Objektorienteeritud programmeerimine

25. veebruar, 3. loeng

Eno Tõnisson

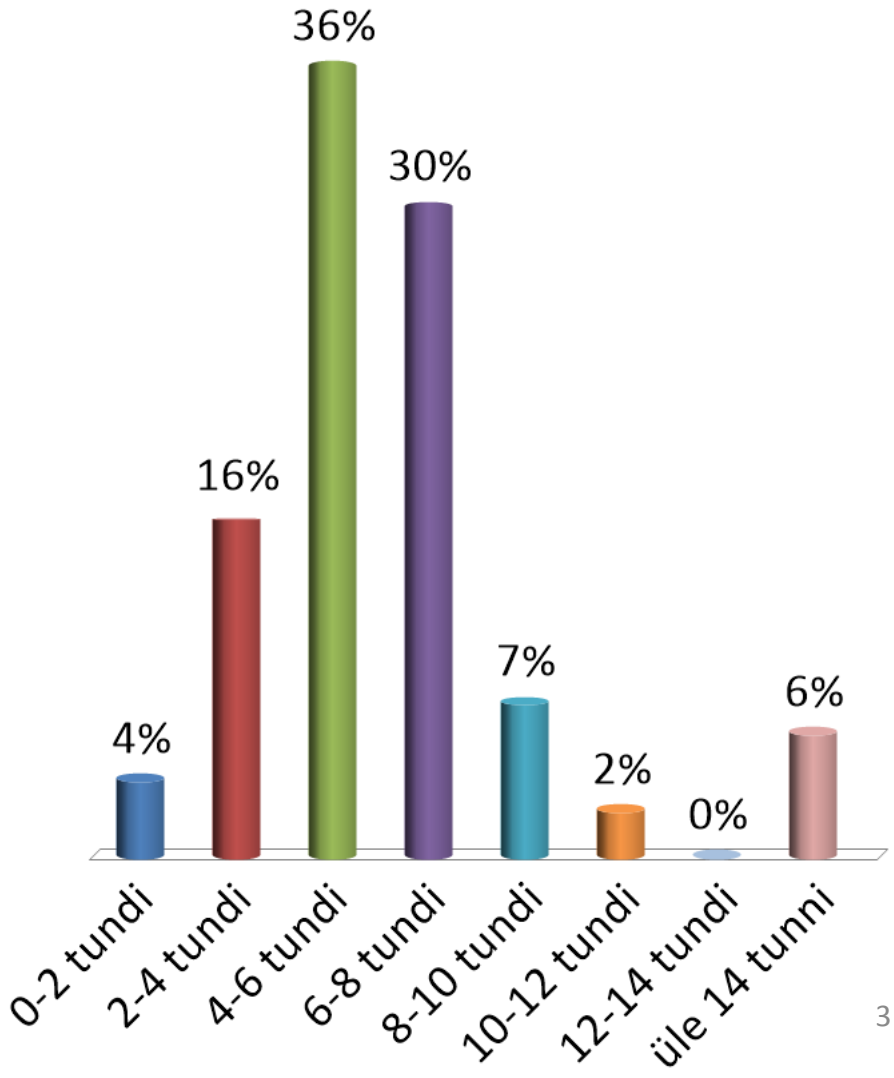
Möödunud nädalal

- Loeng
  - Tingimuslaused. Tsüklid. OOP algus
- Lisapraktikum (konsultatsioon)
- Praktikum
  - Java põhikonstruktsioonid

- Eesti Vabariigi aastapäev

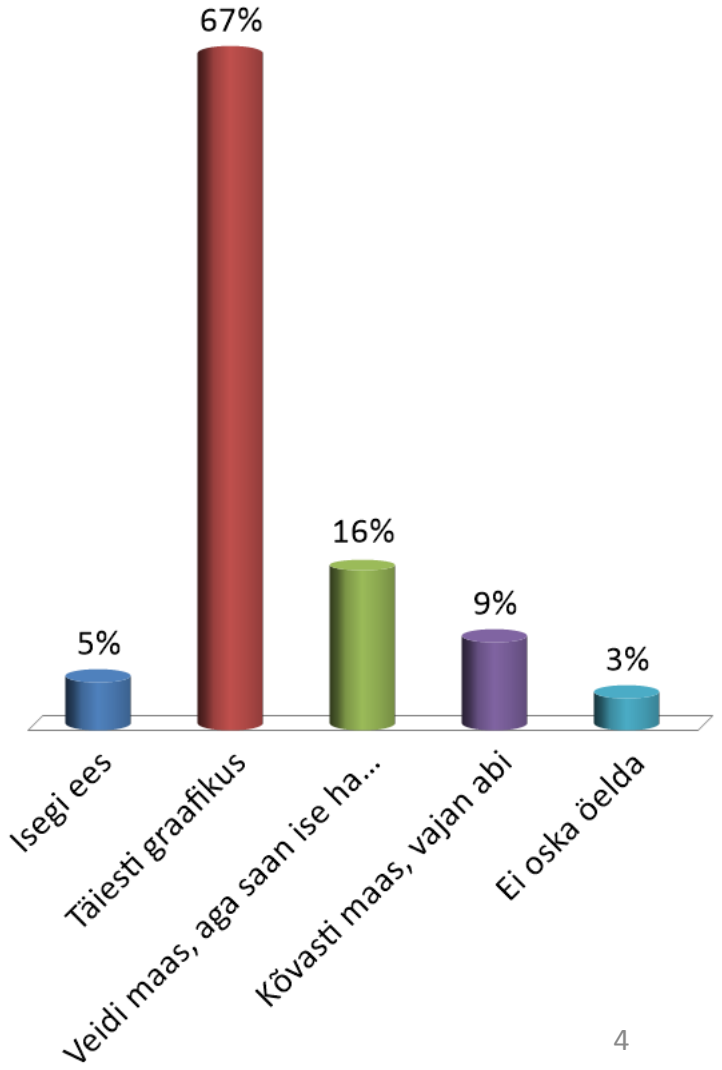
Umbes mitu tundi tegelesite eelmisel nädalal selle ainega (loeng+praktikum+iseseisvalt)?

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

- Isendid, klassid, konstruktorid
- Sõned
  - `String`, `StringBuilder`
- Mähisklassid
  - `char`, `Character`

Klass

Isendiväljad
(Isendimuutujad)

Isendimeetodid

```
public class Kast {
    String nimi;
    double pikkus;
    double laius;
    double kõrgus;

    double ruumala(){
        return pikkus*laius*kõrgus;
    }

    void tervitus(){
        System.out.println("Tere");
    }
}
```

Milleks on vajalikud klassid ja kus neid päriselt tarvis läheb?

- Klass defineerib ära objekti abstraktsed omadused
- Klass on nagu šabloon, mis kirjeldab millegi olemust

```
Kast kast1 = new Kast("Paul",2,3,4);
Kast kast2 = new Kast("Ülo",3,5,3.4);
```

```
String kast1nimi = "Paul";
double kast1pikkus = 2;
double kast1laius = 3;
double kast1kõrgus = 4;
String kast2nimi = "Ülo";
double kast2pikkus = 3;
double kast2laius = 5;
double kast2kõrgus = 3.4;
```

Konstruktor (ingl. *constructor*)

- Konstruktoril väga selge rakendus
  - uue isendi loomisel, **`new`**
    - klassi kehas kirjeldatud eriline protseduur, mida rakendatakse isendiloome käigus (nt. vastloodud isendi väljade algväärtustamiseks)
      - J. Kiho *Väike Java leksikon*

- Nimi langeb kokku klassi nimega
- Sarnane meetodiga, kuid ei oma tagastustüüpi
- Võimalik üledefineerimine
- Kui klassis ei ole konstruktorit defineeritud, siis lisatakse vaikekonstruktor (parameetriteta)
- See, millist konstruktori versiooni kasutama hakatakse, sõltub argumentide arvust ja/või tüübist

Konstruktor

```
public class Kast {
    String nimi;
    double pikkus;
    double laius;
    double kõrgus;

    public Kast(String nimi, double pikkus,
                double laius, double kõrgus) {
        this.nimi = nimi;
        this.pikkus = pikkus;
        this.laius = laius;
        this.kõrgus = kõrgus;
    }
```

Isendiväljad
(Isendimuutujad)

Konstruktor

Meetodid

Mitu konstruktorit

```
public Kast(String nimi, double pikkus, double laius,
double kõrgus) {
    this.nimi = nimi;
    this.pikkus = pikkus;
    this.laius = laius;
    this.kõrgus = kõrgus;
}

public Kast(String antudNimi) {
    nimi = antudNimi;
    pikkus = 1;
    laius = 1;
    kõrgus = 1;
}

public Kast(){
    this("", 0.0, 0.0, 0.0);
}
```

Võtmesõna `this`

- Viitamine objektile endale
  - isendiväljadele viitamisel, kui parameetrite nimed langevad kokku isendiväljade nimedega
  - ühe konstruktori sees teise konstruktori väljakutsumisel

Mitu konstruktorit

```
public Kast(String nimi, double pikkus, double laius,
double kõrgus) {
    this.nimi = nimi;
    this.pikkus = pikkus;
    this.laius = laius;
    this.kõrgus = kõrgus;
}

public Kast(String antudNimi) {
    nimi = antudNimi;
    pikkus = 1;
    laius =
    kõrgus =
}

public Kast(){
    this("", 0.0, 0.0, 0.0);
}
```

```
Kast kast2 = new Kast("Paul", 4.8, 2, 3);
Kast kast3 = new Kast("Ülo");
Kast kast5 = new Kast();
```

Mõisteid

- Üledefineerimine (ingl. *overloading*)
  - olukord, kus klassi kuulub mitu sama nimega, kuid erineva signatuuriga meetodit (ka päriluse teel saadut) või mitu konstruktorit; väljakutse puhul rakendatakse neist väljakutses antud argumentide poolest sobivat.
- Signatuur (ingl. *signature*)
  - meetodi või konstruktori iseloomustus, mis koosneb (meetodi või konstruktori) nimest ning formaalsete parameetrite tüüpide loetelust.

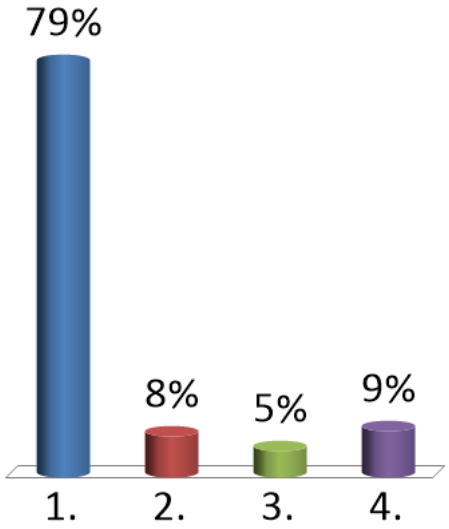
  - J. Kiho *Väike Java leksikon*

Mis väljastatakse ekraanile?

```
public class Paar {
    int i;
    double d;
    Paar(int i, double d) {
        this.i = i * 10;
        this.d = d;
    }
}
```

```
Paar p1 = new Paar(2, 4.1);
System.out.println(p1.i);
```

1. ✔ 20
2. 20.0
3. midagi muud
4. veateade

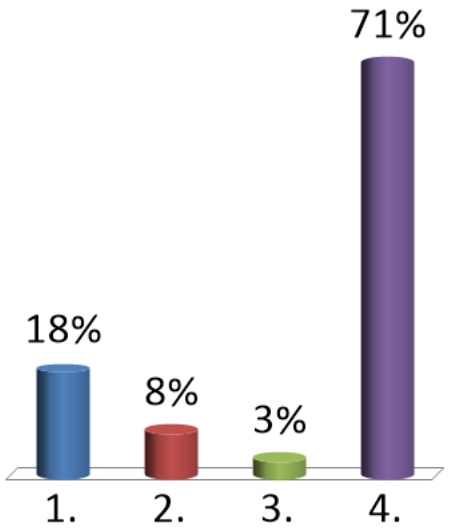


Mis väljastatakse ekraanile?

```
public class Paar {
    int i;
    double d;
    Paar(int i, double d) {
        this.i = i * 10;
        this.d = d;
    }
}
```

```
Paar p1 = new Paar(2.0, 4.1);
System.out.println(p1.i);
```

1. 20
2. 20.0
3. midagi muud
4. ✓ veateade

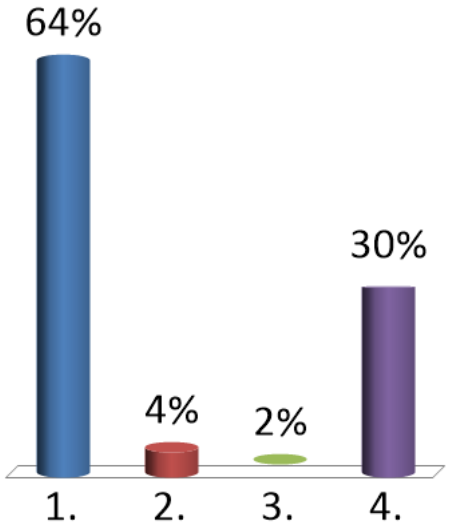


Mis väljastatakse ekraanile?

```
public class Paar {
    int i;
    double d;
    Paar(int i, double d) {
        this.i = i * 10;
        this.d = d;
    }
}
```

```
Paar p1 = new Paar(2, 4);
System.out.println(p1.i);
```

1. ✓ 20
2. 20.0
3. midagi muud
4. veateade

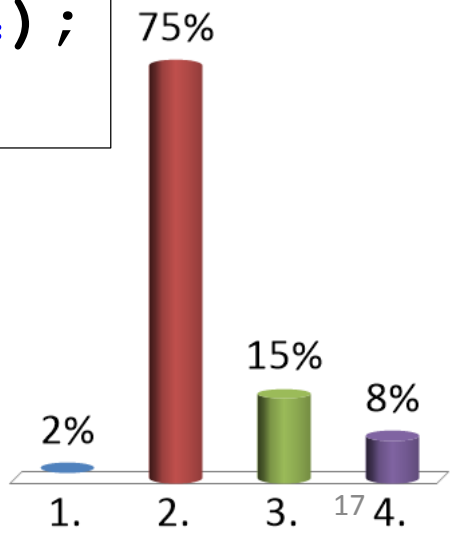


Mis väljastatakse ekraanile?

```
public class Paar {
    int i;
    double d;
    Paar(int i, double d) {
        this.i = i * 10;
        this.d = d;
    }
    Paar(double d, int i) {
        this.i = i;
        this.d = d * 10;
    }
}
```

```
Paar p1 = new Paar(2.0, 4);
System.out.println(p1.d);
```

1. 20
2. ✓ 20.0
3. midagi muud
4. veateade

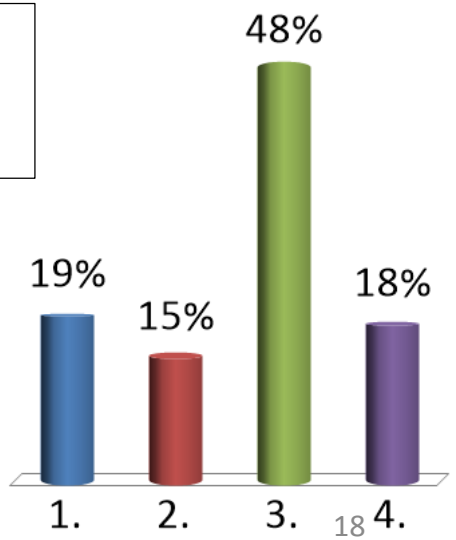


Mis väljastatakse ekraanile?

```
public class Paar {
    int i;
    double d;
    Paar(int i, double d) {
        this.i = i * 10;
        this.d = d;
    }
    Paar(double d, int i) {
        this.i = i;
        this.d = d * 10;
    }
}
```

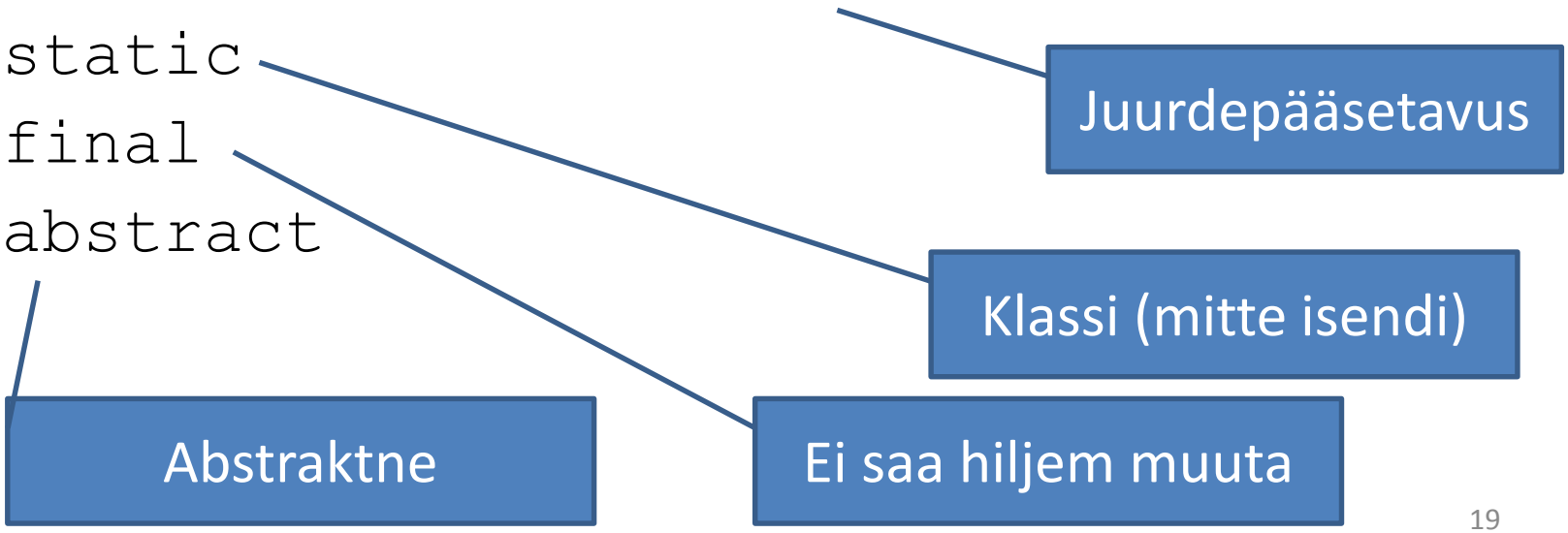
```
Paar p1 = new Paar(2, 4);
System.out.println(p1.d);
```

1. 20
2. 20.0
3. midagi muud
4. ✓ veateade



Piiritleja (ingl. *modifier*)

- kasutusliiki täpsustav kirjelduse alguses paiknev võtmesõna
- järjestikused eraldatakse tühiku(te)ga
- omavaheline järjestus ei oma tähtsust
- Näiteid
  - `public, protected, private`
  - `static`
  - `final`
  - `abstract`



Juurdepääsetavus

- Enamik objektorienteeritud keeli toetab informatsiooni varjamist
- Väljad ja meetodid on jagatud avalikeks ja privaatseteks
- Privaatsed väljad ja meetodid on nähtavad (kättesaadavad) ainult klassi sees; väljastpoolt on nähtavad ainult avalikud väljad ja meetodid
  - Tavaline jaotus: väljad privaatsed ja meetodid avalikud
- Soodustab suurte programmide hallatavust, kuna objekti "kasutaja" ei pea teadma midagi selle sisemistest realisatsioonidetailidest

Juurdepääsetavuse piiritlejad

- `public`, `protected`, `private`, juurdepääsu piiritlejat pole
- Kui juurdepääsu piiritlejat ei kasutata, siis klassid, meetodid ja andmeväljad on kättesaadavad sama paketi kõikides klassides

Paketid (ingl. *package*)

- Kasutatakse klasside rühmitamiseks
- Iga klass kuulub paketti
- Nimekonfliktide vältimiseks
  - `java.util.List`
  - `java.awt.List`
- Võimalik hierarhiline struktuur
- Teistest pakettidest
  - `import java.util.Scanner;`
  - `import java.util.*;`

- Alates Java 9 iga pakett kuulub moodulisse

Samas paketis

```
package p1;

public class K1 {
    public int x;
    int y;
    private int z;

    public void m1(){
    }
    void m2(){
    }
    private void m3(){
    }
}
```

```
package p1;

public class K2 {
    public static void
     main(String[] args) {
        K1 k = new K1();
        int a = k.x;
        int b = k.y;
        //int c = k.z;
        k.m1();
        k.m2();
        //k.m3();
    }
}
```

Erinevates pakettides

```
package p1;

public class K1 {
    public int x;
    int y;
    private int z;

    public void m1(){
    }
    void m2(){
    }
    private void m3(){
    }
}
```

```
package p2;

import p1.*;

public class K3 {
    public static void
     main(String[] args) {
        K1 k = new K1();
        int a = k.x;
        //int b = k.y;
        //int c = k.z;
        k.m1();
        //k.m2();
        //k.m3();
    }
}
```

Get- ja set-meetodid

```
private double pikkus;

public double getPikkus() {
    return pikkus;
}

public void setPikkus(double pikkus) {
    this.pikkus = pikkus;
}
```

Näide isendiväljast, mida ei saa pärast isendi loomist muuta

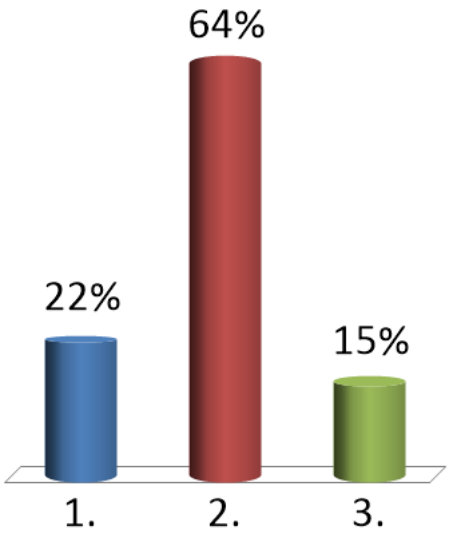
```
public class Isik {
    private long isikukood;

    public Isik(long isikukood) {
        this.isikukood = isikukood;
    }
    public long getIsikukood() {
        return isikukood;
    }
}
```

- Aitaks meetod `setIsikukood`

Kui juurdepääsetavust määravat piiritlejat pole, siis meetod on kättesaadav …

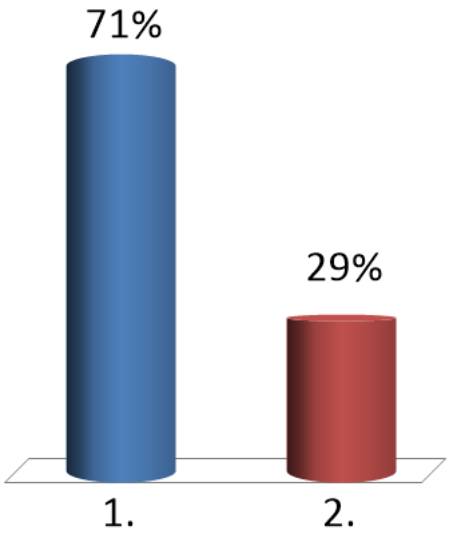
1. ainult samas klassis
2. ainult samas paketis ✓
3. kõikjal



Kas peameetodi päis võib olla `static public void main(String[] args)`

✓ 1. jah

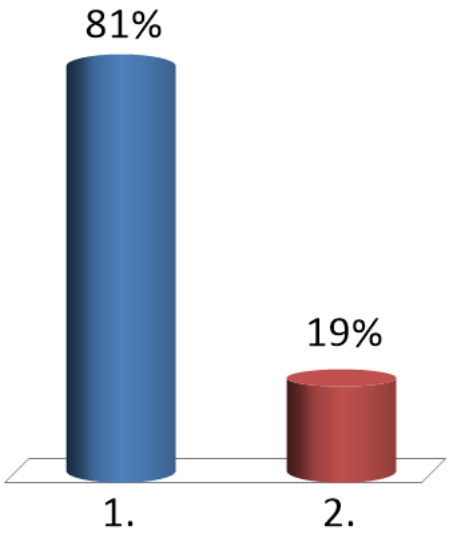
2. ei



Kas peameetodi päis võib olla
`public static void main(String[] a)`

✓ 1. jah

2. ei



Meetod `toString`

- tagastab sõne, mis peaks arusaadaval viisil esitama olulist informatsiooni objekti kohta
  - ilma meetodita `toString`

    Kast@50cbc42f
  - meetodiga `toString`

    Mina olen kast Paul: pikkus=2.0, laius=3.0, kõrgus=4.0

```
public String toString() {
     return "Mina olen kast " + nimi + ": pikkus=" + pikkus +
             ", laius=" + laius + ", kõrgus=" + kõrgus ;
}
```

Isend argumendina

- Klassis `Kast`

```
void tervitus(Kast teineKast) {
    System.out.println("Tere, " + teineKast.nimi);
}
```

- Klassis `KastTest`

```
kast1.tervitus(kast2);
```

Klassi- (staatiline) ja isendimeetod

`static` on või pole

- Klassimeetod

```
static double aritkeskmine(double arv1, double arv2)
```

  - väljakutse on võimalik kõikjalt, kus vastav klass on nähtav (isendeid ei pea olema olemas)
    - peameetod
      `public static void main (String[] args)`
    - meetodid klassist `java.lang.Math`

- Isendimeetod

```
void tervitus()
```

  - väljakutse on võimalik ainult mingi olemasoleva isendi kaudu
  - meetodi nimele lisandub isendi `osuti.` , klassisisesel kasutamisel on selleks vaikimisi `this.`

```
kast1.tervitus()
```

Klassi- (staatiline) ja isendiväli

`static` on või pole

- Klassiväli

  ```
  static int a
  ```

  - kasutamine on võimalik kõikjalt, kus vastav klass on nähtav
  - ei ole isendi osaks
  - klassist `java.lang.Math`
    - `static double E`, kasutamine `Math.E`
- Isendiväli

  ```
  int b
  ```

  - kuulub isendi struktuuri ja kasutamine on võimalik ainult vastava isendi kaudu
  - välja nimele lisandub isendi `osuti.` , klassisisesel kasutamisel on selleks vaikimisi `this.`

    `kast2.pikkus`

Mõlemaid

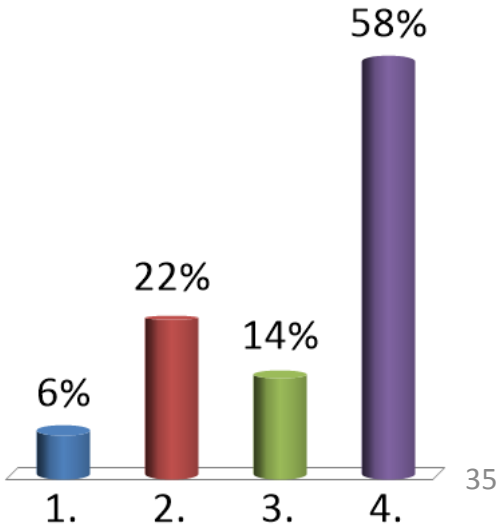
- Ühes klassis võib olla nii klassi- kui isendivälju ja -meetodeid

Mis väljastatakse ekraanile?

```
public class Paar {
    static int kordaja;
    int i;
    double d;
    Paar(int i, double d) {
        this.i = i * kordaja;
        this.d = d;
    }
}
```

```
Paar p = new Paar(2, 4.1);
System.out.println(p.i);
```

1. 2
2. ✓ 0
3. midagi muud
4. veateade



Vaikeväärtused

- Väljadel on vaikeväärtus
- Lokaalsetel muutujatel vaikeväärtust pole

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

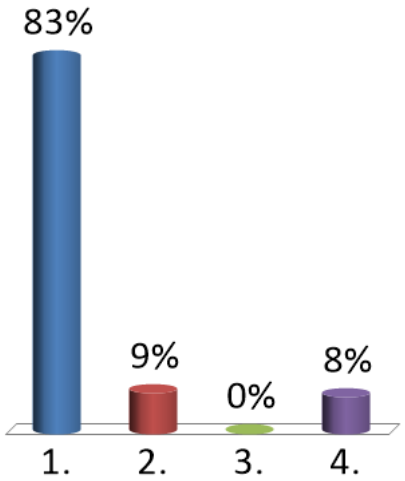
http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Mis väljastatakse ekraanile?

```
public class Paar {
    static int kordaja;
    int i;
    double d;
    Paar(int i, double d) {
        this.i = i * kordaja;
        this.d = d;
    }
}
```

```
Paar.kordaja = 10;
Paar p = new Paar(2, 4.1);
System.out.println(p.i);
```

1. ✓ 20
2. 0
3. midagi muud
4. veateade



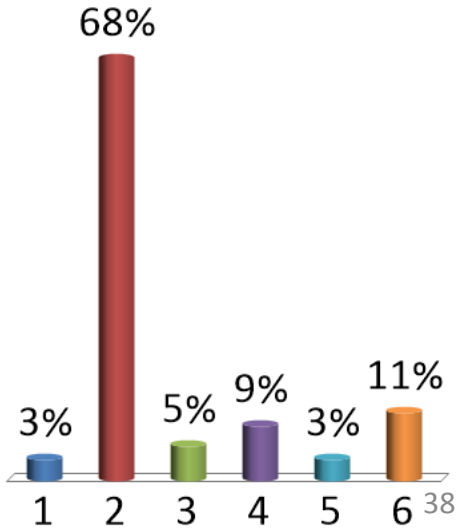
Mida väljastab järgmine programmilõik?

```
class Palk {
    static int tulumaks;
    static int brutopalk;
    int netopalk;
    String töötaja;

    public Palk(int brutopalk,
                String töötaja) {
        this.brutopalk = brutopalk;
        this.töötaja = töötaja;
        this.netopalk = brutopalk*
                (100-tulumaks)/100;
    }
}
```

```
Palk.tulumaks = 20;
Palk p1 = new Palk(1000,"Heli Kopter");
System.out.println(p1.tulumaks);
```

1. 1000
2. ✓ 20
3. 0
4. 800
5. midagi muud
6. veateate



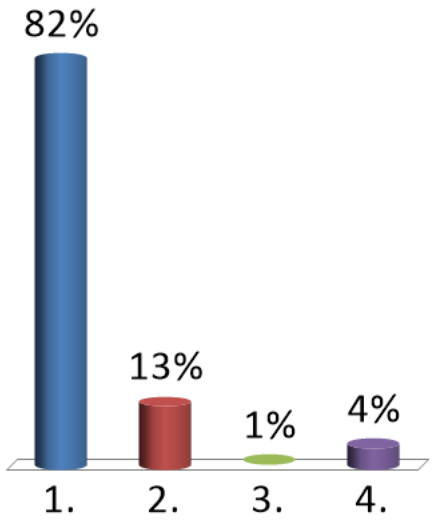
Mida väljastab järgmine programmilõik?

```
class Palk {
    static int tulumaks;
    static int brutopalk;
    int netopalk;
    String töötaja;

    public Palk(int brutopalk,
                    String töötaja) {
        this.brutopalk = brutopalk;
        this.töötaja = töötaja;
        this.netopalk = brutopalk*
                    (100-tulumaks)/100;
    }
}
```

```
Palk.tulumaks = 20;
Palk t1 = new Palk(1000, "Heli Kopter");
Palk t2 = new Palk(500, "Ülle Õpilane");
System.out.println(t1.brutopalk);
```

1. 1000
2. ✓ 500
3. midagi muud
4. veateate



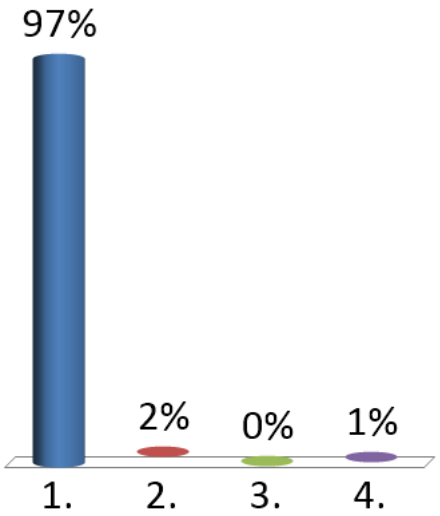
Mida väljastab järgmine programmilõik?

```
class Palk {
    static int tulumaks;
    int brutopalk;
    int netopalk;
    String töötaja;

    public Palk(int brutopalk,
                    String töötaja) {
        this.brutopalk = brutopalk;
        this.töötaja = töötaja;
        this.netopalk = brutopalk*
                    (100-tulumaks)/100;
    }
}
```

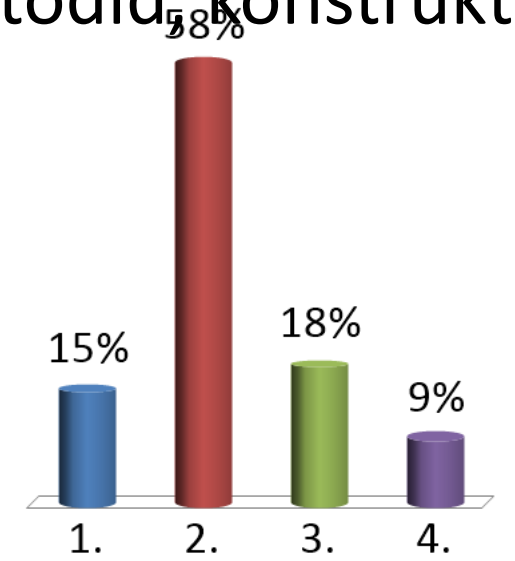
```
Palk.tulumaks = 20;
Palk t1 = new Palk(1000, "Heli Kopter");
Palk t2 = new Palk(500, "Ülle Õpilane");
System.out.println(t1.brutopalk);
```

1. 1000 ✓
2. 500
3. midagi muud
4. veateate



Milline on klassis õige järjekord?

1. isendiväljad, klassiväljad, konstruktorid, meetodid
2. ✓ klassiväljad, isendiväljad, konstruktorid, meetodid
3. klassiväljad, konstruktorid, meetodid, isendiväljad
4. isendiväljad, klassiväljad, meetodid, konstruktorid



Sõned

- Sõne pole algtüüpi
  - Iga sõne on isend
- Javas põhineb sõnetöötlus sisseehitatud klasside kasutamisel
  - `java.lang.String`
  - `java.lang.StringBuffer`
  - `java.lang.StringBuilder`

Loomine, võrdlemine

- `new`
- hulk konstruktoreid

```
char[] tähed = {'a','b','c'};
String sõne1 = new String(tähed);
```

- on eriline

```
String sõne2 = "abc";
```

Sõneliteraal

- võrdlemine

Objekti viida võrdlemine

```
System.out.println(sõne1 == sõne2);
System.out.println(sõne1.equals(sõne2));
```

false

true

Puhvri sisu võrdlemine

Literaal

- konkreetse väärtuse üleskirjutus programmis
- literaale ei tohi poolitada
- literaalina esitatud väärtuse tüüp on määratud kirjakujuga
  - näiteks
    - `0`, `-15`, `2000` `int`-tüüpi kümnendsüsteemis
    - `0b0`, `-0b1111`, `0b0_011_111_010_000` `int`-tüüpi kahendsüsteemis (alates Java 1.7)
    - `00`, `-017`, `03720` `int`-tüüpi kaheksandsüsteemis
    - `0x0`, `-0Xf`, `0xF` `int`-tüüpi kuueteistkümnendsüsteemis
    - `0L`, `-017l`, `-0Xf1`, `0x3E8l` `long`-tüüpi (soovitavalt L, mitte l)
    - `0.`, `-15.`, `60.301`, `20000e-1` `double`-tüüpi
    - `0.D`, `-15.d`, `20e2d` `double`-tüüpi
    - `0.f`, `15.F`, `60.9f`, `20000e-1F` `float`-tüüpi
    - `'a'`,`'%'`,`'\n'`,`'\\'`,`'\u03a8'`,`'\177'` `char`-tüüpi
    - `""`,`"abc"`,`"1. \n 2."` `String`-tüüpi
    - `true`, `false` `boolean`-tüüpi (ainsad)
    - `null` suvalist `viit`-tüüpi (tühiviit)

Sõne

```
char[] tähed = {'a','b','c'};
char[] tähed2 = {'a','b','c'};
String sõne1 = new String(tähed);
String sõne2 = new String(tähed2);
String sõne3 = "abc";
String sõne4 = "abc";
System.out.println(sõne1 == sõne2);

System.out.println(sõne1 == sõne3);

System.out.println(sõne3 == sõne4);
```

false

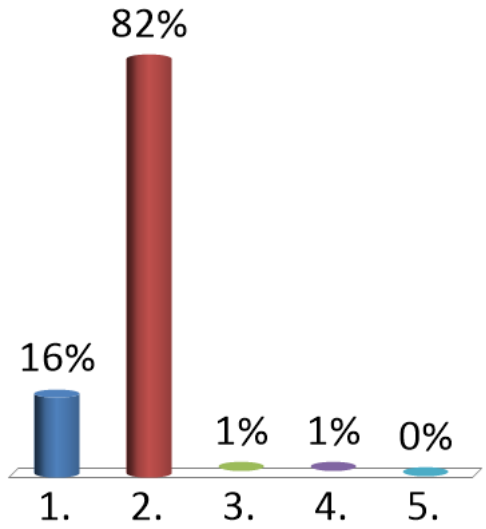
false

true

Mida väljastab järgmine programmilõik?

```
String s1 = "siil";
String s2 = "Siil";
boolean k = s1.equals(s2);
System.out.println(k);
```

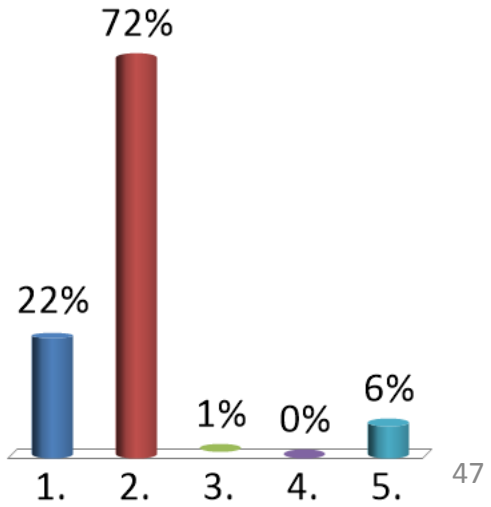
1. true
2. false ✓
3. midagi muud
4. mitte midagi
5. veateate



Mida väljastab järgmine programmilõik?

```
String s1 = "siil";
String s2 = "Siil";
if (s1 == s2)
    System.out.println("võrdsed");
else
    System.out.println("ebavõrdsed");
```

1. võrdsed
2. ebavõrdsed ✓
3. midagi muud
4. mitte midagi
5. veateate



Vaatame APIt

- `charAt`
- `compareTo`
- `equals`
- `equalsIgnoreCase`
- `indexOf`
- `length`
- `replace`
- `toLowerCase`
- `toUppercase`
- `toString`

Leksikograafiline võrdlemine

- compareTo

```
String s1 = "Tartu";
String s2 = "Tallinn";
int j = s1.compareTo(s2);
System.out.println(j);
```

6

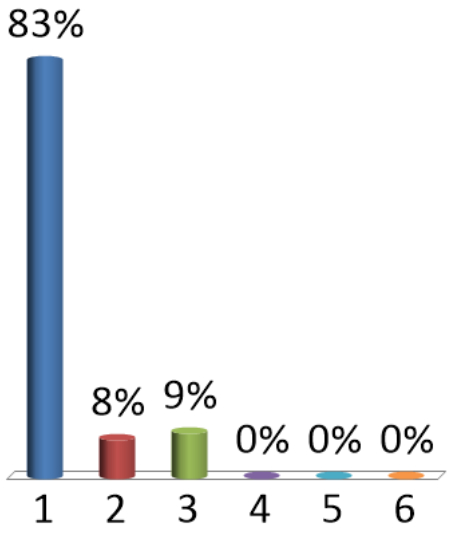
```
String s1 = "Tartu";
String s2 = "Tallinn";
int j = s2.compareTo(s1);
System.out.println(j);
```

-6

Mida väljastab järgmine programmilõik?

```
String str1 = "Piilupart";
String str2 = "piilupart";
String str3 = "Donald";
if (str1.equalsIgnoreCase(str2))
    System.out.println(str1);
else
    System.out.println(str3);
```

1. ✓ Piilupart
2. piilupart
3. Donald
4. midagi muud
5. mitte midagi
6. veateate



Mähisklassid

- 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

- Kolm põhjust, miks kasutada mähisklassi:
  - Andmestruktuurid ja meetodid, mille argument on objekt (nt. listid)

    ```
    ArrayList<Integer>
    ```

  - Selle klassi konstandid, nagu MIN_VALUE ja MAX_VALUE

    ```
    Integer.MAX_VALUE
    Character.CONTROL
    ```

  - Erinevad kontrollid ja teisendamised (sõneks ja sõnest, erinevate arvusüsteemide vahel)

    ```
    Integer.parseInt("123")
    Integer.toHexString(123)
    Character.isLetter('e')
    ```

Klass `Character`

- Java APIst
- Klassimeetodeid
  - `isDigit`
  - `isLetter`
  - `isLetterOrDigit`
  - `isLowerCase`
  - `isUpperCase`
  - `toLowerCase`
  - `toUpperCase`
- Isendimeetodeid
  - `charValue`
  - `compareTo`
  - `equals`
  - `toString`

Klass `Character`

```
char c = 'a';
System.out.println(Character.isLetter(c));

System.out.println(Character.isDigit(c));

System.out.println
      (Character.isLetterOrDigit(c));

Character ch = Character.valueOf(c);
System.out.println(ch.charValue());
```

true

false

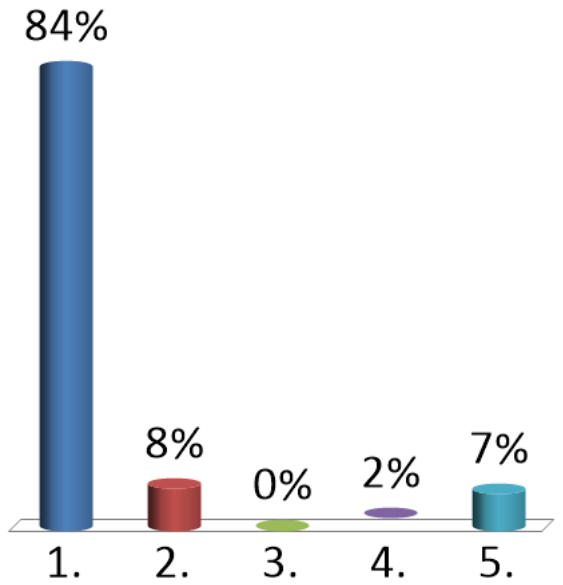
true

a

Mida väljastab järgmine programmilõik?

```
char c1 = 'a';
char c2 = 'a';
System.out.println(c1.equals(c2));
```

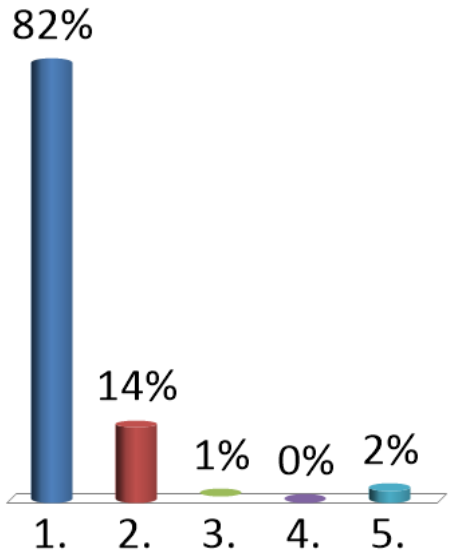
1. true
2. false
3. mitte midagi
4. midagi muud
5. veateate ✓



Mida väljastab järgmine programmilõik?

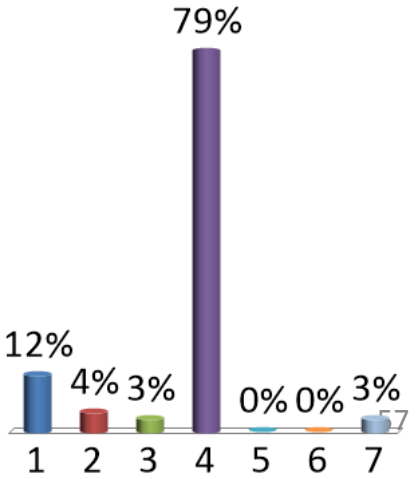
```
char c1 = 'a';
char c2 = 'a';
System.out.println(c1 == c2);
```

1. true ✓
2. false
3. mitte midagi
4. midagi muud
5. veateate



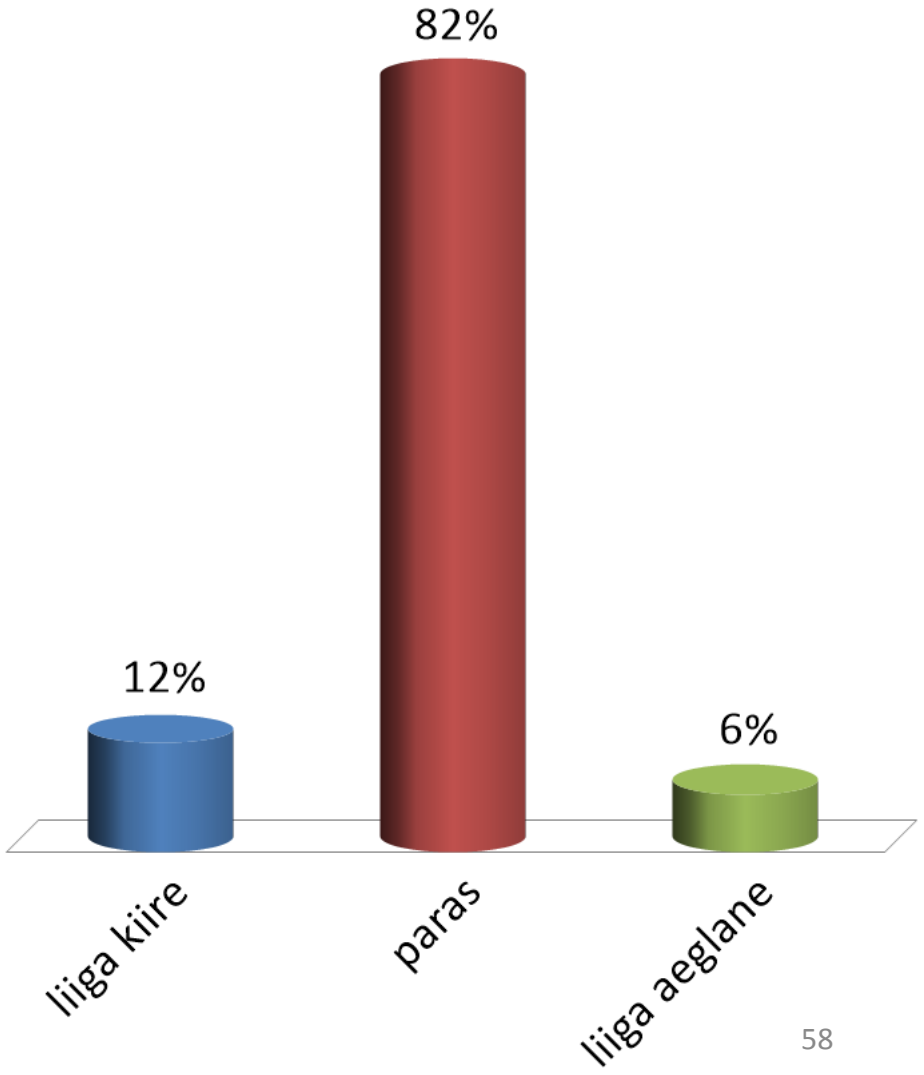
Igal algtüübil on vastav mähisklass. Algtüüpile `char` vastab klass `Character`. Veel ühel algtüübil erineb mähisklassi nimi algtüübi nimest rohkem kui esitähe suuruse poolest.

1. `boolean`
2. `byte`
3. `short`
4. ✓ `int`
5. `long`
6. `float`
7. `double`



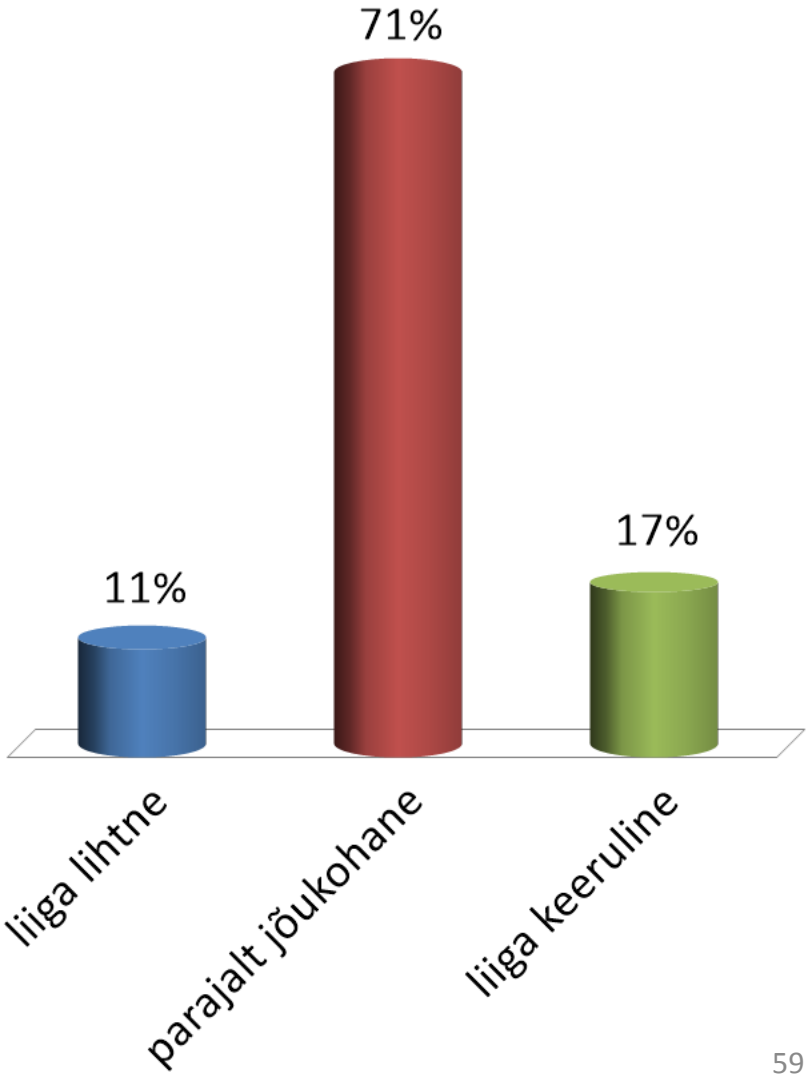
Loengu tempo oli

1. liiga kiire
2. paras
3. liiga aeglane



Materjal tundus

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



Suur tänu osalemast ja kohtumiseni!