AKT 1. vahetest 13. märtsil 2017

12:15-13:45

0. Prelüüd

Ülesannete lahendamiseks vajalikud abi- ja lähteklassid ning mõned JUnit testid on kursuse avalikus repositooriumis (https://bitbucket.org/plas/akt2017/src), Gradle'i projektis nimega „kt1”. Impordi see projekt enda IDE-sse.

1. Regulaaravaldised (5p)

Avaliku repositooriumi *kt1* projektis, kaustas *src/main/java/kt1* on klass `AdjectiveExtractor`, mis tuleb täiendada ja esitada. Vaja on implementeerida meetod `extractAdjectives`, mille sisendiks on sõne. See sõne võib koosneda mitmest lausest ning sellest on vaja regulaaravaldisi kasutades tuvastada omadussõnad mis on omistatud teistele sõnadele. Tulemuseks on vaja koostada `Map<String,Set<String>>`, mille võtmed on sõnad ning väärtused on omadussõnad, mis on sellele sõnale omistatud. Sõnad ja omadussõnad koosnevad ladina suur- ja väiketähtedest (näiteks *koer*, *Lotte*, *AKT*, *aKt*, ...).

Omistamine käib kujul:

- Lihtomistamine - „A on X” - Suvalisele sõnale järgneb sõna „on”, millele järgneb suvaline teine sõna.
- Ja-omistamine - „A on X ja Y” - Sarnane eelmisele, kuid omistatakse kaks omadussõna korraga.
- Komadega omistamine - „A on X, Y, Z ja W” - Siin on omadussõnade järjend, mis on komadega eraldatud ning mille viimane eraldaja on koma asemel sõna „ja”.

Tekstis võib esineda samale sõnale mitu omistamist, mispuhul peaks tagastama mõlema omistamise omadussõnad.

Näide sisendist:

> Näidislause kus esineb muid sõnu, aga Juhan on tubli, tark ja arukas, ning Aadu on tore ja unine. Lisaks mainin, et Juhan on topelt.

Tulemus peaks olema

> Juhan → {tubli, tark, arukas, topelt}
>
> Aadu → {tore, unine}

Võib eeldada, et sisendis on tühikud ja kirjavähemärgid õigesti kasutatud, s.t. sõnade vahel on ainult üks tühik ja koma ees ei ole tühikut, aga koma järel on alati tühik.

2. Automaatide koostamine (5p)

Koosta JFLAP-is järgenevatele kirjeldustele vastavad lõplikud automaadid ja lae failid Moodle'isse. **NB! Kasuta failinimesid kujul *yl<ülesande nr>.jff* nt. *yl0.jff, yl4.jff*.**

0. Sõnad üle tähestiku {a,b}, mis sisaldavad alamsõne "bb".
1. Sõnad üle tähestiku {a,b}, mis ei sisalda alamsõne "bb".
2. Sõnad üle tähestiku {a,b}, mis sobituvad regulaaravaldisega "(a*b)*".
3. Sõnad üle tähestiku {a,b}, mis sobituvad regulaaravaldisega "(a*b)*" või ei sisalda alamsõne "bb". (Automaatide yl1 ja yl2 keelte ühend.)
4. Sõnad üle tähestiku {a,b}, mis sobituvad regulaaravaldisega "(a*b)*" ja sisaldavad alamsõne "bb". (Automaatide yl0 ja yl2 keelte ühisosa)

(viimane ülesanne on järgmisel lehel)

3. Puu läbimine (5p)

Avaliku repositooriumi *kt1* projektis, kaustas *src/main/java/kt1* on klass `RegexAnalyzer`, mis tuleb täiendada ja esitada. Vaja on implementeerida meetod `canUsePlus`, mis võtab argumendiks regulaaravaldise süntaksipuu ja tagastab `true` või `false` vastavalt sellele, kas antud regulaaravaldise või mõne selle alamavaldise saaks kergesti ümber kirjutada +-operaatoriga.

Täpsemalt, meetod peab tuvastama, kas antud regulaaravaldis või mõni selle alamavaldis vastab kujule `XX*` või `X*X` (mis oleksid ekvivalentsed kirjaviisiga `X+`). `X` tähistab siin suvalist alamavaldist.

Mõned näited:

- `a*a` puhul tuleb tagastada `true`, aga `a*b` puhul `false`.
- `(abba)(abba)*` puhul tuleb tagastada `true`, aga `(abba)(akka)*` puhul `false`.
- `abc|a*a|def` puhul tuleb tagastada `true`, samuti `abc(aa*)def` puhul.

NB! Meid huvitab ainult kuju `X*X` või `XX*` esinemine, mitte see, kas mingite muude struktuuriteisenduste tõttu saaks sisse tuua +-operaatori. Näiteks avaldise `(aa)a*` puhul tuleb tagastada false hoolimata sellest, et seda saaks kirjutada ümber + abil, sest see avaldis ei vasta nõutud kujule.

Regulaaravaldise süntaksipuu on moodustatud paketis `regex` oleva `RegexNode` klassi alamklasside isenditest. Konkreetsemalt, huvipakkuvad klassid on:

- `Alternation` – valik kahe alternatiivi (`getLeft()` ja `getRight()`) vahel
- `Concatenation` – kahe regulaaravaldise (`getLeft()` ja `getRight()`) konkatenatsioon
- `Repetion` – regulaaravaldise (`getChild()`) kordamine 0 või enam korda
- `Letter` – kindla tähega (`getSymbol()`) sobituv regulaaravaldis
- `Epsilon` – tühja sõne tähistav regulaaravaldis

Nende klasside kasutamine näeks välja midagi sellist:

```
void teeMidagiRegulaaravaldisega(RegexNode node) {
      if (node instanceof Alternation) {
            // see tipp tähistab valikut
            Alternation altNode = (Alternation)node;
            teeMidagi(altNode.getLeft());
            teeMidagi(altNode.getRight());
      } else if (node instanceof Repetition) {
            ...
      }
      ...
       // kui node'i tüüp ei ole oluline:
       for (child : node.getChildren()) teeMidagi(child);
}
```

NB! Etteantud projektis (kaustas *src/test/java* on mõned testid, aga **soovitame lahenduse kontrollimiseks sinna ise teste juurde lisada.**