TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

**Harri Saar**

# Aine Tehisintellekt I õpitarkvara

**Bakalaureusetöö (9 EAP)**

Juhendaja: Tõnu Tamme

Tartu 2016

## Aine Tehisintellekt I õpitarkvara

**Lühikokkuvõte:**

Käesoleva bakalaureusetöö kirjutamise käigus valmis õpitarkvara Tehisintellekt I aine jaoks. Antud lõputöö lahendus võimaldab luua mugavalt ja kiirelt otsimispuid nupuülesannetele. Tarkvara loomisel kasutati järgnevaid tehnoloogiaid: Java, Haskell, Git, Github, Eclipse. Rakendust saavad kasutada kõik kellel on seadistatud arvuti Java Runtime Environment.

**CERCS:** P170, Arvutiteadus

**Võtmesõnad:**

Tehisintellekt, otsimispuu, süvitsiotsing, laiutiotsing, õpitarkvara

## A courseware for Artificial Intelligence I

**Abstract:**

In the process of writing this Bachelor's Thesis a courseware for Aritificial Intelligence I was developed. This courseware enables creating search trees quickly and easily. The software was built using the following technologies: Java, Haskell, Git, Github, Eclipse. The app can be used by any computer with a Java Runtime Environment.

**CERCS:** P170, Computer Science

**Keywords:**

Artificial intelligence, search tree, depth-first search, breadth-first search, educational software

# Sisukord

# 1 Sissejuhatus

Käesolev bakalaureusetöö on rakendusliku ülesande lahendus. Valmiva programmi sihtgrupp on Tehisintellekt I aine tudengid. Loodud rakendus saab olema kasulik õpilastele kuna aitab õppematerjalile uut moodi läheneda. Eesmärk on luua programm, mis on piisavalt dünaamiline lahendamaks erinevaid nupuülesandeid.

Rakendus peab vastama järgnevatele kriteeriumidele:

1. Kasutajal ei pea olema eelnevaid programmeerimisteadmisi. Isik saab keskenduda ainult teadmuse esitusele. Sarnaselt Tehisintellekt I aine raames harjutuste läbimisel.
2. Programm on suuteline looma otsimispuu ülesannetele, mis on aine Tehisintellekt I raames. Näiteks: “Mees, rebane, hani ja vili tuleb üle jõe viia”, “4x4 kabe mäng”
3. Kasutajal on võimalik valida erinevate algoritmide vahel. Ning nende visuaalne esitus on selge ja arusaadav ning seega võimendab arusaamist erinevatest algoritmidest.
4. Programmi väljundit on võimalik eksportida pildina. Nii, et seda saab mujal õppekeskkonnas kasutada.

Töö on jagatud kaheks suuremaks osaks. Esimese osas kirjutatakse programmi realisatsioonist. Vastav peatükk sisaldab tutvustust kasutatavatest tehnoloogiatest, süsteemi arhitektuuri kirjeldust ning kasutuslugusi rakendusele. Teises osas keskendutakse tarkvara dokumentatsioonile, kus saab täpsemalt lugeda kasutusjuhendist ning näidislahendustest. Tulemuse analüüsi peatükis kirjutatakse sarnastest realisatsioonidest ning valminud programmi kvaliteedist.

# 2 Valdkonna kirjeldus

Intellekti defineerimise kallal on töötatud pikka aega ning on saadud palju erinevaid vastuseid. Raamatus: "A Collection of Definitions of Intelligence" võtsid S. Legg ja M.Hutter intellekti mitmed definitsioonid kokku järgneva lausega: *"Intelligence measures an agent's ability to achieve goals in a wide range of environments."*

Siit jõuame tehisintellekti juurde ja mis see on. Ehk laias laastus saab öelda, et tehisintellekt on loomuliku intellekti jäljendamine [1]. Samal ajal tehisintellekti defineeritakse lisaks ka kui teadusharu, mis tegeleb uurimisega, kuidas luua tehnoloogiat, mis jäljendab inimintellekti, Eesti Entsüklopeedia (1998).

Vastava teadusharuga tehti algust aastal 1956 Dartmouth ülikoolis toimuval konverentsil. Mõned tuntud osalejad olid John McCarthy, Marvin Minsky, Allen Newell, Arthur Samuel, Herbert Simon [2]. Kuidas teada, et loodud edukalt tehisintellekt? Vastuseks on "Turingi test", mille pakkus välja Allan Turing aastal 1950. Masin läbib testi, kui arvuti suudab vastata inimkeeles esitatud küsimustele niimoodi, et küsimusi esitav inimene ei saa aru, kas tegu on inimese või masinaga, siis tehisintellekt on olnud edukas [3]. Tisteerides M. Minsky "Turing test is a joke" ning vastavas allikas Minsky räägib, et Turing test ei olnud kunagi mõeldud otsustamaks, kas masin on intelligentne, küll aga Turing test oli üks meetod hindamaks masinat [4].

Tehisintellekti arendamine on raske ja just seetõttu, et inimestel ei ole väga selget ettekujutust, kuidas inimeste mõtlemine toimib ja mis protsessid seal esinevad probleemide lahendamisel. Võimalik, et inimese intellekti tasemele ei jõuta veel niipea, küll aga on arendatud mitmeid häid meetodeid, kuidas lahendada komplekseid ülesandeid. Näiteks, 2016 märts seisuga on Googli isesõitvad autod läbinud üle kahe miljoni kilomeetri [5] ning juba aastal 1996 võitis arvuti, Deep Blue, male maailmameistri Gerry Kasparovi [6].

Probleemi lahendamisel tuleb määrata selline operatsioonide järjend, et jõutakse soovitud tulemuseni. Selleks defineeritakse olekute ruum, alg- ja lõppolek ning kirjeldatakse tegevused, mida saab antud olekutega teha [1]. Antud järjendi leidmiseks moodustatakse otsimispuu ning kasutatakse süstemaatilisi juhtimisstrateegiaid nagu näiteks laiutiotsing ja süvitsiotsing [1].

Laiutiotsingu algoritm on järgnev [1]:

1. Paiguta algolekule vastav tipp nimestikku L. Kui algolek on ühtlasi lõppolek, siis on lahend leitud, lõpeta.
2. Kui L on tühi, siis lahendit ei leidu, lõpeta.
3. Eemalda esimene tipp (n) nimestikust L.
4. Leia tipu n kõik vahetud järglased. Kui tipul n pole järglasi, siis mine 2.
5. Paiguta tipu n kõik vahetud järglased nimestiku L **lõppu**.
6. Kui mõni tipu n järglastest vastab lõppolekule, siis on lahend leitud, lõpeta. Muidu mine 2.

Süvitsiotsingu algoritm on järgnev [1]:

1. Paiguta algolekule vastav tipp nimestikku L. Kui algolek on ühtlasi lõppolek, siis on lahend leitud, lõpeta.
2. Kui L on tühi, siis lahendit ei leidu, lõpeta.
3. Eemalda esimene tipp (n) nimestikust L.
4. Leia tipu n kõik vahetud järglased. Kui tipul n pole järglasi, siis mine 2.
5. Paiguta tipu n kõik vahetud järglased nimestiku L **algusesse**.
6. Kui mõni tipu n järglastest vastab lõppolekule, siis on lahend leitud, lõpeta. Muidu mine 2.

# 3 Realisatsioon

## 3.1 Kasutatud tehnoloogiad

Selles alapeatükis tutvustatakse lugejat erinevate tehnoloogiatega, mis leidsid kasutust antud bakalaureusetöös. Kirjutatakse ülevaatlikult tehnoloogiast, miks just seda kasutati, mis olid alternatiivid ning kui kulukaks antud valikud läksid.

### 3.1.1 Haskell

Haskell on tuntud funktsionaalne programmeerimiskeel, mille väljalaskeaasta on 1990 ning nime on saanud Haskell Curry järgi. Haskelli põhitunnused on tüübiklassid, tüübi polüformism, näidiste sobitamine, laisk väärtustamine, listikomprehensioon ning kõrvalefektide puudumine [7].

Loodud programmis rakendatakse Haskellit teadmusbaasi parsimiseks ning otsimispuu loomiseks. Lisaks autori poolt loodud koodile, kasutatakse järgnevaid teeke: Data.List, Data.Text, Data.Map, Data.Char, System.Environments.

Haskell osutus valituks tänu oma keele eripäradele, mis võimaldavad tekstitöötlust ja puude moodustamist kiirelt ja efektiivselt sooritada. Alternatiiviks oleks sobinud mitmed teised keeled, millega autor on tuttav, nagu näiteks Java, C, C++, JavaScript, PHP, Prolog, Python. Algselt alustati rakenduse loomist Prologis, aga Prolog ei olnud tekstitöötluses piisavalt efektiivne.

### 3.1.2 Java

Java on objektorienteeritud programmeerimiskeel, mille tugev eelis on see, et kompileeritud Java koodi on võimalik jooksutada kõikidel platvormidel, kus on toetatud Java, hoolimata masina arhitektuurist. Algselt oli Java arendatud James Goslingu poolt, nüüdseks on õigused omandanud Oracle Corporation. Java on hetkel maailma kõige populaarsem programmeerimiskeel. Java süntaks sarnaneb C ja C++ keelega [8].

Lõputöö rakenduse graafiline liides on arendatud Javas. Täpsemalt: programmi käivitamine, teadmusbaasi modifitsseerimine, algoritmide valimine, otsimispuu redigeerimine, väljundi eksportimine. Teekidest kasutati javax.swing, java.awt, java.util.

Valituks osutus Java sellepärast, et lõputöö autor on Javaga kõige rohkem kursis ning Javas on mitmeid efektiivseid mooduleid kasutajaliidese loomiseks. Näiteks pakend java.awt läks kasutusse hiire funktsionaaluse võimaldamiseks ning java.swing läks menüü implementeerimiseks. Alternatiiviks oleks sobinud Python, Haskell, C++.

### 3.1.3 Git ja GitHub

Git on koodihaldus süsteem, mis on laialt kasutatud tarkvara arenduses. Git võimaldab lähtekoodi salvestada ning varasemaid versioone uuesti esile tuua. Antud tehnoloogia võimaldab töötada samaaegselt mitme erineva ülesande kallal [9]. GitHub on Giti veebipõhine keskkond, mis võimaldab Giti funktsionaalsustele ligi saada, ilma et peaks vastavat programmi lokaalselt installeerima [10].

Autor kasutas antud tehnoloogiaid, kuna Git on kõige populaarsem koodihaldus süsteem. Alternatiiviks oleks olnud Mercurial.

### 3.1.4 Eclipse

Eclipse on Integrated Development Environment (IDE) ehk keskkond kus arendadakse tarkvara. Eclipse võimaldab süntaksi kontrolle, programmi kiiret kompileerimist, ülevaadet koodistruktuurist ning lühiteid koodi kirjutamiseks ning modifitseerimiseks. Eclipse on populaarseim Java arenduskeskkond [11].

Lõputöö tarkvara arendus toimus Eclipses, kuna suurem osa koodist on kirjutada Javas. Alternatiiviks oleks sobinud Intellij või mõni kergekaalulisem nagu näiteks Notepad++, Sublime.

## 3.2 Süsteemi kasutuslood

Süsteemi funktsionaalsete nõuete haldamiseks kasutab autor kasutuslugusi. Kasutuslood on rakenduse nõuete edastamine stsenaariumite abil [12].

**Kasutuslugu 1. Teadmusbaasi loomine**

- **Kirjeldus:** Kasutaja saab luua uue teadmusbaasi
- **Tegutseja:** Rakendust kasutatav isik
- **Eeldused:** "Teadmusbaasi" tab on avatud
- **Põhivoog:**
    - Kasutaja valib nuppu "New"
    - Kasutaja kirjutab avanenud teksti alasse teadmusbaasi, vastavalt süntaksile.
- **Tulemus:** Avanenud tekstialas on kasutaja sisestatud teadmusbaas.

**Kasutuslugu 2. Teadmusbaasi salvestamine**

- **Kirjeldus:** Kasutaja saab salvestada teadmusbaasi
- **Tegutseja:** Rakendust kasutatav isik
- **Eeldused:** "Teadmusbaasi" tab avatud
- **Põhivoog:**
    - Kasutaja vajutab menüüs nuppu "Save" või "Save As"
    - Kui valiti "Save" ja algselt polnud teadmusbaas laetud, siis esitatakse kasutajale sisend, kuhu kasutaja saab sisestada soovitud faili nime.
    - Kasutajale esitatakse teade, et ta teadmusbaas on salvestatud.
- **Tulemus:** Teadmusbaasis olev tekst on salvestatud faili.

**Kasutuslugu 3. Teadmusbaasi laadimine**

- **Kirjeldus:** Kasutaja saab laadida teadmusbaasi
- **Tegutseja:** Rakendust kasutatav isik
- **Eeldused:** Rakendus on avatud.
- **Põhivoog:**
    - Kasutaja vajutab “Teadmusbaas” tabile.
    - Kasutaja valib nuppu “Load”
    - Avanenud aknas valib kasutaja faili.
- **Tulemus:** Teadmusbaasi tekstialas esitatakse valitud faili sisu.

**Kasutuslugu 4. Otsimispuu genereerimine**

- **Kirjeldus:** Kasutaja saab genereerida otsimispuu vastavalt ülesandele
- **Tegutseja:** Rakendust kasutatav isik
- **Eeldused:** Korrektne ülesande kirjeldus “teadmusbaas” tabis.
- **Põhivoog:**
    - Kasutaja vajutab “Puu” tabile.
    - Kasutaja valib alogritmi
    - Kasutaja valib menüüst “Run”
- **Tulemus:** Vastavalt algoritmile väljastatakse rakenduse otsimispuu

**Kasutuslugu 5. Otsimispuu modifitseerimine**

- **Kirjeldus:** Kasutaja saab liigutada otsimispuu tippe.s
- **Tegutseja:** Rakendust kasutatav isik
- **Eeldused:** Otsimispuu väljund on “puu” tabis.
- **Põhivoog:**
    - Kasutaja teeb topelt klõpsu tipul või selekteerib mitu tippu korraga hoides all vasakut hiirenuppu.
    - Kasutaja tirib märgistatud tippu/tippe ringi.
- **Tulemus:** Vastavalt kasutaja tirimisele on puu tipud liigutatud uutesse kohtadesse.

**Kasutuslugu 6. Pildi genereerimine väljundist**

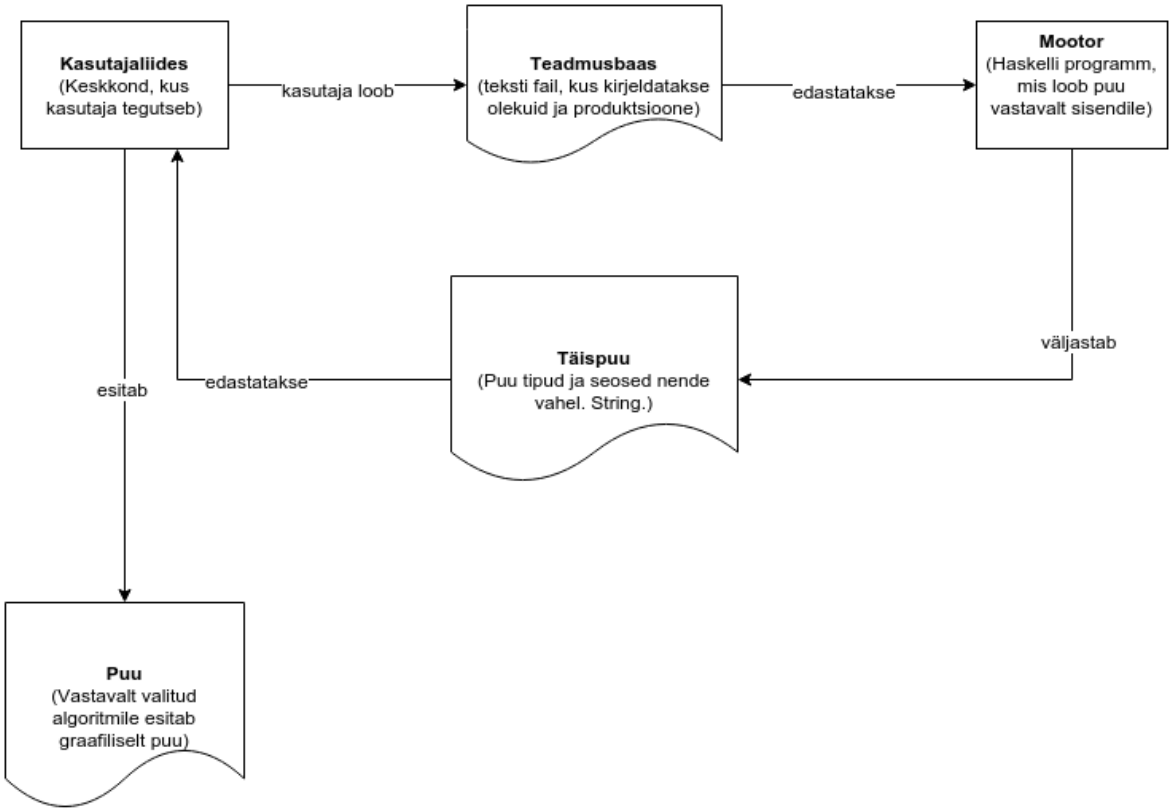
- **Kirjeldus:** Kasutaja saab genereerida pildi otsimispuust
- **Tegutseja:** Rakendust kasutatav isik
- **Eeldused:** Otsimispuu väljund on “puu” tabis.
- **Põhivoog:**
    - Kasutaja valib menüüst “snapshot”
    - Avanenud aknas sisestab faili nime.
- **Tulemus:** Vastavasse kausta luuakse .png fail otsimispuu väljundist.

## 3.3 Süsteemi arhitektuur

Järgnevalt kirjeldatakse rakenduse struktuurset poolt. Esitatakse mitmeid mudeleid ning seletusi nende kohta. Põhimõtteliselt, erinevad klassid ning mis on nende roll programmi koodis.

### 3.3.1 Üldine mudel

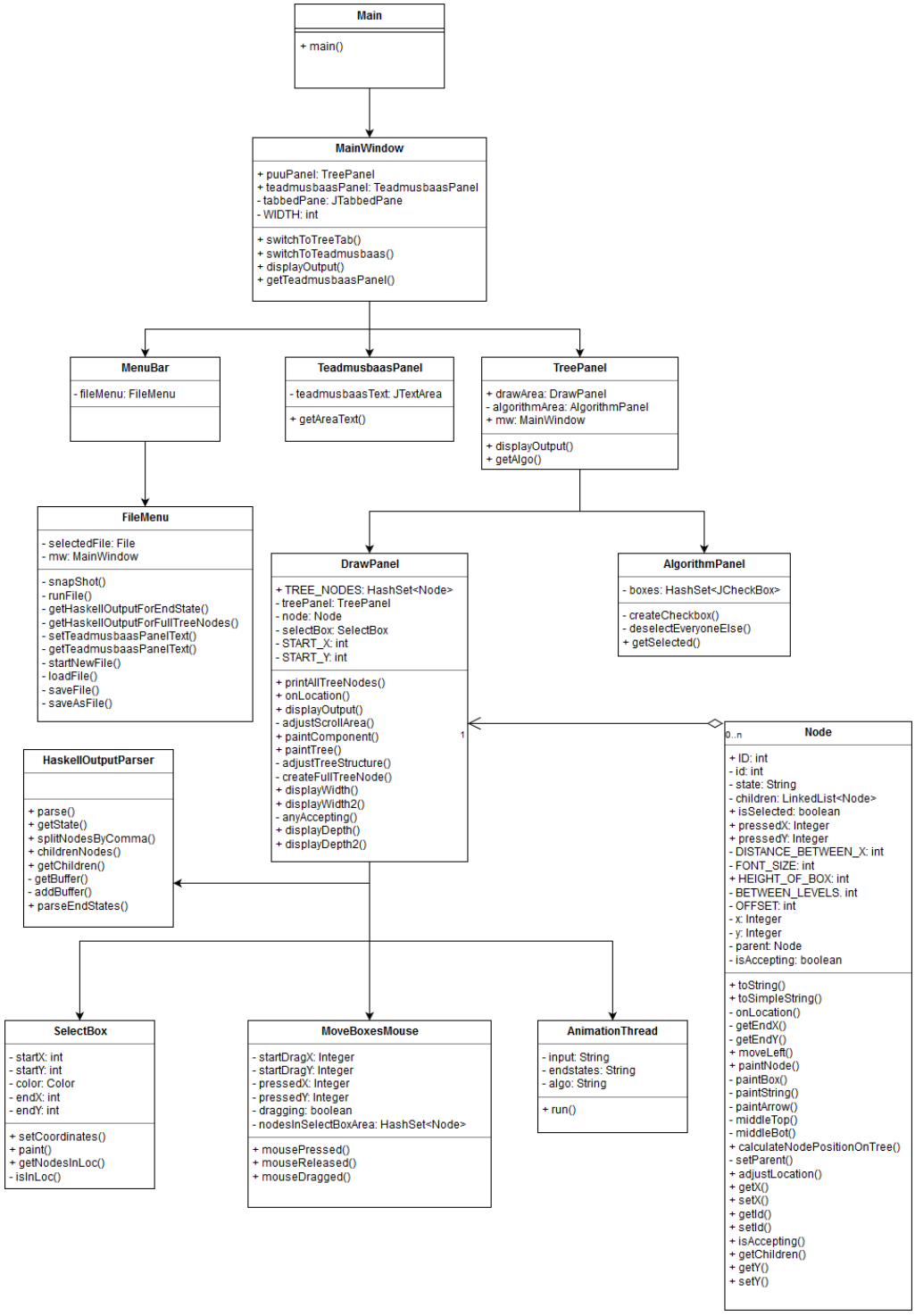
Joonisel 1 on mudel, mis näitab kuidas kasutajaliides, teadmusbaas, väljund ning otsimispuu on omavahel ühenduses.




Joonis 1 Üldine mudel süsteemi arhitektuurist

### 3.3.2 Kasutajaliidese mudel

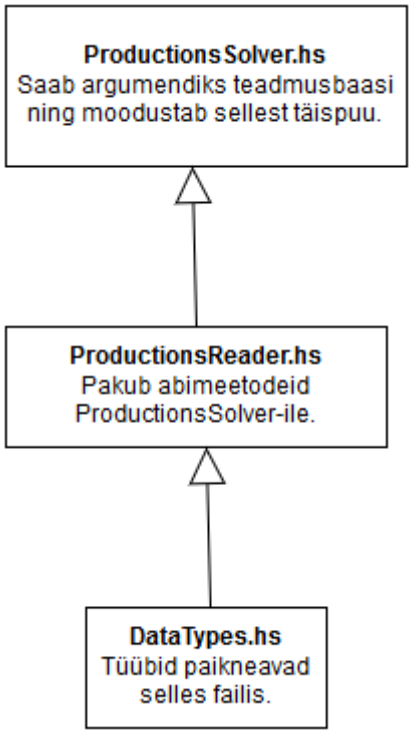
Antud graaf (vt. Joonis 2) esindab kasutajaliidese koodi struktuuri. Kõige mahukamad ja olulisemad klassid on DrawPanel ning Node. DrawPanelis on implementeeritud alogritmid ning animatsiooniga väljundi esitamine. Node klass esindab puu tippu, selles klassis on tipu graafiline esitus, asukoht ekraanil, ühendused teiste tippudega.




Joonis 2 Javas kirjutatud kasutajaliidese mudel

### 3.3.3 Haskelli mudel

Kuna Haskell on küllaltki efektiivne programmeerimiskeel, siis Haskelli osas on vaid kolm faili. Need failid sisaldavad mitmeid meetodeid, kuid lõputöö autor ei näinud põhjust neid selles mudelis (vt. Joonis 3) esitada.




Joonis 3 Üldine mudel Haskelli koodile

# 4 Tarkvara dokumentatsioon

Selles peatükis kirjeldatakse kuidas antud tarkvara kasutada ning kuidas see arvutisse seadistada.

## 4.1 Tarkvara nõuded

1. Java Runtime Environment (JRE) peab olema intalleeritud.
2. Linux. (Windowsis hetkel ei tööta.)

## 4.2 Installeerimisjuhend

1. Tarkvara zip link on leitav järgnevalt lingilt README failis https://github.com/saarthesis/opitarkvara .
2. Paki zip lahti soovitud asukohta.
3. Kasutajaliidese jooksutamiseks on vaja JRE. Kui see puudub, siis intallida saab selle siit: http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
4. Ubuntus käivita käsurealt “java –jar kasutajaliides.jar”.

## 4.3 Kasutamisjuhend

### 4.3.1 Teadmusbaasi süntaks

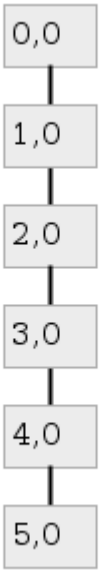
Järgnevalt kirjeldatakse teadmusbaasi süntaksit. Ehk kuidas kirja panna probleemiesitlus. Kõigpealt näidatakse grammatikat ja hiljem mõned näited reaalsete väljunditega rakendusest.

```
S:                Algolek Lõppolek Produktsioonid Not_allowed?
Algolek:          OlekD '\n'
Lõppolek:         OlekD (' ' OlekD)* '\n'
Produktsioonid:   Produktsioon Produktsioon*
Produktsioon:     Tingimus? Olek '->' OlekA '\n'
Tingimus:         'kui ' TingimusEnnikud ' siis '
TingimusEnnikud:  TingimusEnnik ('&&' TingimusEnnik)*
TingimusEnnik:    '(' LiigeA Võrdlus LiigeA  ')'
Võrdlus:          '>'|'<'|'=='|'/='|'>='|'<='
Olek:             '(' Liige  ( ',' Liige )*    ')'
OlekD:            '(' Digit  ( ',' Digit )*    ')'
OlekA:            '(' LiigeA ( ',' LiigeA)*    ')'
Liige:            Digit|Muutuja
LiigeA:           Liige (('+'|'-') Liige)*
Digit:            [0-9]
Muutuja:          [a-z]
Kommentaarid:     Kommentaar Kommentaar* -> skip()
Kommentaar:       '--' .* '\n'
Not_allowed:      'not allowed' OlekD (' ' OlekD)* '\n'
```

### 4.3.2 Teadmusbaasi süntaksi näiteid

Järgneva näite väljund on joonisel 4.

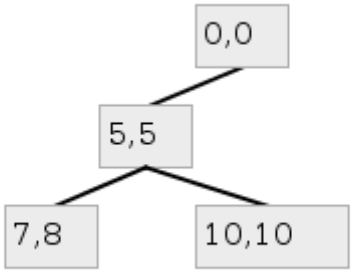
```
(0,0)
-- lõppolek nagu ikka.
(0,1)
-- lihtne tingimus
kui (x<5) siis (x,y)->(x+1,y)
```




Joonis 4 Väljundi näide

Järgneva näite väljund on joonisel 5.

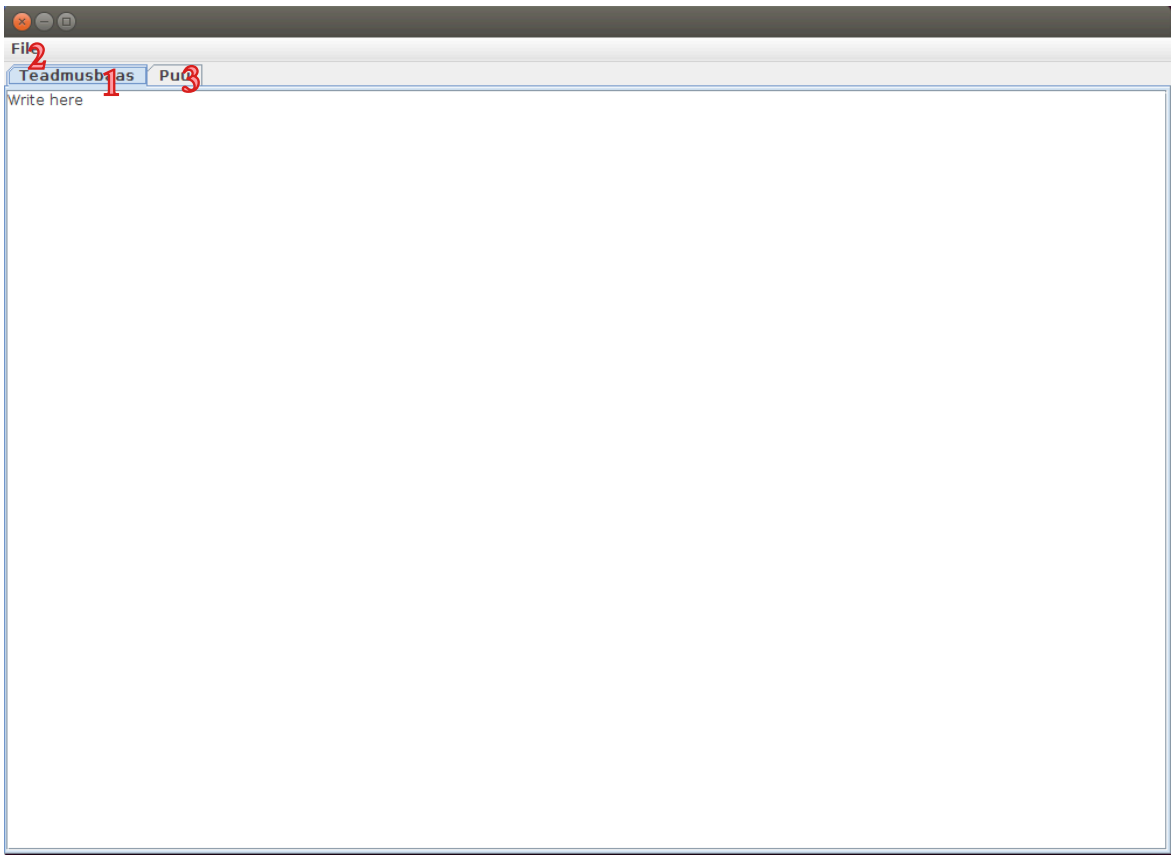
```
(0,0)
-- lõppolek nagu ikka.
(0,1)
-- arvutusega muutuja
kui (x<7)&&(y<7)&&(x==y) siis (x,y)->(x+5,y+5)
(5,5)->(7,8)
```



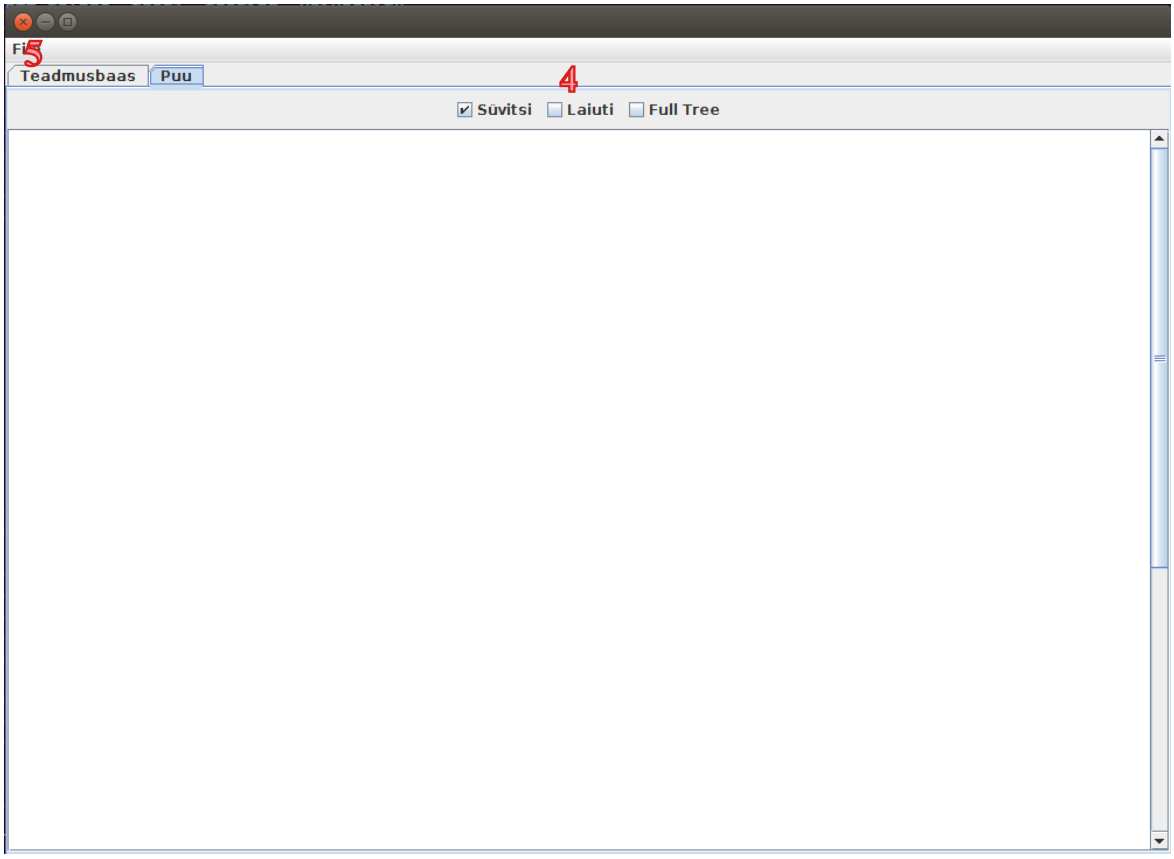

Joonis 5 Väljundi näide

### 4.3.3 Teadmusbaasi ja algoritmi valimine

1. Kliki „Teadmusbaas“ lipikule. (Punkt 1, Joonis 6)
2. Sisesta või lae teadmusbaas.
    - Sisestamiseks alusta trükkimist avanud tekstialas.
    - Laadimiseks kliki menüüs „File“ ja sealt „Load“. (Punkt 2, Joonis 6)
3. Kliki „Puu“ lipikule. (Punkt 3, Joonis 6)
4. Algoritmi valimiseks märgista kastike: „Süvitsi“, „Laiuti“, „Full Tree“ (Punkt 4, Joonis 7)
5. Menüüst „File“ vali „Run“. (Punkt 5, Joonis 7)
6. Menüüst „File“ vali „snapshot“ (Punkt 5, Joonis 7)




Joonis 6 Kasutajaliidese vaade




Joonis 7 Kasutajaliidese vaade

### 4.3.4 Ülesannete näidislahendusi

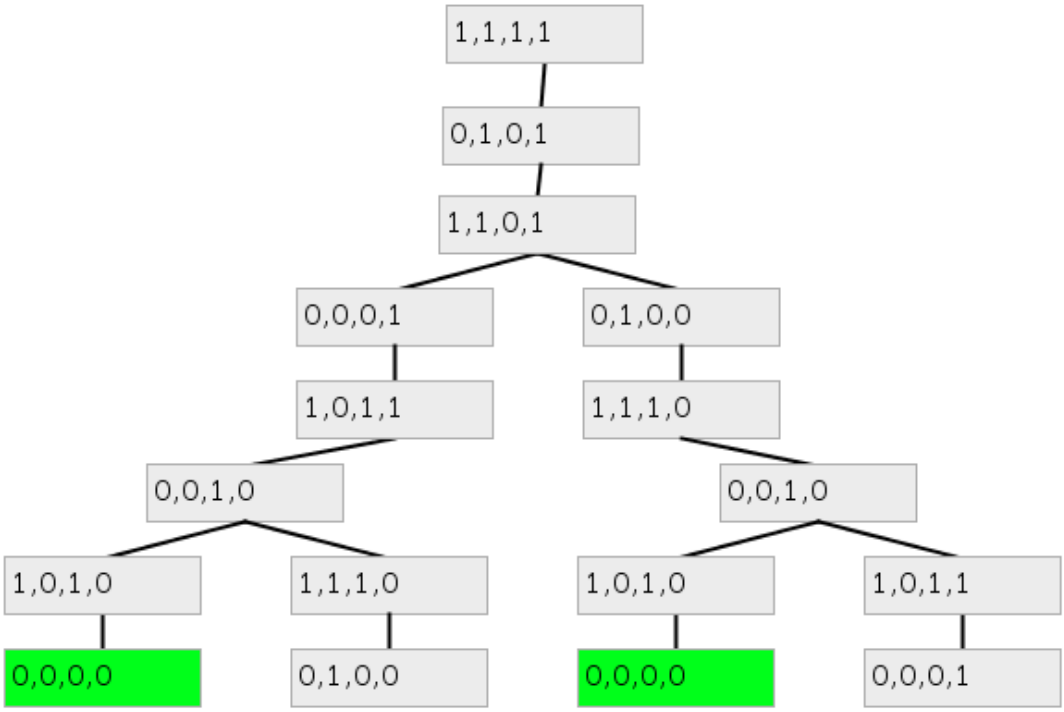
Selles peatükis tuuakse näiteid mõningatest ülesannete lahendustest. Nende teadmusbaas ja otsimispuu väljund. Ühele probleemile võib olla mitmeid erinevaid olekute ruumi definitsioone, seega kaks erinevat kasutajat võivad luua erinevad kirjeldused samale ülesandele. , Puu välimus sõltub, mis järjekorras produktsioonid on esitatud ja kuidas olek on defineeritud.

#### *4.3.4.1 Mees, rebane, hani ja vili*

*Mees peab viima rebase, hane ja vilja üle jõe, aga paati mahub kõige rohkem ainult kaks: tema ja kas vili või üks elukas (ei rebane ega hani ei oska aerutada). Kui rebane hanega üksi jätta, siis ta sööb hane ära. Kui hani viljaga üksi jätta, siis ta sööb vilja ära. Rebane vilja ei söö.*

```
-- defineerime oleku kui (mees,rebane,hani,vili)

-- algolek. Kõik on vasakul kaldal.
(1,1,1,1)

-- lõppolek. Kõik paremal kaldal.
(0,0,0,0)

-- produktsioonid

(1,r,1,v)->(0,r,0,v)
kui (r+h/=2) siis (1,r,h,1)->(0,r,h,0)
kui (h+v/=2) siis (1,1,h,v)->(0,0,h,v)
kui (h+v/=2)&&(r+h/=2) siis (1,r,h,v)->(0,r,h,v)
(0,r,0,v)->(1,r,1,v)
kui (r+h/=0) siis (0,r,h,0)->(1,r,h,1)
kui (h+v/=0) siis (0,0,h,v)->(1,1,h,v)
kui (r+h/=0)&&(h+v/=0) siis (0,r,h,v)->(1,r,h,v)
```

Antud ülesande lahendus on esitatud joonisel 8, tegu on täispuuga, kus lõppolekud on esitatud rohelise kastikesena.



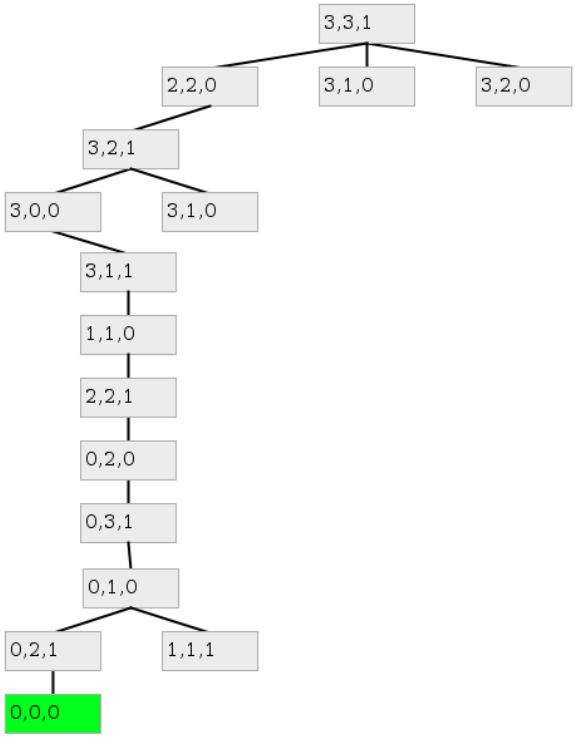

Joonis 8 Ülesande „Mees, rebane, hani, vili“ täispuu

#### *4.3.4.2 Armukadedad purjetajad*

*Armukadedad pühapäevapurjetajad. Kolm abielupaari on rannas ja otsustavad sõita kummipaadiga lähedal asuvale neemele piknikule. Paati mahub kõige rohkem kaks inimest, seega tuleb teha mitu sõitu. Probleem seisneb aga selles, et härrad on väga tundlikud selle suhtes, kes kui palju seltskonnas prouade tähelepanu saab. Seetõttu ei ole nad nõus, et ühelgi hetkel jääks randa või neemele rohkem prouasid kui seal on härraseid. Paadi jagamise suhtes neil pretensioone pole. Prouadel on härrade seebiooperist ükskõik, nemad ei hooli, kui palju kedagi korraga ühel või teisel pool lahte on. Kas sellel seltskonnal on võimalik neemele piknikule minna?*

```
-- first state
(3,3,1)
-- end state
(0,0,0)

-- vasakult:
-- mees
kui (m-1>=0) siis (m,n,1)->(m-1,n,0)
-- naine
kui (n-1>=0) siis (m,n,1)->(m,n-1,0)
-- mees mees
kui (m-2>=0) siis (m,n,1)->(m-2,n,0)
-- naine naine
kui (n-2>=0) siis (m,n,1)->(m,n-2,0)
-- mees naine
kui (m-1>=0)&&(n-1>=0) siis (m,n,1)->(m-1,n-1,0)

-- paremalt:
-- mees
kui (m+1<=3) siis (m,n,0)->(m+1,n,1)
-- naine
kui (n+1<=3) siis (m,n,0)->(m,n+1,1)
-- mees mees
kui (m+2<=3) siis (m,n,0)->(m+2,n,1)
-- naine naine
kui (n+2<=3) siis (m,n,0)->(m,n+2,1)
-- mees naine
kui (m+1<=3)&&(n+1<=3) siis (m,n,0)->(m+1,n+1,1)

not allowed (1,2,1) (1,3,1) (1,2,0) (1,3,0) (1,0,0) (2,3,1) (2,1,0) (2,3,0)
(2,1,1) (2,0,0)
```

Antud ülesande lahendus on joonisel 9, rakendatud on süvitsi otsimise algoritmi, kuni esimese lahenduseni.



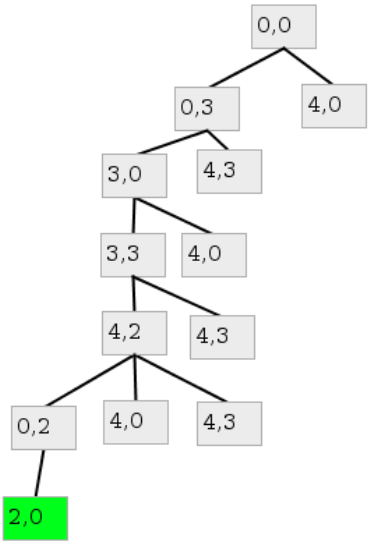

Joonis 9 Ülesande „Armukadedad purjetajad“ süvitsi lahendus

#### 4.3.4.3 Veekannude täitmine

*Antud on kaks kannu, mahuga vastavalt 4 l ja 3 l. Kanne võib täita veega, samuti valada vett ühest kannust teise või maha. Kuidas saavutada, et suuremas kannus oleks täpselt 2 l vett?*

```
-- algolek
(0,0)
-- lõppolek: esimeses kannus on täpselt 2 liitrit vett
(2,0) (2,1) (2,2) (2,3)

-- tühjendame kanne
(s,v)->(0,v)
(s,v)->(s,0)

-- täidame kanne
(s,v)->(4,v)
(s,v)->(s,3)

-- kallame ühest kannust teise
kui (s>=3-v) siis (s,v)->(s-3+v,3)
kui (s<3-v) siis (s,v)->(0,v+s)
kui (v>=4-s) siis (s,v)->(4,v-4+s)
kui (v<4-s) siis (s,v)->(s+v,0)
```

Antud ülesande lahendus on joonisel 10, rakendatud on süvitsi otsimise algoritmi.




Joonis 10 Ülesande „Veekannude täitmine“ süvitsi lahendus

# 5 Tulemuse analüüs

Loodud lõputöö saavutas oma eesmärgid ning kui peaks kellegil esinema soovi seda edasi arendada ja täiendada, siis on see täitsa võimalik. Koodi kvaliteet tagati läbi white-box ja black-box testimise ning õpitarkvara läbis beta- ja alfatestimise. Testimisprotsessi kaasati tarkvara autor ning kolmandad isikud, kes ei olnud varem antud programmiga tuttavad. Tarkvara töötas ja oli suuteline ülesandeid lahendama. Küll aga, programmis on erijuhte, kus kasutaja ebakorrektse käitumise peale, esineb süsteemis error, aga sellest ei teavitata isikut, et mida ta tegi valesti. Samuti, kui isik pole lugenud kasutusjuhendit, siis ta ei pruugi programmi käsitlemisest aru saada.

Üks sarnane realisatsioon on PathFinding.js [13]. Antud programm on tehtud ühele konkreetsele ülesandele, kus saab muuta algolekut ja lõppolekut. Algoritmide valik on mitmekülgne ning samuti on võimalik lisada olekuid, mis ei lubatud. Loodud õpitarkvaras on aga võimalik lahendada rohkem ülesandeid ning lahenduskäik on esitatud puuna. Tegu on sarnaste realisatsioonidega, kuid programmidel on natukene erinev rõhuasetus. Õpitarkvara põhirõhk on eelkõige erinevate nupuülesannete lahendamine, aga toodud lingis on rakenduse eesmärk visualiseerida võimalikult palju erinevaid algoritme.

# 6 Kokkuvõte

Käesoleva bakalaureusetöö käigus valmis tarkvara, mis aitab aine Tehisintellekt I nupuülesandeid lahendada. Rakendus vastab püsitatud eesmärkidele. Kasutajal ei pea olema eelnevaid programmeerimisteadmisi. Isik saab keskenduda ainult teadmuse esitusele. Programm on suuteline looma otsimispuu ülesannetele. Kasutajal on võimalik valida erinevate algoritmide vahel ning nende visuaalne esitus on selge ja arusaadav ning seega võimendab arusaamist antud ainest. Lisaks on võimalik väljundit eksportida ning kasutada seda õppekeskkonnas.

Kuna antud lõputöö maht osutus küllaltki suureks, siis valminud tarkvaras on nii mõndagi edasi arendada. Näiteks, puu tippude esituses võiks olla võimalik kasutada numbrite asemel sõnesi. Samuti oleks võimalik lisada algoritme, hetkel rakendus toetab süvitsi- ja laiutiotsingut. Arhitektuur on loodud, mille peale on võimalik implementeerida uusi algoritme küllaltki lihtsalt. Saab ka täiendada väljundit, osade ülesannete otsimispuu ei ole sümmeetriline, tippe saab küll hiirega liigutades kohendada. Lõpetuseks, tarkvara graafilist liidest saab muuta kasutajasõbralikumaks ning intuitiivsemaks.

Lõputöö kirjutamise käigus, omandas autor mitmeid uusi teadmisi ning mahukat tarkvaraarenduse kogemust. Kuna antud programm osutus väga mahukaks, siis mitmed kohad koodis jäid autoril südamele, et saaks teha paremini. Küll aga lõputöö täitis oma eesmärgid ning valmis konkreetne toode, siis võib olla rahul tulemusega.

# 7 Kasutatud materjal

[1] Mare Koit, Tiit Roosma, “Tehisintellekt”, Tartu, 2011.

[2] Pamela McCorduck, "Machines Who Think", 2004, pp. 111-136.

[3] Alan Turing, “Computing Machinery and Intelligence”, Mind, Vol. 69, pp. 433-460, 1950.

[4] “Marvin Minsky on Singularity 1 on 1: The Turing Test is a Joke!” [Internet] [viidatud 6.mail 2016] Kättesaadav aadressil: https://www.youtube.com/watch?v=3PdxQbOvAlI (min 24).

[5] „Google Self-Driving Car Project Monthly Report – March 2016“

[6] History. Kasparov loses chess game to computer. [Internet] [viidatud 6. mail 2016] Kättesaadav aadressil: http://www.history.com/this-day-in-history/kasparov-loses-chess-game-to-computer.

[7] Haskell Wiki. Introduction. [Internet] [viidatud 6. mail 2016] . Kättesaadav aadressil: https://wiki.haskell.org/Introduction.

[8] James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley, "The Java Language Specification", 2015.

[9] Git. [Internet] [viidatud 6.mail 2016] Kättesaadav aadressil: https://git-scm.com/.

[10] GitHub. [Internet] [viidatud 6.mail 2016] Kättesaadav aadressil: https://github.com/.

[11] Eclipse. [Internet] [viidatud 6.mail 2016] Kättesaadav aadressil: www.eclipse.org.

[12] Martin Fowler, „UML Distilled Third Edition“, 2004, p. 79.

[13] PathFinding.js. [Internet] [viidatud 11.mail 2016] Kättesaadav aadressil: https://qiao.github.io/PathFinding.js/visual/ .

# Lisad

## I. Litsents

**Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks**

Mina, **Harri Saar**,

(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **Aine Tehisintellekt I õpitarkvara**,
   (*lõputöö pealkiri*)

   mille juhendaja on Tõnu Tamme,

   (*juhendaja nimi*)

   1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **05/08/16**