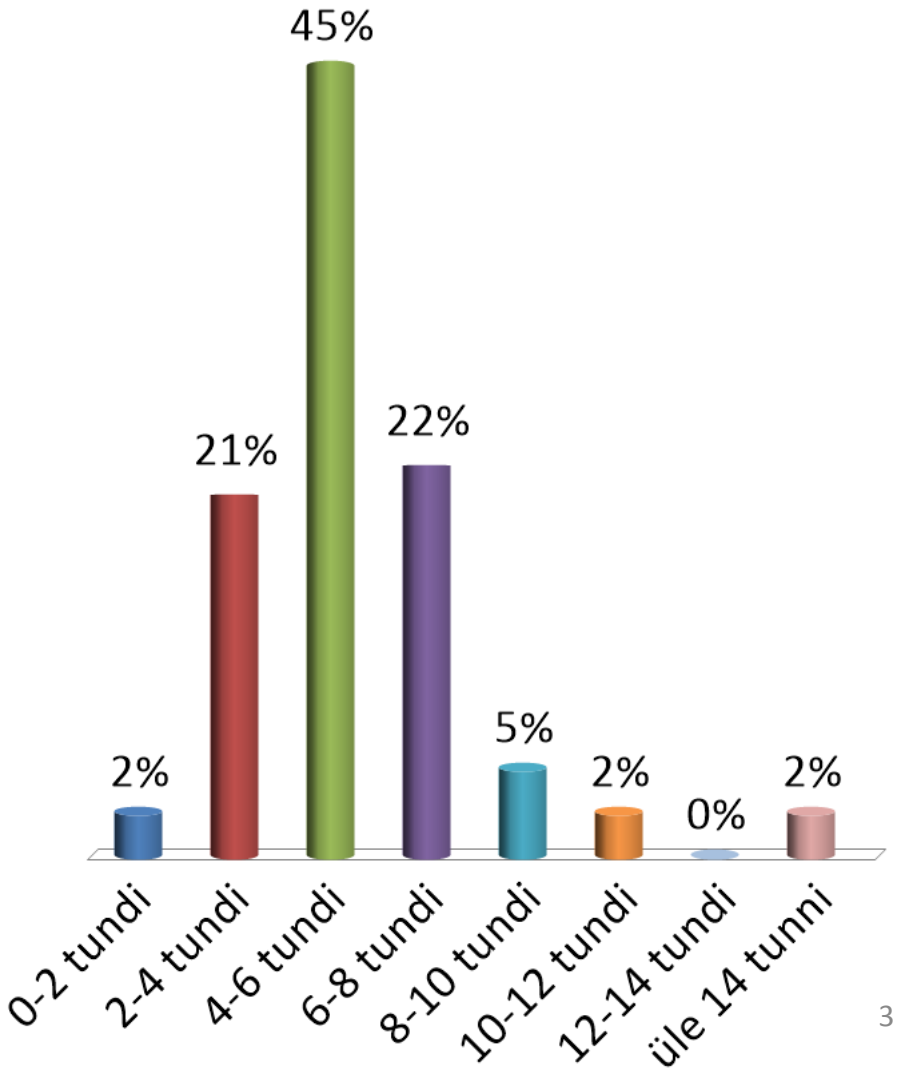
Möödunud nädalal

- Loeng
  - Klassid. Isendid. Konstruktorid. Sõned. Mähisklassid
- Praktikum
  - Objektid ja klassid. Muutujate skoobid. Objektide edastamine meetoditele
- Lisapraktikum

- 27.02 – rahvusvaheline jääkaru päev
- 3.03 – riigikogu valimised

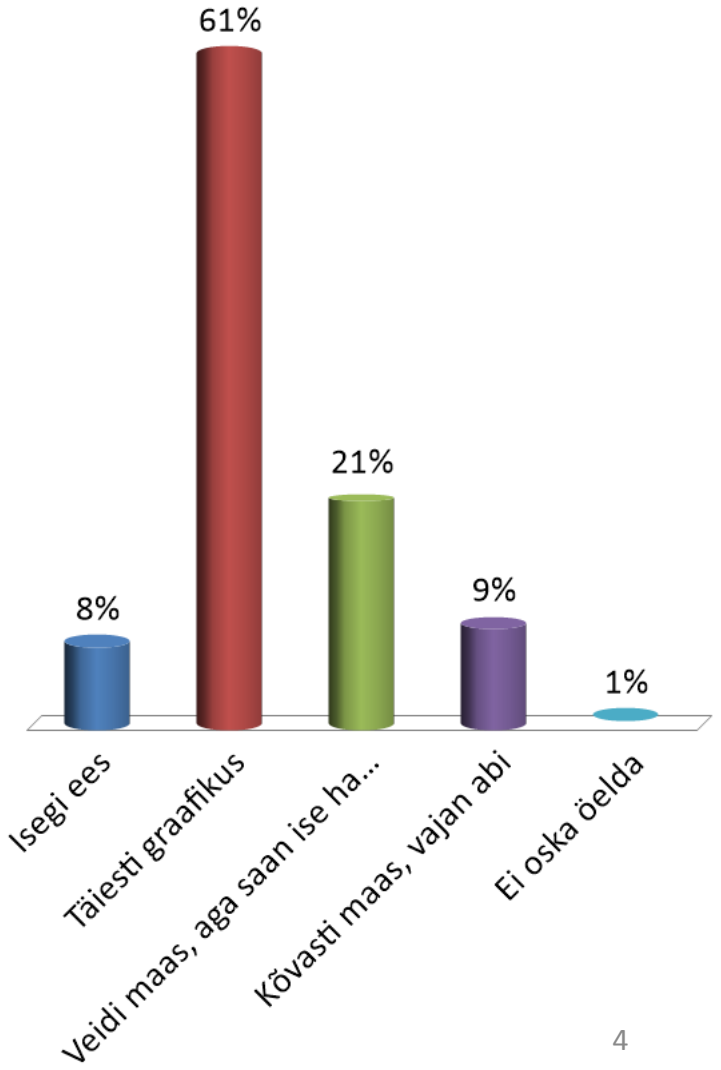
Umbes mitu tundi tegelesite eelmisel nädalal selle ainega (loeng+praktikum+iseseisvalt)?

1. 0-2 tundi
2. 2-4 tundi
3. 4-6 tundi
4. 6-8 tundi
5. 8-10 tundi
6. 10-12 tundi
7. 12-14 tundi
8. üle 14 tunni



Kuivõrd olete selle ainega graafikus?

1. Isegi ees
2. Täiesti graafikus
3. Veidi maas, aga saan ise hakkama
4. Kõvasti maas, vajan abi
5. Ei oska öelda



Täna

- Listid
  - Tüübiparameetrid
  - Mähisklassid
- Liidesed

Massiiv

- Samatüübilised elemendid kindlas järjekorras

`int[]`

- Massiivi korral on kaks olulist piirangut:
  - kui massiiv on juba loodud, siis tema suurust enam muuta ei saa
  - massiiv ise ei toeta elemendi lisamist ja kustutamist

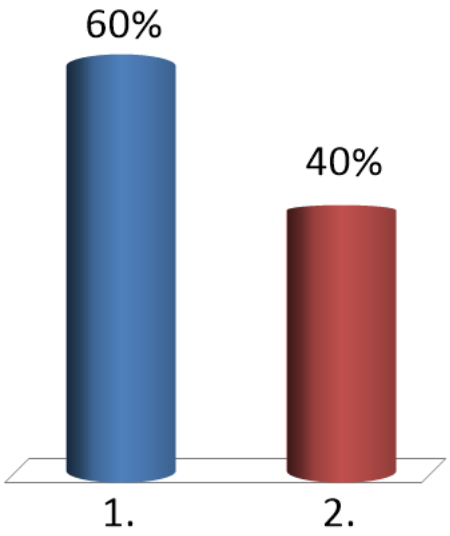
- Massiiv on staatiline andmestruktuur
  - mälukasutus on staatiline: mälu eraldatakse vaid korra (maksimaalne elementide arv on fikseeritud)
    - eraldatud mälu maht on kindel suurus
  - sisemine struktuur on fikseeritud ja ei muutu töö käigus

Kas peameetodi päis võib olla

```
public static void main(String args[])
```

✓ 1. jah

2. ei



Dünaamilised andmestruktuurid

- Muutuva suurusega
- Võimaldavad elemente lisada ja eemaldada
- Võimaldavad teha päringuid suuruse (elementide arvu), konkreetse elemendi sisalduvuse kohta

- Põhitüübid:
  - list (*list*)
  - magasin (*stack*)
  - järjekord (*queue*)
  - kuhi (*heap*)
  - puu (*tree*)
  - graaf (*graph*)
- …

Mis on olemas?

- Mitte niivõrd põhjalik ülevaade kõigest, kuivõrd ettevalmistus ise olulistele asjadele tähelepanu pööramiseks
- Ajalooliselt
  - massiiv (nt. **`int[]`**), **`Vector`**, **`Hashtable`**
  - on ka praegu
- Nüüd
  - *Java Collections Framework*
    - *https://docs.oracle.com/javase/tutorial/collections/index.html*
  - Väga palju erinevaid võimalusi
    - Liidesed
    - Abstraktsed klassid
    - Klassid

List

- Andmestruktuur, milles andmed on kindlas järjekorras
- Saab
  - elemendi võtta
    - `get`
  - elemendi lisada
    - `add`
  - elemendi eemaldada
    - `remove`

- Saab mitut moodi realiseerida
  - Näiteks klassid `ArrayList`, `LinkedList`
  - Liides *List*

Klass *ArrayList*

- `add`
- `get`
- `remove`
- `size`
- …

- API

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ArrayList.html

Listide jaoks on teisi klasse ka ja saame neid juurde teha

Uus aspekt

```
class ArrayList<E>
```

```
import java.util.ArrayList;
public class GeneeriliseNaide {
    public static void main(String[] args) {
        ArrayList loend1 = new ArrayList();
        loend1.add(18);
        int i1 = (int)loend1.get(0);
        System.out.println(i1);
    }
}
```

18

```java
import java.util.ArrayList;
public class GeneeriliseNaide {
    public static void main(String[] args) {
        ArrayList loend1 = new ArrayList();
        loend1.add("18");
        int i1 = (int)loend1.get(0);
        System.out.println(i1);
    }
}
```

Exception in thread "main" java.lang.ClassCastException:

ArrayList<Integer>

GeneeriliseNaide.java

```
1  import java.util.ArrayList;
2  public class GeneeriliseNaide {
3      public static void main(String[] args) {
4          ArrayList<Integer> loend1 = new ArrayList<>();
5          loend1.add("18");
6          int i1 = (int)loend1.get(0);
7          System.out.println(i1);
8      }
9  }
10
```

Tüübiparameeter, tüübimuutuja

Kompileerimisaegne viga!

Kas viga on viga?!

- Miks kompileerimisaegne viga parem on?

http://docs.oracle.com/javase/tutorial/java/generics/

Klass **`ArrayList`**

- Listid ei saa sisaldada algtüüpi elemente
  - massiivid saavad
  - mähisklasside abil

Mähisklass

- ingl. k. *wrapper class*
- klass, mille põhiülesandeks on seostada mingi objekti või väärtusega täiendavaid meetodeid
- on olemas algtüüpide jaoks
  - `Character`, `Boolean`, `Byte`, `Short`, `Integer`, `Long`, `Float`, `Double`

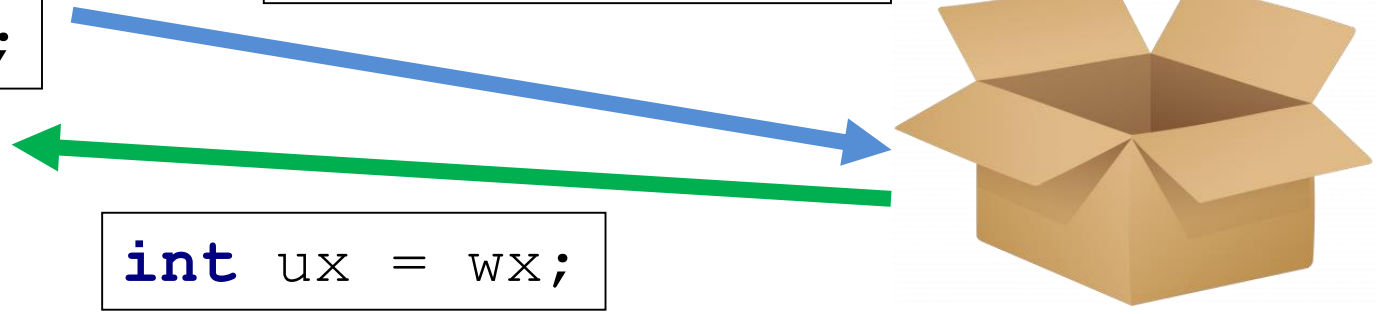
Mähisklassid

- Alates Java 5 versioonist tehakse paljud teisendused mähisklassi ning algtüübi vahel automaatselt

```
Integer wx2 = Integer.valueOf(x);
```

```
Integer wx = x;
```

```
int x = 100;
```



```
int ux = wx;
```

```
int ux2 = wx2.intValue();
```

- *Miks üldse algtüübid? Kõik objektidena!!!*
  - Kui on vaja efektiivsust

Massiiv

- int[] arvud = new int[5];
- int x = arvud[4];
- arvud[4] = 35;
- arvud.length

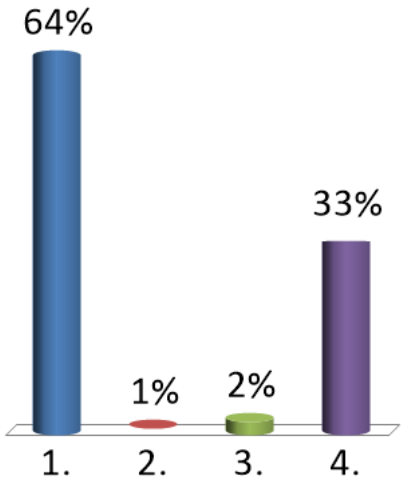
List

- ArrayList<Integer> arvud = new ArrayList<>();
- int x = arvud.get(4);
- arvud.set(4,35);
- arvud.size()
- arvud.add(35);
- arvud.remove(4);

Mida väljastab järgmine programmilõik?

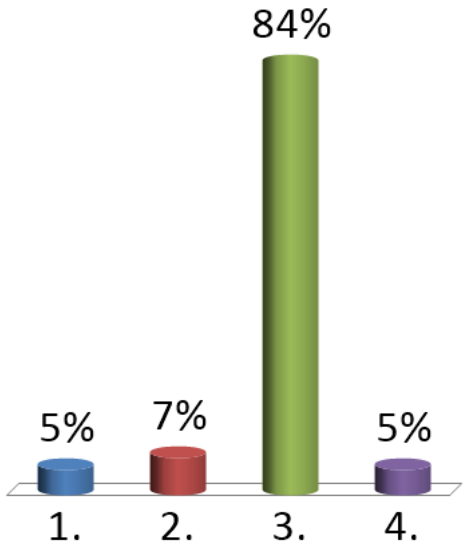
```
ArrayList<String> nimed;
nimed.add("Ülle");
System.out.println(nimed.get(0));
```

1. Ülle
2. 0
3. midagi muud
4. ✓ veateate



Milline on õige variant listi loomiseks?

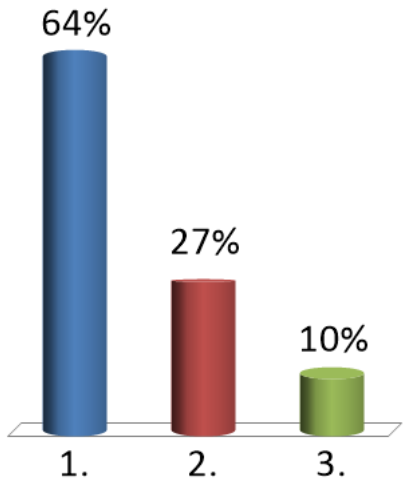
1. `ArrayList(Integer) list = new ArrayList();`
2. `ArrayList[Integer] list = new ArrayList[]();`
3. ✓ `ArrayList<Integer> list = new ArrayList<>();`
4. `ArrayList<int> list = new ArrayList<>();`



Mida väljastab järgmine programmilõik?

```
ArrayList<Integer> arvud = new ArrayList<>(5);
System.out.println(arvud.get(0));
```

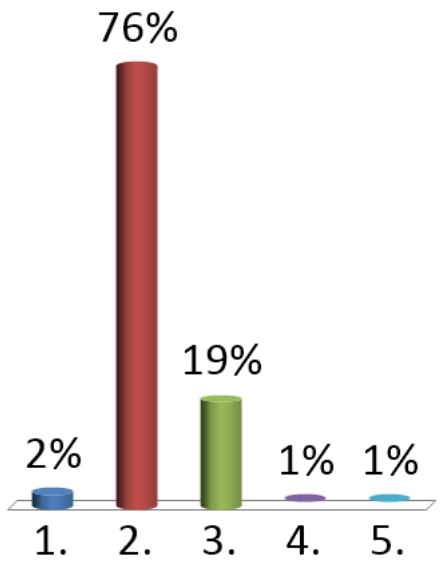
1. 0
2. midagi muud
3. ✓ veateate



Mida väljastab järgmine programmilõik?

```
ArrayList<Integer> list = new ArrayList<>();
list.add(2);
list.add(2);
for (int elem: list){
    System.out.print(elem + " ");
}
```

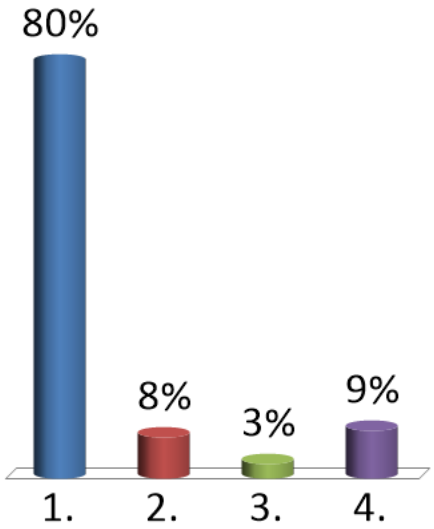
1. 22
2. ✓ 2 2
3. 2
   2
4. midagi muud
5. veateate



Mida väljastab järgmine programmilõik?

```
ArrayList<Integer> list = new ArrayList<>();
list.add(2);
list.add(2);
for (int elem: list){
    elem = 1;
    System.out.print(elem + " ");
}
```

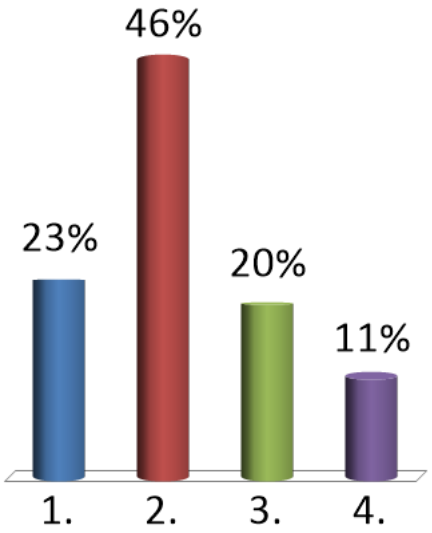
1. 1 1 ✓
2. 2 2
3. midagi muud
4. veateate



Mida väljastab järgmine programmilõik?

```
ArrayList<Integer> list = new ArrayList<>();
list.add(2);
list.add(2);
for (int elem: list){
    elem = 1;
}
System.out.println(list);
```

1. [1, 1]
2. ✓ [2, 2]
3. midagi muud
4. veateate



Liidesed (*interfaces*)

- Liides on klassitaoline konstruktsioon
  - nagu klass, kus ei ole realisatsiooni
- Liides kirjeldab komplekti meetodeid, kus puuduvad meetodite sisud
  - Liides võib sisaldada ka konstante, vaike- ja privaatseid meetodeid
- Liides võimaldab määrata, mida klass peab tegema, jättes täpsustamata, kuidas seda teha

```
piiritlejad interface LiideseNimi {
...
}
```

- Liidese realiseerimiseks klassis kasutatakse võtmesõna **`implements`**
  - Sellega nagu antakse lubadus, et kõik realiseeritakse

Liides

- Liidese realiseerimisel peab klass realiseerima kõik liideses deklareeritud meetodid
  - vastasel juhul tuleb klass kuulutada abstraktseks

- Liideses deklareeritud
  - meetodid on piiritlejatega **`public abstract`**
  - konstandid piiritlejatega **`public static final`**

  aga seda ei pea eraldi märkima

Kaubanduskeskuse näide

```
public interface Söödav {
    public String kuidasSüüa();
}
```

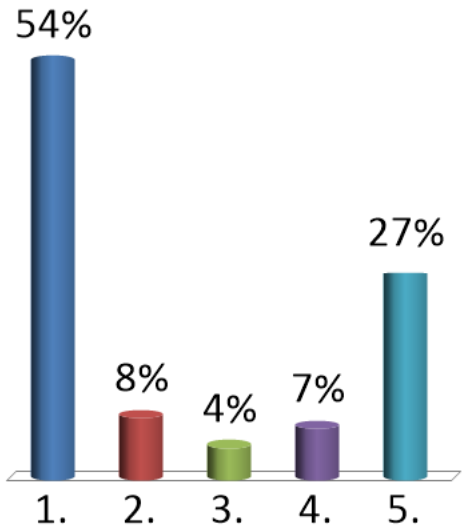
```
public class Riis implements Söödav {
    public String kuidasSüüa(){
        return "keedetult";
    }
}
```

```
public class Kartul implements Söödav {
    public String kuidasSüüa() {
        return "keedetult või praetult";
    }
}
```

Mida väljastab järgmine programmilõik?

```
public class SöögiTest {
    public static void main(String[] args) {
        System.out.println(new Riis().kuidasSüüa());
    }
}
```

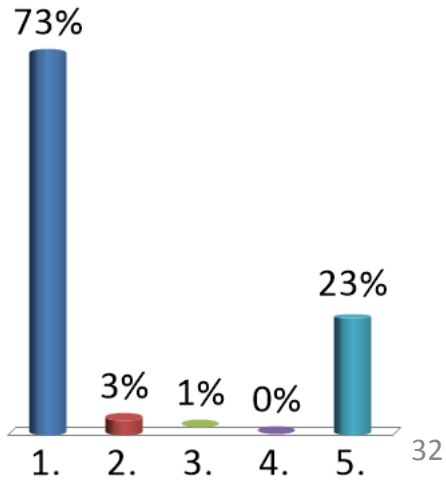
✔ 1. keedetult
2. keedetult või praetult
3. mitte midagi
4. midagi muud
5. veateate



Mida väljastab järgmine programmilõik?

```
public class SöögiTest {
    public static void main(String[] args) {
        Söödav t1 = new Riis();
        System.out.println(t1.kuidasSüüa());
    }
}
```

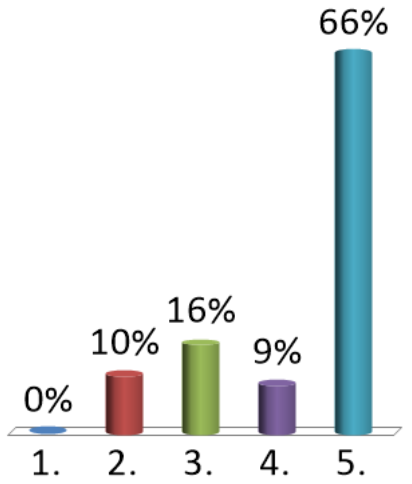
✓ 1. keedetult
2. keedetult või praetult
3. mitte midagi
4. midagi muud
5. veateate



Mida väljastab järgmine programmilõik?

```
public class SöögiTest {
    public static void main(String[] args) {
        Söödav t2 = new Söödav();
        System.out.println(t2.kuidasSüüa());
    }
}
```

1. keedetult
2. keedetult või praetult
3. mitte midagi
4. midagi muud
5. ✓ veateate



Liides `java.lang.Comparable`

Module java.base
Package java.lang

Interface Comparable<T>

Type Parameters:

T - the type of objects that this object may be compared to

All Known Subinterfaces:

AnnotationTypeDoc, AnnotationTypeElementDoc, ArrayType, ByteValue, CharValue, C
ExecutableMemberDoc, Field, FieldDoc, FloatValue, IntegerValue, InterfaceType, I
RunnableScheduledFuture<V>, ScheduledFuture<V>, SerialFieldTag, ShortValue

All Known Implementing Classes:

AbstractChronology, AbstractRegionPainter.PaintContext.CacheMode, AccessMode
BigDecimal, BigInteger, Boolean, Byte, ByteBuffer, Calendar, CardTerminals.State
ClientInfoStatus, CollationKey, Collector.Characteristics, Component.Baseline
Diagnostic.Kind, Diagnostic.Kind, Dialog.ModalExclusionType, Dialog.ModalityT
Duration, ElementKind, Elements.Origin, ElementType, Enum, File, FileTime, FileV
GraphicsDevice.WindowTranslucency, GregorianCalendar, GroupLayout.Alignment,
IsoEra, JapaneseChronology, JapaneseDate, JavaFileObject.Kind, JConsoleContext
LinkOption, LocalDate, LocalDateTime, Locale.Category, Locale.FilteringMode, Lo

```
public class KaksArvu {
    private int a;
    private int b;

    public KaksArvu(int a, int b) {
        this.a = a;
        this.b = b;
    }

    int summa(){
        return a + b;
    }

    public String toString() {
        return "KaksArvu [a=" + a + ", b=" + b +
                        ", summa()=" + summa() + "]";
    }
}
```

```
public class TestKaksArvu {
    public static void main(String[] args) {
        KaksArvu ka1 = new KaksArvu(4, 3);
        KaksArvu ka2 = new KaksArvu(5, 0);
        KaksArvu ka3 = new KaksArvu(2, 4);
        KaksArvu[] kad = {ka1, ka2, ka3};
        for (KaksArvu ka : kad)
            System.out.println(ka);
    }
}
```

```
KaksArvu [a=4, b=3, summa()=7]
KaksArvu [a=5, b=0, summa()=5]
KaksArvu [a=2, b=4, summa()=6]
```

```
public class KaksArvu
                    implements Comparable<KaksArvu>{
```

```
@Override
public int compareTo(KaksArvu o) {
    if (this.summa() > o.summa()) {
        return 1;
    }
    if (this.summa() < o.summa()) {
        return -1;
    }
    return 0;
}
```

```
java.util.Arrays.sort(kad);
for (KaksArvu ka : kad)
    System.out.println(ka);
```

```
KaksArvu [a=5, b=0, summa()=5]
KaksArvu [a=2, b=4, summa()=6]
KaksArvu [a=4, b=3, summa()=7]
```

```
public int compareTo(KaksArvu o) {
 return this.summa() - o.summa();
}
```

```
public int compareTo(KaksArvu o) {
 return Integer.compare(this.summa(),o.summa());
}
```

```
public int compareTo(KaksArvu o) {
   return Integer.valueOf(this.summa()).
          compareTo(
          Integer.valueOf(o.summa()));
}
```

```
List<KaksArvu> l = new ArrayList<>();
l.addAll(Arrays.asList(ka1,ka2,ka3));
System.out.println(l);
java.util.Collections.sort(l);
System.out.println(l);
```

```
[KaksArvu [a=4, b=3, summa()=7], KaksArvu [a=5, b=0, summa()=5],
KaksArvu [a=2, b=4, summa()=6]]
[KaksArvu [a=5, b=0, summa()=5], KaksArvu [a=2, b=4, summa()=6],
KaksArvu [a=4, b=3, summa()=7]]
```

```
public class Inimene {
    private String nimi;
    private int pikkus;
    public Inimene(String nimi, int pikkus) {
        this.nimi = nimi;
        this.pikkus = pikkus;
    }
    public String toString() {
        return "Inimene [nimi=" + nimi +
                ", pikkus=" + pikkus + "]";
    }
}
```

```
public class TestInimene {
    public static void main(String[] args) {
        Inimene i1 = new Inimene("Eva",160);
        Inimene i2 = new Inimene("Ülle",170);
        Inimene i3 = new Inimene("Ada",165);
        Inimene[] id = {i1, i2, i3};
        for (Inimene i: id)
            System.out.println(i);
    }
}
```

```
Inimene [nimi=Eva, pikkus=160]
Inimene [nimi=Ülle, pikkus=170]
Inimene [nimi=Ada, pikkus=165]
```

```
public class Inimene
                implements Comparable<Inimene>{
```

```
sõne1.compareTo(sõne2);
```

public final class **String**
extends Object
implements Serializable, Comparable<String>,

int **compareTo**(**String** anotherString)
Compares two strings lexicographically.

```
public int compareTo(Inimene i){
    return this.nimi.compareTo(i.nimi);
}
```

```
java.util.Arrays.sort(id);
for (Inimene i: id)
    System.out.println(i);
```

Inimene [nimi=Ada, pikkus=165]
Inimene [nimi=Eva, pikkus=160]
Inimene [nimi=Ülle, pikkus=170]

Liides *List*

Module java.base
Package java.util

Interface List<E>

Type Parameters:

E - the type of elements in this list

All SuperInterfaces:

`Collection<E>`, `Iterable<E>`

All Known Implementing Classes:

`AbstractList`, `AbstractSequentialList`, `ArrayList`, `AttributeList`,

```
public interface List<E>
extends Collection<E>
```

An ordered collection (also known as a *sequence*). The user of this inte

Veel liideseid

Module java.base
Package java.lang

Interface Iterable<T>

Type Parameters:

T - the type of elements returned by the iterator

All Known Subinterfaces:

BeanContext, BeanContextServices, BlockingDeque<E>, Bloc
ObservableSetValue<E>, Path, Queue<E>, SecureDirectorySt

All Known Implementing Classes:

AbstractCollection, AbstractList, Al
ConcurrentHashMap.KeySetView, Concu
LinkedBlockingDeque, LinkedBlockin
PriorityQueue, ReadOnlyListPropert
SetBinding, SetExpression, SetPrope
SQLIntegrityConstraintViolationExc
SQLTransactionRollbackException, S

Module javafx.base
Package javafx.event

Interface EventHandler<T extends Event>

Type Parameters:

T - the event class this handler can handle

All Superinterfaces:

EventListener

All Known Implementing Classes:

WeakEventHandler

Veel liidestest

- Üks klass võib realiseerida mitu liidest, kusjuures liidesed võivad sisaldada sama meetodit

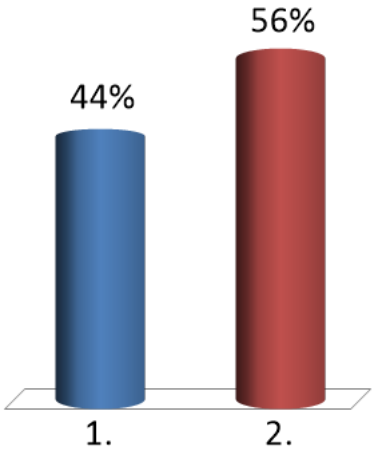
```
implements Liides1, Liides2
```

- Liidesel võib olla alamliideseid, oma hierarhia

Kas on lubatud?

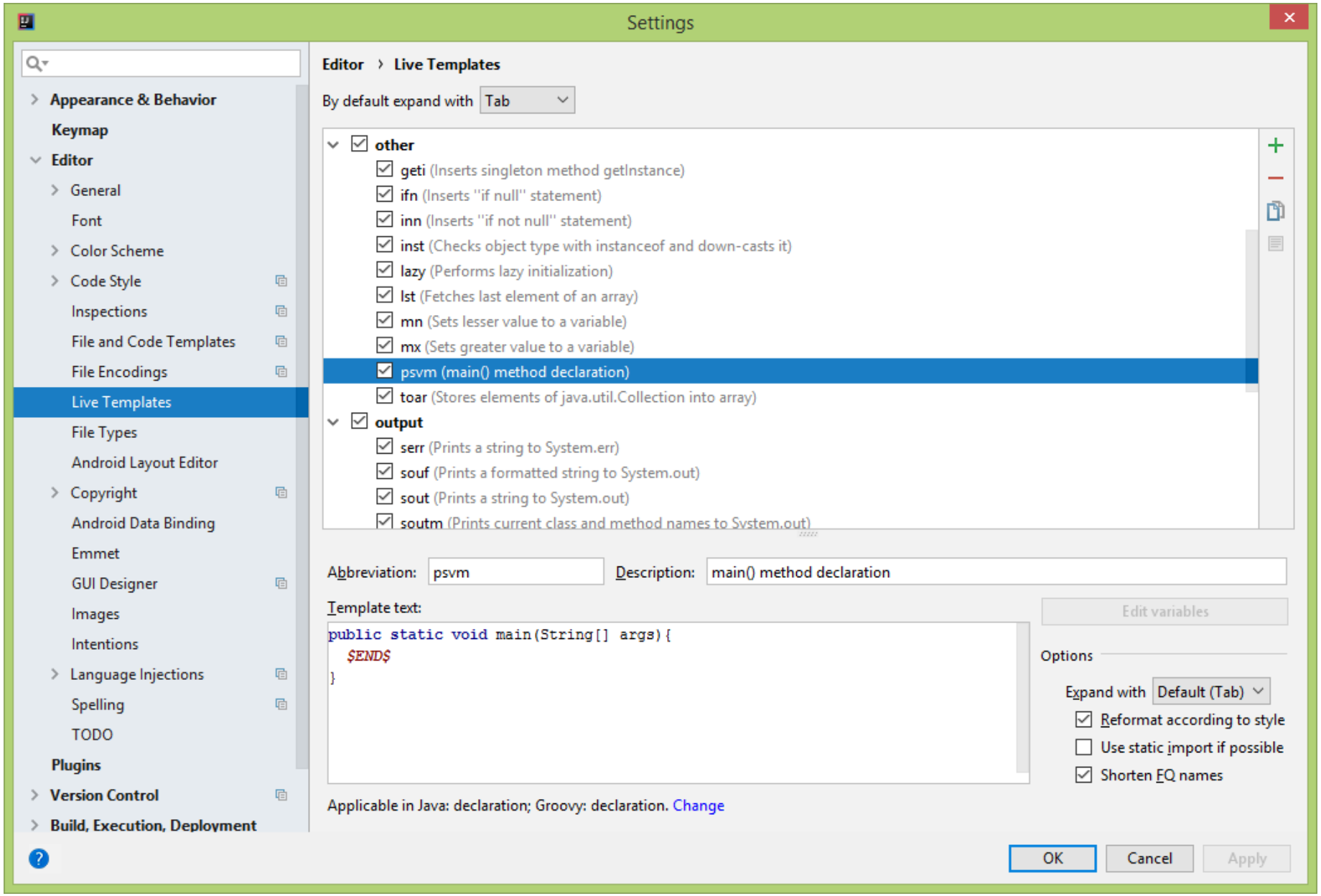
```
public interface Nuputiri {
   String laul = "mina olen nupp ja sina oled nupp";
   public void nuputa(){
      System.out.println(laul);
   }
}
```

1. Jah
2. ✓ Ei



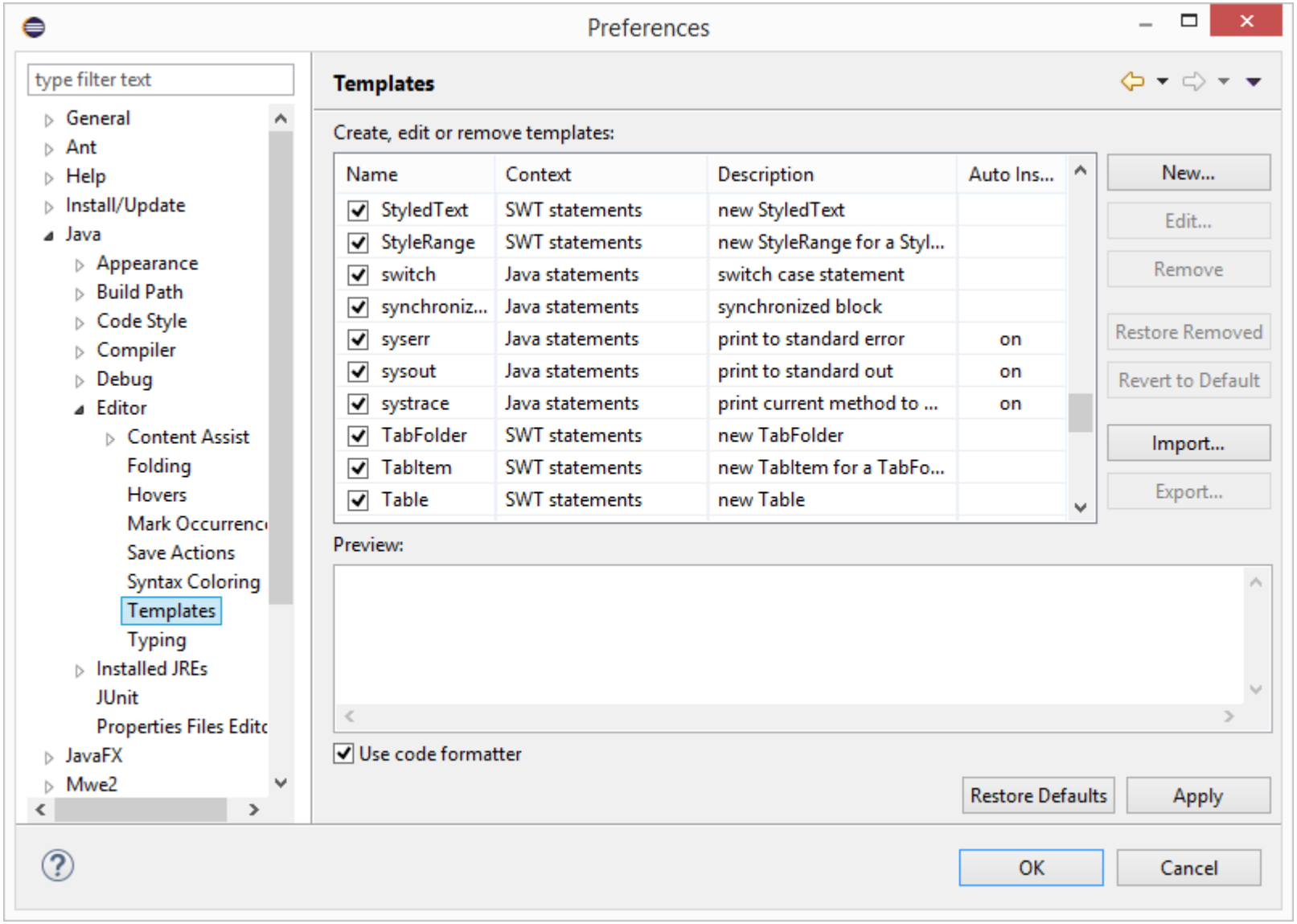
IntelliJ

- Palju võimalusi, vaatame mõningaid
  - Treppimine
    - Ctrl + Alt + I
  - Importimine
    - Alt + Enter
  - Konstruktorite, get-, set-meetodite genereerimine
  - Lühemalt, mallid
    - File → Settings → Editor → Live Templates


Settings
Editor › Live Templates
By default expand with Tab
other
geti (Inserts singleton method getInstance)
ifn (Inserts "if null" statement)
inn (Inserts "if not null" statement)
inst (Checks object type with instanceof and down-casts it)
lazy (Performs lazy initialization)
lst (Fetches last element of an array)
mn (Sets lesser value to a variable)
mx (Sets greater value to a variable)
psvm (main() method declaration)
toar (Stores elements of java.util.Collection into array)
output
serr (Prints a string to System.err)
souf (Prints a formatted string to System.out)
sout (Prints a string to System.out)
soutm (Prints current class and method names to System.out)
Abbreviation: psvm
Description: main() method declaration
Template text:
public static void main(String[] args){
END
}
Edit variables
Options
Expand with Default (Tab)
Reformat according to style
Use static import if possible
Shorten FQ names
Applicable in Java: declaration; Groovy: declaration. Change
Appearance & Behavior
Keymap
Editor
General
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Android Layout Editor
Copyright
Android Data Binding
Emmet
GUI Designer
Images
Intentions
Language Injections
Spelling
TODO
Plugins
Version Control
Build, Execution, Deployment
OK
Cancel
Apply

Eclipse

- Palju võimalusi, vaatame mõningaid
  - Treppimine
    - Ctrl + I
  - Importimine
    - Ctrl + Shift + O
  - Konstruktorite, get-, set-meetodite genereerimine
  - Lühemalt, mallid
    - Window → Preferences → Java →Editor → Templates


Preferences
type filter text
General
Ant
Help
Install/Update
Java
Appearance
Build Path
Code Style
Compiler
Debug
Editor
Content Assist
Folding
Hovers
Mark Occurrenc
Save Actions
Syntax Coloring
Templates
Typing
Installed JREs
JUnit
Properties Files Editc
JavaFX
Mwe2
Templates
Create, edit or remove templates:
Name
Context
Description
Auto Ins...
StyledText
SWT statements
new StyledText
StyleRange
SWT statements
new StyleRange for a Styl...
switch
Java statements
switch case statement
synchroniz...
Java statements
synchronized block
syserr
Java statements
print to standard error
on
sysout
Java statements
print to standard out
on
systrace
Java statements
print current method to ...
on
TabFolder
SWT statements
new TabFolder
TabItem
SWT statements
new TabItem for a TabFo...
Table
SWT statements
new Table
New...
Edit...
Remove
Restore Removed
Revert to Default
Import...
Export...
Preview:
Use code formatter
Restore Defaults
Apply
OK
Cancel

Privaatsed isendiväljad ja get- ja set-meetodid

this.nimi = nimi

- On kaks ühesuguse nimega suurust

```
public class Kool {
    int õpilasteArv; //opilasteArv
    String nimi;
    public Kool(int õpilasteArv, String nimi){
        this.õpilasteArv = õpilasteArv;
        this.nimi = nimi;
    }
}
```

Klassiväli, isendiväli

- Klassiväli **`static int a`**
- Isendiväli **`int b`**

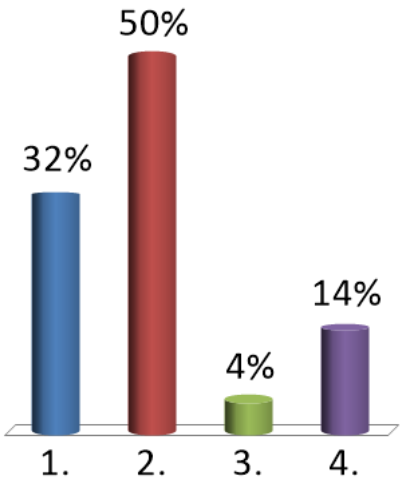
```
int õpilasteArv;
static int õpilasteArv;
```

Mida väljastab järgmine programmilõik?

```
static int õpilasteArv;
```

```
Kool kool1 = new Kool(100, "Morna Gümnaasium");
Kool kool2 = new Kool(90, "Simuvere Gümnaasium");
System.out.println(kool1.õpilasteArv);
```

1. 100
2. ✓ 90
3. midagi muud
4. veateate

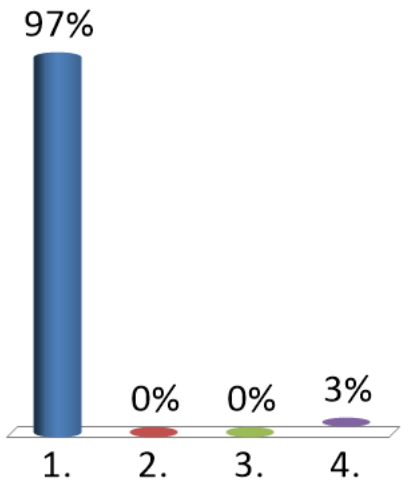


Mida väljastab järgmine programmilõik?

```
int õpilasteArv;
```

```
Kool kool1 = new Kool(100, "Morna Gümnaasium");
Kool kool2 = new Kool(90, "Simuvere Gümnaasium");
System.out.println(kool1.õpilasteArv);
```

✔ 1. 100
2. 90
3. midagi muud
4. veateate



Klassimeetod, isendimeetod

- Klassimeetod **`static int meetoda`**
- Isendimeetod **`int meetodb`**

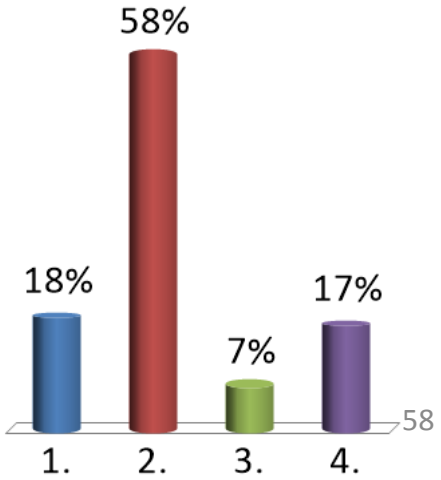
```
void lisaÕpilasi(int arv){
    õpilasteArv += arv;
}

static void lisaÕpilasi(int arv){
    õpilasteArv += arv;
}
```

Mida väljastab järgmine programmilõik?

```
int õpilasteArv;
...
static void lisaÕpilasi(int arv){
   õpilasteArv += arv;
}
```

```
Kool kool1 = new Kool(100, "Morna Gümnaasium");
Kool kool2 = new Kool(90, "Simuvere Gümnaasium");
kool1.lisaÕpilasi(20);
System.out.println(kool1.õpilasteArv);
```

1. 100
2. 120
3. midagi muud
4. ✓ veateate

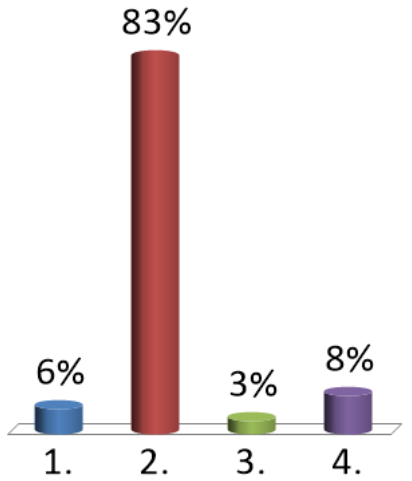


Mida väljastab järgmine programmilõik?

```
int õpilasteArv;
...
void lisaÕpilasi(int arv){
   õpilasteArv += arv;
}
```

```
Kool kool1 = new Kool(100, "Morna Gümnaasium");
Kool kool2 = new Kool(90, "Simuvere Gümnaasium");
kool1.lisaÕpilasi(20);
System.out.println(kool1.õpilasteArv);
```

1. 100
2. ✓ 120
3. midagi muud
4. veateate



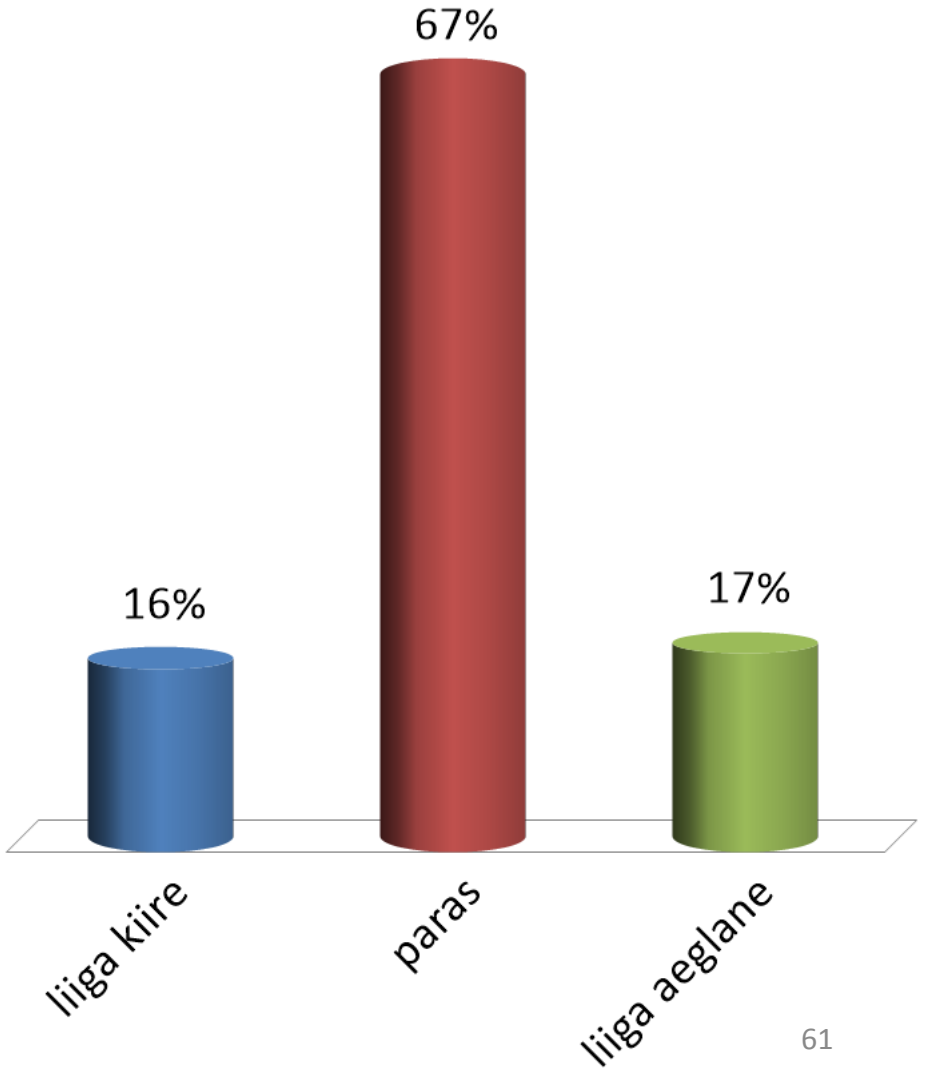
`==` või `equals()`

- `==`
  - algtüübi puhul võrdleb väärtusi
  - viittüübi puhul võrdleb viitasid
    - kas on sama isend?
- `equals()`
  - viittüübi jaoks
  - meetod klassis `Object`
    - võrdleb viitasid
  - pigem ikkagi sisu võrdlemiseks
    - üle katta

  Vt. ka `hashCode()`

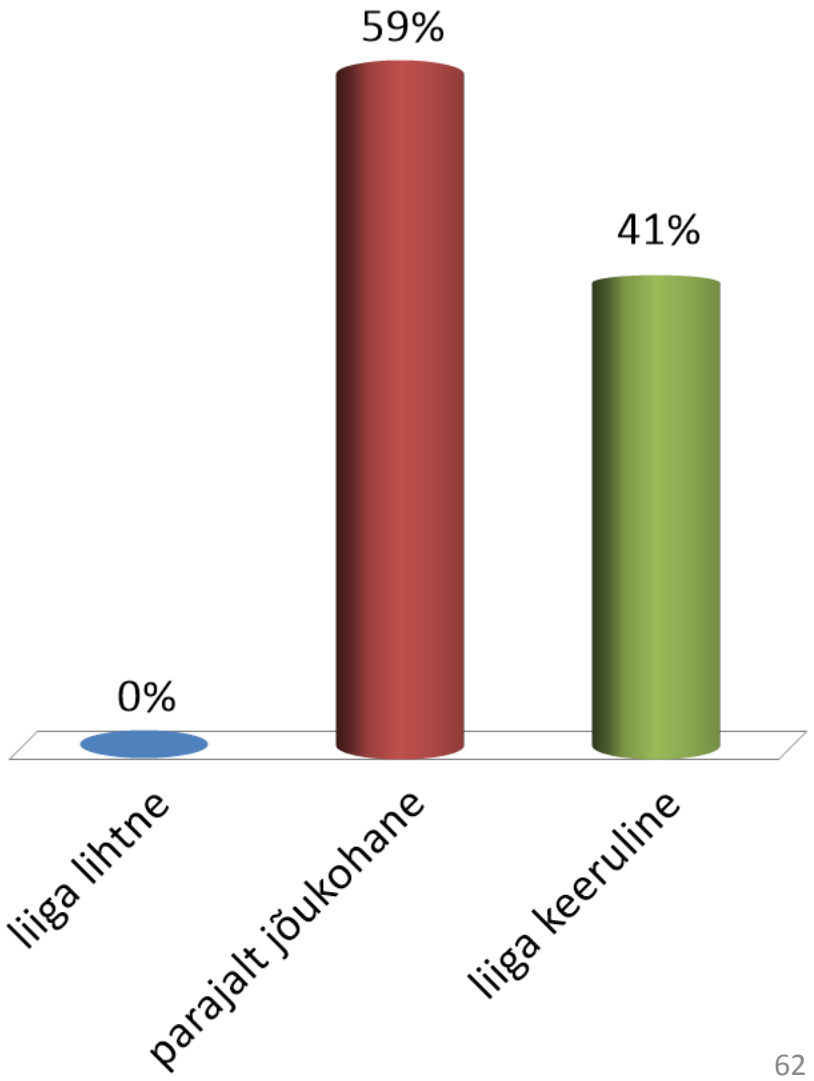
Loengu tempo oli

1. liiga kiire
2. paras
3. liiga aeglane



Materjal tundus

1. liiga lihtne
2. parajalt jõukohane
3. liiga keeruline



Suur tänu osalemast ja kohtumiseni!