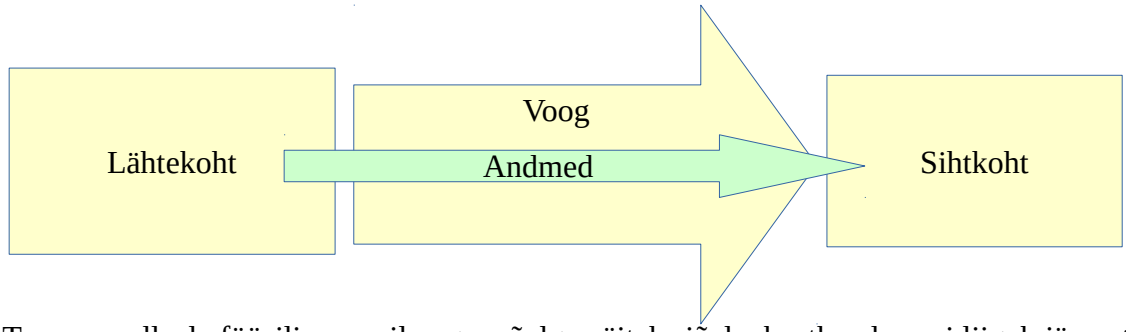
IDK0051 8. nädala praktikum

Sügis 2015

Voog (stream)

Voog on kontseptsioonilt andmete liikumise kanal. Voog ei salvesta andmeid, ta edastab neid.



Tuues paralleele füüsilise maailmaga mõelge näiteks jõele, kustkaudu vesi liigub järvest merre. Lähtekohaks on järv ja sihtkohaks on meri. Jõgi (täpsemalt jõe säng) on aga voog, mida mööda vesi merre voolab. Jõgi ise ei salvesta vett. Samamoodi ei salvesta ka voog andmeid.

1. Uurige ja mõistke voo (streami) olemust kõige lihtsama süsteemse voo System.out peal. System.out on *PrintStream*-tüüpi voog, mis on disainitud andmete liikumiseks väljundisse. Sellel vool on meetodid, mida saab välja kutsuda, nt olete kõige rohkem kasutanud println() meetodit.

PrintStream dokumentatsioon:
http://docs.oracle.com/javase/8/docs/api/java/io/PrintStream.html

Muutke System.out voo tüüp ära nii, et väljundit ei trükita mitte konsooli, vaid hoopis faili. Selleks looge programmikoodis uus voo objekt. PrintStream asemel kasutage FileOutputStream, nt nii:

```
FileOutputStream out = new FileOutputStream("C:\\out.txt");
või
FileOutputStream out = new FileOutputStream("resources/out.txt");
```

Meetodi voo muutmiseks leiate Systemi dokumentatsioonist:

http://docs.oracle.com/javase/8/docs/api/java/lang/System.html

NB! Sellele meetodile ei saa otse argumendiks anda FileOutputStream-tüüpi voogu, vaid te peate looma uue PrintStream voo, mille initsialiseerite FileOutputStream-tüüpi vooga. See tähendab, et teil on üks voog teise sees – PrintStream kasutab sisemas FileOutputStreami.

Kasutaja jaoks ei muutu aga midagi – peale väljundvoo muutmist kutsutakse meetodit ikka samamoodi välja, kuid tulemus kirjutatakse teise kohta. Testige, et System.out.println väärtus tekiks nüüd teie määratud faili.

2. Lisaks System.out-ile on veel 2 olulist süsteemset voogu: System.in ja System.err. Esimest neist kasutatakse sisendi tarvis (vaikimisi klaviatuurilt) ja teist vigade väljutamiseks.

Võrrelge, mida teeb System.err.println() erinevalt vaikeseadistustes System.out'iga

3. Konfigureerige nüüd oma programm nii, et System.out töötaks vaikimisi vooga, kuid System.err kirjutaks faili.

4. Muutke failikirjutamise käitumist nii, et igakordsel faili lisamisel ei kirjutata faili üle, vaid uus tekst lisatakse faili lõppu. Selleks muutke ühe juba kasutatud klassi konstruktori väljakutset – uurige Java dokumentatsioonist, millise.

Dependency injection

Sõltuvuse sisestamine ehk *dependency injection* on viis anda tarkvaramoodulile ette osa, mida ta peaks oma tegevuses kasutama. Erinevalt tavalisest argumendist eeldab sõltuvuse sisestamine, et sisestatakse mingi keerukam objekt, mille konkreetne implementatsioon võib varieeruda.

Tuues paralleeli autoga – auto vajab töötamiseks mootorit (sõltuvus), kuid see mootor võib sama automudeli korral olla erinevat tüüpi (nt diisel- või bensiinimootor). Auto ise ei otsusta, millist mootorit ta kasutab, see mootor sisestatakse talle tehases. Samuti on ka sõltuvuse sisetamisega objektidesse – objekt teab, et tal on mingit liiki sõltuvus, kuid see, milline alamtüüp või realisatsioon tal konkreetselt on, jäetakse objekti looja või kasutaja otsustada.

Sõltuvuse saab sisestada kas konstruktoris või eraldi setteriga. Java raamistikes kasutatakse sõltuvuse sisestamiseks annotatsioone.

4. Looge TDD meetodil tudengi klass, kus on olemas meetod eksamitulemuse lisamiseks tudengile. Ainekood ja tulemus salvestatakse kollektsiooni.

5. Lisage sellele meetodile logimise funktsionaalsus, logimise koht sisestatakse sõltvusena objekti loomisel või setteriga, kas:

- faili
- konsooli

Kirjutage kood nii, et üks objekt oleks suuteline logima faili ja teine samal ajal konsooli. Logi tekst võiks olla: „Mari Maasikas sai aines IDK0051 hindeks 5” - kus nime, ainekoodi ja hinde asendate tegelike väärtustega.