Objektorienteeritud programmeerimine

14. loeng, 13. mai

Marina Lepp

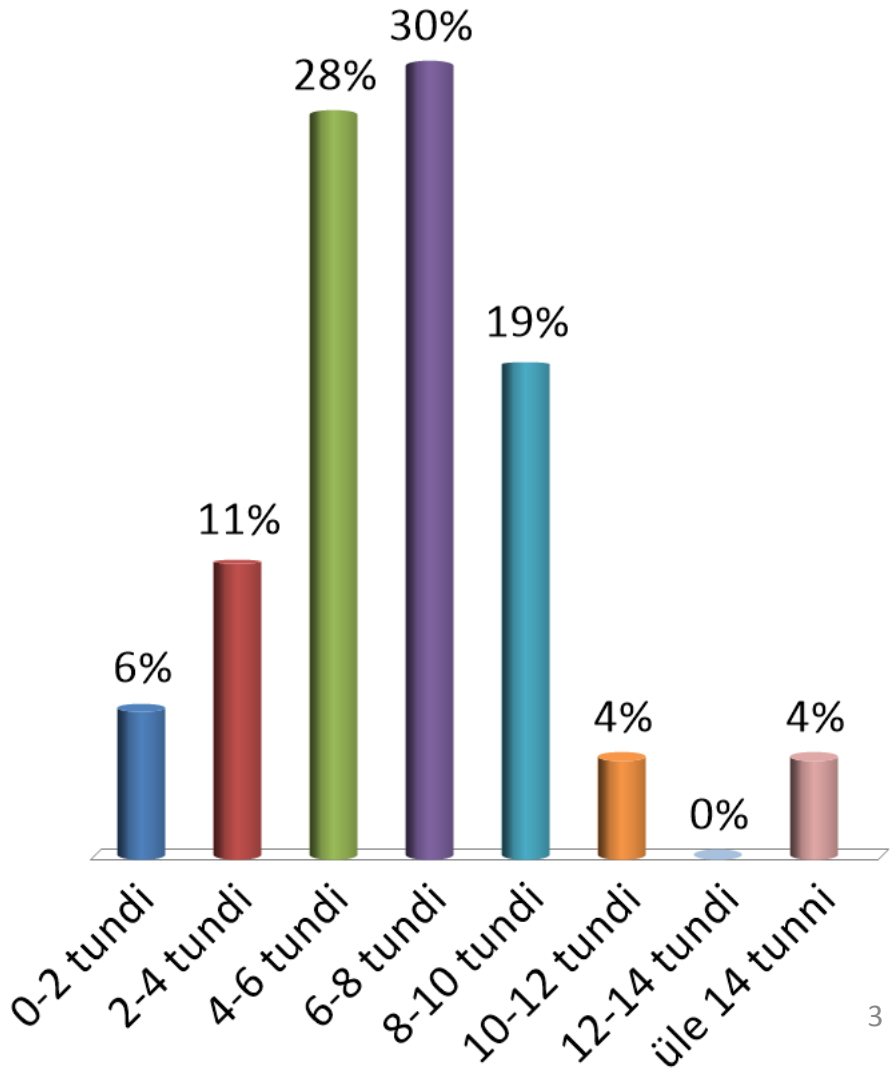
Eelmisel nädalal

- Loeng
  - lõimed
- Lisapraktikum
- Praktikum
  - 2. kontrolltöö

- Euroopa päev
- Emadepäev

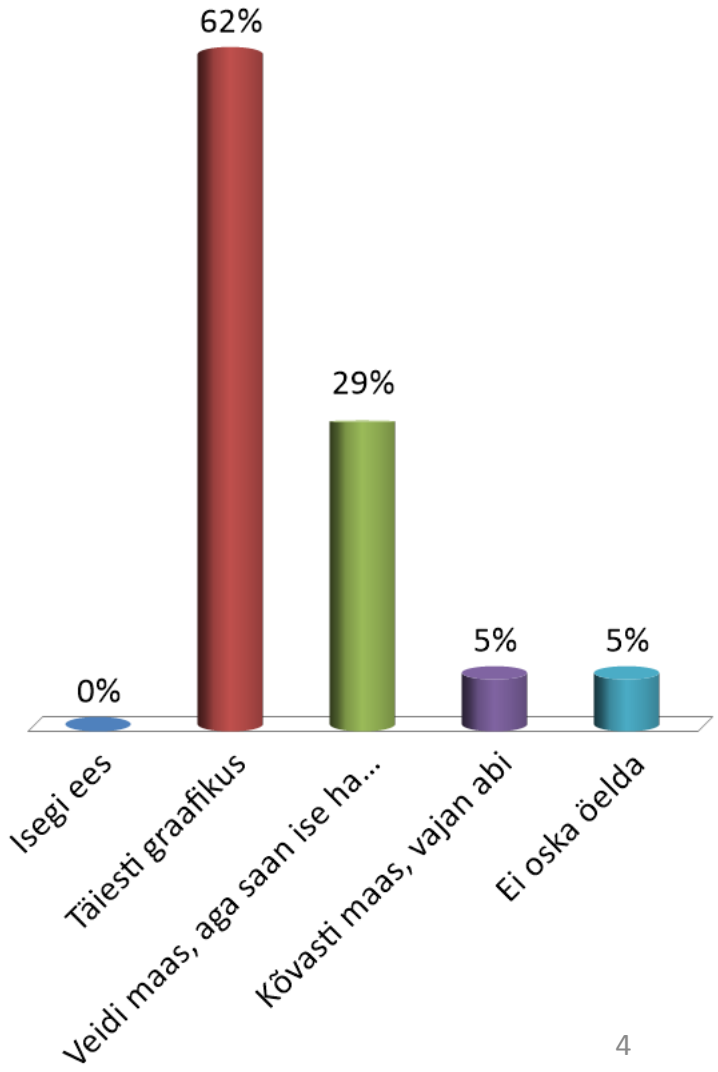
Umbes mitu tundi tegelesite eelmisel nädalal selle ainega (loeng+praktikum+iseseisvalt)?

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

- Organisatoorset
- Sünkroniseerimine
- Ujukomaarvud
- Internatsionaliseerimine, lokaliseerimine
- Võrguprogrammeerimine

Organisatsioonilisi küsimusi

- Lõpusirge
  - https://courses.cs.ut.ee/2019/OOP/spring/Main/Lopusirge
- 2. rühmatöö – 16.05
  - https://courses.cs.ut.ee/2019/OOP/spring/Main/Ruhm2
- Rühmatööde lõpuesitlused – 23.05
  - https://courses.cs.ut.ee/2019/OOP/spring/Main/RuhmEsitlus
- 2. kontrolltöö analüüs – 23.05
  - https://courses.cs.ut.ee/2019/OOP/spring/Main/KTAnaluus2

Lõimed

- Lõimed
  - programmi täitmisel täidetavad "kergekaalulised" paralleelsed protsessid, mis võivad olla sisuliselt seotud ning teha koostööd
- Mitmelõimelisus on Java platvormi omadus
  - `Object`
- Igal Java programmil on vähemalt üks lõim – `main`
  - tegelikult rohkemgi

Javas

- Javas iga lõim on seotud klassi `Thread` isendiga
- Lõime loomiseks on kaks põhilist moodust:
  - luua klassi `Thread` alamklass ja kirjutada lõime tegevus meetodisse `run()`
  - luua liidest `Runnable` realiseeriv klass ja kirjutada lõime tegevus meetodisse `run()`. Lõime loomiseks luua klassi `Thread` isend, mille konstruktorile anda ette antud klassi isend
- Lõime käivitamiseks kasutada `Thread` meetodit `start()`

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Thread.html
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Runnable.html

2. moodus lõimede tegemiseks

```
public class PrindiArvud implements Runnable{

    private int esimeneArv;
    private int viimaneArv;

    public PrindiArvud(int n1, int n2) {
        esimeneArv = n1;
        viimaneArv = n2;
    }

    public void run() {
        for (int i = esimeneArv; i <= viimaneArv; i++) {
            System.out.print(i + "; ");
        }
    }
}
```

2. moodus

```
public class TestPrindiArvud {
    public static void main(String[] args) {
        Runnable r1 = new PrindiArvud(1, 10);
        Runnable r2 = new PrindiArvud(20, 30);
        Runnable r3 = new PrindiArvud(50, 60);
        Thread t1 = new Thread(r1);
        Thread t2 = new Thread(r2);
        Thread t3 = new Thread(r3);
        t1.start();
        t2.start();
        t3.start();
    }
}
```

1; 2; 3; 20; 21; 22; 23; 24; 25; 26; 27; 28; 50; 51; 52; 53; 54; 55; 56; 29; 4; 57; 30; 58; 59; 60; 5; 6; 7; 8; 9; 10;

Prioriteedid

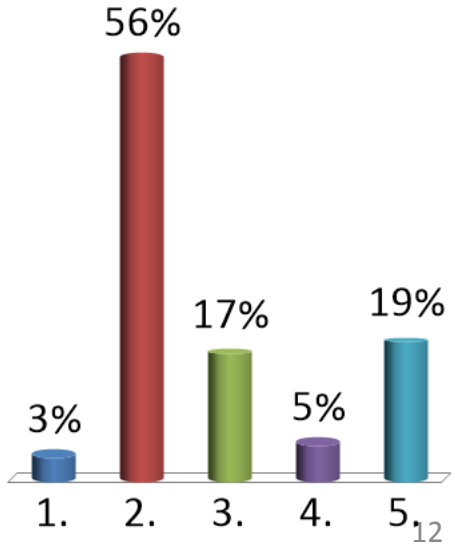
- Igal lõimel on prioriteet – täisarv 1 kuni10
- Kõrgema prioriteediga lõimedel on eelis madalama prioriteediga lõimede ees
- Vaikimisi prioriteet on 5
- Meetodid
  - **`setPriority()`**
  - **`getPriority()`**
- Konstandid
  - MAX_PRIORITY
  - MIN_PRIORITY
  - NORM_PRIORITY

- Lõimede juhtimise konkreetne strateegia on platvormist sõltuv

Millised võivad ekraanile ilmuda?

```
class Lõim1 extends Thread {
    public void run() {
        System.out.print("A");
        System.out.print("B");
    }
}
class Lõim2 extends Thread {
    public void run() {
        System.out.print("1");
        System.out.print("2");
    }
}
public class AB12 {
    public static void main(String[] args) {
        Lõim1 l1 = new Lõim1();
        Lõim2 l2 = new Lõim2();
        l2.setPriority(10);
        l1.start();
        l2.start();
    }
}
```

1. ABAB
2. ✓ 12AB
3. A12B
4. BA12
5. A1B2



Tahaks kasutada samu andmeid

- Mured
  - lõimede vastastikune mõju (*thread interference*)
    - ka üks direktiiv võib JVM mõttes olla mitmesammuline
      - näiteks `c++` korral 3 sammu
      - kui teine lõim teeb `c--`
  - mälu terviklikkus (*memory consistency*)
    - kui erinevate lõimede “arusaam” andmetest ei ole sama
    - juhtub-varem seos (*happens-before*)
- Lahenduseks on sünkroniseerimine

https://docs.oracle.com/javase/tutorial/essential/concurrency/sync.html

Sünkroniseerimine

- Anda ühele lõimele juurdepääs ja blokeerida samal ajal teisi
- “Lukustada” ühe lõime jaoks

Ei sünkroniseeri

```
public class Helistamine {
    void helista(String jutt) {
        //helistamise algus
        System.out.print("[" + jutt);
        try{
            Thread.sleep(1000);
        } catch(InterruptedException e) {
            System.out.println("Katkestatud");
            throw new RuntimeException(e);
        }
        System.out.println("]");
        //helistamise lõpp
    }
}
```

Ei sünkroniseeri

```
public class Helistaja implements Runnable {
    private String jutt;
    private Helistamine h;

    public Helistaja(Helistamine h, String s) {
        this.h = h;
        jutt = s;
        Thread t = new Thread(this);
        t.start();
    }

    public void run() {
        h.helista(jutt);
    }
}
```

Ei sünkroniseeri

```
public class TestHelistamine {
    public static void main(String[] args) {
        Helistamine h = new Helistamine();
        Helistaja ob1 = new Helistaja(h, "Esimene");
        Helistaja ob2 = new Helistaja(h, "Teine");
        Helistaja ob3 = new Helistaja(h, "Kolmas");
    }
}
```

```
[Teine[Esimene[Kolmas]
]
]
```

Sünkroniseerimine

Java koodi saab sünkroniseerida kahel viisil:

- meetodi sünkroniseerimine

```
synchronized void meetod(){
        ...
}
```

- koodilõigu sünkroniseerimine

```
synchronized(objekt){
        ...
}
```

Sünkroniseerime

```
public class Helistamine {
    synchronized void helista(String jutt) {
        //helistamise algus
        System.out.print("[" + jutt);
        try{
            Thread.sleep(1000);
        } catch(InterruptedException e) {
            System.out.println("Katkestatud");
            throw new RuntimeException(e);
        }
        System.out.println("]");
        //helistamise lõpp
    }
}
```

```
[Teine]
[Kolmas]
[Esimene]
```

Sünkroniseerime teistmoodi

```
public class Helistamine {
     void helista(String jutt) {
         //helistamise algus
          synchronized(this) {
               System.out.print("[" + jutt);
               try {
                    Thread.sleep(1000);
               } catch (InterruptedException e) {
                    System.out.println("Katkestatud");
                    throw new RuntimeException(e);
               }
               System.out.println("]");
               //helistamise lõpp
          }
     }
}
```

```
[Teine]
[Esimene]
[Kolmas]
```

Aga mis juhtus?

```
Helistamine h = new Helistamine();
Helistaja ob1 = new Helistaja(h, "Esimene");
```

```
public Helistaja(Helistamine h, String s) {
    this.h = h;
    jutt = s;
    Thread t = new Thread(this);
    t.start();
}
public void run() {
    h.helista(jutt);
}
```

```
synchronized void helista(String jutt) {
```

- `helista` on klassi `Helistamine` isendimeetod
- `h` on lukustatud `ob1` lõime jaoks

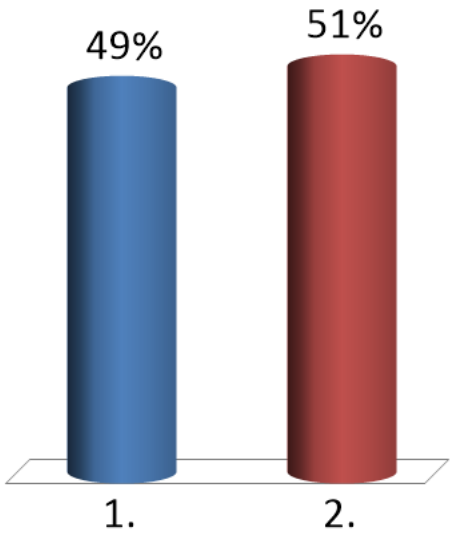
Monitor

- Sünkroniseerimine on korraldatud monitoride abil
  - *monitor*, *intrinsic lock*, *monitor lock*
- Iga objektiga on seotud oma monitor
  - ka klassiga, kui klassi **`Class`** isendiga
- Kui lõim kutsub välja sünkroniseeritud meetodi (või jõuab sünkroniseeritud koodilõiguni), siis saab see lõim vastava objekti monitori endale. Kuni meetodist naasmiseni (või koodilõigu lõpuni).
- Ühel hetkel saab konkreetne monitor olla vaid ühel lõimel
- Kõik ülejäänud lõimed, mis tahavad monitori, peavad ootama

Kas **`helista`** tehakse kindlasti ühe hooga lõpuni?

```
Helistamine h = new Helistamine();
Helistamine h1 = new Helistamine();
Helistamine h2 = new Helistamine();
Helistaja ob1 = new Helistaja(h, "Esimene");
Helistaja ob2 = new Helistaja(h1, "Teine");
Helistaja ob3 = new Helistaja(h2, "Kolmas");
```

1. Jah
2. ✓ Ei



Loendur

```
public class Loendur {
    private int c = 0;
    public void ühikLiitmine() {
        int algneC = c;
        c++;
        if (c > algneC + 1)
            System.out.println("Anomaalia: " +
                "algne = " + algneC + ", nüüd = " + c);
    }
    public void ühikLahutamine() {
        c--;
    }
    public int väärtus(){
        return c;
    }
}
```

Võidujooks (*race condition*)

```
public class Võidujooks implements Runnable{
    private Loendur loendur;
    public Võidujooks(Loendur loendur) {
        this.loendur = loendur;
    }
    public void run() {
        for (int i = 0; i < 100000000; i++) {
            loendur.ühikLiitmine();
        }
    }
}
```

Võidujooks (*race condition*)

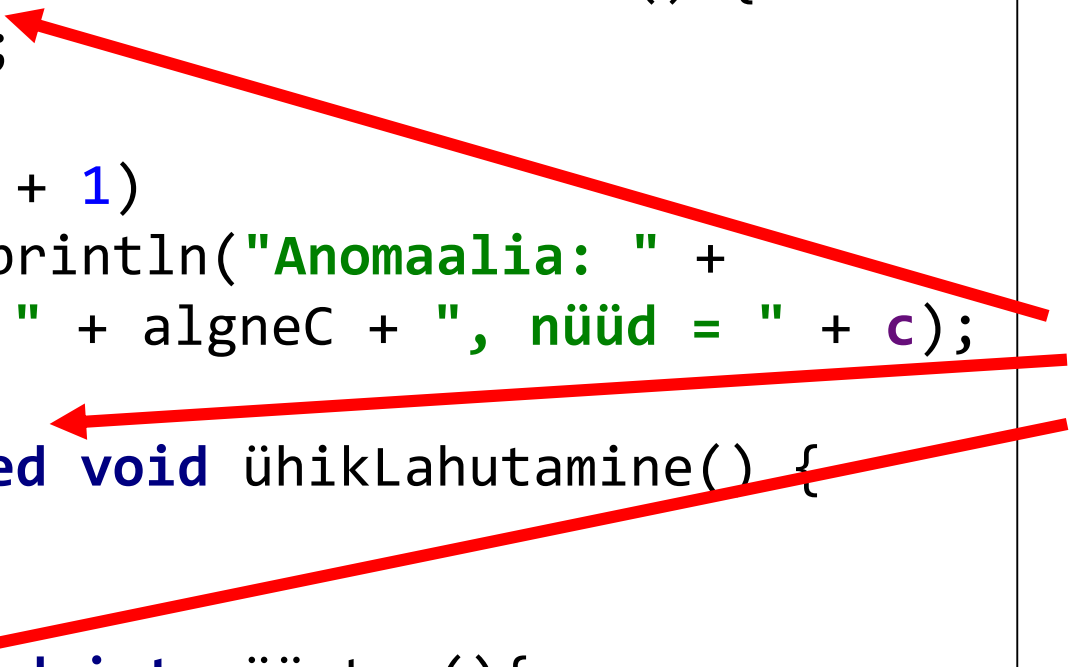
```
public class TestVõidujooks {
    public static void main(String[] args) {
        Loendur loendur = new Loendur();
        Runnable r1 = new Võidujooks(loendur);
        Runnable r2 = new Võidujooks(loendur);
        Runnable r3 = new Võidujooks(loendur);
        Thread t1 = new Thread(r1);
        Thread t2 = new Thread(r2);
        Thread t3 = new Thread(r3);
        t1.start();
        t2.start();
        t3.start();
    }
}
```

Anomaalia: algne = 29, nüüd = 246
Anomaalia: algne = 23, nüüd = 126972

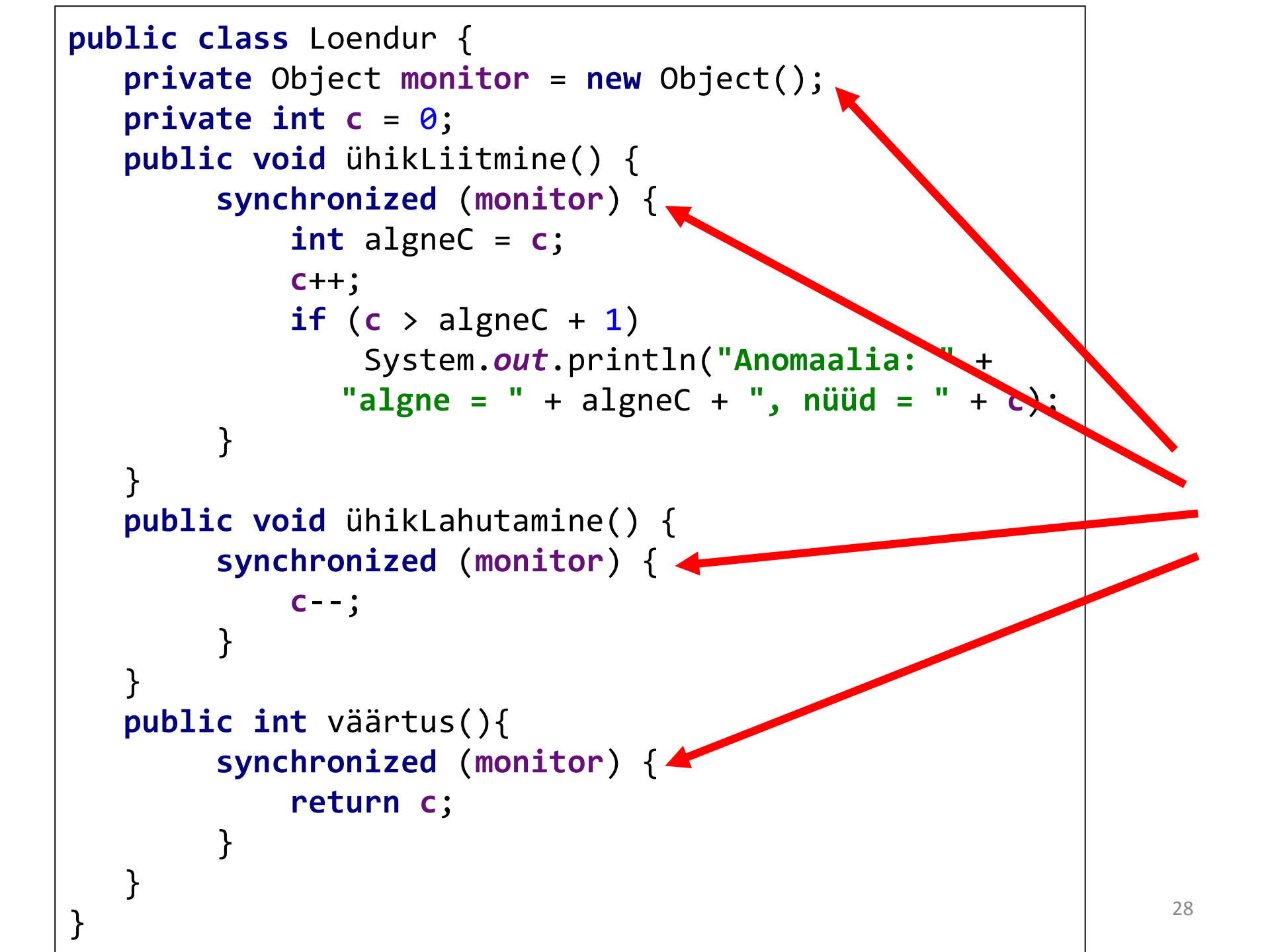
Anomaalia: algne = 1196, nüüd = 1450

Sünkroniseeritud loendur

```
public class Loendur {
    private int c = 0;
    public synchronized void ühikLiitmine() {
        int algneC = c;
        c++;
        if (c > algneC + 1)
            System.out.println("Anomaalia: " +
                "algne = " + algneC + ", nüüd = " + c);
    }
    public synchronized void ühikLahutamine() {
        c--;
    }
    public synchronized int väärtus(){
        return c;
    }
}
```



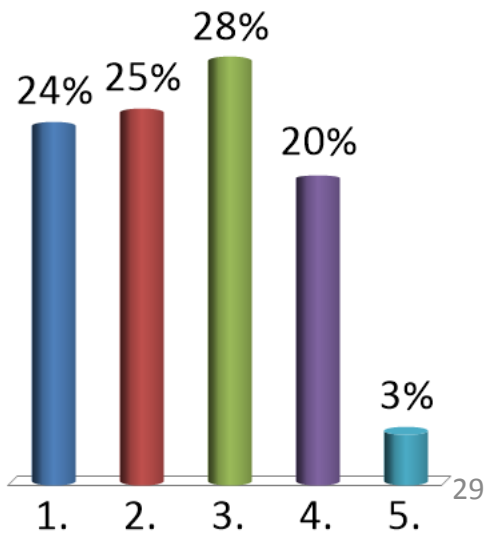
```
public class Loendur {
    private Object monitor = new Object();
    private int c = 0;
    public void ühikLiitmine() {
        synchronized (monitor) {
            int algneC = c;
            c++;
            if (c > algneC + 1)
                System.out.println("Anomaalia: " +
                "algne = " + algneC + ", nüüd = " + c);
        }
    }
    public void ühikLahutamine() {
        synchronized (monitor) {
            c--;
        }
    }
    public int väärtus(){
        synchronized (monitor) {
            return c;
        }
    }
}
```



Millised võivad ekraanile ilmuda?

```
class Arvud{
    private int n1; private int n2;
    public Arvud(int n1, int n2) {
        this.n1 = n1; this.n2 = n2;          }
    public void prindi() {
        for (int i = n1; i <= n2; i++)
            System.out.print(i);             }
}
class Lõim extends Thread{
    private Arvud a;
    public Lõim(int n1, int n2) {
        a = new Arvud(n1, n2);               }
    public void run() {
        a.prindi();                          }
}
public class ArvudLõim {
    public static void main(String[] args) {
        Lõim l1 = new Lõim(1, 3);
        Lõim l2 = new Lõim(5, 7);
        l1.start(); l2.start();              }
}
```

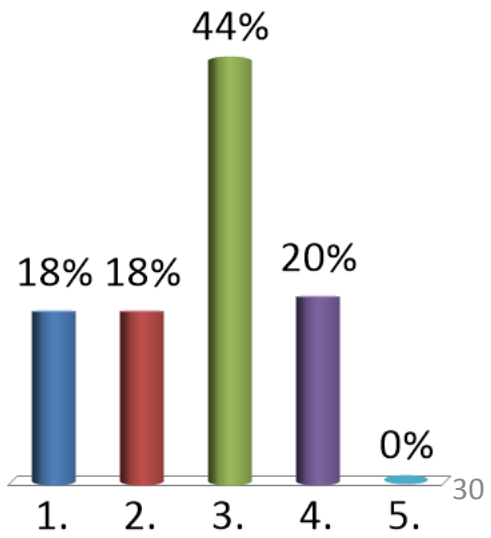
- ✓ 1. 512637
- ✓ 2. 156723
- ✓ 3. 123567
- ✓ 4. 152367
- 5. 132567



Millised võivad ekraanile ilmuda?

```
class Arvud{
    private int n1; private int n2;
    public Arvud(int n1, int n2) {
        this.n1 = n1; this.n2 = n2;          }
    public synchronized void prindi() {
        for (int i = n1; i <= n2; i++)
            System.out.print(i);             }
}
class Lõim extends Thread{
    private Arvud a;
    public Lõim(int n1, int n2) {
        a = new Arvud(n1, n2);               }
    public void run() {
        a.prindi();                          }
}
public class ArvudLõim {
    public static void main(String[] args) {
        Lõim l1 = new Lõim(1, 3);
        Lõim l2 = new Lõim(5, 7);
        l1.start(); l2.start();              }
}
```

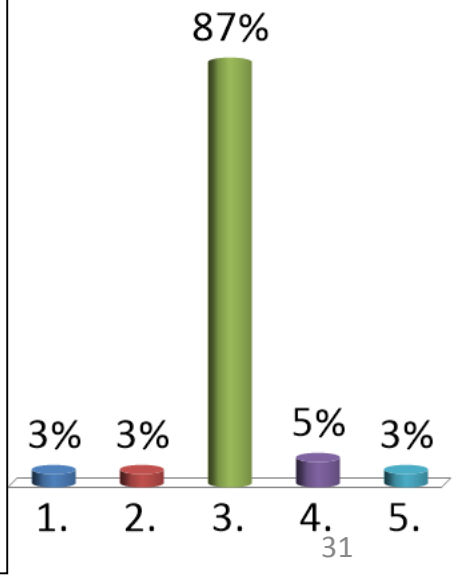
1. 512637 ✓
2. 156723 ✓
3. 123567 ✓
4. 152367 ✓
5. 132567



Millised võivad ekraanile ilmuda?

```
class Arvud{
    public synchronized void prindi(int n1, int n2) {
        for (int i = n1; i <= n2; i++)
            System.out.print(i);                    }
}
class Lõim extends Thread{
    private Arvud a; private int n1; private int n2;
    public Lõim(Arvud a, int n1, int n2) {
        this.a = a;
        this.n1 = n1; this.n2 = n2;                 }
    public void run() {
        a.prindi(n1, n2);                           }
}
public class ArvudLõim {
    public static void main(String[] args) {
        Arvud a = new Arvud();
        Lõim l1 = new Lõim(a, 1, 3);
        Lõim l2 = new Lõim(a, 5, 7);
        l1.start();
        l2.start();                                 }
}
```

1. 512637
2. 156723
3. 123567 ✓
4. 152367
5. 132567



Lõimedevaheline suhtlemine

- Klassi **`Object`** isendimeetodid
  - **`wait`**
  - **`notify`**
  - **`notifyAll`**

- ***`obj`*`.wait()`**
  - peatab jooksva lõime töö niikauaks, kuni mingi teine lõim saadab teate ***`obj`*`.notify()`** või ***`obj`*`.notifyAll()`**
  - on ka **`wait(`*`millisekundeid`*`)`** versioon, mille toimel lõim "ärkab" ka etteantud aja pärast

Töö näide

```
public class Töö {
    private String nimi;

    Töö(String nimi) {
        this.nimi = nimi;
    }

    String getNimi() {
        return nimi;
    }
}
```

```
public class TööJärjekord {
    LinkedList<Töö> järjekord = new LinkedList <>();

    synchronized void lisaTöö(Töö t) {
        järjekord.addLast(t);
        notify();
    }

    synchronized Töö võtaTöö() {
        while (järjekord.isEmpty()) {
            try {
                wait();
            }
            catch (Exception e) {
                throw new RuntimeException(e);
            }
        }
        return järjekord.removeFirst();
    }
}
```

```
public class Tööline implements Runnable {
   private TööJärjekord järjekord;

   Tööline(TööJärjekord järjekord, int nr) {
      this.järjekord = järjekord;
      new Thread(this, "Tööline-" + nr).start();
   }

   public void run() {
      while (true) {
         Töö t = järjekord.võtaTöö();
         teeÄra(t);
      }
   }

   void teeÄra(Töö x) {
      System.out.println(Thread.currentThread().getName() +
                  " tegi töö nr " + x.getNimi());
   }
}
```

```
public class TestTöö {
    public static void main(String[] args) {
        TööJärjekord järjekord = new TööJärjekord();
        Tööline[] töötajad = new Tööline[4];
        for (int i = 0; i < töötajad.length; i++) {
            töötajad[i] = new Tööline(järjekord, i+1);
        }

        for (int i = 1; i <= 100; i++) {
            järjekord.lisaTöö(new Töö(String.valueOf(i)));
        }
    }
}
```

Spetsiaalne järjekord

- **`BlockingQueue`**
  - **`put`**
  - **`take`**
  - **`poll`**

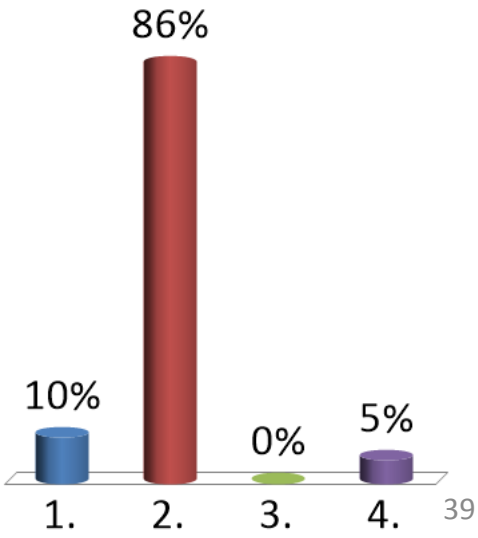
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/BlockingQueue.html

```
public class TööJärjekord {
    BlockingQueue<Töö> järjekord =
                                new ArrayBlockingQueue<>(10);

     void lisaTöö(Töö t) {
         try {
             järjekord.put(t);
         } catch (InterruptedException e) {
             throw new RuntimeException(e);
         }
    }

     Töö võtaTöö() {
         try {
             return järjekord.take();
         } catch (InterruptedException e) {
             throw new RuntimeException(e);
         }
    }
}
```

Mis ilmub ekraanile?

```
System.out.println(0.03-0.02);
```

1. 0.01
2. ✓ 0.009999999999999998
3. midagi muud
4. veateade



Ujukomaarvude täpne esitus

```
double a = 0.02;
double b = 0.03;
double c = b - a;
System.out.println(c);

BigDecimal a1 = new BigDecimal("0.02");
BigDecimal b1 = new BigDecimal("0.03");
BigDecimal c1 = b1.subtract(a1);
System.out.println(c1);
```

0.009999999999999998
0.01

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/math/BigDecimal.html

Internatsionaliseerimine

- *Internationalization*
  - *i18n*
- Rakenduste projekteerimine selliselt, et seda saab kohandada (lokaliseerida) erinevatele keeltele ja piirkondadele muutmata rakenduse arhitektuuri
  - Lokaliseerimisel lisatakse piirkonnaspetsiifilised komponendid (nt. tõlgitud tekst, andmete piirkonnaspetsiifiline esitus, töötlus jne.)
    - *Localization*
      - *l10n*

https://docs.oracle.com/javase/tutorial/i18n/index.html

Erinevad stiilid

- Kuupäev (nt. 13/05/2019, 2019/05/13, 13.05.2019, 2019-05-13)
- Kellaaeg
- Arvud (10,000)
- Valuuta (nt. $, £, ¥, €, SEK)
- Mõõtühikud (nt. pikkus, mass, ruumala, temperatuur, …)
- Telefoninumbrid
- Postiaadressid
- Sõnumid
- Märgised kasutajaliidese komponentidel
- …

i18n

- Tekstielemendid, nt. seisundisõnumid ja GUI komponentide sildid ei tohi olla programmi otse sisse kirjutatud (*hardcoded*). Nad peavad olema väljaspool lähtekoodi ja dünaamiliselt kättesaadavad
- Uue keele toetamine ei tohi nõuda koodi kompileerimist
- Teised kultuurist sõltuvad andmed, nt kuupäevad näidatakse lõppkasutaja piirkonna formaadis
- …

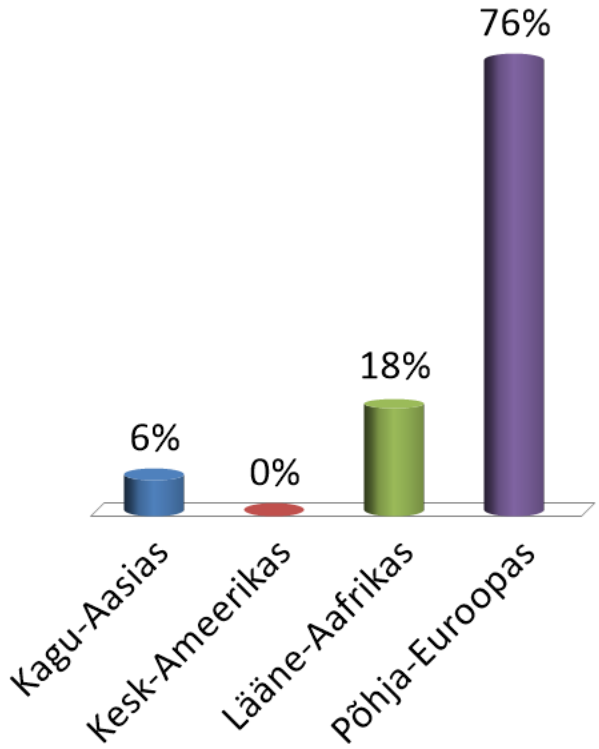
Javas

- klass **`Locale`**
  - **lo.kaat** <22e: -kaadi, -kaati> *kultuurispetsiifiliste elementide kogum*
- keel
  - ISO 639
  - http://www.loc.gov/standards/iso639-2/php/English_list.php
- skript
  - ISO 15924
  - http://unicode.org/iso15924/iso15924-codes.html
- riik (regioon)
  - ISO 3166
  - http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

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

Millises piirkonnas räägitakse keelt, mille kahetäheline ISO 639 kood on „ee“?

1. Kagu-Aasias
2. Kesk-Ameerikas
3. ✓ Lääne-Aafrikas
4. Põhja-Euroopas



Lokaat

```
System.out.println(Locale.getDefault());
```

```
et_EE
```

keel ja riik

```
Locale lo = new Locale("et", "EE");
System.out.println(lo.getDisplayCountry());
System.out.println(lo.getDisplayLanguage());
System.out.println(lo.getDisplayName());
```

```
Eesti
eesti
eesti (Eesti)
```

Lokaat

```
System.out.println(
   Locale.GERMANY.getDisplayCountry(Locale.FRANCE));
```

Allemagne

```
Locale[] availableLocales =
                    Locale.getAvailableLocales();
```

Lokaaditundlik

- *locale-sensitive*
- kuupäeva, arvu formaatimine
- lokaaditundlikud objektid
  - klass **`ResourceBundle`**

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

Veel mitte i18n

```
System.out.println("Tere!");
System.out.println("Kuidas läheb?");
System.out.println("Head aega!");
```

```
import java.util.Locale;
import java.util.ResourceBundle;
public class JubaI18n {
    static public void main(String[] args) {
        String keel;
        String maa;
        if (args.length != 2) {
            keel = new String("et");
            maa = new String("EE");
        } else {
            keel = new String(args[0]);
            maa = new String(args[1]);
        }

        Locale currentLocale = new Locale(keel, maa);
        ResourceBundle teated = ResourceBundle.getBundle(
                                    "Teated", currentLocale);
        System.out.println(teated.getString("tervitus"));
        System.out.println(teated.getString("huvi"));
        System.out.println(teated.getString("lahkumine"));
    }
}
```

```
Hallo.
Wie geht's?
Tschüß.
```

Teated_et_EE.properties

```
tervitus = Tere!
lahkumine = Head aega!
huvi = Kuidas läheb?
```

Teated_fr_FR.properties

```
tervitus = Bonjour.
lahkumine = Au revoir.
huvi = Comment allez-vous?
```

Teated_de_DE.properties

```
tervitus = Hallo.
lahkumine = Tschüß.
huvi = Wie geht's?
```

Liitsõnum

- Liitsõnum võib sisaldada erinevaid muutujaid, nt. kuupäev, kellaaeg, valuuta, protsent jne

- Pühapäeval, 26.05.19 kell 10:15 algab 38. Tartu Rattaralli.

```
import java.util.*;
import java.text.*;
public class Liitsõnum {
    static void displayMessage(Locale currentLocale) {
        System.out.println("Lokaat = " + currentLocale.toString());
        ResourceBundle messages = ResourceBundle.getBundle(
                                                "Rall",currentLocale);
        GregorianCalendar gc = new GregorianCalendar(2019, 4, 26, 10, 15);
        Object[] messageArguments = {
              messages.getString("day"),
              38,
              new Date(gc.getTimeInMillis())
        };
        MessageFormat formatter = new MessageFormat("");
        formatter.setLocale(currentLocale);
        formatter.applyPattern(messages.getString("template"));
        String output = formatter.format(messageArguments);
    }
    p
    }
}
```

```
Lokaat = et_EE
Pühapäeval, 26.05.19 kell 10:15 algab 38. Tartu Rattaralli.
Lokaat = en_US
On Sunday May 26, 2019 at 10:15 AM the Tartu 38th Bicycle Rally will
start.
```

Rall_et_EE.properties

```
template = {0}, {2,date,short} kell
{2,time,short} algab
{1,number,integer}. Tartu Rattaralli.
day = Pühapäeval
```

Rall_en_US.properties

```
template = On {0} {2,date,long} at
{2,time,short} the Tartu
{1,number,integer}th Bicycle Rally will
start.
day = Sunday
```

Arvuformaat

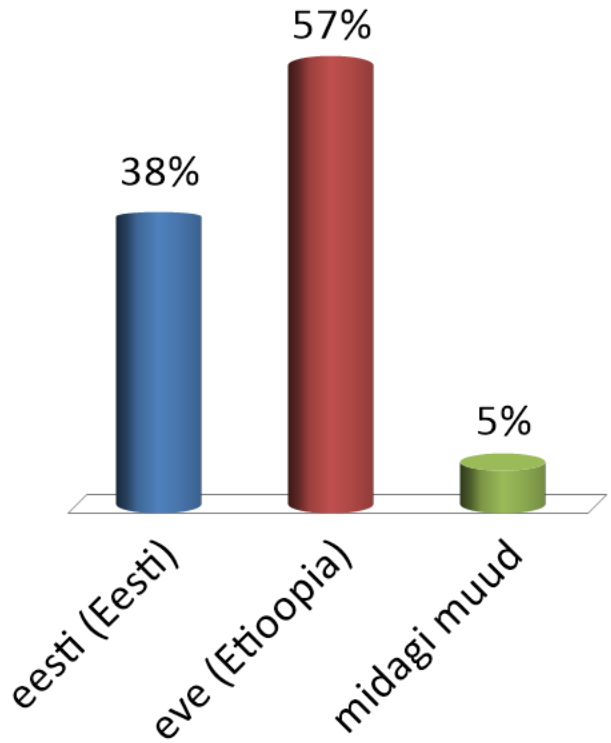
```
import java.util.*;
import java.text.*;
public class ArvuFormaat {
   static void reformat(double num, String[] locales) {
      for (String loc : locales) {
         Locale pl = Locale.forLanguageTag(loc.replace("_","-"));
         NumberFormat fmt = NumberFormat.getInstance(pl);
         System.out.print(fmt.format(num));
         System.out.println("\t" + pl.getDisplayName());
      }
   }
   public static void main(String[] args) {
      String [] locales = {"et_EE", "de_DE", "fr_FR",
                                    "en_US", "ru_RU", "ee_ET"};
      reformat(12345.6789, locales);
   }
}
```

```
12 345.679      eesti (Eesti)
12.345,679      saksa (Saksamaa)
12 345,679      prantsuse (Prantsusmaa)
12,345.679      inglise (Ameerika Ühendriigid)
12 345,679      vene (Venemaa)
12,345.679      eve (Etioopia)
```

Mis ilmub ekraanile?

```
Locale lo = new Locale("ee", "ET");
System.out.println(lo.getDisplayName());
```

1. eesti (Eesti)
2. ✓ eve (Etioopia)
3. midagi muud

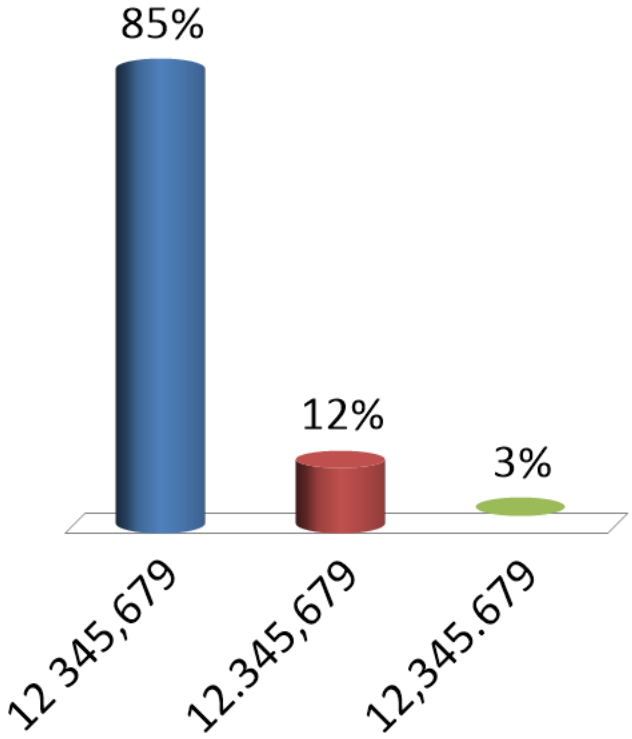


Milline on eestipärane formaat?

✓ **a. 12 345,679**

b. 12.345,679

c. 12,345.679



Võrguprogrammeerimine
Host

- Internetti ühendatud seade: enamasti arvuti, kuid võib olla ka ruuter, printer, faks jne
- IP (*Internet Protocol*) aadress
  - Kasutatakse punktidega eraldatud neli märgita täisarvu 0 kuni 255
    - nt 199.1.32.90
- Domeeninimi IP aadressiks
  - www.ut.ee →193.40.5.73

Klass `java.net.InetAddress`

Isendi loomiseks

```
public static InetAddress getByName(String host)
public static InetAddress[] getAllByName(String host)
public static InetAddress getLocalHost()

        throws UnknownHostException
```

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/net/InetAddress.html

```java
import java.net.InetAddress;
import java.net.UnknownHostException;
public class Võrk {
   public static void main (String[] args)
                                        throws UnknownHostException{
      InetAddress me = InetAddress.getLocalHost();
      System.out.println("Minu nimi on: " + me.getHostName());
      System.out.println("Minu aadress on: " +me.getHostAddress());
      InetAddress address = InetAddress.getByName("www.ut.ee");
      System.out.println("Hosti nimi: " + address.getHostName());
      System.out.println("IP aadress: "+ address.getHostAddress());
   }
}
```

```
Minu nimi on: aurelius
Minu aadress on: 192.168.1.69
Hosti nimi: www.ut.ee
IP aadress: 193.40.5.73
```

Pordid

- Tavaliselt on hostil ainult üks Interneti aadress
- See aadress jagatakse 65 536 pordi vahel
  - Pordid on loogilised abstraktsioonid, mis lubavad ühel hostil suhelda samaaegselt paljude teiste hostidega
  - Osa teenuseid on seotud kindlate pordinumbritega (0…1023), nt
    - HTTP – 80, telnet – 23, finger – 79, SMTP – 25, …

Protokollid

- Protokoll defineerib, kuidas kaks hosti omavahel suhtlevad
- Protokoll määrab:
  - andmetihendusmeetodid
  - kuidas saatev seade annab teada, et sõnumi edastamine on lõpetatud
  - kuidas vastuvõtja teavitab saatjat
  - …
- IP
  - *Internet Protocol*
  - internetiaadresside tasemel
- UDP
  - *User Datagram Protocol*
  - reeglid sõnumite vahetamiseks teiste internetipunktidega andmepakettide tasemel
- TCP
  - *Transmission Control Protocol*

Protokollid rakenduse tasemel

- http
  - hüperteksti edastamise protokoll HyperText Transfer Protocol
- telnet
  - kaugterminaliga suhtlemise protokoll
- ftp, ftp-data
  - failide ülekande protokollid (file transfer protocol)
- smtp
  - kirjavahetuse protokoll (simple mail transfer protocol)
- nntp
  - võrguuudiste edastamise protokoll (network news transfer protocol)
- finger
  - protokoll lühiinformatsiooni edastamiseks kasutaja või süsteemi kohta
- rpc
  - hajussüsteemide tööks vajalik kaugprotseduuri väljakutse protokoll (remote procedure call)
- snmp
  - võrgu teeninduseks vajalik protokoll (simple network management protocol)
- ntp
  - ajateenistusprotokoll (network time protocol)
- nfs
  - protokoll hajusa välismäluga töötamiseks (network file system)
- …

URL

- *Uniform Resource Locator*
- URL on ressursi identifitseerimiseks Internetis e. Internetiaadress. Igale dokumendile või muule ressursile Internetis vastab oma unikaalne internetiaadress

- Internetiaadressi esimene osa näitab ära kasutatava protokolli (näit. HTTP), sellele järgneb domeeninimi, alamkataloogi nimi ja failinimi
  - http://www.oracle.com/technetwork/java/index.html
  - ftp://ftp.info.apple.com/pub/
  - telnet://utoopia.pol.edu
  - ftp://mp3:mp3@138.247.121.61:21000/c3a/stuff/mp4/

Veebist - klass **`java.net.URL`**

```
String aadress =
           "https://courses.cs.ut.ee/2019/OOP/spring/";

InputStream sisse =  new URL(aadress).openStream();

OutputStream välja = new FileOutputStream("uus.html");
```

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/net/URL.html

Klass **`java.net.URL`**

- Klass **`java.net.URL`** sisaldab meetodeid:
  - uue URL loomiseks
  - URL-ist osade eraldamiseks
  - sisendvoo saamiseks URL-ist (andmete lugemine serverist)
- Serverist sisu saamiseks *Java* objektina
- Toetab protokolle:
  - http, https, ftp, file, …

Klass **`java.net.URL`**

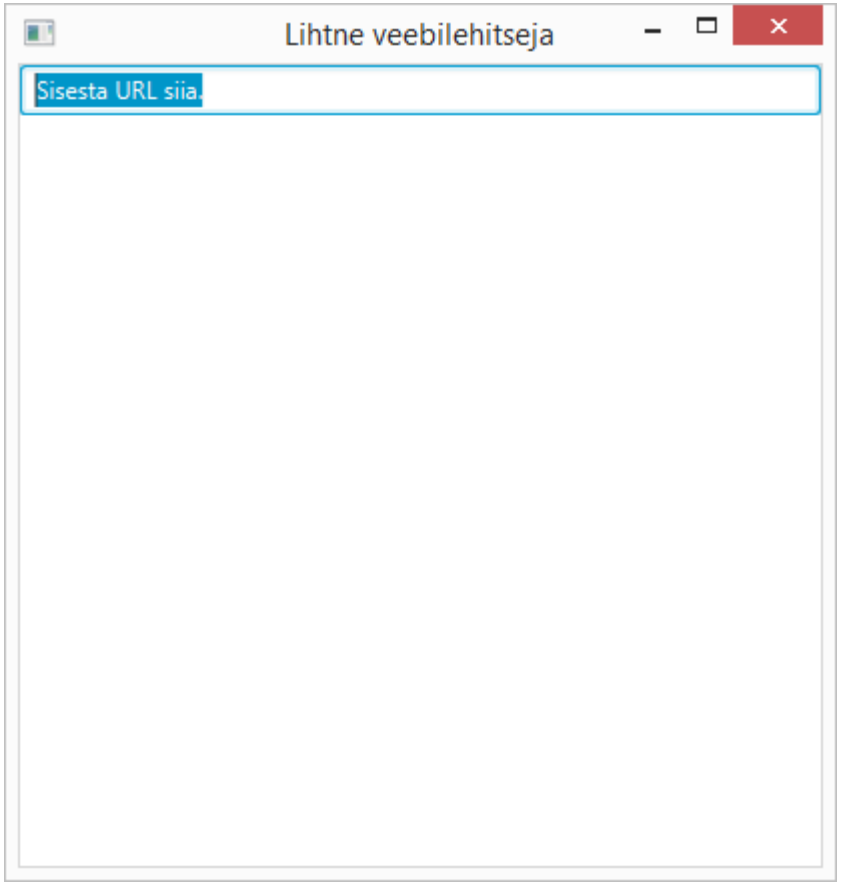
- Konstruktoreid:
  - **`URL(String spec)`**
  - **`URL(String protocol, String host, int port, String file)`**
  - **`URL(String protocol, String host, String file)`**
- Meetodeid:
  - **`public String getProtocol()`**
  - **`public String getHost()`**
  - **`public int getPort()`**
  - **`public String getFile()`**

Veebilehe näitamine

```
WebView lehitseja = new WebView();
WebEngine veebimootor = lehitseja.getEngine();
veebimootor.load(aadress.getText());
```

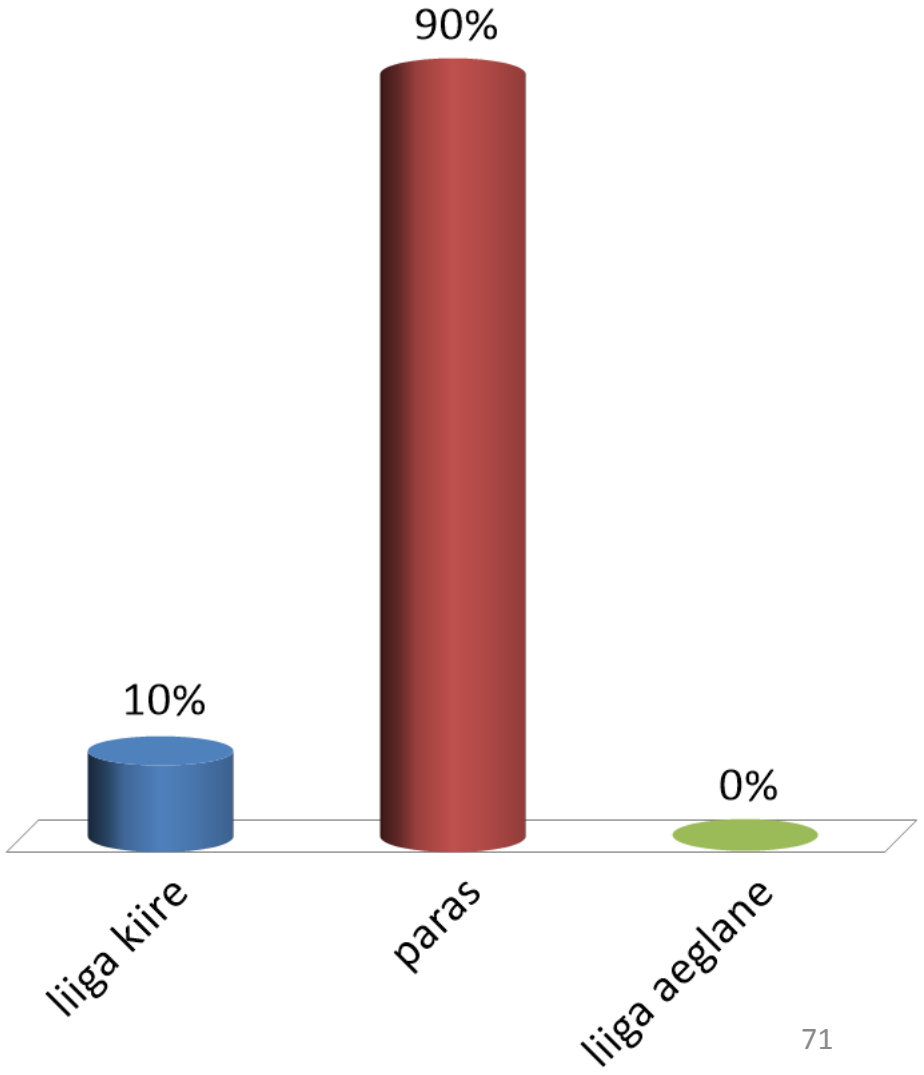
https://openjfx.io/javadoc/11/javafx.web/javafx/scene/web/WebView.html
https://openjfx.io/javadoc/11/javafx.web/javafx/scene/web/WebEngine.html

```
public void start(Stage lava) {
    BorderPane juur = new BorderPane();
    Scene scene = new Scene(juur,400,400);
    TextField aadress = new TextField("Sisesta URL siia.");
    WebView lehitseja = new WebView();
    WebEngine veebimootor = lehitseja.getEngine();

    aadress.setOnKeyPressed(new EventHandler<KeyEvent>() {
        @Override
        public void handle(KeyEvent event) {
            if(event.getCode().equals(KeyCode.ENTER)) {
                veebimootor.load(aadress.getText());
            }
        }
    });

    juur.setTop(aadress);
    juur.setCenter(lehitseja);
    lava.setTitle("Lihtne veebilehitseja");
    lava.setScene(scene);
    lava.show();
}
```


Lihtne veebilehitseja
Sisesta URL siia.

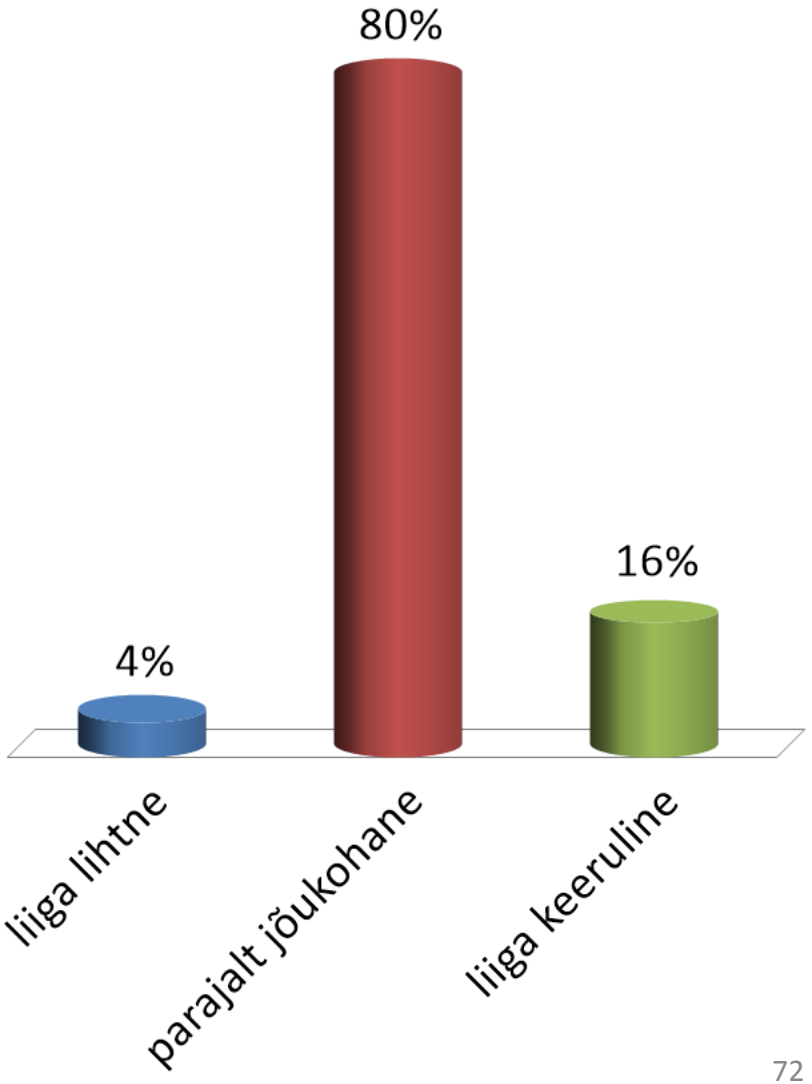
Loengu tempo oli

1. liiga kiire
2. paras
3. liiga aeglane



Materjal tundus

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



Suur tänu osalemast!
Kohtumiseni!