TARTU ÜLIKOOL

Arvutiteaduse instituut

Informaatika õppekava

Kristen Surva

Praktiliste kodutööde loomine ainele “Automaadid, keeled ja translaatorid”

Bakalaureusetöö (9 EAP)

Juhendaja: Vesal Vojdani

Tartu 2023

Praktiliste kodutööde loomine ainele “Automaadid, keeled ja translaatorid”

Lühikokkuvõte:

See bakalaureusetöö kirjeldab alternatiivsete praktiliste kodutööde loomisprotsessi Tartu Ülikooli kursusele “Automaadid, keeled ja translaatorid”. Kodutööde laiem eesmärk on näidata, et kursusel käsitlevatel teemadel on ka kasutust väljaspool programmeerimiskeelte konteksti. Töö käigus loodi kaks kodutööd. Esimene kodutöö õpetab JSON parsimist kasutades Java GSON teeki. Teine kodutöö tutvustab ärireeglite mõistet ning kasutust läbi JsonLogicu ja Easy Rulesi. Mõlemad kodutööd on käesoleval semestril kasutusele võetud. Ainekorraldajate ning tudengite tagasiside põhjal täidavad loodud kodutööd oma eesmärki. Suurem osa kursusel osalenud tudengite arvates on alternatiivsete kodutööde pakkumine hea täiendus kursusele.

Võtmesõnad: õppematerjalide koostamine, JSON parsimine, ärireeglid

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria); S270 Pedagoogika ja didaktika

Creating practical assignments for the course “Automata, Languages and Compilers”

Abstract:

This bachelor thesis describes the process of creating alternative practical homework assignments for the University of Tartu course “Automata, Languages and Compilers”. The broader purpose of the assignments is to show that the topics covered in the course have uses outside of the context of creating programming languages. During the work, two homework assignments were created. The first assignment teaches JSON parsing using the Java GSON library. The second assignment introduces the concept and use of business rules through JsonLogic and Easy Rules. Both assignments have been introduced into the course this semester. Based on the feedback of the course organizers and students, the created assignments fulfill their purpose. According to the majority of students participating in the course, offering alternative assignments is a good addition to the course.

Keywords: creating learning materials, JSON parsing, business rules

CERCS: P170 Computer science, numerical analysis, systems, control; S270 Pedagogy and didactics

Sisukord

Sissejuhatus

Tartu Ülikooli informaatika õppekava kursuse “Automaadid, keeled ja translaatorid” üks peamiseid eesmärke on arendada sügavam arusaamine arvutiprogrammide ülesehitusest ja täitmisest [1]. Selline sügav mõttemudel programmide tähendusest on oluliseks eelduseks, et osata hästi programmeerida. Samas on sellise mõttemudeli loomine abstraktne ning pikaajaline eesmärk.

Paljudel kursustel käsitletavatel teemadel on reaalses elus laiem kasutus ja seda soovib aine korraldaja ka rõhutada. Aines küll käsitletakse mõningaid praktilisemaid teemasid, kuid muutuvas maailmas võiks nende teemade praktilisust rohkem demonstreerida. Näiteks regulaaravaldiste teema kodutöös on praktiline näide nimede ning telefoninumbrite sobitamisest, mis tudengite tagasiside järgi on selle aine kõige kasulikum kodutöö [2]. Sarnast praktilist kogemust tahab aine korraldaja pakkuda ka süntaksanalüüsi ja interpretaatori teemade kohta.

Eeltoodut silmas pidades, on selle töö eesmärk luua 2 alternatiivset praktilist kodutööd, mida AKT (Automaadid, keeled ja translaatorid) tudengid saavad huvi korral valida. Esimene kodutöö on süntaksanalüüsi alternatiiviks, ning tuvustab JSON formaati ning selle parsimist. Teine kodutöö on interpretaatori alternatiiviks ning tutvustab ärireeglite kasutust. Alternatiivsed kodutööd on laialdasemalt seotud juba olemasolevate kodutööde teemadega ning aitaksid demonstreerida ja seostada nende teemade praktilisemat poolt.

Peale kodutöö valmimist integreeriti need kodutööd kohe kursusesse. Tudengid, kes otsustasid lahendada neid kodutöid, said pärast lahendamist täita tagasiside küsitluse. Tagasiside küsitluse vastuste põhjal analüüsitakse kodutööde edukust.

Selle töö esimeses peatükis antakse ülevaade kursusest endast ja põhjendatakse vajadust koostada praktilisemaid kodutöid. Teises peatükis põhjendatakse uute teemade valikut ning õpetatavate tehnoloogiate valikut. Kolmandas peatükis kirjeldatakse materjalide koostamise põhimõtteid ning metoodikat. Neljandas peatükis analüüsitakse kodutööde edukust tagasiside küsitluse põhjal.

Esimest kodutööd kutsun edaspidi JSON kodutööks ja teist kodutööd ärireeglite kodutööks.

1. Kursuse ülevaade ja töö motivatsioon

Tartu Ülikooli kursus “Automaadid, keeled ja translaatorid LTAT.03.006” (edaspidi AKT) toimub igal kevadsemestril ning on mõeldud teise õppeaasta Informaatika tudengitele. Kursus käsitleb peamiselt kompileerimisega seotud teemasid nagu regulaaravaldised, kontekstivabad grammatikad, süntaksipuud ja koodi genereerimine. Läbi nende teemade annab kursus tudengitele võimaluse süvendada oma arusaamist arvutiprogrammide ülesehitusest ja täitmisest. See arusaamine omakorda aitab programmeerijal usaldusväärsemat tarkvara luua. Kursuse põhieesmärk ongi ennetada halvast programmeerimisest tekitatud kahju ühiskonnale [1].

Selle abstraktse eesmärgi kõrval, tahavad aine korraldajad demonstreerida teemade praktilisemaid kasutuskohti. Näiteks regulaaravaldiste teema kodutöös on praktiline ülesanne nimede ning vastavate telefoninumbrite sobitamisest Java regex APIga. Aine korraldajad väidavad, et tudengite meelest on see kodutöö kõige meeldivam kogu aines, sest seostab teoreetilist praktilistega [2]. Sarnast praktilist kogemust tahavad aine korraldajad pakkuda ka süntaksanalüüsi ja interpretaatori teemade kohta.

Lisaks aine praktilisele poolele on ka süntaksanalüüsi teema juures mure, et tutvustatakse vaid üldist lahendust ja töövahendeid grammatikatest parserite loomiseks, aga käsitlemata jäävad meetodid konkreetsete laialt levinud failiformaatide töötlemiseks nagu näiteks JSON (JavaScript Object Notation). Sellega kaasneb see oht, et õpitakse asju leiutama ainult nullist, selle asemel, et kasutada paremaid ja töökindlamaid lahendusi.

Kursuse interpretaatori teema puhul on eesmärk näidata, et interpreteerimine on informaatikas laiem mõiste kui ainult programmeerimiskeelte interpreteerimine. Interpretaatorite laialdasema kasutuse demonstreerimiseks, soovitaksegi aines tutvustada juurde ka ärireegleid, sest see täiendab omal moel interpreteerimise teemat, mille kõik olemasolevaid näited on seotud programmeerimiskeelte avaldiste väärtustamisega.

AKT kursus annab arusaama programmide ülesehitusest ja täitmisest. Põhiprogramm keskendub üldise programmeerimisoskuse arendamisele, aga jääb puudu konkreetsete probleemide lahendamise võtteid, mis oleks rakendatavad väljaspool programmeerimiskeelte realiseerimist. Idee on pakkuda valikuna praktilisemaid kodutöid, mis aitaksid tudengitel paremini toime tulla ka reaalelus ettetulevate programmeerimise probleemidega. Lisaks võib

valikuvõimalus tõsta motivatsiooni õppimise vastu, sest lubab tudengitel valida kodutöö, mis neile kõige rohkem huvi pakub.

2. Teemade valiku ja õpetatavate tehnoloogiate põhjendus

Selles peatükis antakse ülevaade sel kursusel õpetatavate teemade valikust ning ka täpsemalt tehnoloogiatest, mida uutes kodutöödes kasutati.

Nagu ka eelnevalt mainitud, siis motivatsioon oli luua süntaksanalüüsi ja interpretaatori teemade kohta sarnased praktilised ülesanded nagu on regulaaravaldiste kodutöös Java regex APIga. Praktilisuse all on mõeldud seda, et õpitu oleks miski, mis ka väljaspool programmeerimiskeelte loomise konteksti kasutusel oleks. Selliste ülesannete loomiseks on tarvilik enne uurida, mida nendes süntaksanalüüsi ja interpretaatori teemades juba kursusel käsitletakse.

2.1 Süntaksanalüüs

AKT aines algab süntaksanalüüsi teema nii lekseri kui parseri käsitsi kirjutamise kodutöödega. Kursusel on viies kodutöö käsitsi lekseri kirjutamine ja kuues kodutöö käsitsi parseri kirjutamine. Mõlemas kodutöös luuakse õppe eesmärgil kursuse jaoks loodud lihtsustatud programmeerimiskeelele AKTK (AKT keel) vastavalt lekser ja parser. Edasi õpitakse selle teema all kursusel kasutama ANTLRit (ANother Tool for Language Recognition). ANTLR võimaldab keele grammatika defineerimise kaudu genereerida nii parseri kui ka lekseri. See lihtsustab oluliselt keele loomist, sest lekseri ja parseri loomise asemel peab lihtsalt keskenduma sellele, et keele grammatika oleks korrektne.

Lekseri ja parseri teemad on omavahel tihedalt seotud, sest moodustavad ühe translaatori ehk kompilaatori kaks esimest osa. Tihti kutsutakse seda esiosa koos tüübikontrolliga (laiemalt semantilise analüüsiga) kompilaatori *front-endiks* [3]. Lekser ehk leksiline analüsaator võtab sisendiks programmi teksti ning teisendab selle lekseemideks (ingl *tokens*), mida siis parser saab kasutada süntaksipuude loomisel [3]. Parser võtab sisendiks lekseri poolt loodud lekseemide jada ning koostab vastavalt keele grammatika ja lekseemide põhjal programmi süntaksipuu. Süntaksipuu omakorda peegeldab programmi struktuuri [3].

Praktilise teemana sobib nende kodutööde alternatiiviks väga hästi JSON parsimine. See on loomulik teema, kus näidata juba olemasolevate töövahendite kasutust, sest JSON on väga populaarne formaat ja selle töötlemiseks on juba olemas palju töökindlaid ja testitud lahendusi. Lihtsalt töötava JSON parseri loomine on küll üpris lihtne, eriti ANTLRit kasutades, kuid selle parseri loomine töökindlaks ning turvaliseks on märksa keerulisem

ülesanne. Ka JSON formaadi ebaregulaarne dokumentatsioon teeb korraliku rakendamise veelgi keerulisemaks [4]. Kodutöö võiks olla eriti kasulik, kuna JSON parsimine on vägagi aktuaalne ja praktiline oskus. Seega keskendubki uus kodutöö juba olemasoleva JSON teegi õpetamisele.

JSON teeke on Javale mitmeid, aga antud töö jaoks kaaluti vaid Maven Repository andmetel kahte kõige populaarsemat - GSONi ja Jacksoni [5].

GSON on Google'i poolt loodud JSONi parsimise teek. GSONi loojate sõnul on teegi üheks peamiseks eesmärgiks olla lihtne. Lihtsus seisneb selles, et piisab kahest meetodist, *toJson* ja *fromJson*, et Java objekte serialiseerida JSONiks ja JSONit deserialiseerida Java objektideks. Meetodid töötavad ka juba olemasolevatel objektidel, mille lähtekoodile ei ole ligipääsu. GSONil on ainult üks teek [6].

Jackson on avatud lähtekoodiga projekt, mis on jagatud kolmeks peamiseks mooduliks: jackson-core, jackson-annotations ja jackson-databind. Jacksoni *ObjectMapper* klass on võrreldav GSONi lihtsusega, kuid on siiski veidi keerulisem kui GSONi *toJson* ja *fromJson* meetodid. Jacksonil on ka palju lisafunktsionaalsust võrreldes GSONiga. Leidub ka palju lisamooduleid, mis lubavad parsida teisi andmetüüpe peale JSONi nagu näiteks YAML, XML ja CSV [7].

Kuna JSON kodutöö eesmärk on anda tudengitele algteadmised JSONi serialiseerimise ja deserialiseerimise kohta, siis kõiksugused muud lisafunktsioonid, mida Jackson pakub, on ebavajalikud. Samuti võib segadust tekitada asjaolu, et Jackson on jagatud eri mooduliteks. Kõiki neid fakte arvesse võttes, on käesoleva töö raames valitud JSON parsimise õpetamiseks GSON.

Kokkuvõtvalt kursuse teema "Süntaksanalüüs" praktiliseks kodutööks sobib JSON parsimist õpetav kodutöö, sest see teema sobib täpselt kursuse süntaksanalüüsi teema alla. Samuti kasutades GSON teeki täidab see kodutöö algset eesmärki näidata juba valmisolevate lahenduste kasutamist improviseeritud lahenduste asemel.

2.2 Interpretaator

Interpretaatori õpetamise teema peamine eesmärk on kodutöö käigus luua kursuse õppekeelele AKTK interpretaator.

Peale lekseri ja parseri tööd, on tulemuseks programmi süntaksipuu. Programmi peab aga olema võimalik ka käivitada ja lihtsaim viis selleks on interpretaatori kasutamine. Interpretaator töötleb järk-järgult süntaksipuud, kuni kogu programm on täidetud [3].

Kursuse lisalugemise all demonstreeritakse ka interpretaatori üldisemat kasutamist lihtsa *interpreter pattern* ülesandega. Selle ülesande edasiarendamise võimalustest mõtlemisel jõuti ärireeglite teema juurde.

Ärireeglid on otsustusjuhiste kogum, mida kasutatakse süsteemi käitumise reguleerimiseks. Ärireeglite implementatsioon kaasab tavaliselt mingis vormis ka interpretaatorit ja seda just käskude täitmiseks. Selle tõttu sobibki interpretaatori kodutöö alternatiiviks ärireeglite temaatika. Lisaks annab ärireeglite tutvustamine tudengitele kogemuse domeenispetsiifliste keeltega, mis on informaatika tööstuses üsna tihti kasutusel. Ärireeglite demonstreerimiseks valiti siin JsonLogic ja Easy Rules. Järgnevalt nende tutvustus.

2.2.1 JsonLogic

JsonLogic on JSON formaadis ja väga piiratud funktsioonidega keel. JsonLogic põhineb reeglite defineerimisel, mida saab siis andmete sisestamisel kasutada, et otsuseid teha. JsonLogicu eelisteks on välja toodud tema napisõnalisus, järjepidevus, turvalisus ja paindlikkus [8].

Turvalisuse juures tuuakse välja, et ei kasutada *eval()* funktsiooni. *Eval* funktsiooni kasutamine on teadaolev turvarisk – seda funktsiooni ei tohiks kunagi kasutada [9]. Lisaks on reeglitel lugemisõigus ainult nendele andmetele, mis muutujatena sisestatakse ning reeglitel puudub kirjutamisõigus [8].

2.2.2 Easy Rules

Easy Rules on Javal põhinev lihtne “reeglimootor” või “reeglimasin”, mis on inspireeritud Martin Fowleri artiklist “Should I use a Rules Engine?”. Reeglimootor koosneb reeglitest, millel on tingimused ja tegevused/käsud (*actions*). Reeglimasinale saab sisestada “fakte”, mis on Easy Rulesi viis, kuidas muutuvaid andmeid reeglitele anda [10, 11].

Easy Rules lihtsus seisneb selles, et reeglimootor itereerib üle reeglite ning evalueerib need. Kui reeglid vastavad tingimustele, siis käivitab reeglimootor reeglile vastava käsu [10]. Reeglite enda koostamine on võrreldav tingimuslausete koostamisega.

Nii JsonLogic kui ka Easy Rules on sobilikud tehnoloogiad, mida kasutada ärireeglite õpetamiseks, kuna nad lubavad luua reegleid ja nendele vastavaid käske. Mõlemad tehnoloogiad on oma olemusest samuti väga lihtsad – nende kohta ei ole palju õpetada. Seega, et ärireeglite kodutöö oleks piisavalt mahukas, on ärireeglite kodutöös kasutusel nii JsonLogic kui ka Easy Rules.

3. Kodutööde koostamine

Õppematerjalide loomisel toetusin peamiselt Greg Wilsoni raamatus “Teaching Tech Together” välja toodud tagurpidi disaini (ingl *backward design*) printsiipidele [12]. Tagurpidi disain on mõeldud terve kursuse loomiseks, kuid osa sellest on võimalik ka kasutada individuaalsete kodutööde ja praktikumide loomisel. Tagurpidi disaini lihtsustatud sammud, mida kodutööde loomisel kasutati, on:

- *Learner Persona* loomine. Selles töös on analüüsi lihtsustamiseks see samm asendatud lihtsalt sihtgrupi analüüsimisega.
- Algsete visandite loomine, mis aitab ettekujutuse saada sellest, milliseid teemasid käsitleda ja kuidas neid teemasid käsitleda. Need on algsed mõtted enne kodutöö valmimist.
- Kokkuvõtva hindamise (ingl *summative assessment*) loomine. Aitab määratleda ülesande lõppeesmärki. Siin on käsitletud kodutöö eesmärgid oma lõppvormis.
- Kujundava hindamise (ingl *formative assessment*) ehk ülesannete loomine. Annab õppijatele võimaluse harjutada õpitut. Samuti aitab nii õppijatel kui õpetajal määratleda, kas nad edendavad õpitust arusaamist.

Edasi antakse lähemalt neist sammudest ülevaade.

3.1 Sihtgrupp

Sihtgrupi analüüsimine aitab õppematerjalide koostamisel mõista õppijate oskusi ja vajadusi. Kuigi käesolevas töös koostatud kodutööd on kaudselt seotud kursuse AKT teemadega, on need kodutööd siiski mõeldud olema alternatiivsed ning ei haaku otseselt aines õpetatuga. Seda enam on kasulik analüüsida AKT õpilaste omadusi, sest sihtgrupi tundmine aitab luua selle aine tudengitele asjakohased kodutööd.

Võib oletada, et enamik tudengitest, kes võtavad osa AKT ainest, on vähemalt teise aasta Tartu Ülikooli Informaatika eriala tudengid [13]. Teise aasta tudengid on läbinud mitmeid programmeerimisega seotud aineid, nagu “Programmeerimine”, “Objekt orienteeritud programmeerimine”, “Programmeerimine II”, “Algoritmid ja andmestruktuurid” ja “Sissejuhatus andmeteadusesse”. Olles läbinud mitmeid kursuseid võiksid AKT tudengid

juba olla üpris tuttavad peamiste keeltega, mida TÜ informaatika õppekavas õpetatakse – Java ja Python.

Teise aasta tudengid on läbinud ka aine "Andmebaasid" [14]. Aines andmebaasid õpetatakse SQL-i, mida võib klassifitseerida domeenispetsiifiline keeleks. Aine läbinul on vähemalt mingi arusaam sellest, et domeenispetsiifilised keeled on olemas ja reaalselt kasutusel. Seda on hea tähele panna ärireeglite kodutöö loomisel, sest mõlemaid käsitletud tehnoloogiatest (JsonLogic ja Easy Rules) võib samuti liigitada domeenspetsiifilisteks keelteks.

Võib oletada, et TÜ informaatika tudeng on peale lõpetamist huvitatud edasi tegelema IT-valdkonnas. Järelikult on õppijate eesmärk omandada praktilisi oskusi ja teadmisi, mis neid IT-valdkonnas aitab.

JSON on üks populaarsemaid andmevahetus formaate, mida kasutatakse. Teine üks kõige populaarseim andmevahetus formaat XML (Extensible Markup Language) on ajapikku populaarsust kaotanud. Seda on ka näha Maven teekide populaarsuse edetabelist, kus GSON on 13nes kõige populaarsem teek, ning samal ajal kõige populaarsem XML teek JAXB, on alles 82ndal kohal [5]. Sellest võime järeldada, et JSON formaadi õpetamine läheb kokku õppijate eesmärgiga omandada praktilisi oskusi.

Samuti on kasutusel ärireeglitel põhinevaid tehnoloogjaid nagu JsonLogic, mida kasutati Euroopa Liidu digitaalsete COVID-tõendite ärireeglite loomisel [15]. Ärireeglite ja valitud domeeni spetsiifiliste keelte õpetamine avardab silmaringi ja aitab õppijatel harjuda uute tehnoloogiate õppimisega.

3.2 Kodutööde visandite loomine

Selles peatükis kirjeldatakse kodutööde visandite loomist. Visandid toovad välja üldisemad punktid, mida kodutöös peaks käsitlema ja annavad kodutööle laiema eesmärgi. Laiema eesmärgi defineerimine teeb ka hiljem õpiväljundite defineerimise lihtsamaks.

3.2.1 JSON kodutöö visand

JSON kodutöö peamised eesmärgid on õpetada:

- Mis on JSON objekt.
- Kuidas JSON-it serialiseerida.
- Kuidas JSON-it deserialiseerida.

Täpsemalt on kodutöö eesmärk anda tudengitele ainult algsed teadmised JSON formaadist ja selle parsimisest, seega teemat ei peaks käsitlema liiga süvitsi. On oluline keskenduda JSON-i serialiseerimise ja deserialiseerimise õpetamisele ja hoiduda tudengite segadusse ajamist teemadega, mis on vähem olulised. Erijuhud ja teegi omapärad ei ole kindlasti esiplaanil.

3.2.2 Ärireeglite kodutöö visand

Ärireeglite kodutöö peamised eesmärgid:

- Selgitada, mis on ärireeglid.
- Selgitada, millal ärireeglid kasulikud on.
- Õpetada valitud teeke kasutama, et tudengid saaksid praktilise kogemuse ärireeglite kasutamisest.

Ärireeglite kodutöö keskendub rohkem konkreetsete tehnoloogiate õpetamisele ja vähem suurema pildi edasiandmisele. See tuleneb sellest, et erinevad ärireeglite põhimõtteid kasutatavad tehnoloogiad on väga erinevad üksteisest. JSON kodutöö puhul erinevad teegid töötlesid siiski sama formaati, kuid ärireeglite puhul sellist ühtset dokumentatsiooni ei leidu, sest tegemist on pigem laiema mõistega kui kindla formaadiga.

3.3 Kodutööde kokkuvõttev ja kujundav hindamine

Kokkuvõtva hindamise all on ära kirjeldatud täpsed õpiväljundid ning ülesande lõppvorm. Kujundav hindamine kirjeldab valminud ülesandeid kodutööde lõpus, mida tudengid peavad lahendama.

3.3.1 JSON kodutöö seletava osa kokkuvõte

Ülevaade kodutöös õpetatavast:

- Mis on JSON.
- Mis on GSON teek.
- Kuidas serialiseerida Java objekte JSON objektideks, ToJson meetodiga.
- Kuidas GSON parseri seadeid kohandada, kasutades GsonBuilder klassi.
- Kuidas deserialiseerida JSON objekte Java objektideks, FromJson meetodiga.
- Annotatsioonide kasutamine lähteklassis, et kohanda FromJson meetodi käitumist.

- Kuidas deserialiseerida individuaalseid JSON atribuute, JSON puud läbides.
- Mis on kirjetüübid (ingl *record*) Javas.

Selleks, et kodutöö maht liiga väike ei oleks, on kodutööle materjali juurde lisatud. Nimelt on kodutöös lisaks kajastatud GSON parseri seadete kohandamist, GSON annotatsioonide kasutamist ning Java *Record* andmetüüp. Parseri seadete kohandamine ja annotatsioonide kasutamine aitab õpilasel GSON teeki veidi paremini mõista. Java *Record* andmetüüp otsustati kodutöösse lisada, sest see on sobiv tüüp, mida muutumatute andmete puhul kasutada.

Kodutöö materjal alustab JSON formaadi tutvustusega. Tutvustuses on kirjeldatud, et JSON on JavaScripti objekt süntaksil põhinev ning väga tihti kasutatakse veebirakendustes. Sealjuures antakse ka ülevaade JSON andmetüüpidest ning tutvustatakse võtme-väärtuse paare, mis on JSON-i põhiliseks struktuuri elemendiks.

Tutvustatakse väga lühidalt GSON teeki - mis see on ning kuidas paigaldada. Samuti on välja toodud, et kõikides näidetes on kasutatud lihtsat Student klassi, millel on vaid kolm atribuuti ning konstruktor (joonis 1).

GSON

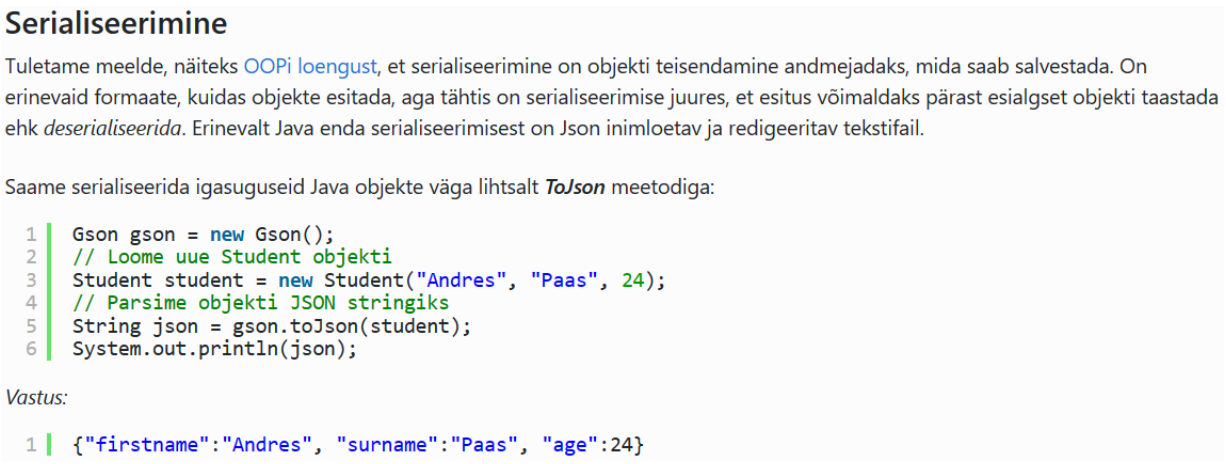
Selle kodutöö piires õpime GSON-i. See on Google poolt loodud teek JSON-i parsimiseks. Selle lisame sõltuvus järgmisele teegile Gradle'i konfiguratsioonifailis: `implementation 'com.google.code.gson:gson:2.10.1'` (See on kursuse repositooriumis juba tehtud.)

Näidetes kasutame lihtsat ***Student*** klassi, millel on vaid paar atribuuti ja konstruktor:

```
1  public class Student {
2      String firstname;
3      String surname;
4      int age;
5
6      public Student(String firstname, String surname, int age) {
7          this.firstname = firstname;
8          this.surname = surname;
9          this.age = age;
10     }
11 }
```

Joonis 1. GSON teegi tutvustus

Kodutöö peamine osa algab Java objektide serialiseerimisega JSON objektideks. Tutvustatakse toJson meetodit, mis lubab väga lihtsalt igasuguseid Java objekte serialiseerida, isegi kui selle klassi lähtekoodile puudub ligipääs (joonis 2). Serialiseerimise all tutvustatakse ka lühidalt kuidas GSON parserit kohandada, kasutades GsonBuilder klassi.



Joonis 2. Näide toJson meetodi kasutusest

Järgmise teemana käsitletakse deserialiseerimist. Esimesena tutvustatakse fromJson meetodit, mis on sarnaselt lihtne eelnevalt tutvustatud toJson meetodiga (joonis 3). Rõhutatakse ka, et JSON atribuute, mida parsitavaks klassis ei ole, ignoreeritakse.

Joonis 3. fromJson meetodi tutvustus koos näitega.

Järgmisena tutvustatakse õppijatele lühidalt GSONi annotatsioonide võimalusi (joonis 4). Kuigi üks peamisi GSONi eeliseid on see, et JSONi parsimiseks pole vaja kasutada annotatsioone, võib see teadmine siiski kasulik olla eriti juhtudel, kus andmed on keerulisemalt esitatud.

Annotatsioonid. GSON on küll loodud põhimõttega, et annotatsioone pole meie klassides vaja kasutada. Kuid mõnedel juhtudel on siiski objektide parsimist vaja kohandada ja seega on selleks olemas mitmeid annotatsioone. Näiteks on olukord, kus meie JSON sisendandmed võivad olla mitmes eri keeles. Eelnevalt sai mainitud, et *fromJson* meetod toimib ainult siis kui klassi atribuudid on samade nimedega mis JSON-i atribuudid. Sellest piirangust on võimalik ümber saada kui kasutame *@SerialisedName* annotatsiooni, et defineerida võimalikud aliased meie atribuutidele.

```java
public class Student {
    @SerializedName(value = "firstname", alternate = {"nimi", "eesnimi"})
    String firstname;
    @SerializedName(value = "surname", alternate = {"perekonnanimi", "perenimi"})
    String surname;
    @SerializedName(value = "age", alternate = "vanus")
    int age;

    public Student(String firstname, String surname, int age) {
        this.firstname = firstname;
        this.surname = surname;
        this.age = age;
    }
}
```

Nüüd saab GSON ka parsida järgnevat JSON stringi ilma probleemideta.

```json
{
  "nimi": "Pipi",
  "perenimi": "Pikksukk",
  "vanus": 20,
}
```

Joonis 4. Annotatsioonide tutvustus ja näide

Järgmisena tutvustatakse teist lähenemist JSON-i parsimiseks, mis on JSON puu meetod (joonis 5).

2. JSON puu meetod

Kui me ei taha eraldi klassi luua (nagu meie näidetes kasutatud Student klass) selleks, et JSON-ist andmeid kätte saada, siis me võime ka individuaalseid atribuute JSON-ist välja noppida. Selleks peame esiteks parsima JSON-i *JsonObject*'iks.

```java
String jsonString = Files.readString(Paths.get("input.json")); // Input faili sisu sama mis enne
JsonObject jsonTree = JsonParser.parseString(jsonString).getAsJsonObject();
```

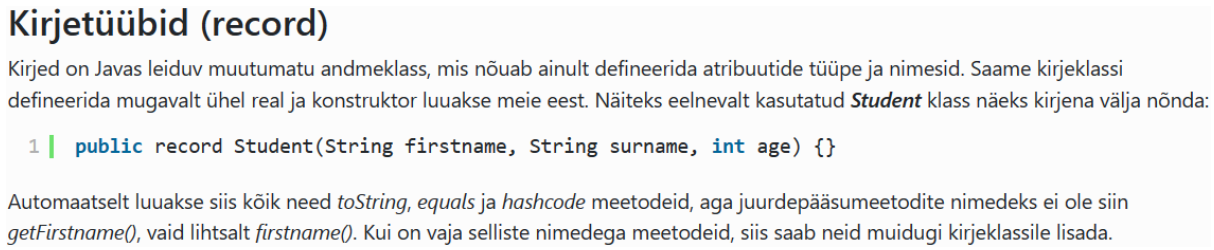
Paneme tähele, et tahame individuaalseid atribuute JSON-ist kätte saada, ja selle jaoks on vaja käsitleda *JsonObject* tüüpi, mitte *JsonElement* tüüpi. Kuna *parseString* tagastab *JsonElement* tüübi, siis peame kutsuma sellel veel *.getAsJsonObject* meetodit.

Peale seda on võimalik kõiki atribuute käsitleda lihtsalt nende atribuutide nimede kaudu, kasutades *get* meetodit. Näiteks kui tahame saada tudengi eesnime ja kursuseid kätte, saame seda teha järgnevalt:

```java
String firstname = jsonTree.get("firstname").getAsString();
System.out.println(firstname);
// Kuna kursused on massiivis, peame neid veidi teistmoodi käsitlema
JsonArray courses = jsonTree.getAsJsonArray("courses");
for (Object course : courses ) {
    System.out.println(course.toString());
}
```

Joonis 5. JSON puu meetodi tutvustus ja näide.

Viimane teema, mida seletatakse, on Java *Record* tüübid (joonis 6).

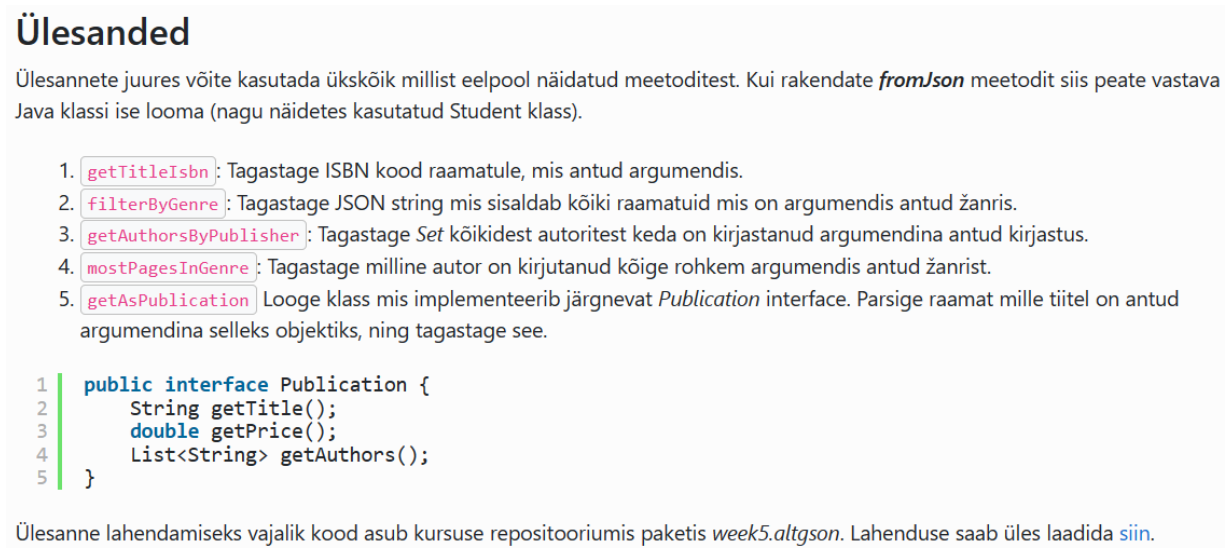


Joonis 6. Kirjetüüpide tutvustus ja näide.

Peale JSON parsimise õpetamist, on tudengitel vaja kodutöö ülesanded täita.

3.3.2 JSON kodutöö ülesannete kokkuvõte

Kujundava hindamise jaoks on koostatud ülesanded (joonis 7). JSON kodutöö hindamiseks on 5 ülesannet, kus iga ülesanne väärt 0,2 punkti. Ülesanded on järjestatud põhimõttega, et iga järgnev ülesanne on keerulisem kui eelmine. Iga ülesanne nõuab meetodi loomist, mille signatuur on tudengitele ette antud.



Joonis 7. JSON kodutöö ülesanded.

Kindlustamaks tudengite arusaama teemast, nõuavad ülesanded eri lähenemisi. Näiteks esimene, kolmas ja neljas ülesanne nõuavad JSONi parsimist, et algandmed kätte saada ja tagastada. Ülesanne kaks aga nõuab ka lisasammuna tagasi JSON formaadiks parsimist. Ülesanne viis on kõige mahukam, nõudes tudengitel implementeerida etteantud liidest rahuldava klassi, mida siis hiljem andmete tagastamisel kasutada.

3.3.3 Ärireeglite kodutöö seletava osa kokkuvõte

Ülevaade kodutöös õpetatavast:

- Mis on ärireeglid.
- Mis on JsonLogic.
- Kuidas kirjutada JsonLogic reegleid.
- Kuidas kasutada JsonLogicut lihtsate probleemide lahendamiseks.
- Mis on Easy Rules.
- Kuidas kirjutada Easy Rules reegleid.
- Kuidas kasutada Easy Rulesi lihtsate probleemide lahendamiseks.

Kodutöö alustab ärireeglite kontseptsiooni sissejuhatusega. Tarkvaraarenduses on ärireeglid otsustusjuhiste kogum, mida kasutatakse süsteemi käitumise reguleerimiseks, võttes arvesse erinevaid sisendeid ja sündmusi. Ärireeglid aitavad otsustusprotsesse automatiseerida.

Nagu ka eelnevalt mainitud, käsitletakse ärireeglite kodutöös kahte tehnoloogiat – JsonLogic ja Easy Rules. Peale selle, et mõlemad tehnoloogiad põhinevad ärireeglitel, ei ole nad omavahel seotud. Seetõttu on kodutöö jagatud kaheks osaks, esimene osa JsonLogic jaoks, teine osa Easy Rules jaoks.

JsonLogicu süntaks on väga omapärane, niisiis on selle osa jaoks näiteid rohkem toodud. Näidete rohkus võiks tagada, et tudengitele ei jää ükski samm segaseks.

Peale sissejuhatust algab JsonLogic osa kodutööst. Esmalt lühitutvustus JsonLogicu kohta.

Järgnevalt algab JsonLogicu süntaksi tutvustus. Siin õpetatakse, kuidas JsonLogicu reeglite loomisel kasutada operaatoreid, muutujaid ja tingimuslauseid (joonis 8). Samuti on antud link JsonLogicu kodulehele, kus on kirjas kõik võimalikud operatsioonid, mida JsonLogic toetab [16].

Operaatorid

JsonLogicus on kõik operaatorid ja võtmesõnad defineeritud välja võtme osas ja välja väärtuse osas operandid. Näiteks liitmise operatsioon `4+6`, võtmeosa `"+"` ja operandid esitatud väärtuse osas massiivina. Kõiki opratsioone saab näha siit.

```
{"+": [4,2]}
```

```
6
```

Muutujad

Muutujaid märgime `"var"` võtmesõnaga, ja väärtuseks on muutuja nimi. Muutujate väärtustamiseni jõuame hiljem.

```
{"+": [{"var": "number"}, 2]}
```

If lause

Üksikutel `if` lausetel on 3 argumenti, mis on esitatud massiivis:

1. Tingimus (`true`)
2. Mida teha kui tingimus True (`"yes"`)
3. Mida teha kui tingimus False (`"no"`)

```
{"if" : [true, "yes", "no" ]}
```

```
yes
```

`if` lauseid saab ka üksteise järele panna. Sel juhul on massiivi esimene element esimese if lause tingimus ja massiivi teine element tagastatav väärtus kui tingimus on `true`. Samamoodi kolmanda ja neljanda elemendiga massiivis jne. Viimane element on samaväärne `else` lausega.

Järgnevalt näide kus olenevalt muutjast `"temp"`, tagastatakse kas “freezing”,“liquid” või “gas”:

```
{"if" : [
  {"<": [{"var":"temp"}, 0] }, "freezing",
  {"<": [{"var":"temp"}, 100] }, "liquid",
  "gas"
]}
```

Võrdsus "==="

Võrdsuse operaatorit kasutame võtmega `"==="` ja väärtuseks massiiv, kus massiivi esimene element võrdsuse vasak pool ja teine element parem pool. Näitena `5==6`:

```
{"if" : [{"===": [5,6]}, "yes", "no" ]}
```

```
no
```

Joonis 8. JsonLogic operaatorite kasutamise tutvustus.

JsonLogicu seletav osa lõpeb õpetusega JsonLogicu jooksutamisest, kasutades eelnevalt loodud reegleid (joonis 9).

JsonLogic jooksutamine

Selleks, et meie JsonLogic kood midagi teeks, peame JsonLogic interpretaatori käivitama `apply` meetodiga. Kuna meil on ilmselt ka muutujad meie JSON koodis, siis peame need enne jooksutamist JsonLogicule andma. Muutujaid anname edasi Mapiga, kus `key` on muutuja nimi ja `value` muutuja väärtus. Siin olevas näites on muutujaks `"temp"`:

```
// Loeme sisse json faili stringi
String jsonString = Files.readString(Path.of("JSON-logic/waterstate.json"));

// Loome instantsi JsonLogicust
JsonLogic jsonLogic = new JsonLogic();
// Muutujate map
Map<String, Integer> data = new HashMap<>();
// Paneme andmed mapi
int temperature = 10;
data.put("temp", temperature);
// Käivitame JsonLogicu interpretaatori apply meetodiga, kus esimene argument on
// meie json string ja teine argument meie muutujad mapis.
System.out.println(jsonLogic.apply(jsonString, data));
```

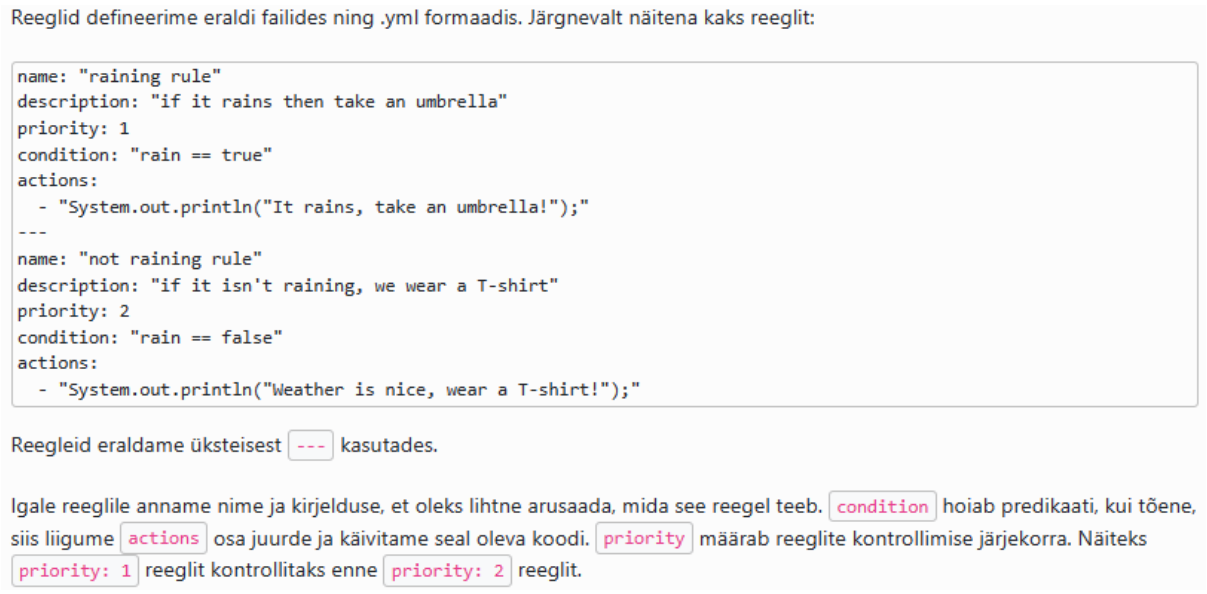
```
liquid
```

Waterstate.json sisu:

```
{"if" : [
  {"<": [{"var":"temp"}, 0] }, "freezing",
  {"<": [{"var":"temp"}, 100] }, "liquid",
  "gas"
]}
```

Joonis 9. JsonLogicu jooksutamine.

Easy Rules osa kodutööst algab lühikese tutvustusega, millele järgnevad õpetlikut näited reeglite kirjutamisest (Joonis 10).



Reeglid defineerime eraldi failides ning .yml formaadis. Järgnevalt näitena kaks reeglit:

```
name: "raining rule"
description: "if it rains then take an umbrella"
priority: 1
condition: "rain == true"
actions:
  - "System.out.println("It rains, take an umbrella!");"
---
name: "not raining rule"
description: "if it isn't raining, we wear a T-shirt"
priority: 2
condition: "rain == false"
actions:
  - "System.out.println("Weather is nice, wear a T-shirt!");"
```

Reegleid eraldame üksteisest `---` kasutades.

Igale reeglile anname nime ja kirjelduse, et oleks lihtne arusaada, mida see reegel teeb. `condition` hoiab predikaati, kui tõene, siis liigume `actions` osa juurde ja käivitame seal oleva koodi. `priority` määrab reeglite kontrollimise järjekorra. Näiteks `priority: 1` reeglit kontrollitaks enne `priority: 2` reeglit.

Joonis 10. Easy Rules reeglite kirjutamise õpetus ning näited.

Peatüki lõpus on seletus reeglite töötlemisest (joonis 11).

Selleks, et reegleid evalueerida, peame enne looma mõned instantsid.

- `Facts` instants - hoiab meie "fakte". Faktid on kuidas me anname andmeid reeglitele. Faktid ise on muutumatud aga sellest piirangust on võimalik ümber saada kui sisestame mingisuguse objekti faktidesse, millel on muutuvad atribuudid ning getter ja setter meetodid.
- `MVELRuleFactory` instants - Factory mis loob meile `Rule` objekte.
- `Rules` instants - hoiab kõiki `Rule` objekte.
- `RulesEngine` instants - jooksutab reeglimasinat, meie antud reeglite ja faktidega.

```
public static void demo1() throws Exception {
    // Defineerime "faktid"
    Facts facts = new Facts();
    facts.put("rain", true);
    // Loeme sisse reeglid
    MVELRuleFactory ruleFactory = new MVELRuleFactory(new YamlRuleDefinitionReader());
    Rules rules = ruleFactory.createRules(new FileReader("easyrules/demo/rain-rule.yml"));
    // Käivitame reeglimasina reeglite ja faktidega
    RulesEngine rulesEngine = new DefaultRulesEngine();
    rulesEngine.fire(rules, facts);
}
```

Joonis 11. Reeglite töötlemine ning eelnevad sammud.

Peale mõlema JsonLogic ja Easy Rules õpetavat osa, järgnevad kodutöö ülesanded.

3.3.4 Ärireeglite kodutöö ülesannete kokkuvõte

Nagu eelnevalt oli mainitud, on kodutöö jagatud kaheks osaks. Niisiis on ka ülesanded eraldi, JsonLogic ülesanded peale JsonLogic seletavat osa ja Easy Rules ülesanded peale Easy Rules seletavat osa. Kokku on viis ülesannet, igaühe edukas lahendamine annab 0.2 punkti.

JsonLogic sai keerulise süntaksi ja pikema seletava osa tõttu 3 ülesannet. Easy Rules osa sai 2 ülesannet (Joonis 12, 13).

JsonLogic ülesanded

Lahendustes kasutage Java tingimuslausete asemel JsonLogicut. Teie koodis ei tohiks ühtegi `if` lauset leiduda.

1. Implementeerige meetod, mis kontrollib kas argumendina antud stringis leidub alamstringi "aa". Kui leidub, tagastage `true`, vastasel juhul `false`. `static public boolean task1(String string) throws IOException, JsonLogicException`
2. Implementeerige meetod, mis eemaldab argumendina antud listist negatiivsed arvud. Tagastage sama list kus on alles ainult positiivsed arvud (0 kaasa arvatud). Saate kasutada filter meetodit. `static public ArrayList task2(ArrayList<Double> numbers) throws IOException, JsonLogicException`
3. Implementeerige meetod, mis olenevalt kliendi eelnevatest kultustest, määrab allahindluse ostule. Täpsemalt, kui klient on eelnevalt kulutanud kas 1000€ või rohkem poes, siis määratakse 10% allahindlus. Kui klient on kulutanud alla 1000€ aga 200€ või rohkem, siis 5% allahindlus. Kui klient on kulutanud vähem kui 200€, siis allahindlust ei ole. Tagastage ostu korrektne lõpphind. `static public double task3(double spent, double cost) throws IOException, JsonLogicException`

Joonis 12. JsonLogic ülesanded.

Easy Rules ülesanded

Ülesannetes oodatakse teil tagastada väärtusi. Selleks on teil vaja muuta fakte Easy Rules reegleid kasutades ning tagastada need peale reeglimasina jooksutamist. Java `if` lauseid kasutada ei tohi.

1. Implementeerige lihtsustatud FizzBuzz meetod. Argumendina antakse teile arv. Kui arv jagub 3-ga, tagastage Fizz. Kui arv jagub 5-ga, tagastage Buzz. Kui arv jagub nii 3 kui 5-ga, tagastage FizzBuzz. Võite oletada, et alati jagub antud arv vähemalt 3 või 5-ga. Vihje: peate koostama 3 eraldi reeglit. `public static String task1(int number) throws Exception`
2. Implementeerige meetod, mis olenevalt kliendi eelnevatest kultustest, määrab allahindluse ostule. Täpsemalt, kui klient on eelnevalt kulutanud poes kas 1000€ või rohkem, siis määratakse 10% allahindlus. Kui klient on kulutanud alla 1000€ aga 200€ või rohkem, siis 5% allahindlus. Kui klient on kulutanud vähem kui 200€, siis allahindlust ei ole. Tagastage ostu korrektne lõpphind. Selleks, et oleks võimalik andmeid ka reeglite siseselt muuta, on teile antud `Cashier` klass. Saate selle `Cashier` instantsi faktina sisestada. `public static double task2(double spent, double cost) throws Exception`

Joonis 13. Easy Rules ülesanded.

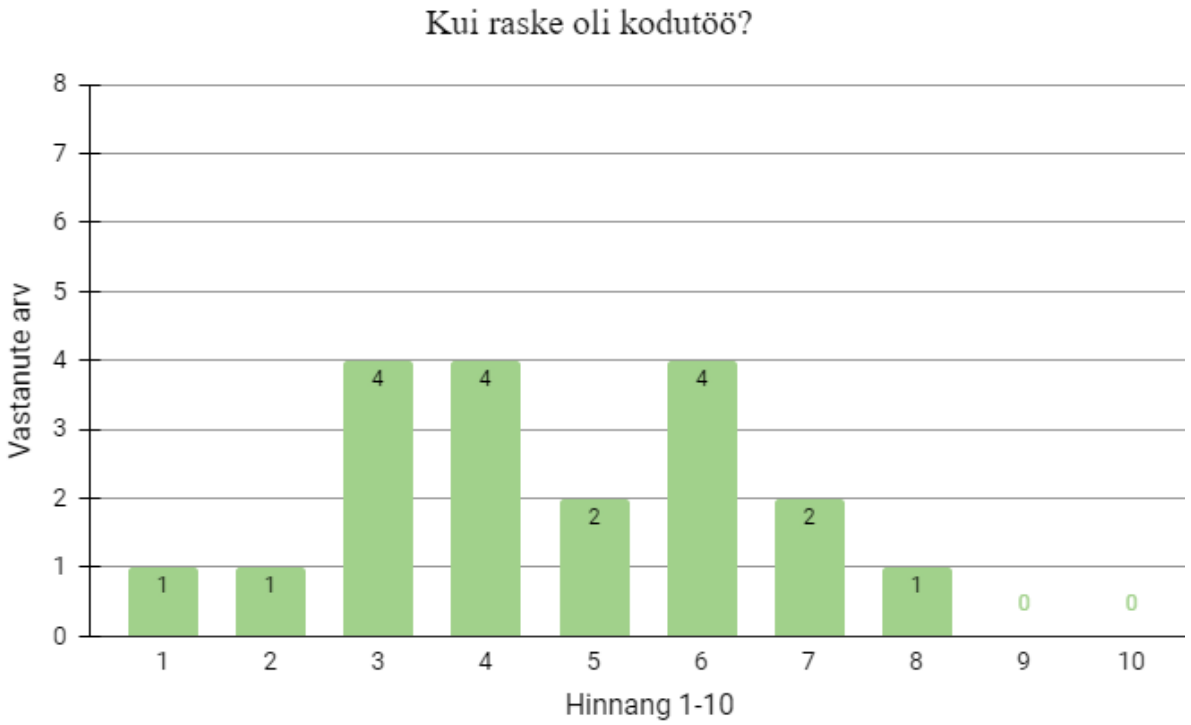
Ülesanded võiksid eri osadel samad olla, kuid tehnoloogiate ehituse ning dokumentatsioonide omapärasuste tõttu ei ole see ideaalne. Näiteks JsonLogic kodulehel on FizzBuzz ülesanne juba näitena lahendatud ning esile toodud. Kuna loodud kodutöö lahendamine nõuab JsonLogic kodulehe külastamist, siis on väga tõenäoline, et tudengid näeksid leheküljel olemasolevat lahendust. On parem kui ülesannetel ei leidu veebis lihtsasti leitavaid lahendusi, et tagada tudengite iseseisev ülesannete lahendamine. Niisiis on enamik ülesandeid JsonLogic ja Easy Rules osade vahel erinevad. Ainult viimane ülesanne on mõlema osa puhul sama, sest see ülesanne lubab mõistliku raskustasemega lahenduse loomist nii JsonLogicu kui Easy Rulesi puhul.

4. Kodutööde tagasiside analüüs

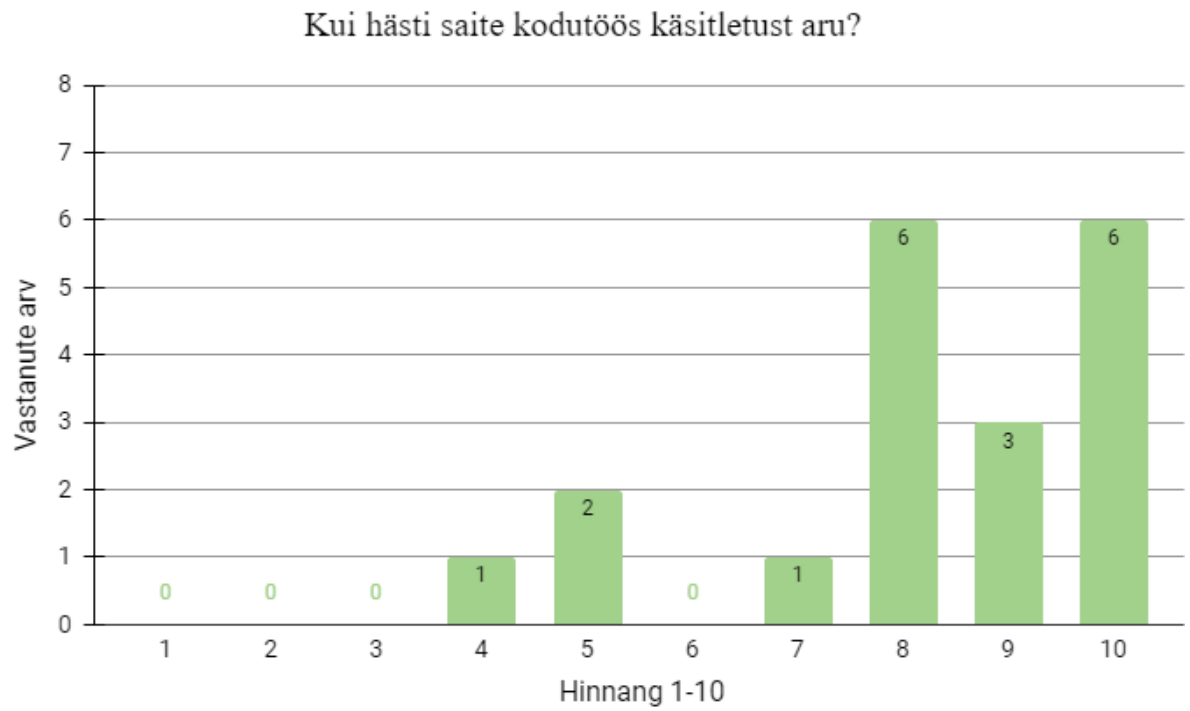
Kodutööde efektiivsuse hindamiseks koostati ka tagasiside vorm (Lisa II ja III). Tagasiside vorm on mõeldud täitmiseks tudengitele, kes lahendasid loodud kodutöid. Esimese kodutöö puhul oli piisavalt aega, et tudengitelt tagasisidet saada. Teise kodutöö esitamise tähtaeg on 1. juuni, mis on peale bakalaureusetöö esitamise tähtaega. Sellepärast oli teise kodutöö puhul ainult võimalik analüüsida õppejõu kirjaliku tagasisidet.

4.1 JSON kodutöö tagasiside

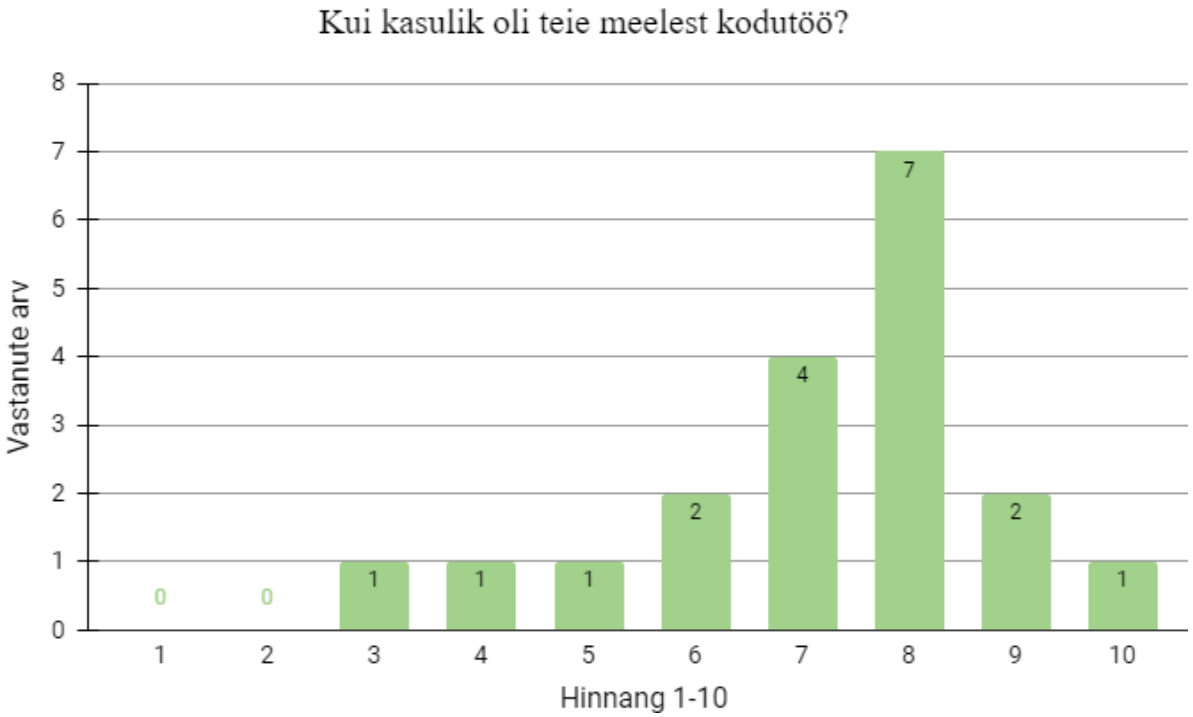
JSON kodutöö lahendanud 19st tudengist täitsid küsimustiku kõik 19. Järgnevalt loetletud küsimused olid kõik hinnatavad punkti skaalal ühest kümneni. JSON kodutöö raskuse hinnangu keskmine oli 4.6 ning mediaan 4 (joonis 14). Tudengite kodutööst arusaamise hinnangu keskmine oli 8.2 ja mediaan 8 (joonis 15). Kodutöö kasulikuse hinnangu keskmine oli 7.2 ning mediaan 8 (joonis 16).



Joonis 14. Kodutöö raskuse hinnangu jaotus.



Joonis 15. Kodutööst arusaamise hinnangu jaotus.



Joonis 16. Kodutöö kasulikuse hinnangu jaotus.

Seitsmel tudengil kulus kodutöö lahendamiseks alla tunni. Ajavahemikus üks kuni kaks tundi lahendasid kodutöö samuti seitse tudengit. Ajavahemikus kaks kuni kolm tundi lahendasid neli tudengit ning ühel tudengil kulus kodutöö lahendamiseks neli tundi. Keskmiselt kulutasid tudengid kodutöö lahendamiseks 84 minutit.

Kokkuvõtvalt võiks järeldada, et JSON kodutöö oli edukas. Kodutöö sisust said enamik tudengid hästi aru ning kasulikust hindasid nad samuti kõrgelt. Kodutöö raskus oli enamik tudengite jaoks kas paras või kohati lihtne. Kodutöö liiga lihtne olemine ei ole suur probleem, sest kodutöö eesmärk oli õpetada ainult algsed teadmised JSON parsimise kohta. Tudengite jaoks kellele jäi teema veidi segaseks, võib kodutöö muudatusena kaaluda rohkemate selgituste lisamist.

4.2 Ärireeglite kodutöö tagasiside

Kursuse ajakava tõttu ei olnud võimalik tudengite tagasisidet ärireeglite kodutöö kohta analüüsida, kuna tudengid ei olnud veel kodutööd täitnud. Sellepärast oli võimalik ainult kursuse õppejõult tagasisidet saada.

Õppejõu tagasisidest selgus, et kodutöö üldine materjal ja ülesanded on asjakohased, kuid oleks võinud rohkem ärireeglite töötlemist demonstreerida[1]. Selle demonstreermine aitaks põhjendada, milleks üldse ärireegleid kasutada selle asemel, et lahendusi lihtsalt näiteks Javas kirjutada. Õppejõu mõtted, kuidas seda kodutööd nõnda täiendada on:

- Lisada JsonLogic ülesanne, kus peab JSON reeglit analüüsima, kasutades juba eelnevas kodutöös õpetatud GSON teeki.
- Easy Rules ülesandeid muuta, et paremini näidata kuidas Easy Rules käivitab mitu reeglit järjest.
- Lisada Easy Rules ülesanne, milles peab olenevalt sisendist dünaamiliselt reeglid looma.

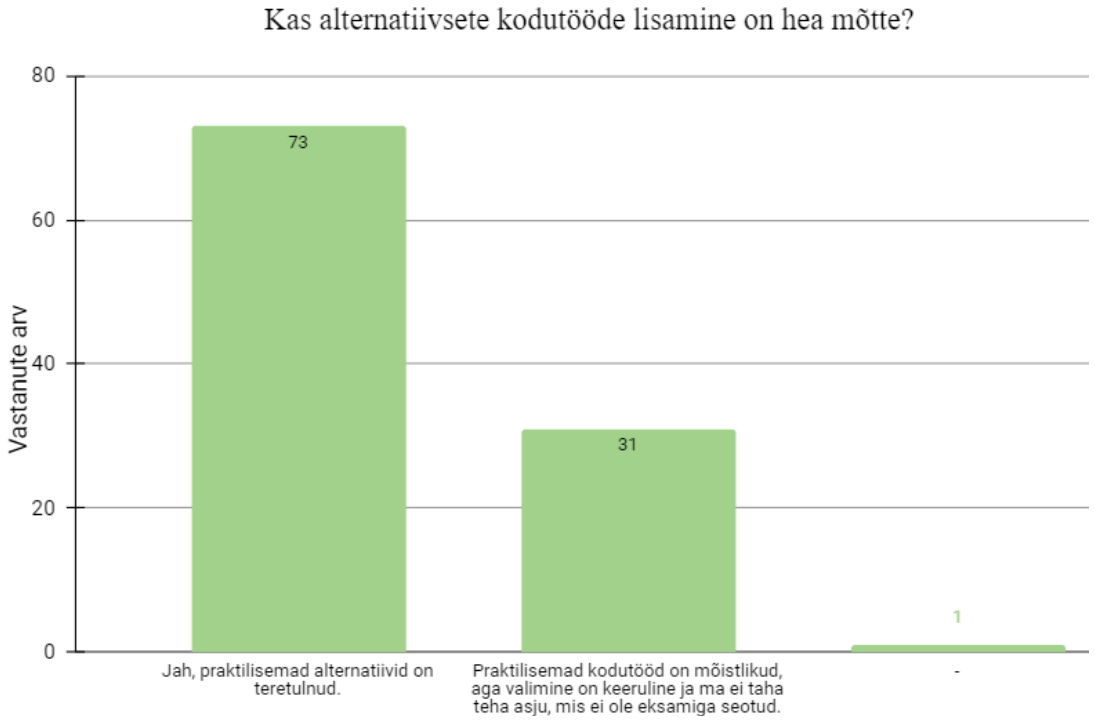
Vajalikud täiendused tehti ärireeglite kodutööle koostöös õppejõuga. Täiendused jäid aga ajapuuduse tõttu käesolevast tööst välja. Valminud täiendatud kodutöid on võimalik näha kursuse lehel (Lisa I).

4.3 Kursuse poolne tagasiside

Aine tudengite üldisema arvamuse määratlemiseks, koostas kursuse korraldaja lühiküsitluse (Lisa IV). Küsitlusele vastas 105 tudengit. 70% tudengitele meeldis praktilisemate kodutööde lisamine kursusele (joonis 17). Ülejäänud 30% meelest on kodutööd mõistlikud, kuid ei

[1] Tagasiside Vesal Vojdani kirjaliku kommentaari põhjal (09.05.2023).

soovinud neid lahendada, sest tahtsid pigem keskenduda kursuse põhi teemadele, mis võivad eksamil ilmuda.



Joonis 17. Tudengite vastused küsimusele "Kas alternatiivsete kodutööde lisamine on hea mõtte?"

Küsimuse vastustest selgub, et kuigi alternatiivsed kodutööd olid loodud kodutöödele, mille teemad ei olnud eksamil, jäi see asjaolu paljude tudengite jaoks segaseks. Sama selgus ka kirjutatud tagasisidest, kus mitmed tudengid mainisid, et ei soovi alternatiivseid kodutöid lahendada kuna tahtsid pigem eksami teemade õppimisele keskenduda. Selle segaduse vähendamiseks oleks võimalik selgemalt ja rohkem mainida, et tudengid ei jää millestki ilma kui nad valivad lahendamiseks alternatiivsed kodutööd.

5. Kokkuvõte

Töö eesmärk oli luua Tartu Ülikooli informaatika õppekava kursusele “Automaadid, keeled ja translaatorid” alternatiivseid praktilisi kodutöid. Kodutööde laiem eesmärk on pakkuda tudengitele valikut ning õpetada kursusel käsitlevate teemade praktilisemaid pooli. Töö käigus loodi kaks eri kodutööd. Esimene kodutöö sisaldab JSON formaadi tutvustust ning parsimist kasutades Java GSON teeki. Teine kodutöö hõlmab ärireeglite keelte JsonLogic ning Easy Rules kasutamist. Kodutööde tagasisidest selgus, et 19 tudengi keskmise arvamuse järgi oli JSON kodutöö kasulik. Kursuse õppejõu tagasiside põhjal, oli ärireeglite kodutöö asjakohane, kuid vajas mõningaid täiendusi. Täiendused said tehtud koostöös õppejõuga. Suurem osa kursusel osalevate tudengite arvates on alternatiivsete kodutööde pakkumine hea täiendus kursusele.

Viidatud kirjandus

[1] Automaadid, keeled ja translaatorid LTAT.03.006 - Tartu Ülikooli Õppeinfosüsteem. https://ois2.ut.ee/#/courses/LTAT.03.006/details (19.04.2023)

[2] Automaadid, keeled ja translaatorid - regulaaravaldiste kodutöö. https://courses.cs.ut.ee/t/akt/Main/JavaRegex (20.04.2023)

[3] Torben Ægidius Mogensen. Introduction to Compiler Design. London: Springer. 2011.

[4] Seriot N. Parsing JSON is a Minefield, 2016 https://seriot.ch/projects/parsing_json.html (25.04.2023)

[5] Maven Repository andmed eri teekide populaarsuse kohta. https://mvnrepository.com/popular (10.04.2023)

[6] GSON. https://github.com/google/gson (10.04.2023)

[7] Jackson. https://github.com/FasterXML/jackson (10.04.2023)

[8] JsonLogic. https://jsonlogic.com/ (10.04.2023)

[9] Mozilla dokumentatsioon https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval (10.04.2023)

[10] Easy Rules. https://github.com/j-easy/easy-rules (10.04.2023)

[11] Fowler, Martin 2019. RulesEngine artikkel “Should I use a rules engine?” https://martinfowler.com/bliki/RulesEngine.html (10.04.2023)

[12] Wilson, Greg 2019. Teaching Tech Together (A Lesson Design Process) https://teachtogether.tech/en/index.html (10.04.2023)

[13] 2021/2022 aasta tudengite näidisõppekava
https://cs.ut.ee/sites/default/files/2022-02/infbak2021_2022.pdf (10.04.2023)

[14] Andmebaasid LTAT.03.004 - Tartu Ülikooli Õppeinfosüsteem.
https://ois2.ut.ee/#/courses/LTAT.03.004/details (19.04.2023)

[15] *Digital COVID Certificates: Business Rules*
https://github.com/ehn-dcc-development/eu-dcc-business-rules (10.04.2023)

[16] JsonLogic - Supported Operations
https://jsonlogic.com/operations.html (24.04.2023)

Lisad

I Materjalide asukohad

Koostatud kodutööd on leitavad AKT kursuse arvutiteaduse instituudi veebilehel. Kodutööd võivad aja vältel paranduste ning muudatuste tõttu muutuda.

- JSON kodutöö: https://courses.cs.ut.ee/t/akt/Main/Gson
- Ärireeglite kodutöö: https://courses.cs.ut.ee/t/akt/Main/BusinessRules

II JSON kodutöö tagasiside vorm

Kui raske oli kodutöö? *

	1	2	3	4	5	6	7	8	9	10	
Liiga lihtne	○	○	○	○	○	○	○	○	○	○	Liiga raske

Umbes kui mitu minutit võttis kodutöö lahendamine? *

Teie vastus

Kui hästi saite kodutöös käsitletust aru? *

	1	2	3	4	5	6	7	8	9	10	
Ei saanud üldse aru	○	○	○	○	○	○	○	○	○	○	Kõik oli selge

Kui mingid asjad jäid segaseks, siis miks? Kas selgitused ei olnud piisavalt detailsed või selged? Kas polnud piisavalt näiteid?

Teie vastus

Kui kasulik oli teie meelest see kodutöö? *

	1	2	3	4	5	6	7	8	9	10	
Mitte üldse kasulik	○	○	○	○	○	○	○	○	○	○	Väga kasulik

Kas tuli ette tehnilisi probleeme?

Teie vastus

Kui teil tuli mõttesse veel midagi mida võiks muuta või parandada kodutöös siis võite seda siia kirjutada.

Teie vastus

III Ärireeglite kodutöö tagasiside vorm

Kui raske oli JsonLogic osa? *

	1	2	3	4	5	6	7	8	9	10	
Liiga lihtne	○	○	○	○	○	○	○	○	○	○	Liiga raske

Kui raske oli Easy Rules osa? *

	1	2	3	4	5	6	7	8	9	10	
Liiga lihtne	○	○	○	○	○	○	○	○	○	○	Liiga raske

Umbes kui mitu minutit võttis kodutöö lahendamine? *

Teie vastus

Kui hästi saite kodutöös JsonLogic käsitletust aru? *

	1	2	3	4	5	6	7	8	9	10	
Ei saanud üldse aru	○	○	○	○	○	○	○	○	○	○	Kõik oli selge

Kui hästi saite kodutöös Easy Rules käsitletust aru? *

	1	2	3	4	5	6	7	8	9	10	
Ei saanud üldse aru	○	○	○	○	○	○	○	○	○	○	Kõik oli selge

Kui mingid asjad jäid segaseks, siis miks? Kas selgitused ei olnud piisavalt detailsed või selged? Kas polnud piisavalt näiteid?

Teie vastus

Kui kasulik oli teie meelest JsonLogic osa kodutööst? *

	1	2	3	4	5	6	7	8	9	10	
Mitte üldse kasulik	○	○	○	○	○	○	○	○	○	○	Väga kasulik

Kui kasulik oli teie meelest Easy Rules osa kodutööst? *

	1	2	3	4	5	6	7	8	9	10	
Mitte üldse kasulik	○	○	○	○	○	○	○	○	○	○	Väga kasulik

Kas tuli ette tehnilisi probleeme?

Teie vastus

Kui teil tuli mõttesse veel midagi mida võiks muuta või parandada kodutöös siis võite seda siia kirjutada.

Teie vastus

IV Kursuse sisene küsitlus

Question 1
Not complete
Marked out of 1.00
Flag question
Edit question

Kui lahendasid Gsoni kodutöö, siis oleks väga tore, kui saaksid täita järgnev (anonüümne) tagasiside vorm:

https://docs.google.com/forms/d/e/1FAIpQLSfyfsmPd3OydNfsJRH-wBTeom3cJRY-GTAhZhMDtCjTJhIP9Q

Kas tegid seda?

- ○ Jah, ma täitsin ta ära!
- ○ Ei, ma teen seda homme.
- ○ Ei, ma ei taha üldse seda teha.
- ○ Ma ei lahendanud seda kodutööd.

Check

Question 2
Not complete
Marked out of 1.00
Flag question
Edit question

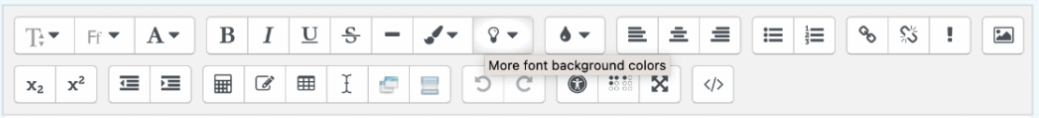
Kas alternatiivsete kodutööde lisamine on hea mõtte?

- ○ Jah, praktilisemad alternatiivid on teretulnud.
- ○ Praktilisemad kodutööd on mõistlikud, aga valimine on keeruline ja ma ei taha teha asju, mis ei ole eksamiga seotud.
- ○ Ei, aine peaks keskenduma oma põhieesmärgile arendada sügavama arusaamise arvutiprogrammide tähendusest. Praktilisemaid asju võib jätta ChatGPT/Copilotile.

Check

Question 3
Not yet answered
Marked out of 1.00
Flag question
Edit question

Kas tahad midagi öelda alternatiivsete kodutööde kohta või mõni muu idee pakkuda, kuidas teha aine just sinu jaoks väärtuslikumaks. (Kui ei taha, siis kirjuta "ei taha" vms, muidu hindab moodle seda automaatselt.)



Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Kristen Surva**,

1) annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose
   Praktiliste kodutööde loomine ainele "Automaadid, keeled ja translaatorid",
   mille juhendaja on Vesal Vojdani,
   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2) Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3) Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4) Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Kristen Surva

09.05.2023