Praktikum 3

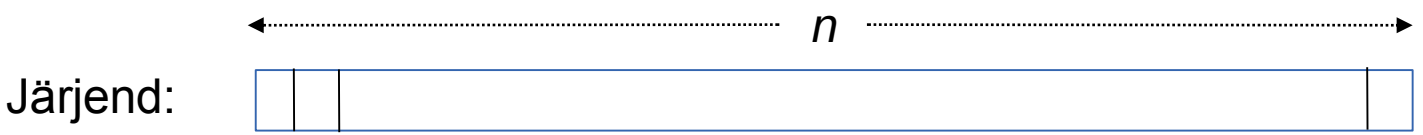
Kombinatoorika. Rekursioon (I)

Kombinatsioonide mitterekursiivselt leidmise idee
Kombinatsioonide rekursiivselt leidmise idee
Permutatsioonide rekursiivselt leidmise idee
Variatsioonid
Generaatorid (Java)
Kombinatoorsete valimite genereerimine
Indeksite kombinatsioonide generaator (Java)
Ennikute generaator (Java)

Harjutusülesanded

- **KMB_0.0. Algarvukaksikute loendamine**
- **KMB_0.1. Algarvukaksikutest tühja lõigu otsimine**
- **KMB_1. Kombinatsioonid mitterekursiivselt**
- **KMB_2. Permutatsioonid rekursiivselt**
- **KMB_3. Trepist üles (rekursiivne meetod)**
- **KMB_4. Täisruudud arvude 12345678 ... 87654321 seas (permutatsioonid)**

Kombinatsioonide mitterekursiivselt leidmise idee



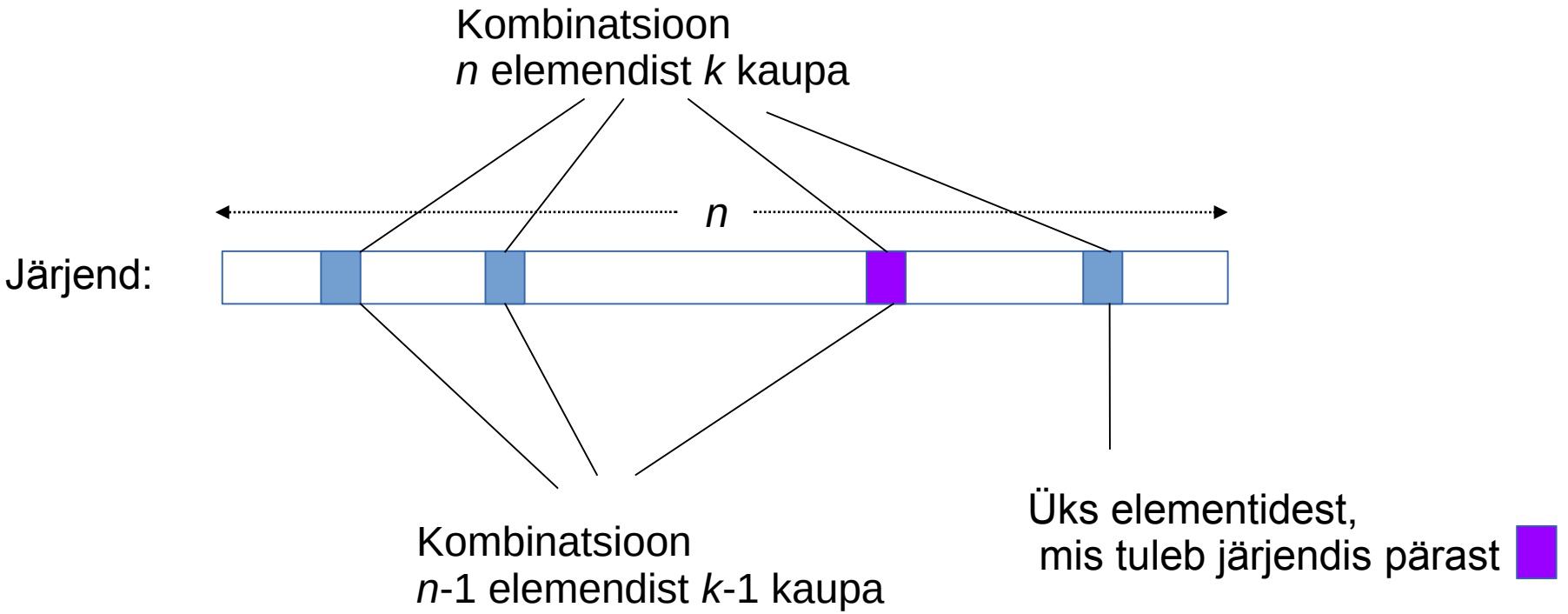
Iga kombinatsioon *n*-elemendilistest järjendist *k* kaupa on määratud järjendi ühe maskiga. Maskideks on bitijärjendid pikkusega *n*, milles igas *k* ühte. Näiteks maskid *n* = 5 ja *k* = 3 korral:

00111
01011
01101
01110
10011
10101
10110
11001
11010
11100

A B C D E	järjend
1 0 1 1 0	mask
A C D	maskile vastav kombinatsioon (5-st 3 kaupa)

Kõikvõimalikud bitijärjendid pikkusega *n* on arvude
0, 1, 2, ..., 2^{n}-1
kahendkujude *n* viimast numbrit; n = 5 korral siis 00000, 00001, 00010, ..., 11111.

Kombinatsioonide rekursiivselt leidmise idee



See kombinatsioon
7 elemendist *3* kaupa (B C E):

A B C D E F G H

annab

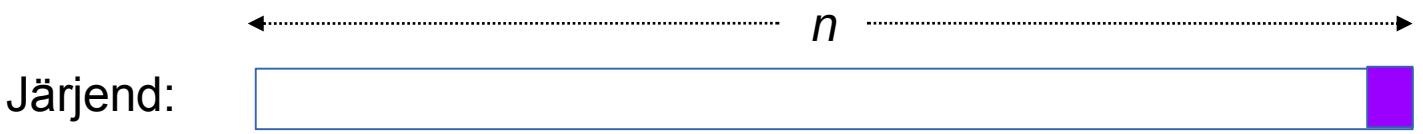
3 kombinatsiooni
8 elemendist *4* kaupa:

B C E F

B C E G

B C E H

Permutatsioonide rekursiivselt leidmise idee



Iga permutatsioon *n* elemendist on saadav sel teel, et mingile permutatsioonile *n*-1 (esimesest) elemendist on mingisse positsiooni lisatud järjendi viimane element.
Näiteks 6-liikmelise järjendi

A B C D E F

esimese 5 liikme üks permutatsioone on B A E D C

Sellest saame järgmised antud 6-liikmelise järjendi permutatsioonid:

F B A E D C
B F A E D C
B A F E D C
B A E F D C
B A E D F C
B A E D C F

Variatsioonid

Variatsioonideks järjendi elementidest *k* kaupa on selle järjendi kõigi *k*-kaupa kombinatsioonidest saadavad permutatsioonid.

Generaatorid (Java)

Üldiselt esitatakse generaator klassina, milles on kindlasti isendimeetodid *next* ja *hasNext* [*Ülesannete kogu*, lk 118, 119].

Käesolevas vaadeldavate generaatorite korral on meetodi *next* poolt välja antav objekt alati viidatüüpi (nt massiiv, sõne vms). Niisugusel puhul võib loobuda meetodist *hasNext*: meetodist *next* antakse välja kas viit järjekordsele genereeritud objektile või, ammendatuse signaalina, tühiviit **`null`**.

Näide (algarvude kaksikute generaator) – vt programm *Gen_Algarvukaksikud.java.*

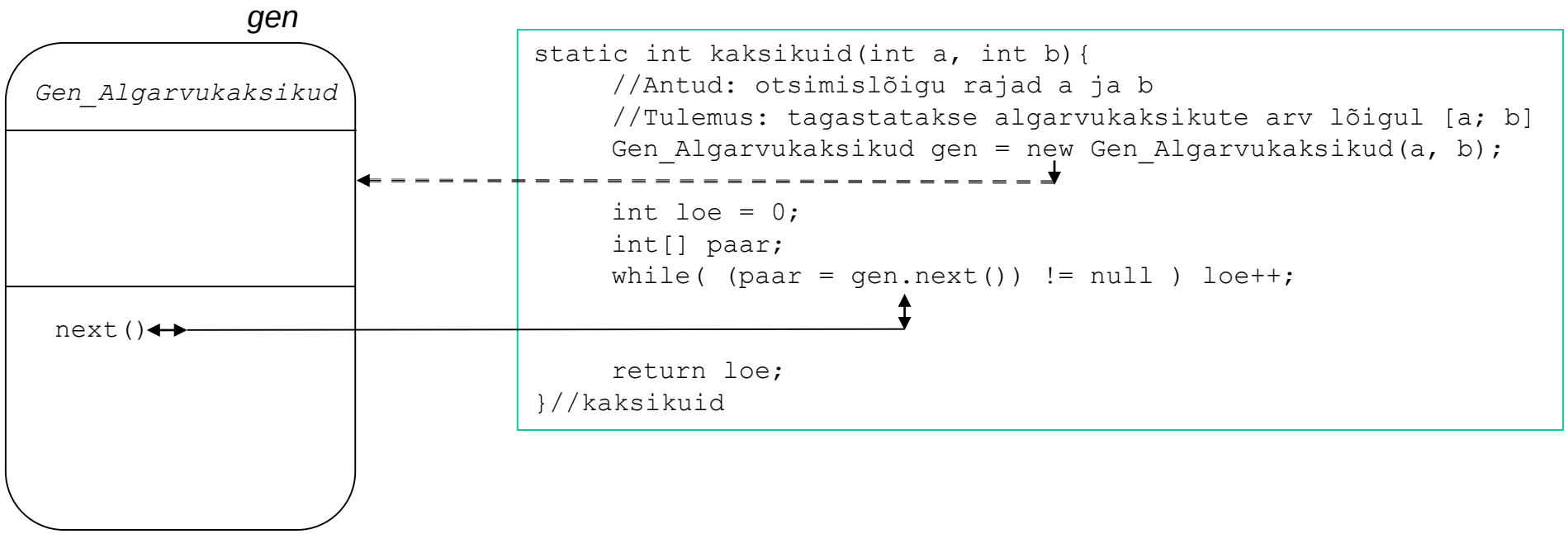
Selle spetsifikatsioon:

```
* Konstruktori sisendiks on otsimislõigu rajad a ja b.
*
* Meetod next()
*    annab välja järjekordse algarvukaksikute paari lõigul [a; b] kahe-elemendilise massiivina;
*    ammendatuse tunnusena tagastatakse null.
*
```

Näide

Generaatori *Gen_Algarvukaksikud.java* rakendamine

Generaatorit *Gen_Algarvukaksikud* kasutav meetod



Meetodi poolt loodud
generaatori klassi isend

Kombinatoorsete valimite genereerimine

Olgu lähtehulgaks *n*-elemendiline indekseeritud andmekogum (massiiv, list, sõne vmt). Sellest kombinatoorsete valimite (kombinatsioonide, permutatsioonide, variatsioonide) genereerimiseks on otstarbekohane kasutada indeksite (0, 1, …, *n*-1) komplektide generaatorit. Viimane ei sõltu lähtehulga ega selle elementide tüübist.

Kombinatsioonide *n* elemendist *k* kaupa leidmiseks sobib generaator, mis annab välja indeksite (0, 1, …, *n*-1) kombinatsioone *k* kaupa. Näiteks n= 5, k = 3 korral genereeritakse komplektid {2, 3, 4} {1, 3, 4} {1, 2, 4} {1, 2, 3} {0, 3, 4} {0, 2, 4} {0, 2, 3} {0, 1, 4} {0, 1, 3} {0, 1, 2}. Kui komplekti tüübiks (Javas) on *int*[ ], siis järjekordsele komplektile *kompl* vastavaks kombinatsiooniks (valimiks) lähtehulga *L* elementidest on

L[*kompl*[0]], *L*[*kompl*[1]], *L*[*kompl*[2]] – kui *L* on massiiv
ja
L.get(*kompl*[0]), *L*.get(*kompl*[1]), *L*.get(*kompl*[2]) – kui *L* on *ArrayList*
ja
L.charAt(*kompl*[0]), *L*.charAt(*kompl*[1]), *L*.charAt(*kompl*[2]) – kui *L* on sõne (*String*).

Permutatsioonide *n* elemendist leidmiseks sobib generaator, mis annab välja indeksite (0, 1, …, *n*-1) permutatsioone.

Indeksite kombinatsioonide generaator (Java)

Antud juhul on generaatoriks klass, mille isendi konstrueerimisel antakse parameetritena arv *n* – mitmest indeksist, ja arv *k* – mitme kaupa genereerida.
Klass sisaldab isendimeetodit *next*() tagastustüübiga *int*[ ], mille järjekordsel rakendamisel antakse välja järjekordne indeksite komplekt – üks kombinatsioon *k*-kaupa arvudest 0, 1, … *n*-1 (või tagastatakse *null*, kui kombinatsioone rohkem ei ole).

Klass: *Gen_Komb.java*.

Ennikute generaator (Java)

Antud juhul on generaatoriks klass, mille isendi konstrueerimisel antakse parameetrina ette järkude arv *n* >= 1, järkude maksimumid *maks*[0], *maks*[1], ... , *maks*[*m*] (*m* >= *n-1*), ning tunnus *kordusedLubatud*.
Klass sisaldab isendimeetodit *next*() tagastustüübiga *ArrayList<Integer>,* mille järjekordsel rakendamisel antakse välja ennik – *n*-kohaline järjend, mille *i*-ndas järgus (kohas, elemendis) on arv lõigult [0; *maks*[*i*]], *i* = 0, 1, ..., *n*-1 (või tagastatakse *null*, kui ennikuid rohkem ei ole). Kui kordused ei ole lubatud (tunnus *kordusedLubatud* on *väär*), siis korduvaid elemente sisaldavaid järjendeid välja ei anta.

Klass: *Gen_Ennikud.java*.

Näide: n = 3, *maks*[0]=3, *maks*[1]=1, *maks*[2]=2.

Ennikud kordustega:

```
[0, 0, 0]
[0, 0, 1]
[0, 0, 2]
[0, 1, 0]
[0, 1, 1]
[0, 1, 2]
[1, 0, 0]
[1, 0, 1]
[1, 0, 2]
[1, 1, 0]
[1, 1, 1]
[1, 1, 2]
[2, 0, 0]
[2, 0, 1]
[2, 0, 2]
[2, 1, 0]
[2, 1, 1]
[2, 1, 2]
[3, 0, 0]
[3, 0, 1]
[3, 0, 2]
[3, 1, 0]
[3, 1, 1]
[3, 1, 2]
```

Ennikud kordusteta:

```
[0, 1, 2]
[1, 0, 2]
[2, 0, 1]
[2, 1, 0]
[3, 0, 1]
[3, 0, 2]
[3, 1, 0]
[3, 1, 2]
```

Harjutusülesanded

KMB_0.0. Algarvukaksikute loendamine

Kirjutada meetod loendamaks, mitu algarvukaksikut leidub etteantud naturaalarvude lõigul.
Meetodis tuleb rakendada generaatorit *Gen_Algarvukaksikud.*
Koostada ka vastav testimisotstarbeline peameetod.

KMB_0.1. Algarvukaksikutest tühja lõigu otsimine

Kirjutada ja testida programm, et leida selline naturaalarvude lõik antud pikkusega *p*,
millel ei esine ühtegi algarvukaksikut. Programmi alguses fikseeritakse veel esimese lõigu algus.
Programmis tuleb rakendada generaatorit *Gen_Algarvukaksikud.*

Programmi väljundi näiteid:

```
Otsimise algus: 10
Kontrollitava lõigu pikkus: 10
Lõigul [20; 29] ei esine algarvukaksikuid.

Otsimise algus: 100000
Kontrollitava lõigu pikkus: 1000
Lõigul [2498000; 2498999] ei esine algarvukaksikuid.
```

KMB_1. Kombinatsioonid mitterekursiivselt
Programmeerida mitterekursiivne generaator-klass indeksite hulgast {0, 1, 2, ... , *n*-1}
k kaupa kombinatsioonide leidmiseks.
Programm: *Gen_Komb.java*

Java
Ühtede arv täisarvu *m* kahendkujus:
```
Integer.bitCount(m)
Long.bitCount(m)
```

Kui *m* on indeksite järjendi {0, 1, 2, ... , *n*-1} üks maskidest, siis indeks *i* kuulub selle maskiga määratud valikusse, kui
Java avaldis
```
((m>>(n-1-i)) % 2 == 1)
```
on tõene.
Järjend {0, 1, 2, ... , *n*-1} on virtuaalne, seda ei tule konstrueerida.

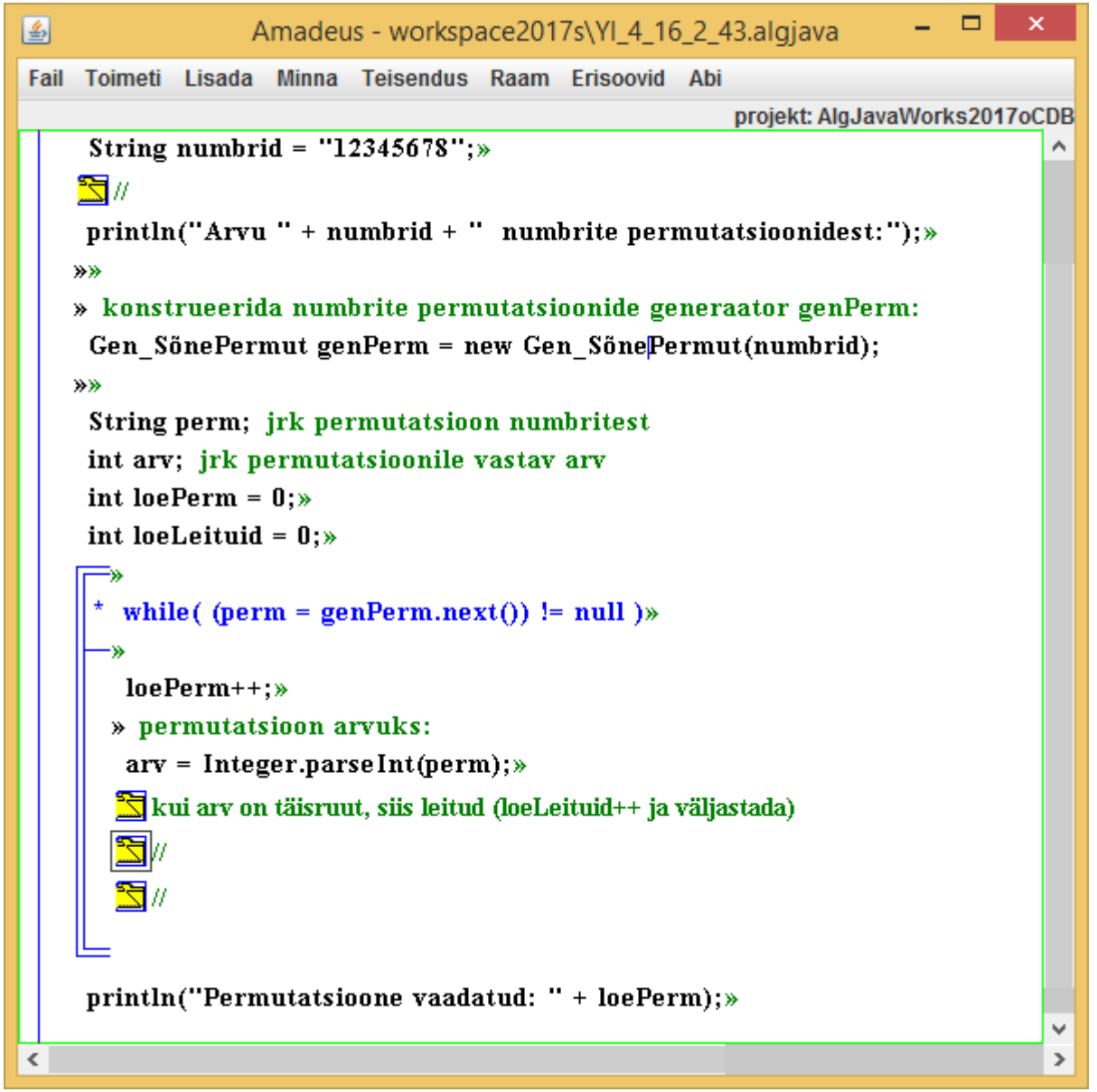
KMB_2. Permutatsioonid rekursiivselt

KMB_2.1 Programmeerida rekursiivne generaator-klass sõne sümbolite permutatsioonide leidmiseks.(Programm: *Gen_SõnePermut.java.*)

KMB_2.2 Kirjutada ja testida programm sõne sümbolite permutatsioonide leidmiseks, rakendades generaatorit *Gen_Ennikud.*

KMB_3. Trepist üles (rekursiivne meetod)
[*Ülesannete kogu*, ül nr 4.12].

KMB_4. Täisruudud arvude 12345678 ... 87654321 seas (permutatsioonid)
[*Ülesannete kogu*, ül nr 4.16].
Täisruudud arvude seas, mis on saadud arvu 1234567**8** numbrite permuteerimisel.



Märkus.
Täisruute 123456789 permutatsioonide seas on 30.