Sakari Mäkinen

Havainto- ja sensomotoristen taitojen testausohjelma

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

21.11.2013

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Sakari Mäkinen Havainto- ja sensomotoristen taitojen testausohjelma 51 sivua + 2 liitettä 21.11.2013
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Simo Silander Yliopettaja Erja Nikunen

Insinöörityössä toteutettiin tietokoneohjelma, jolla voidaan testata sekä harjoittaa erilaisia havainto- ja sensomotorisia taitoja. Ohjelma sisältää pääosa-alueinaan kirjoitusnopeustestin, rämpytysnopeustestin, reaktionopeustestin, muistitestin ja sekuntitarkkuustestin. Muita osa-alueita ovat tekstieditori, ennätyslistojen ylläpito, henkilökohtaiset tulokset, ohjeet ja ohjelman tiedot.

Ohjelma toteutettiin Java-ohjelmointikielellä MVC-suunnittelumallin mukaisesti työasemaympäristössä käytettäväksi. Ohjelma käyttää tietojen eli ennätyslistojen, tekstien, asetusten ja ohjeiden pysyvään tallennukseen tietokoneen kovalevyn tietokantaa.

Insinöörityön tulosta eli testausohjelmaa voitaisiin hyödyntää viihdetarkoituksessa sen pelimäisen luonteen vuoksi sekä myös mahdollisesti kuntoutustarkoituksessa. Ohjelmaa voitaisiin kehittää eteenpäin yhteistyössä hyvinvointiteknologian asiantuntijoiden kanssa.

Avainsanat	havaintomotoriikka, sensomotoriikka, tietokoneohjelma, testausohjelma, tietokanta

Author(s) Title Number of Pages Date	Sakari Mäkinen A Testing Software for Perceptual and Sensorimotor Skills 51 pages + 2 appendices 21 November 2013
Degree	Bachelor of Engineering
Degree Program	Information Technology
Specialization Option	Software Engineering
Instructor(s)	Simo Silander, Senior Lecturer Erja Nikunen, Principal Lecturer

The Bachelor's thesis implemented a computer software that can be used to test and practice different kinds of perceptual and sensorimotor skills. The software's main areas are a writing speed test, a clicking speed test, a reaction speed test, a memory test, and a time accuracy test. Other areas include a text editor, maintenance of the records lists, personal results, and the guidelines and basic information for the software.

The software was implemented according to the Model-View-Controller design model using the Java programming language and is used in a workstation environment. The software uses a database on the hard drive to permanently store its data in. The saved data includes record lists, texts, preferences, and guidelines.

The testing software, the end product of the Bachelor's thesis, could, because of its game-like nature, be utilized for entertainment purposes as well as possibly for rehabilitation purposes. The software could be further developed together with experts from the field of medical engineering.

Keywords	perceptual motor, sensorimotor, computer software, testing software, database

Sisällys

1 Johdanto 1

2 Toiminnallinen määrittely 2

2.1 Toiminnalliset vaatimukset 2

2.2 Tilannekuvaukset ja käyttötapaukset 4

2.3 Ominaisuudet 4
2.3.1 Testit 5
2.3.2 Tekstieditori 6
2.3.3 Testien ennätyslistat 6
2.3.4 Ohjeet 6
2.3.5 Ohjelman tiedot 6

3 Tekninen määrittely 7

3.1 Tekniikat 7

3.2 Ohjelmistot 7

3.3 Ohjelmassa käytettävät valmiit Javan kirjastot 8
3.3.1 Swing ja AWT 8
3.3.2 Muut kirjastot 9

3.4 Menetelmät 10
3.4.1 MVC-suunnittelumalli 10
3.4.2 Observer-suunnittelumalli 12
3.4.3 Säikeet 12

3.5 Valmiita rajapintoja ja luokkia 13

4 Ohjelman testien tarkoitukset ja tulokset 16

4.1 Kirjoitusnopeustesti 16

4.2 Rämpytysnopeustesti 17

4.3 Reaktionopeustesti 17

4.4 Muistitesti 17

4.5 Sekuntitarkkuustesti 18

5 Tietokanta 18

5.1 Tietokanta-ajurit 20

5.2 DAO-suunnittelumalli 20
5.2.1 DAO-suunnittelumallin rakenne 21
5.2.2 Ohjelman yhteinen DAO-luokka 21

5.3 Tietokantataulut 22
5.3.1 Ennätystaulut 22
5.3.2 Nimitaulu-taulu 23
5.3.3 Tekstitaulut 23
5.3.4 Asetukset-taulut 24
5.3.5 OhjeetTaulu-taulu 24
5.3.6 Taulujen nimiä sisältävät taulut 25

6 Toteutus 25

6.1 Luokkakaavio 26

6.2 Käyttöliittymä 29
6.2.1 Tervetuloa-välilehti 29
6.2.2 Kirjoitusnopeustesti-välilehti 29
6.2.3 Rämpytysnopeustesti-välilehti 31
6.2.4 Reaktionopeustesti-välilehti 32
6.2.5 Muistitesti-välilehti 34
6.2.6 Sekuntitarkkuustesti-välilehti 35
6.2.7 Ennätyslistan nimikysely 36
6.2.8 Ennätyslistat-ikkuna 37
6.2.9 Henkilökohtaiset tulokset -ikkuna 37
6.2.10 Tekstieditori-ikkuna 39
6.2.11 Ohjeet-ikkuna 40
6.2.12 Tiedot-ikkuna 41

7 Testaus 41

7.1 Yleistä 41

7.2 Ohjelman testaus 42

8 Pohdintaa 42

8.1 Toteutetun ohjelman arviointia 42
8.1.1 Pääikkuna 43
8.1.2 Muut ikkunat 45

8.2 Toteutusongelmia 46

8.3 Ohjelman jatkokehitys 47
8.3.1 Hoitoalalle toteutettu ohjelma 47
8.3.2 Internetin pilvipalveluja hyödyntävä ohjelma 48

9 Yhteenveto 48

Lähteet 50

Liitteet

Liite 1. Tilannekuvaukset

Liite 2. Käyttötapaukset

1 Johdanto

Tässä luvussa esitellään insinöörityön tarkoitus ja tavoite. Insinöörityön tarkoituksena on luoda ohjelma, jolla nuoriso ja muut kiinnostuneet voisivat testata sekä harjoittaa tietoteknisiä taitojaan, muistiaan ja tarkkuuttaan erilaisilla testeillä. Ohjelma voisi olla hyödyllinen myös esimerkiksi terapiana, jolla hoidetaan potilaiden erilaisia vaivoja, kuten sormien jäykkyyttä, muistia ja reaktiokykyä. Ohjelmassa on viisi pääosa-aluetta: kirjoitusnopeustesti, rämpytysnopeustesti, reaktionopeustesti, muistitesti ja sekuntitarkkuustesti. Testeillä testataan käyttäjän havainto- ja sensomotorisia taitoja. Havaintomotoriikalla tarkoitetaan tapahtumasarjaa, jossa ihminen käsittelee tietoa itsestään ja ympäristöstään eri aistiensa avulla tuottaakseen tilanteeseen sopivan motorisen vasteen, esimerkiksi käden puristuksen. Havaintomotorinen toiminta on pääosin automatisoitunutta ja tiedostamatonta. [1.] Sensomotoriikka taas on aistien, aivojen, tukirangan ja hermolihasjärjestelmän monitieteistä toimintaa [2]. Muut ohjelman osat liittyvät edellä mainittuihin testeihin: Tekstieditori-ikkunassa on mahdollisuus muokata kirjoitusnopeustestin harjoitusmoodin harjoitustekstejä. Ennätyslistat-ikkunassa voi tarkastella ja tarvittaessa nollata testien ennätyslistoja. Henkilökohtaiset tulokset -ikkunassa voi tarkastella ohjelman käyttäjien tuloksia ja sijoituksia testien ennätyslistoissa. Ohjeet-ikkunassa voi perehtyä ohjelman käyttöohjeisiin ja Tiedot-ikkunassa tarkistaa ohjelman nimen, version, toteutuspäivämäärän ja tekijän. Insinöörityön aihe on lähtöisin omista mielenkiinnon kohteistani.

Tavoitteena on, että ohjelma hyödyntää tietojen tallennuksessa tietokoneen kovalevyn tietokantaa. Ohjelman hyödyntämä tietokanta mahdollistaa sen, että tietokoneen eri käyttäjillä on käytössään samat tietokantaan keskitetysti tallennetut tiedot eli testien ennätyslistat, kirjoitusnopeustestissä kysyttävät tekstit, rämpytys- ja reaktionopeustestin asetukset sekä ohjelman ohjeet. Testien ennätyslistat antavat käyttäjille mahdollisuuden asettaa tavoitteita ja kilpailla keskenään. Ne myös lisäävät ohjelman kiinnostavuutta ja käyttöikää.

2 Toiminnallinen määrittely

Tässä luvussa esitellään toiminnallinen määrittely. Luvussa vastataan kysymykseen, mitä toteutettava ohjelma tekee. Luvussa kootaan toteutettavan ohjelman toiminnalliset vaatimukset, esitetään käyttäjän tyypilliset tilannekuvaukset ja käyttötapaukset sekä kerrotaan ohjelman ominaisuudet.

2.1 Toiminnalliset vaatimukset

Insinöörityö toteutetaan koululle. Alkuvaiheessa mietitään, mitä toteutettavalla ohjelmalla tulisi voida tehdä. Ajatukset kirjataan ja kootaan oheinen luettelo toiminnallisista vaatimuksista, jotka toteutettavalla ohjelmalla tulee voida suorittaa ja jotka ohjelmassa tulee olla. Oheisen luettelon tilannekuvaukset ovat liitteessä 1:

- Käyttäjä voi valita suoritettavan testin (Tilannekuvaus 1: Testin valitseminen).
- Käyttäjä voi testata kirjoitusnopeuttaan (Tilannekuvaus 2: Kirjoitusnopeustestin suorittaminen).
- Käyttäjä voi testata sorminäppäryyttään (Tilannekuvaus 3: Rämpytysnopeustestin suorittaminen).
- Käyttäjä voi testata reaktionopeuttaan (Tilannekuvaus 4: Reaktionopeustestin suorittaminen).
- Käyttäjä voi testata muistiaan (Tilannekuvaus 5: Muistitestin suorittaminen).
- Käyttäjä voi testata ajantajuaan (Tilannekuvaus 6: Sekuntitarkkuustestin suorittaminen).
- Käyttäjä voi katsella testien ennätyslistoja tietokoneen kovalevyltä tietokannasta (Tilannekuvaus 7: Ennätyslistojen tarkasteleminen).
- Käyttäjä voi nollata testien ennätyslistoja tietokoneen kovalevyltä tietokannasta (Tilannekuvaus 8: Ennätyslistojen nollaaminen).

- Käyttäjä voi muokata kirjoitusnopeustestin harjoitustekstejä tekstieditorilla (Tilannekuvaus 10: Tekstin muokkaaminen tekstieditorilla tietokoneen kovalevyltä).

- Käyttäjä voi katsella ohjelman ohjeita (Tilannekuvaus 11: Ohjelman ohjeiden tarkasteleminen).

- Käyttäjä voi katsella ohjelman tietoja (Tilannekuvaus 12: Ohjelman tietojen tarkasteleminen).

- Käyttäjä voi valita kirjoitusnopeustestin harjoitus- ja kilpailumoodin (Tilannekuvaus 2: Kirjoitusnopeustestin suorittaminen).

- Käyttäjä voi valita reaktionopeus- ja muistitestin vaikeusasteen (Tilannekuvaus 4: Reaktionopeustestin suorittaminen) (Tilannekuvaus 5: Muistitestin suorittaminen).

- Käyttäjä voi valita kirjoitusnopeus-, rämpytysnopeus- ja sekuntitarkkuustestin keston (Tilannekuvaus 2: Kirjoitusnopeustestin suorittaminen) (Tilannekuvaus 3: Rämpytysnopeustestin suorittaminen) (Tilannekuvaus 6: Sekuntitarkkuustestin suorittaminen).

- Käyttäjä voi tutkia kirjoitus- ja rämpytysnopeustestin nopeuden keskiarvokäyrää (Tilannekuvaus 2: Kirjoitusnopeustestin suorittaminen) (Tilannekuvaus 3: Rämpytysnopeustestin suorittaminen).

- Käyttäjä voi suorittaa rämpytys- ja reaktionopeustestin näppäimistöllä (Tilannekuvaus 3: Rämpytysnopeustestin suorittaminen) (Tilannekuvaus 4: Reaktionopeustestin suorittaminen).

- Käyttäjä voi katsella henkilökohtaisia tuloksiaan tietokoneen kovalevyltä tietokannasta (Tilannekuvaus 9: Henkilökohtaisten tulosten tarkasteleminen).

- Käyttäjälle näytetään kirjoitusnopeustestin keskinopeus (Tilannekuvaus 2: Kirjoitusnopeustestin suorittaminen).

- Käyttäjälle näytetään rämpytysnopeustestin reaktioaika ja nopeuden keskihajonta (Tilannekuvaus 3: Rämpytysnopeustestin suorittaminen).

- Ohjelma käyttää tietokoneen kovalevylle tallennettua .properties-tiedostoa, jonka sisältämiä lähdekoodin muuttujia muokkaamalla käyttäjä voi kalibroida ohjelman toimintaa haluamakseen.

2.2 Tilannekuvaukset ja käyttötapaukset

Edellä esitettyjen toiminnallisten vaatimusten pohjalta kirjoitetaan ohjelman käyttäjän tyypillisiä tilannekuvauksia. Toteutettavan ohjelman on tarkoitus olla sellainen, että tilannekuvaukset voidaan tarvittaessa toistaa saavuttaen ennalta sovittu lopputulos. Ohjelman tilannekuvaukset ovat liitteessä 1.

Käyttötapausmalli kuvaa käyttäjän ja ohjelman välistä vuorovaikutusta [3]. Käyttötapauskaavioita mallinnettaessa ei vielä tiedetä, miten ohjelman toiminnallisuus toteutetaan, eikä mallinnusvaiheessa myöskään välitetä ohjelman rakenteesta. Mallin suhteen ohjelma on musta laatikko. [4.] Ohjelman käyttötapaukset ovat liitteessä 2.

2.3 Ominaisuudet

Insinöörityössä luodaan TestYourself-ohjelma, jolla jokainen kiinnostunut voi testata sekä harjoittaa esimerkiksi tietoteknisiä taitojaan, muistiaan ja tarkkuuttaan erilaisilla testeillä. Ohjelma voisi olla hyödyllinen myös terapiana, jolla hoidetaan potilaiden erilaisia vaivoja, kuten sormien jäykkyyttä, muistia ja reaktiokykyä. Ohjelma hyödyntää tietojen tallennuksessa tietokoneen kovalevyn tietokantaa. Ohjelman hyödyntämä tietokanta mahdollistaa tietokoneen eri käyttäjille samat ennätyslistat, kirjoitusnopeustestissä kysyttävät tekstit, rämpytys- ja reaktionopeustestin asetukset sekä ohjelman ohjeet. Ohjelma sisältää pääosa-alueinaan viisi testiä: kirjoitusnopeustestin, rämpytysnopeustestin, reaktionopeustestin, muistitestin ja sekuntitarkkuustestin.

2.3.1 Testit

Kirjoitusnopeustestissä testataan käyttäjän taitoja lukea ja kirjoittaa pyydettyä tekstiä mahdollisimman nopeasti ja virheettömästi. Testissä on harjoitusmoodi, jossa testattava harjoitusteksti voidaan valita, ja kilpailumoodi, jossa ohjelma arpoo testattavan tekstin. Testin kestoksi voidaan valita 1, 5 tai 10 minuuttia. Kirjoitettaessa tekstiä virheellisesti väärät sanat värjätään punaisella ja virheet voidaan korjata heti tai myöhemmin näppäimistöllä tekstissä liikkuen. Hiirellä ei ole mahdollista liikkua tekstissä. Testi lopetetaan keston ollessa nolla, tai kun teksti on kirjoitettu kokonaisuudessaan oikein.

Rämpytysnopeustestissä testataan käyttäjän taitoja rämpyttää haluamaansa painiketta mahdollisimman nopeasti. Testin kestoksi voidaan valita 10 s, 30 s, 1, 5 tai 10 minuuttia. Testissä voidaan käyttää myös asetuksissa määriteltävää näppäimistön painiketta, joka tallennetaan tietokoneen kovalevylle tietokantaan. Määritelty painike näytetään käyttöliittymän selitteessä. Testi ilmoittaa käyttäjän reaktionopeuden lähtölaskennan jälkeen ja testin päätyttyä myös rämpytysnopeuden keskihajonnan. Kirjoitus- ja rämpytysnopeustestin edetessä piirretään reaaliaikaisesti keskiarvokäyrää, jossa esitetään käyttäjän kirjoitusnopeutta ja rämpytyksen tasaisuutta. Rämpytysnopeustesti lopetetaan keston ollessa nolla.

Reaktionopeustestissä testataan käyttäjän reagointikykyä ja nopeutta. Testissä on neljä eriväristä painiketta, joita valaistaan satunnaisessa järjestyksessä jatkuvasti nopeutuvassa tahdissa. Testin vaikeusasteeksi voidaan valita normaali tai helppo, mikä vaikuttaa testin nopeuteen. Testissä voidaan käyttää myös asetuksissa määriteltäviä näppäimistön painikkeita, jotka tallennetaan tietokoneen kovalevylle tietokantaan. Määritellyt painikkeet näytetään käyttöliittymän selitteessä. Testi lopetetaan, jos valaistuja painikkeita klikataan liian hitaasti, jos klikataan väärää painiketta tai jos painikkeita ei klikata ollenkaan.

Muistitestissä testataan käyttäjän lyhytkestoista muistia keskustelulla ohjelman kanssa. Testissä näytetään alfanumeerisia merkkejä käyttöliittymässä noin 10 – 2 sekuntia nopeutuen sekunnilla joka kierroksella. Merkkien näyttämisen jälkeen kirjoitetaan noin 30 sekunnin kuluessa näytetyt merkit samassa järjestyksessä ohjelman ruudukkoon. Alfanumeeriset merkit ovat joko kirjaimia tai numeroita [5]. Testin vaikeusasteeksi voidaan valita normaali tai helppo, jolloin kysyttyjen merkkien lukumäärä on joko 16 tai 8. Testin ja kierroksen oikeiden vastausten lukumäärät näytetään prosentteina joka kierroksella.

Prosenttien väheneminen ilmaistaan punaisella ja kasvaminen vihreällä värillä. Testin kokonaiskesto vähenee jokaisella kierroksella väärien vastausten lukumäärän mukaan, ja testi lopetetaan keston ollessa nolla.

Sekuntitarkkuustestissä testataan käyttäjän tarkkuutta tunnistaa ajanjaksoja. Testissä klikataan painiketta testin alussa ja uudestaan päässä arvioidun ajan jälkeen, jolloin testi lopetetaan. Testin kestoksi voidaan valita 2 s, 10 s tai 1 minuutti. Testin lopuksi näytetään valitun keston ja käyttäjän aika-arvion välinen erotus sekunteina.

2.3.2 Tekstieditori

Ohjelmassa on tekstieditori kirjoitusnopeustestin harjoitusmoodin harjoitustekstien muokkaamiseen. Harjoitustekstit voidaan hakea tietokoneen kovalevyltä tietokannasta ja tallentaa sinne myöhempää käyttöä varten. Tekstieditorissa tietokannassa jo oleva teksti voidaan ylikirjoittaa tai teksti voidaan tallentaa uutena.

2.3.3 Testien ennätyslistat

Jokaisella testillä on omat ennätyslistansa, jotka vaihtelevat testin keston, reaktioajan tai vaikeusasteen mukaan. Testien ennätyslistat tallennetaan tietokantaan tietokoneen kovalevylle. Tietokantaan tallennettuja ennätyslistoja voidaan tarkastella ja myös nollata suoraan ohjelmasta. Ohjelmassa näytetään myös käyttäjien henkilökohtaiset tulokset käyttäjän nimen perusteella. Tällöin jokaisesta ennätyslistasta haetaan kyseisen käyttäjän mahdolliset tulokset ja näytetään ne listattuna käyttöliittymässä.

2.3.4 Ohjeet

Ohjelmassa on oma osionsa myös ohjeille, jotka näytetään järjestysnumeroin käyttöliittymässä. Ohjeet tallennetaan tietokantaan tietokoneen kovalevylle.

2.3.5 Ohjelman tiedot

Ohjelmassa näytetään myös tiedot, jotka ovat ohjelman nimi, versio, toteutuspäivämäärä ja tekijä.

3 Tekninen määrittely

Tässä luvussa esitellään tekninen määrittely. Luvussa vastataan kysymykseen, miten toteutettava ohjelma tekee toimintonsa. Luvussa esitellään ensin käytettävät tekniikat ja ohjelmistot. Sen jälkeen esitellään ohjelmassa käytettävät valmiit Javan kirjastot ja toteutusmenetelmät. Lopuksi vielä käydään läpi käytettäviä valmiita rajapintoja ja luokkia.

3.1 Tekniikat

Insinöörityössä luotava ohjelma toteutetaan Java-ohjelmointikielellä, koska sillä ohjelmoiminen on selkeää, suoraviivaista ja käyttöliittymien tekeminen helppoa. Java on Sun Microsystemsin vuonna 1995 julkaisema alustariippumaton ohjelmointikieli. Alustariippumattomuudella tarkoitetaan sitä, että ohjelmointikieli ei ole sidoksissa tiettyyn laitteistoalustaan tai käyttöjärjestelmään, kuten esimerkiksi Microsoft Windows -ohjelmistoon [6]. Sun Microsystems oli Javan kehittäjä vuoteen 2009 saakka, jolloin yritys myytiin Oraclelle. [7.]

Toteutettava ohjelma käyttää kovalevyllä olevaa tietokantaa tietojen pysyvään tallennukseen. Tietokannan hallinnassa käytetään SQL eli Structured Query Language -kieltä, joka on IBM:n vuonna 1986 julkaisema ja kehittämä standardoitu kyselykieli, jolla tietokantaan voidaan tehdä hakuja, lisäyksiä ja muutoksia [8].

3.2 Ohjelmistot

Ohjelman toteutuksessa ohjelmointiympäristönä käytetään Eclipse-ohjelmiston versioita Eclipse Juno 4.2, Eclipse Kepler 4.3 ja Eclipse Kepler 4.3.1. Eclipse on ilmainen ohjelmisto, ja se on julkaistu avoimen lähdekoodin lisenssille. Avoimen lähdekoodin ohjelmistoissa käyttäjälle annetaan mahdollisuus muokata ohjelman lähdekoodia haluamallaan tavalla. Eclipsen kehityksen aloitti IBM vuonna 1993, mutta vuonna 2001 ohjelmisto julkaistiin avoimen lähdekoodin lisenssille. Ohjelmiston kehityksestä vastaa nykyisin Eclipse Foundation. [9.]

Ohjelman käyttämien relaatiotietokantojen toteutukseen ja ylläpitoon käytetään Microsoft Access 2003 -ohjelmiston versiota 11.8321.8405. Microsoft julkaisi ensimmäisen versionsa Access-ohjelmastaan marraskuussa 1992. Ohjelman uusin vakaa versio on vuonna 2010 julkaistu 14.0. [10.]

Käyttötapaus- ja luokkakaavioiden toteuttamiseen käytetään Umlet-ohjelman versiota 11.4. Se on julkaistu avoimen lähdekoodin lisenssille.

3.3 Ohjelmassa käytettävät valmiit Javan kirjastot

Ohjelmassa käytetään monia valmiita Javan kirjastoja. Pääasiallisia kirjastoja ovat Swing ja AWT eli Abstract Windowing Toolkit. Muita hyödynnettäviä kirjastoja ovat Util, IO, SQL ja Text.

3.3.1 Swing ja AWT

Ohjelman graafinen käyttöliittymä luodaan Swing-kirjastolla. Swing on kevyt kirjasto Javalle, joka tarjoaa kokoelman valmiiksi toteutettuja käyttöliittymäkomponentteja, joilla graafinen käyttöliittymä rakentuu kätevästi. Swing-luokka tarjoaa myös työkaluja esimerkiksi JTextPane-komponentissa olevan tekstin reaaliaikaiseen seurantaan ja muokkaamiseen. Kyseisistä työkaluista hyödynnetään DocumentListener-rajapintaa kirjainten ja sanojen laskemiseen sekä tekstin sivujen vaihtamiseen, DocumentFilter-luokkaa tekstin sisällön reaaliaikaiseen suodattamiseen ja kirjoittamisen estämiseen sekä virheiden tarkistukseen ja lisäksi Document-rajapintaa JTextPane-komponenttiin kirjoitetun tekstin reaaliaikaiseen muokkaamiseen ja sanojen värittämiseen. Lisäksi hyödynnetään ChangeListener-rajapintaa, joka kuuntelee JTabbedPane-komponentin välilehtien vaihtumista. Swing perustuu aikaisemmin kehitettyyn AWT-käyttöliittymäkirjastoon, ja monet Swing-kirjaston luokat perivätkin aikaisemmat AWT-luokat [11, s. 203]. Swingin käyttöliittymäkomponentit ovat kevyitä, eli ne on kirjoitettu kokonaan Javalla. Tästä syystä ne eivät käytä käyttöjärjestelmän omia graafisia komponentteja. Swing-kirjaston etuna on, että käyttöliittymäkomponentit ovat samannäköisiä ja toimivat samalla tavalla käyttöjärjestelmästä riippumatta. [12.]

Ohjelmassa käytetään Swing-kirjaston lisäksi pääasiassa myös AWT-kirjastoa. Swing-komponentit aiheuttavat tapahtumia, jotka käsitellään käyttöliittymäluokkien anonyy-

meissä sisäluokissa. Lisäksi AWT-kirjasto tarjoaa työkalut Graphics- ja Graphics2D-luokat komponenttien, kuten kirjoitus- ja rämpytysnopeustestin keskiarvokäyrän ja reaktionopeustestin pyöreiden JButton-painikkeiden piirtämiseen sekä BorderLayout-luokan komponenttien asemointiin käyttöliittymässä. AWT-käyttöliittymäkomponentit ovat raskaita eli ne käyttävät käyttöjärjestelmän omia graafisia komponentteja [12]. Tästä syystä graafinen käyttöliittymä vaihtelee eri käyttöjärjestelmissä [13].

3.3.2 Muut kirjastot

Ohjelmassa käytetään myös muita kirjastoja. Niitä ovat mm. Util, IO, SQL ja Text. Util-kirjastosta hyödynnetään Util-kirjaston List-rajapinnan toteuttavia listoja eli dynaamisia taulukoita. Niitä käytetään tietorakenteina esimerkiksi tietokannan taulujen sisällön väliaikaiseen tallentamiseen. Dynaamisuus tarkoittaa, että listojen sisältämien alkioiden lukumäärät muuttuvat ohjelman ajon aikana. Util-kirjastossa on myös Observer-rajapinta ja Observable-luokka, joita hyödynnetään kirjoitus- ja rämpytysnopeustestin ajanoton ja keskiarvokäyrän toteuttamiseen Observer-suunnittelumallin mukaisesti. Lisäksi hyödynnetään Properties-luokkaa, jonka avulla käsitellään tietokoneen kovalevylle tallennettua Map-rajapintaa toteuttavaa .properties-tiedostoa, johon tallennetaan ohjelman käyttämien muuttujien arvoja avain-arvo-pareina. Myös Calendar-luokkaa käytetään ennätyslistojen päivämäärätiedon saamiseksi.

Properties-tiedoston sisältö luetaan ohjelmaan tietovirtana. Tämän toteuttamiseen tarvitaan IO-kirjaston FileInputStream-luokkaa. Ohjelma hyödyntää tietokoneen kovalevyllä olevaa tietokantaa tietojen pysyvään tallentamiseen, ja tietokannan käsittely onnistuu vain, kun käytetään SQL-kirjaston tarjoamia Connection-, PreparedStatement- ja ResultSet-rajapintoja. Rajapinnoista Connection huolehtii tietokantayhteyksistä, PreparedStatement tietokannan käsittelyyn tarvittavista SQL-lauseista ja ResultSet tietokantakyselyn tulosjoukon käsittelystä. Ohjelmassa käytetään myös Text-kirjastoa, joka tarjoaa BreakIterator-luokan ohjelman ohjeiden pilkkomiseen lauseittain ja DecimalFormat-luokan liukulukujen pyöristämiseen ja muotoiluun.

3.4 Menetelmät

3.4.1 MVC-suunnittelumalli

Ohjelma toteutetaan MVC-suunnittelumallin mukaisesti. Ohjelman lähdekoodi jaetaan tiedonkäsittelykerroksen lisäksi kolmeksi toisistaan mahdollisimman riippumattomaksi kerrokseksi. MVC-suunnittelumallin lyhenne tulee sanoista malli eli Model, näkymä eli View ja ohjain eli Controller. Ohjelmassa näkyminä ovat käyttöliittymäluokat MainForm, WordsForm, RecordsForm, PersonalRecordsForm. HelpForm ja AboutForm, ohjaimena Controller-luokka ja malleina Model-luokka sekä tietokantaa käsittelevä ModelDatabase-luokka. Sovelluslogiikkakerroksia eli malleja on kaksi, jotta tietokantaa hyödyntävä lähdekoodi olisi mahdollisimman erillään muusta toteutettavasta laskennasta. Näin tietokanta olisi mahdollisimman helposti vaihdettavissa. Ohjelman MVC-kerrosarkkitehtuuri esitetään kuvassa 1.

Kuva 1. Ohjelman MVC-kerrosarkkitehtuuri

Näkymä on rajapinta käyttäjän ja ohjelman toiminnallisuuden välillä. Se sisältää erilaisia käyttöliittymäkomponentteja, kuten esimerkiksi painikkeita, joita painamalla syntyy tapahtumia. Tapahtumat käsitellään käyttöliittymäluokkien anonyymeissä sisäluokissa, jotka kutsuvat ohjaimen tavallisia metodeja. Ohjain toimii näkymän ja mallien välissä. Ohjain voisi esimerkiksi pyytää ModelDatabase-malliluokan hakemaan tietoja tiedonkäsittelykerroksen eli TestYourselfDatabase-luokan kautta tietovarastosta toisin sanoen tietokannasta, jonka luokan TestYourselfDatabaseDAO-rajapinnan vain ModelDatabase-luokka tuntee. Edellä mainitut tiedot ohjain välittää takaisin näkymälle käyttäjälle näytettäväksi. Model-malliluokka sisältää ohjelman sovelluslogiikan, kuten esimerkiksi

tietorakenteina List-rajapinnan toteuttavia listoja, joihin ohjelmassa käytettävät tiedot tallennetaan väliaikaisesti. ModelDatabase-malliluokan kautta käsiteltävään tietokantaan ohjelmassa käytettävät tiedot tallennetaan pysyvästi.

3.4.2 Observer-suunnittelumalli

Kirjoitus- ja rämpytysnopeustesti sisältää ajanoton. Tämän lisäksi kirjoitus- ja rämpytysnopeustestissä käyttöliittymään piirretään keskiarvokäyrää reaaliaikaisesti testin aikana. Jotta ajanoton ja käyrän päivittäminen käyttöliittymään testin aikana joka sekunti olisi mahdollista, ohjelmoitaessa hyödynnetään Observer-suunnittelumallia. Observer-suunnittelumallissa on tarkkailija eli Observer ja tarkkailtava eli Observable. Tarkkailtavaan eli subjektiin voidaan liittää rajattomasti tarkkailijoita, jotka eivät tunne toisiaan. Observable-luokan perivässä luokassa on jäsenmuuttujana lista tarkkailijoista, joita käsitellään Observer-rajapinnan kautta. Tästä syystä subjekti ei tunne tarkkailijoita. [14, s. 294 & 296.] Aluksi tarkkailijat liitetään tarkkailemaan tarkkailtavaa. Tämän jälkeen tarkkailtava päivittää jokaisen siihen liitetyn tarkkailijan tietyin väliajoin, kunnes ohjelman toiminto on suoritettu loppuun.

Ohjelman tarkkailijaluokat ovat TimeCounterUpdater, TimeCounterUpdater2 ja GraphUpdater sekä subjektina on säieluokka Countdown. Testin alussa tarkkailijaluokat liitetään Model-luokan metodissa tarkkailtavaan säieluokkaan ja säie käynnistetään. Tämän jälkeen tarkkailtava säieluokka päivittää käyttöliittymän ajanoton kerran sekunnissa ja keskiarvokäyrän 25 kertaa sekunnissa, kunnes testin ajanotto saavuttaa nollan. Kyseinen käyrän piirtotiheys mahdollistaa pehmeämmin etenevän käyrän näyttämisen käyttöliittymässä.

3.4.3 Säikeet

Osassa ohjelman testeistä käytetään säikeitä. Ohjelman säieluokkia ovat Countdown, LightFlasher ja MemoryRandom. Countdown-luokassa toteutetaan testien ajanotto ja keskiarvokäyrän piirtäminen. LightFlasher-luokassa taas toteutetaan reaktionopeustestin eteneminen ja MemoryRandom-luokassa muistitestin eteneminen. Säikeitä kutsutaan niin sanotuiksi kevytprosesseiksi ja niillä toteutetaan ohjelmaan rinnakkaisuutta eli moniajoa [11, s. 394]. Säikeitä käyttämällä ohjelmassa voidaan suorittaa useita toimintoja samanaikaisesti ilman, että ohjelman muu toiminta keskeytyisi. Säikeet eroavat

prosesseista niin, että niillä ei ole omia resursseja, vaan ne kuuluvat prosesseihin, joiden resursseja ne käyttävät. Jokaisessa prosessissa on siten yksi tai useampia säikeitä. [15.]

Ohjelman säikeistä Countdown-luokka toteuttaa Runnable-rajapinnan. Tästä syystä säikeen käynnistyksen yhteydessä luodaan ensin Thread-luokan olio niin, että parametriksi annetaan Runnable-rajapinnan toteuttava Countdown-luokka. Säie käynnistetään kutsumalla Thread-luokan start()-metodia. LightFlasher- ja MemoryRandom-luokat taas perivät Thread-luokan, mistä syystä säie käynnistetään yksinkertaisesti kutsumalla Thread-luokasta perittyä start()-metodia. Edellä mainitut start()-metodit kutsuvat säieluokkien run()-metodeita, jotka sisältävät varsinaiset säikeiden toteutukset. [11, s. 396.]

Säikeet pitää myös muistaa lopettaa, koska usein säieluokkien run()-metodiin kirjoitetaan ikuinen while-toistorakenne. Tästä syystä Runnable-rajapintaa toteuttavassa Countdown-luokan säikeessä while-toistorakenteen sisällä on ehtolause, joka toteutuu boolean-tyyppisen muuttujan arvon muuttuessa. Kyseisen ehtolauseen sisällä on while-toistorakenteen keskeyttävä break-lause. Thread-luokan perivissä LightFlasher- ja MemoryRandom-luokissa taas while-toistorakenteen toistoehto määritellään boolean-tyyppisen muuttujan avulla. Edellä mainittuja boolean-tyyppisten muuttujien arvoja voidaan muuttaa kutsumalla säieluokkien lopetusmetodia. [11, s. 397.]

3.5 Valmiita rajapintoja ja luokkia

Ohjelman käyttöliittymä toteutetaan pääasiassa Swing-kirjastossa olevilla käyttöliittymäkomponenttiluokilla. Ohjelman avautuessa ensimmäisenä näytetään JFrame-käyttöliittymäluokan perivä TestYourself-pääikkuna ja sen sisältämä JTabbedPane-säiliökomponentilla toteutettava Tervetuloa-välilehti, joka voidaan sulkea. Myös muut ohjelman ikkunat toteutetaan perimällä JFrame-luokka. Ohjelman ikkunat keskitetään kuvaruudulle Dimension-luokan avulla. Ohjelma sisältää myös viisi muuta välilehteä, jotka sisältävät testit. Nämä välilehdet näytetään aina. Ohjelman jokaisessa ikkunassa on JMenuBar-luokalla toteutettava valikkorivi, jota kautta avataan tekstieditori Tekstieditori-ikkunaan, ennätyslistojen ylläpito Ennätyslistat-ikkunaan, henkikohtaisten tulosten listaus Henkilökohtaiset tulokset -ikkunaan, ohjelman ohjeet Ohjeet -ikkunaan ja ohjelman tiedot Tiedot-ikkunaan.

Kirjoitusnopeustestiin toteutetaan kaksi JTextPane-luokan tekstialuetta. Ylemmässä tekstialueessa näytetään kirjoitettava teksti ja alempaan kirjoitetaan vastaava teksti mahdollisimman nopeasti ja virheettömästi. Lisäksi käytetään vieritettävää JScrollPane-säiliöluokkaa, jolle JTextPane-komponentti annetaan parametrina. JScrollPane-säiliöluokka mahdollistaa sen, että tekstialueen sisällön ylittäessä käytössä olevan tilan oikean reunan riviä vaihdetaan automaattisesti ja alareunan pystysuora vieritysruutu ilmestyy tekstialueen reunalle. [11, s. 233.] Hiiri otetaan JTextPane-komponentista pois toiminnasta luomalla oman JTextPane-luokan perivän aliluokan ja ylikirjoittamalla processMouseEvent()- ja processMouseMotionEvent()-metodit tyhjinä.

JTextPane-luokkaa käytetään JTextArea-luokan sijaan siksi, että voitaisiin suorittaa tekstin virheiden tarkistusta reaaliajassa testin aikana muuttamalla tarvittaessa virheellisesti kirjoitettujen sanojen väri punaiseksi. Jotta kirjoitetun tekstin muokkaaminen reaaliaikaisesti olisi mahdollista, käytetään Document-rajapinnan alirajapintaa StyledDocument, joka ylläpitää JTextPane-komponenttiin kirjoitettavaa tekstiä. Liittämällä JTextPane-komponenttiin DocumentFilter-luokka mahdollistetaan tekstin sisällön reaaliaikainen suodattaminen ja kirjoittamisen estäminen sekä virheiden tarkistus. Jotta suodatinluokkien käyttäminen komponenteissa olisi mahdollista, käytetään AbstractDocument-luokan setDocumentFilter()-metodia. DocumentListener-tapahtumakuuntelija on myös hyödyllinen työkalu tekstialueiden sisällön seurantaan. Sitä käytetäänkin, jotta tekstin sivuja voidaan vaihtaa edestakaisin sekä kirjaimia ja sanoja laskea tekstiä kirjoitettaessa. Sen avulla reaaliaikainen tekstin muokkaaminen ei ole kuitenkaan mahdollista. Myös JComboBox-yhdistelmäruutua hyödynnetään, jotta voitaisiin valita kilpailu- tai harjoitusmoodi, harjoitusmoodin tekstejä ja testin kestoja. JComboBox-yhdistelmäruutua käytetään myös muiden testien kestojen tai vaikeusasteiden listaamiseen. Arvottaessa satunnaisten lukujen avulla esimerkiksi kilpailumoodin tekstejä hyödynnetään Random-luokkaa.

Kirjoitus- ja rämpytysnopeustestiin toteutetaan keskiarvokäyrä. Siinä näytetään nopeuden keskiarvoja, jotka tallennetaan List-tietorakenteeseen. Nopeuden keskiarvoja lasketaan ja käyrä piirretään JPanel-luokan paneelin piirtopinnalle 25 kertaa sekunnissa. Käyrä animoidaan kaksoispuskuroinnilla ja näytetään punaisena viivana. Koordinaatiston kehykset ja akselit piirretään Graphic-luokan piirtopinnalle. Varsinaisen käyrän piirtämiseen taas käytetään Graphics2D-luokkaa. Rämpytys- ja reaktionopeustestin Asetukset-näkymä toteutetaan JPanel-luokan paneeliin. Asetukset näkymän JRadioButton-luokan valintapainikkeilla voidaan valita näppäimistö testien suoritukseen. Tämä mah-

dollistetaan KeyboardFocusManager-luokalla ja KeyEventDispatcher-rajapinnalla. KeyEventDispatcher-rajapinta tekee yhteistyötä KeyboardFocusManager-luokan olion kanssa kohdentaakseen ja suorittaakseen kaikki KeyEvent-tapahtumaluokan näppäimistötapahtumat. KeyEventDispatcher-rajapinta hallinnoi näppäimistötapahtumia.

Reaktionopeustestissä on neljä pyöreää ja eriväristä JButton-painiketta. Jotta painikkeet saadaan pyöreiksi ja erivärisiksi, luodaan oma JButton-luokan perivä luokka. Omassa JButton-luokassaan painike piirretään Graphics2D-luokan piirtopinnalle repaint()-metodilla. Lisäksi yliluokalle ilmoitetaan contains()-metodi ylikirjoittamalla, että JButton-painike on muodoltaan pyöreä ja näin estetään tapahtumien syntyminen pyöreän painikkeen reunojen ulkopuolelta. Pyöreä painike piirretään Ellipse2D-luokan avulla.

Muistitestissä kysyttävät alfanumeeriset merkit näytetään JTable-komponentissa, jonka solujen sisältö keskitetään DefaultTableCellRenderer-luokalla. JTable-komponentissa merkit näytetään selkeämmin järjestyksessä toisin kuin, jos käytettäisiin JTextArea- tai JTextPane-tekstialueita. JTable-komponentin solujen lukumäärää muutetaan vaikeusasteen vaihtuessa. Poistettavat komponentin TableColumn-luokan sarakkeet tallennetaan väliaikaisesti listaan ja palautetaan myöhemmin takaisin komponenttiin. Sekuntitarkkuustestissä on JButton-painike, jota painetaan testiä aloitettaessa ja lopetettaessa. Jokaisella testillä on omat ennätyslistansa. Päästessä ennätyslistalle kysytään käyttäjän nimeä ja se toteutetaan JDialog-luokan dialogilla. Kirjoitettavien merkkien lukumäärää eri kentissä on kuitenkin hyvä rajoittaa ja suodatus toteutetaan luvussa mainitulla DocumentFilter-luokalla. Merkkien lukumäärän ylittäessä sallitun rajan käytetään Toolkit-luokkaa tuottamaan äänimerkin.

Tekstieditori on omassa Tekstieditori-ikkunassaan. Tekstieditorissa on tekstialue, joka toteutetaan JTextPane-luokalla, joka sisällytetään JScrollPane-säiliökomponenttiin. Myös JComboBox-yhdistelmäruutua hyödynnetään valittaessa ylikirjoitetaanko vanha teksti vai tallennetaan uusi teksti sekä valittaessa haettavaa kirjoitusnopeustestin harjoitustekstiä.

Ennätyslistojen ylläpito on omassa Ennätyslistat-ikkunassaan. Ennätyslistat näytetään JTable-komponentissa, jonka solujen sisältö keskitetään DefaultTableCellRenderer-luokan avulla. Henkilökohtaiset tulokset näytetään omassa Henkilökohtaiset tulokset -ikkunassaan. Henkilökohtaiset tulokset listataan JTextArea-komponenttiin. Myös

JComboBox-yhdistelmäruutua hyödynnetään valittaessa käyttäjän nimi, jonka tuloksia halutaan tarkastella. Ohjelman ohjeet näytetään omassa Ohjeet-ikkunassaan. Ohjeet näytetään järjestysnumeroin lauseittain käyttämällä BreakIterator-luokkaa. Tiedot näytetään omassa Tiedot-ikkunassaan.

Ohjelmassa hyödynnetään Properties-luokkaa, jonka avulla käsitellään tietokoneen kovalevylle tallennettua Map-rajapintaa toteuttavaa .properties-tiedostoa. Siihen tallennetaan keskitetysti ohjelman käyttämien muuttujien arvoja avain-arvo-pareina. [11, s. 492.] Properties-tiedoston käyttäminen lisää ohjelman kalibroitavuutta eli muokattavuutta, kun muuttujien arvoja ei kirjoiteta suoraan ohjelman lähdekoodiin. Tällöin käyttäjälle annetaan mahdollisuus muokata ohjelman toimintaa haluamakseen.

4 Ohjelman testien tarkoitukset ja tulokset

Tässä luvussa pohditaan ohjelman testien tarkoituksia ja luetellaan niissä saatavia tuloksia.

4.1 Kirjoitusnopeustesti

Kirjoitusnopeustestin tarkoituksena on mitata käyttäjän tekstin lukunopeutta sekä kirjoittamisen nopeutta ja virheettömyyttä. Testi on hyödyksi esimerkiksi harjoiteltaessa kymmensormijärjestelmää tai tekstin kirjoittamista ilman näppäimistöön katsomista.

Testin tuloksina saadaan kirjainten, sanojen, kirjoitusvirheiden ja oikeiden sanojen lukumäärät halutussa ajassa. Vuonna 1998 tehdyn tutkimuksen perusteella tietokoneen käyttäjä kirjoittaa keskimäärin 33 sanaa minuutissa. Keskimääräinen ammattikirjoittaja taas yltää tavallisesti 50 - 80 sanaan ja jotkut 80 - 95 tai jopa 120 sanaan minuutissa. [16.] Testin tuloksena saadaan myös kirjoittamisen keskinopeus numeroarvona. Keskinopeus ilmaisee oikeiden sanojen lukumäärän sekunnin aikana. Testiä suoritettaessa ohjelma piirtää reaaliaikaisesti käyrää, josta ilmenee visuaalisesti kirjoitusnopeuden keskiarvo ajan suhteen.

4.2 Rämpytysnopeustesti

Rämpytysnopeustestin tarkoituksena on mitata käyttäjän nopeutta, sorminäppäryyttä ja reaktioaikaa. Testi on hyödyksi esimerkiksi terapiana, jolla hoidetaan potilaiden sormien jäykkyyttä ja reagoimiskyvyn hitautta.

Testin tuloksena saadaan painikkeen painallusten lukumäärä halutussa ajassa. Ohjelman testauksen perusteella hyvä tulos testissä on noin 100 painallusta 10 sekunnissa. Tuloksina saadaan myös testin aloittamisen reaktioaika ja rämpytysnopeuden keskihajonta eli painikkeen rämpyttämisen tasaisuus numeroarvona. Ihmisen keskimääräinen reaktioaika on 0,215 sekuntia [17]. Keskihajonta taas ilmaisee miten paljon 25 kertaa sekunnissa lasketut rämpytyksen keskinopeudet vaihtelevat testin aikana [18]. Mitä pienempi keskihajonta on, sitä tasaisempaa rämpytys on. Testiä suoritettaessa ohjelma piirtää reaaliaikaisesti käyrää, josta ilmenee visuaalisesti rämpyttämisen tasaisuus ajan suhteen.

4.3 Reaktionopeustesti

Reaktionopeustestin tarkoituksena on mitata käyttäjän reagointi- ja vastaamisnopeutta painikkeiden valaisemiseen jatkuvasti kiihtyvässä tahdissa. Testi on hyödyksi esimerkiksi harjoiteltaessa reagoimiskykyä.

Testin tuloksena saadaan painikkeiden painallusten lukumäärä ilman ennalta määriteltyä aikarajaa. Ohjelman testauksen perusteella hyvä tulos testissä normaalilla vaikeusasteella on noin 100 ja helpolla noin 150 painallusta.

4.4 Muistitesti

Muistitestin tarkoituksena on mitata käyttäjän lyhytkestoista muistia alfanumeerisilla merkeillä. Testi on hyödyksi esimerkiksi muistiongelmista kärsiville ja sillä on hyvä harjoitella asioiden muistamista.

Testin tuloksina saadaan yhden kierroksen ja koko testin oikeiden vastausten lukumäärä prosentteina ilmaistuna kysytyistä alfanumeerisista merkeistä.

4.5 Sekuntitarkkuustesti

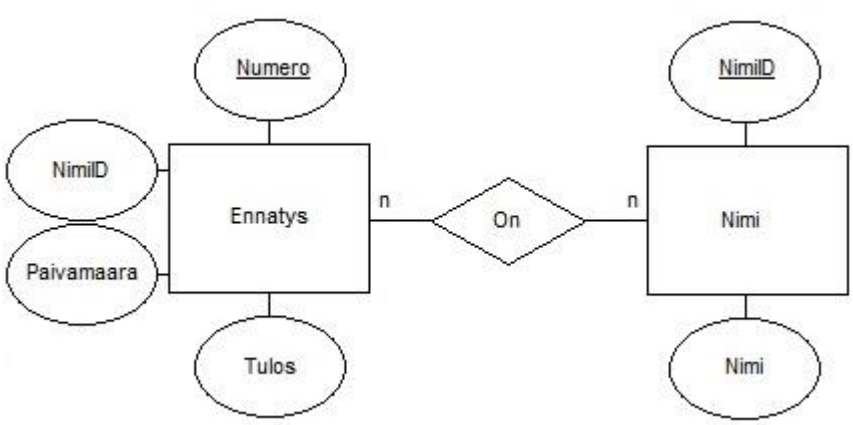
Sekuntitarkkuustestin tarkoituksena on mitata, miten tarkka käyttäjän niin sanottu sisäinen kello on. Testi on hyödyksi esimerkiksi ajantuntemusta harjoiteltaessa.

Testin tuloksena saadaan valitun keston ja käyttäjän aika-arvion välinen erotus sekunteina.

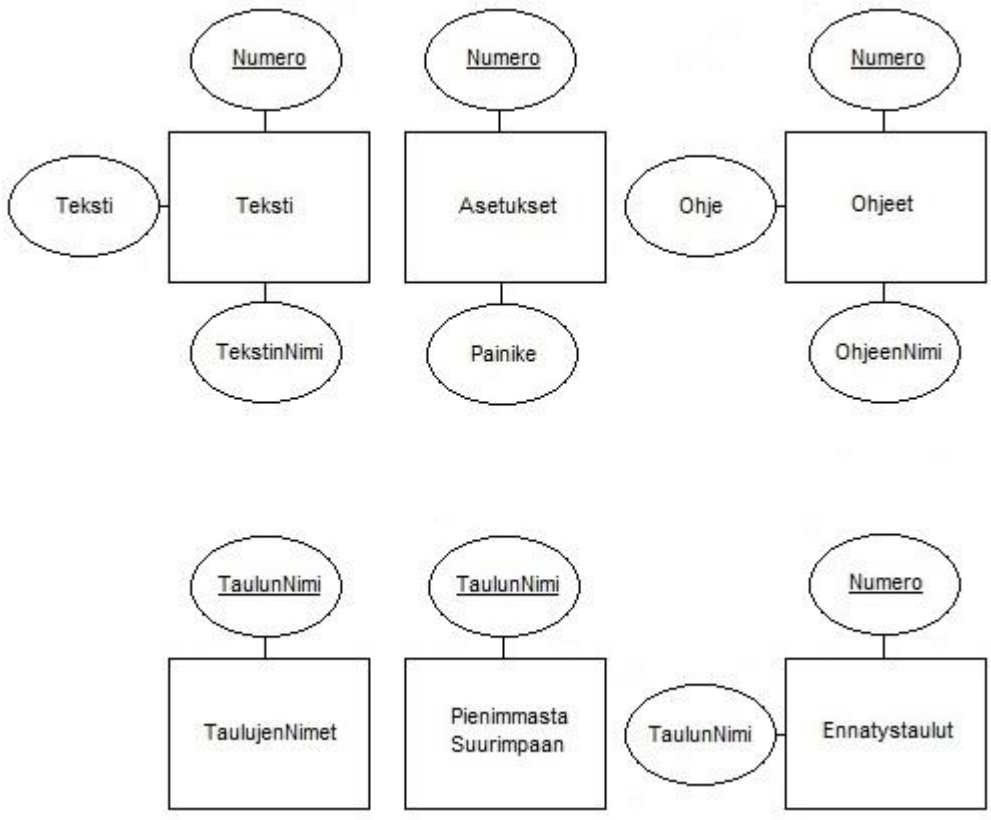
5 Tietokanta

Tämän luvun aluksi esitellään ohjelman käyttämän tietokannan taulut ER-kaavioin. Sen jälkeen kerrotaan teoriaa tietokanta-ajureista ja DAO-suunnittelumallista, joita hyödynnetään toteutetussa ohjelmassa. Lopuksi vielä esitellään tietokantataulut tarkemmin taulukoin. Ohjelmissa käsitellään usein tietoja, jotka halutaan tallentaa tietokoneen kovalevylle myöhempää käyttöä varten. Relaatiotietokanta on yksi vaihtoehto tietojen tallentamiseen, ja se on edelleen usein käytetty tietokantatekniikka. Tietokantaan tallennetaan ohjelman hyödyntämä varsinainen tieto tietokantatauluihin. Lisäksi tallennetaan tietohakemisto eli tietojen määrittelyt, esimerkiksi kenttien nimet ja tietotyypit. Rellaatiotietokantamallin ja käsittelyteorian kehitti Edgar Frank Codd IBM:n tutkimuslaboratiossa vuonna 1970 [19].

Tiedonhallintajärjestelmä on erillinen ohjelma, jonka avulla tietokantaa hyödyntävät ohjelmat voivat suorittaa haku-, tallennus-, poisto- ja muokkausoperaatioita tietokantaan esimerkiksi SQL-kyselykielellä. Tiedonhallintajärjestelmiä ovat esimerkiksi Oracle MySQL ja Microsoft Access. Toteutettavassa ohjelmassa käytetään Microsoft Access -ohjelmistolla toteutettua relaatiotietokantaa tietojen tallentamiseen tietokoneen kovalevylle. Ohjelmassa on monentyyppistä tietoa sisältäviä tietokantatauluja, jotka esitetään kuvien 2 ja 3 ER-kaavioissa. ER-kaavioilla mallinnetaan käsitteellisesti tietokantoja [20].



Kuva 2. Tietokannan Ennätyslista-taulujen ja Nimitaulu-taulun välinen yhteys



Kuva 3. Tietokannan muut taulut

Kuvien ER-kaavioissa suorakulmiot ovat yksilötyyppejä eli kohteita, ellipsit ominaisuuksia ja timanttikuviot suhteita [20]. Yksilötyypit liitetään suhteeseen viivalla, jonka yläpuolelle merkitään lukumääräsuhteet. Kuvan 1 ER-kaavion lukumääräsuhteiden merkinnät voidaan ilmaista sanoilla monen suhde moneen. Yksilötyypit ovat tietokannan

tauluja ja ominaisuudet taulujen sarakkeita, joista alleviivatut ovat taulujen perusavaimia.

5.1 Tietokanta-ajurit

Tietokanta-ajuri on ohjelman ja tietokannan välissä oleva ohjelmisto, joka piilottaa tietokannan hallintajärjestelmän erityispiirteet Java-ohjelmalta. Tietokanta-ajuri on tarpeellinen, jotta ohjelma voi keskustella tiedonhallintajärjestelmän kanssa. Ohjelman ei tarvitse tietää, millainen on käytettävän tietokannan toteutus. Tietokanta-ajuri on eräänlainen tulkki, joka tulkkaa tietokantaan kohdistuvat palvelupyynnöt tietokannan ymmärtämiksi pyynnöiksi. [11, s. 301.]

Ohjelmassa käytetään tiedonhallintajärjestelmänä Microsoft Access -ohjelmistoa. Kyseistä ohjelmistoa käytettäessä tietokantakohtaista JDBC- eli Java Database Connectivity -ajuria ei ole saatavilla, joten käytetään niin kutsuttua JDBC-ODBC-siltaa. Tästä syystä tietokantayhteyden muodostamisessa hyödynnetään erillistä ODBC- eli Open Database Connectivity -ajuria, joka toimii sillan ja tietokannan välissä. Kyseinen silta on välikerros, joka muuntaa Java-ohjelman JDBC-operaatiot vastaaviksi ODBC-operaatioiksi. Tällaisen sillan käyttäminen on kuitenkin toissijainen vaihtoehto, jos ensisijaista JDBC-ajuria ei ole. JDBC-ajurin ollessa käytettävissä sitä käytetään mieluummin, koska JDBC-ajuri on luotettavampi, tehokkaampi ja nopeampi vaihtoehto tietokannan käyttämiseksi. [11, s. 302.]

5.2 DAO-suunnittelumalli

Tietokantaa käyttävä ohjelma sisältää kaikenlaista tietokantaan liittyvää toiminnallisuutta. Ohjelmassa on TestYourselfDatabase-tiedonkäsittelykerros, joka on rajapintana tietokannan ja Java-ohjelman välillä. Tietokannan käsittelyä varten luodaan TestYourselfDatabaseDAO-rajapinnan toteuttava olio, jonka tuntee ainoastaan ModelDatabase-luokka. Tietokannan tietojen käsittely aloitetaan tietokantayhteyden muodostamisella. Ohjelmassa rakennetaan tiedonhallintajärjestelmälle lähetettävät SQL-kyselyt ohjelmallisesti hyödyntäen valmisteltuja lauseita. Ohjelman vastuulla on SQL-tulosjoukkojen käsitteleminen ja sen tulostietueiden muuttaminen takaisin Java-kielen olioiksi. Kun tietokannan tietoja ei enää käsitellä, tietokantayhteys suljetaan. [11, s. 438.]

Ilman erillistä tiedonkäsittelykerrosta edellä mainitut tietokannan käsittelylauseet sijaitsisivat lähdekoodin monessa kohdassa. Tämä tekisi ohjelmasta haastavamman muokata ja kehittää pidemmälle, kun esimerkiksi tietokannan vaihtaminen aiheuttaisi muutoksia moneen eri paikkaan lähdekoodissa. Tästä syystä tietokantatoiminnallisuus on selkeämpää eristää lähdekoodissa omaksi kerroksekseen, jolloin muun ohjelman ei tarvitse tietää juuri mitään taustalla olevasta tietokantatoteutuksesta. Tällaiseen tiedonkäsittelykerroksen eristämiseen käytetään DAO- eli Data Access Objects -suunnittelumallia. Käytettäessä DAO-suunnittelumallia relaatiotietokannan rakennetta voidaan muuttaa tai tietokannan vaihtaa helpommin eikä tarvittavia muutokset kohdistu kuin vain rajattuun osaan ohjelmaa. [11, s. 438.] Tietokantaoperaatiot suoritetaan siis vain ModelDatabase-luokan kautta, jolloin tietokanta on mahdollisimman erillään ohjelman muusta toiminnasta ja siten ohjelmasta tulee modulaarisempi sekä halvempi ja helpompi ylläpitää.

5.2.1 DAO-suunnittelumallin rakenne

DAO-suunnittelumallin pohjana on kohdealueen luokkien ja tietokantatoiminnallisuudesta vastaavan luokan erottaminen toisistaan. Tämä tarkoittaa sitä, että ohjelmassa on erillinen tiedonkäsittelykerros eli Data Access Layer, mikä sisältää tarpeelliset metodit tietokantaoperaatioiden toteuttamiseksi. Tiedonkäsittelykerros lisätään kerrosarkkitehtuurissa sovelluslogiikkakerroksen ja JDBC-sovellusrajapinnan väliin. Näin ohjelmasta tulee modulaarisempi, mutta toisaalta myös hieman hitaampi, koska saman toiminnallisuuden toteuttamiseksi tarvitaan enemmän metodikutsuja. [11, s. 438.]

DAO-suunnittelumalli antaa suunnittelulle suuntaviivat, mutta samalla se sallii melko vapaan toimintatavan, jolla sovelluslogiikka- ja tiedonkäsittelykerros kommunikoivat keskenään [11, s. 439]. TestYourself-ohjelman tiedonkäsittelykerros koostuu vain yhdestä TestYourselfDatabaseDAO-rajapinnan toteuttavasta TestYourselfDatabase-luokasta. Luokka vastaa kaikista tietokantaoperaatioista tietokannan monenlaisista käsiteltävistä tauluista tai operaatioiden kohdealueen luokista riippumatta.

5.2.2 Ohjelman yhteinen DAO-luokka

TestYourself-ohjelman tiedonkäsittelykerros koostuu vain yhdestä TestYourselfDatabase-luokasta, joka vastaa kaikista tietokantaoperaatioista. Jokaiseen tietokantatau-

luun kohdistuu erityyppisiä haku-, lisäys-, poisto- ja päivitysoperaatioita, joten metodeja on suuri määrä. Ohjelman DAO-luokan toteutus perustuu Microsoft Access -relaatiotietokantaohjelmiston käyttämään SQL-kielen murteeseen, joten esimerkiksi tietokantayhteyden muodostamislauseet palvelintietoineen, portteineen ja tietokantanimineen ovat erilaiset kuin esimerkiksi käytettäessä MySQL-ohjelmistoa. [11, s. 440.] Nämä erot ovat syy siihen, että tietokantaoperaatiot toteuttavan DAO-luokan sijaan ModelDatabase-luokka tuntee tietokantajärjestelmästä riippumattoman TestYourselfDatabaseDAO-rajapinnan ja kutsuu sen metodeja, jolloin varsinainen DAO-luokka on mahdollista vaihtaa tarvittaessa helpommin.

DAO-luokan metodeille voidaan antaa parametrina tiedonsiirto-olio eli Data Transfer Objects, jonka tarkoituksena on helpottaa tietojen välittämistä sovelluslogiikkakerroksen ja DAO-kerroksen välillä. Koska sen käyttäminen ei ole kuitenkaan välttämätöntä ja DAO-suunnittelumallin mukainen tiedonkäsittelykerros voidaan toteuttaa myös ilman sitä, TestYourself-ohjelmassa kyseistä oliota ei käytetä. [11, s. 441 - 442.]

5.3 Tietokantataulut

TestYourself-ohjelmassa on monenlaisia erisisältöisiä tietokantatauluja ennätyslistoille, nimille, teksteille, asetuksille, ohjeille ja taulujen nimille. Seuraavassa kerrotaan jokaisesta tarkemmin.

5.3.1 Ennätystaulut

Jokaisella ohjelman testillä on oma ennätyslistansa, mutta jokainen ennätystaulu sisältää samat ominaisuudet eli järjestysnumeron, ennätyksen tekijän id:n eli tunnistenumeron, päivämäärän ja tuloksen. Taulujen perusavaimena on järjestysnumero, joka identifioi eli yksilöllistää taulun rivin yksikäsitteisesti. Järjestysnumero on automaattisesti generoitu kokonaisluku yhdestä kymmeneen. Ennätystaulujen vierasavaimena on ennätyksen tekijän id eli tunnistenumero, minkä avulla ohjelman tietokannan Nimitaulu-taulusta haetaan ennätyksen tekijän nimi ja näin ennätyslistojen ja Nimitaulu-taulun välillä on yhteys eli riippuvuus. Toisin sanoen taulujen välillä suoritetaan luonnollinen liitos, joka ilmaistaan SQL-lauseessa sanoilla inner join. Ennätyksen tekijän id eli tunnistenumero on automaattisesti generoitu kokonaisluku. Ennätystaulujen päivämäärä

generoidaan myös automaattisesti. Tulos on kokonais- tai liukuluku riippuen testistä. Ennätystaulujen kenttien nimet ja tietotyypit esitetään taulukossa 1.

Taulukko 1. Ennätystaulut

Kentän nimi:	**Tietotyyppi:**
Numero	AutoNumber
NimiID	Number
Paivamaara	Date/Time
Tulos	Number

5.3.2 Nimitaulu-taulu

Ohjelman testien ennätysten tekijöillä on nimet, jotka tallennetaan Nimitaulu-tauluun. Nimitaulu-taulun ominaisuudet ovat ennätyksen tekijän id eli tunnistenumero ja ennätyksen tekijän nimi. Taulun perusavaimena on ennätyksen tekijän id eli tunnistenumero ja se on automaattisesti generoitu kokonaisluku. Ennätyksen tekijän nimen tyyppi on merkkijono. Nimitaulu-taulun kenttien nimet ja tietotyypit esitetään taulukossa 2.

Taulukko 2. Nimitaulu-taulu

Kentän nimi:	**Tietotyyppi:**
NimiID	AutoNumber
Nimi	Text

5.3.3 Tekstitaulut

Ohjelman kirjoitusnopeustestin kilpailu- ja harjoitusmoodissa sekä tekstieditorissa käytetään tekstejä, jotka on tallennettu tekstitauluihin. Tekstitauluilla on samat ominaisuudet eli numero, teksti ja tekstin nimi. Taulujen perusavaimena on numero, joka on automaattisesti generoitu kokonaisluku. Tekstin ja tekstin nimen tyyppi on merkkijono. Tekstitaulujen kenttien nimet ja tietotyypit esitetään taulukossa 3.

Taulukko 3. Tekstitaulut

Kentän nimi:	**Tietotyyppi:**
Numero	AutoNumber
Teksti	Memo
TekstinNimi	Text

5.3.4 Asetukset-taulut

Ohjelman rämpytys- ja reaktionopeustestin näppäimistön painikkeet tallennetaan Asetukset-tauluihin. Asetukset-taulujen ominaisuudet ovat painikkeen järjestysnumero ja painike. Taulujen perusavaimena on painikkeen järjestysnumero, joka on automaattisesti generoitu kokonaisluku väliltä 1 ja 4 testien painikkeiden lukumäärän mukaan. Painikkeen tyyppi on merkkijono. Asetukset-taulujen kenttien nimet ja tietotyypit esitetään taulukossa 4.

Taulukko 4. Asetukset-taulut

Kentän nimi:	**Tietotyyppi:**
Numero	AutoNumber
Painike	Text

5.3.5 OhjeetTaulu-taulu

Ohjelman osa-alueiden ohjeet tallennetaan OhjeetTaulu-tauluun. OhjeetTaulu-taulun ominaisuudet ovat numero, ohje ja ohjeen nimi. Taulujen perusavaimena on numero, joka on automaattisesti generoitu kokonaisluku. Ohjeen ja ohjeen nimen tyyppi on merkkijono. OhjeetTaulu-taulun kenttien nimet ja tietotyypit esitetään taulukossa 5.

Taulukko 5. OhjeetTaulu-taulu

Kentän nimi:	**Tietotyyppi:**
Numero	AutoNumber

Ohje	Memo
OhjeenNimi	Text

5.3.6 Taulujen nimiä sisältävät taulut

Ohjelmassa tarvitaan tauluja, jotka sisältävät tietokantataulujen nimiä. TaulujenNimet-taulussa on listattu kaikki tietokantataulut. Ennatystaulut-taulu taas sisältää kaikki ennätyslista-taulut ja PienimmastaSuurimpaan-taulu ne ennätyslista-taulut, joissa tulokset, esimerkiksi reaktioajat, listataan pienimmästä suurimpaan. Jokainen taulujen nimiä sisältävä taulu sisältää taulun nimen. Ennatystaulut-taulu sisältää lisäksi numeron. TaulujenNimet- ja PienimmastaSuurimpaan-taulujen perusavaimena on taulun nimi, joka on merkkijono. Ennatystaulut-taulun perusavaimena on numero. TaulujenNimet- ja PienimmastaSuurimpaan-taulujen kenttien nimet ja tietotyypit esitetään taulukossa 6 ja Ennatystaulut-taulun taulukossa 7.

Taulukko 6. TaulujenNimet- ja PienimmastaSuurimpaan-taulut

Kentän nimi:	**Tietotyyppi:**
TaulunNimi	Text

Taulukko 7. Ennatystaulut-taulu

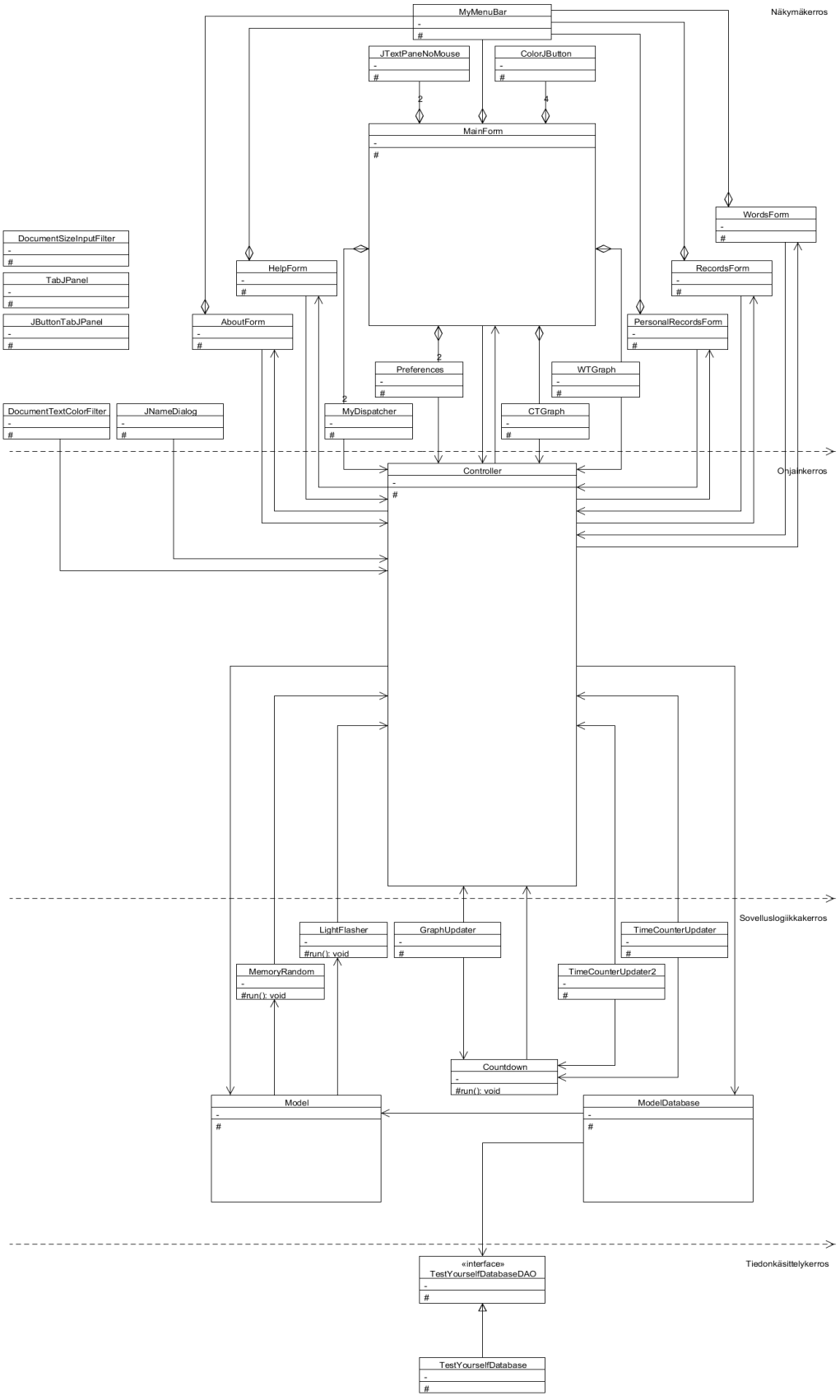
Kentän nimi:	**Tietotyyppi:**
Numero	AutoNumber
TaulunNimi	Text

6 Toteutus

Tässä luvussa kerrotaan ohjelman toteutuksesta. Luvun aluksi esitellään ohjelman luokkakaavio ja kerrotaan toteutetut luokat. Sen jälkeen esitellään toteutetun ohjelman käyttöliittymät ja ominaisuudet.

6.1 Luokkakaavio

Luokkakaavio on UML- eli Unified Modeling Language -mallinnuskielen kaavio, jolla kuvataan ohjelman luokat, niiden rakenteet ja niiden väliset suhteet. Luokkaa merkitään laatikolla, joka sisältää luokan nimen sekä tarvittaessa tärkeimmät jäsenmuuttujat ja -metodit. [14, s. 363.] Luokkakaaviossa esitetään 29 luokkaa ja säikeiden run()-metodit. Ohjelman luokkakaavio esitetään kuvassa 4.



Kuva 4. Ohjelman luokkakaavio

Ohjelma toteutettiin MVC-suunnittelumallin mukaisesti. Tämä tarkoittaa sitä, että malli eli Model, näkymä eli View ja ohjain eli Controller ovat toisistaan mahdollisimman riippumattomia kerroksia. Ohjelmassa on kaksi malliluokkaa eli Model ja ModelDatabase, jotka eivät tunne käyttöliittymäluokkia MainForm, WordsForm, RecordsForm, PersonalRecordsForm, HelpForm tai AboutForm, vaan niiden välissä on ohjainluokka Controller. Käyttöliittymäluokista MainForm on ohjelman pääikkuna, joka näytetään käynnistettäessä. Mainform tuntee Controller-luokan, jonka ilmentymän se sisältää. MainForm-luokka koostuu myös monien muiden luokkien olioista, jotka ovat osa luokkaa. Ilmentymät ovat osaolioita, jotka eivät voi olla olemassa ilman MainForm-luokkaa. Tällaista koostetta kutsutaan aidoksi ja myös kompositioksi. [11, s. 145 - 146.] Niitä ovat WTGraph- ja CTGraph-paneelit keskiarvokäyrille, Preferences-paneelit asetuksille, JTextPaneNoMouse-tekstialueet tekstille, ColorJButton-painikeluokat pyöreille painikkeille, MyMenuBar-luokka valikkoriville ja MyDispatcher-luokat näppäimistön kuuntelemiselle. Muut käyttöliittymäluokat koostuvat myös Controller-luokan oliosta ja sisältävät MyMenuBar-luokan osaolion.

Controller-luokka tuntee kaikki käyttöliittymä- ja malliluokat. Käyttöliittymäluokkien anonyymit sisäluokat käsittelevät käyttäjän synnyttämät tapahtumat ja kutsuvat Controller-luokan tavallisia metodeja pyytäen esimerkiksi tarvittaessa ModelDatabase-luokan hakemaan tietoja tietokannasta käyttöliittymässä näytettäväksi. Model-luokka sisältää laskentaa, ja se tuntee LightFlasher- ja MemoryRandom-säieluokkien oliot. Säieluokilla toteutetaan reaktio- ja muistitestin eteneminen. ModelDatabase-luokka taas tuntee Model-luokan olion. Lisäksi ModelDatabase-luokka sisältää TestYourselfDatabaseDAO-rajapinnan tyyppisen olion, johon viitaten käytetään rajapinnan toteuttavaa tiedonkäsittelykerrosta TestYourselfDatabase. Näin saadaan myös tiedonkäsittelykerros mahdollisimman erilliseksi kerroksekseen.

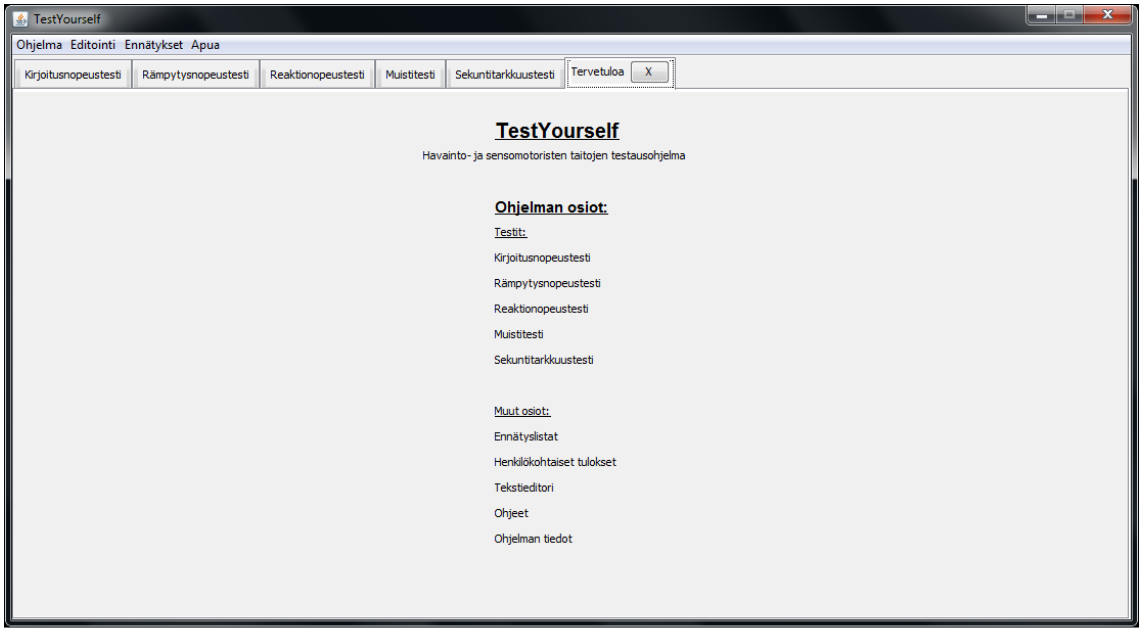
Osa ohjelman testeistä sisältää ajanoton, joka toteutettiin Observable-luokan perivällä Countdown-säieluokalla. Countdown-säieluokka tuntee Controller-luokan. Sitä tarkkailevat Observer-rajapinnan toteuttavat luokat GraphUpdater, TimeCounterUpdater ja TimeCounterUpdater2. Edellä mainitut luokat sisältävät myös Controller-luokan ilmentymän. Luokkakaaviossa esitetään myös Controller-luokan ilmentymän sisältävä JNameDialog-dialogiluokka nimen kysymiseen ja DocumentTextColorFilter-suodatinluokka kirjoitusvirheiden tarkistamiseen. Lisäksi esitetään kolme erillistä luokkaa DocumentSizeInputFilter, TabJPanel ja JButtonTabJPanel, jotka eivät koostu muista luokkakaavion luokkien olioista.

6.2 Käyttöliittymä

Ohjelman graafinen käyttöliittymä luotiin Swing-kirjastolla. Swing tarjoaa kokoelman valmiiksi toteutettuja käyttöliittymäkomponentteja, joilla graafinen käyttöliittymä rakentuu kätevästi ja johdonmukaisesti. Seuraavassa on selostettu tarkemmin ohjelman jokaisen osa-alueen toiminta.

6.2.1 Tervetuloa-välilehti

Ohjelman käynnistyessä käyttäjälle avautuu ensimmäisenä kuvassa 5 oleva Tervetuloa-välilehti.

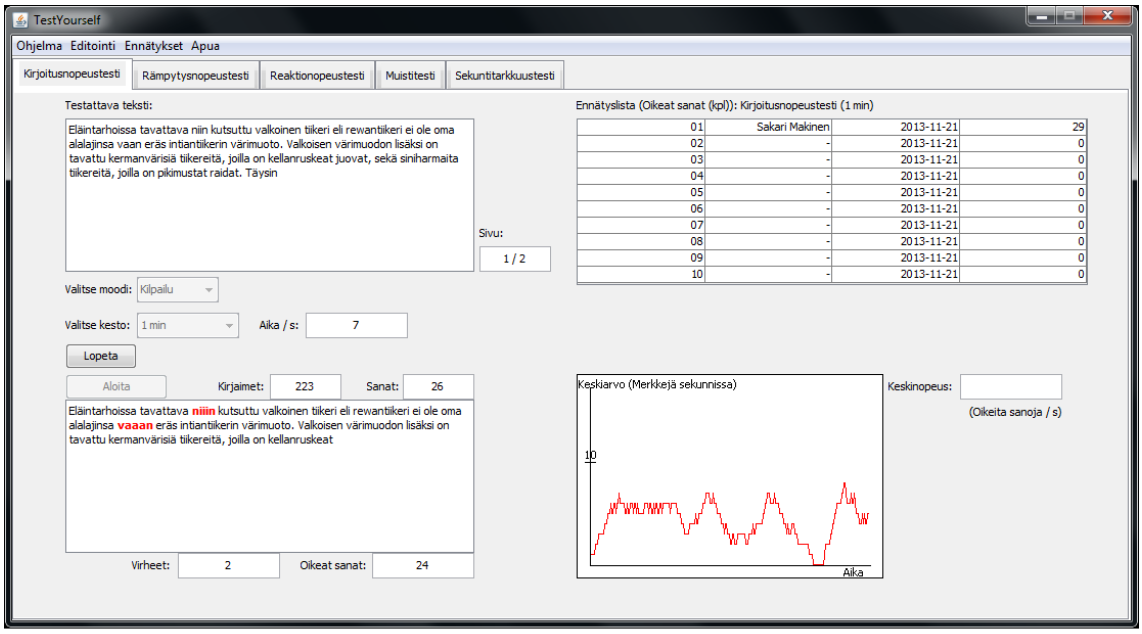


Kuva 5. Tervetuloa-välilehti

Tervetuloa-välilehdellä kerrotaan ohjelman osiot. Välilehti on mahdollista sulkea kuvassa 5 näkyvällä painikkeella.

6.2.2 Kirjoitusnopeustesti-välilehti

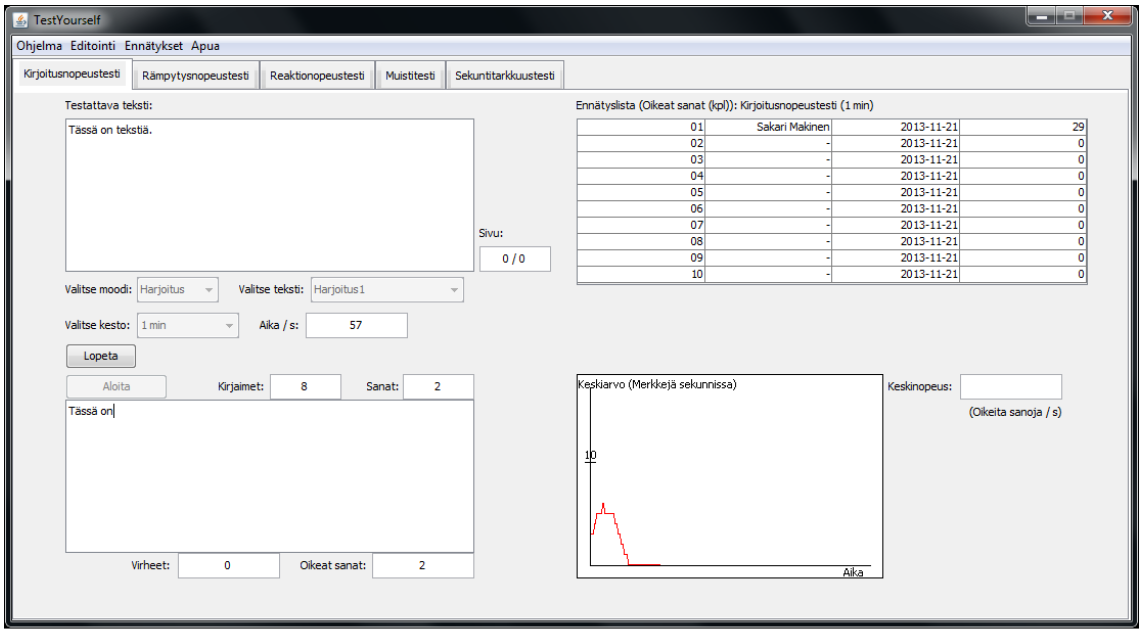
Suljettaessa Tervetuloa-välilehden siirrytään automaattisesti kuvassa 6 esitetylle kirjoitusnopeustestin välilehdelle.



Kuva 6. Kirjoitusnopeustesti-välilehti: Kilpailumoodi

Kirjoitusnopeustesti-välilehdellä käyttäjä voi testata esimerkiksi kymmensormijärjestelmää tai kirjoittamista ilman näppäimistöön katsomista. Ennen testin alkua valitaan kilpailu- tai harjoitusmoodi. Seuraavaksi valitaan testin kesto ja klikataan Aloita-painiketta. Tämän jälkeen näytetään lähtölaskenta, jonka loputtua ajanoton alkaessa kohdistin siirtyy alemmalle tekstialueelle ja testi alkaa. Ylemmällä tekstialueella on testattava teksti ja sen vieressä tekstin sivumäärä. Testissä kirjoitetaan testattava teksti mahdollisimman nopeasti ja virheettömästi alemmalle tekstialueelle. Käyttäjän kirjoittaman tekstin sanoja tarkistetaan reaaliaikaisesti ja mahdolliset kirjoitusvirheet näytetään punaisella kuvan 6 mukaisesti. Testin edetessä ohjelma laskee ja näyttää kirjoitettujen kirjainten, sanojen, virheiden ja oikeiden sanojen lukumäärät. Lisäksi käyttäjä voi seurata kirjoitusnopeutta testin aikana reaaliaikaisesti päivittyvästä kirjoitusnopeuden keskiarvokäyrästä. Testin lopuksi ohjelma laskee ja näyttää kirjoituksen keskinopeuden. Testi lopetetaan, kun ajanotto on nolla tai testattava teksti on kirjoitettu kokonaisuudessaan oikein.

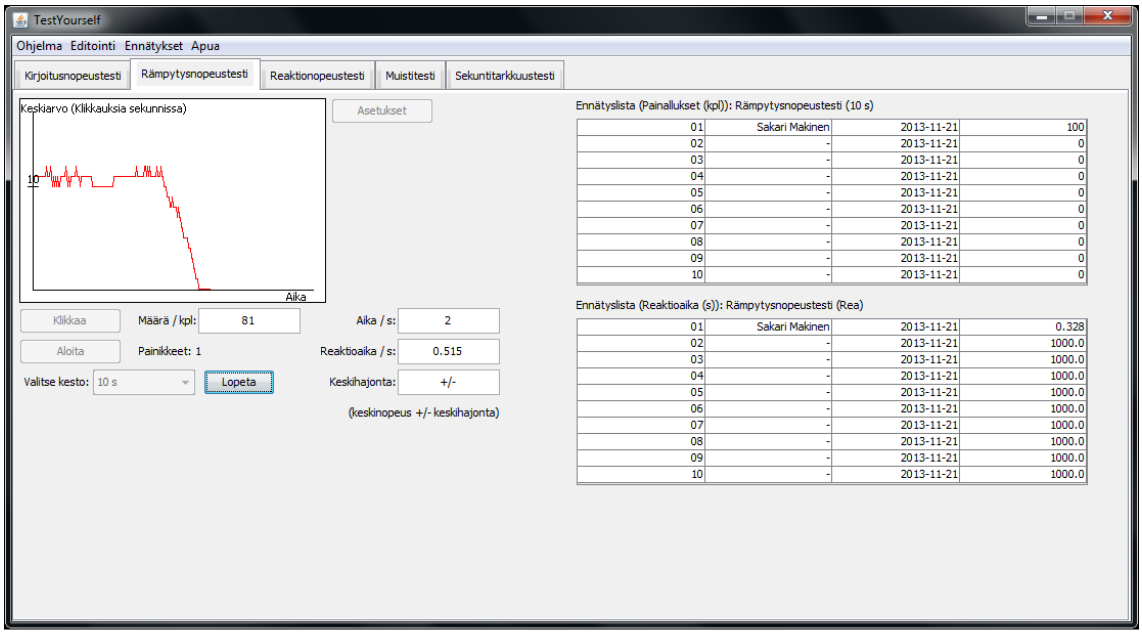
Harjoitusmoodissa testattava harjoitusteksti valitaan yhdistelmäruudusta ja klikataan Aloita-painiketta. Tämän jälkeen testi etenee samalla tavalla kuin kilpailumoodissa kuvan 7 mukaisesti paitsi, että testissä ei ole ennätyslistaa.



Kuva 7. Kirjoitusnopeustesti-välilehti: Harjoitusmoodi

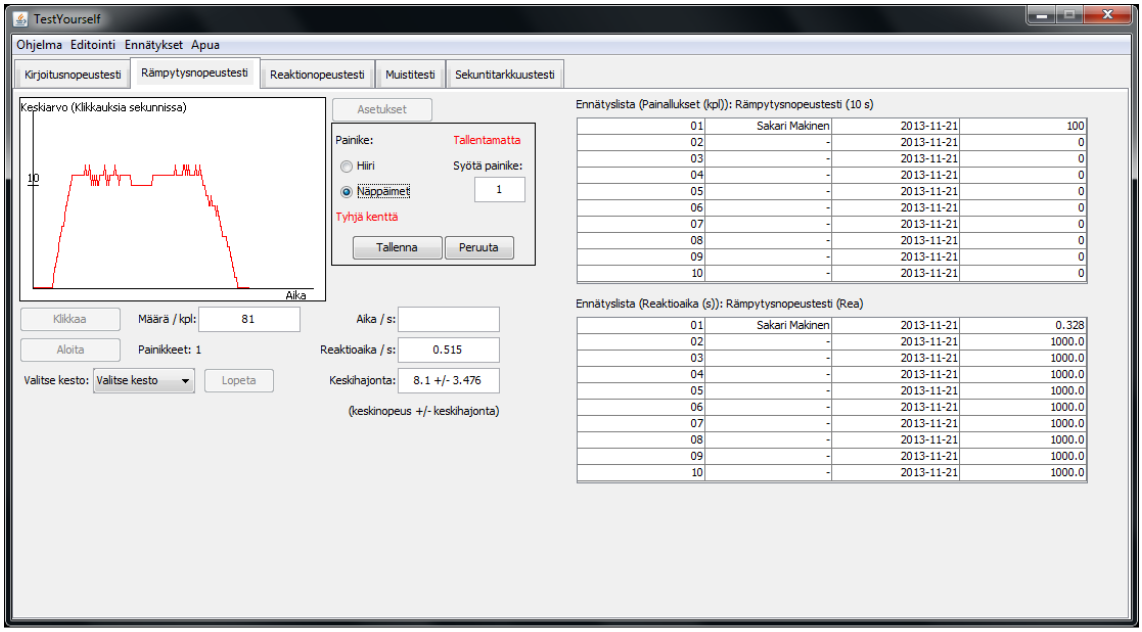
6.2.3 Rämpytysnopeustesti-välilehti

Klikattaessa Rämpytysnopeustesti-välilehteä avautuu kuvan 8 mukainen käyttöliittymä.



Kuva 8. Rämpytysnopeustesti-välilehti

Rämpytysnopeustesti-välilehdellä käyttäjä voi testata esimerkiksi nopeutta, sorminäppäryyttä ja reaktioaikaa. Ennen testin alkua valitaan testin kesto ja klikataan Aloita-painiketta. Tämän jälkeen näytetään lähtölaskenta, jonka loputtua ajanoton alkaessa testi alkaa. Testissä klikataan Klikkaa-painiketta tai kuvassa 9 esitetyn Asetukset-paneelin kautta valittua näppäimistön painiketta mahdollisimman nopeasti. Jos rämpyttäminen aloitetaan liian aikaisin, testi lopetetaan ja käyttäjälle ilmoitetaan varaslähdöstä.

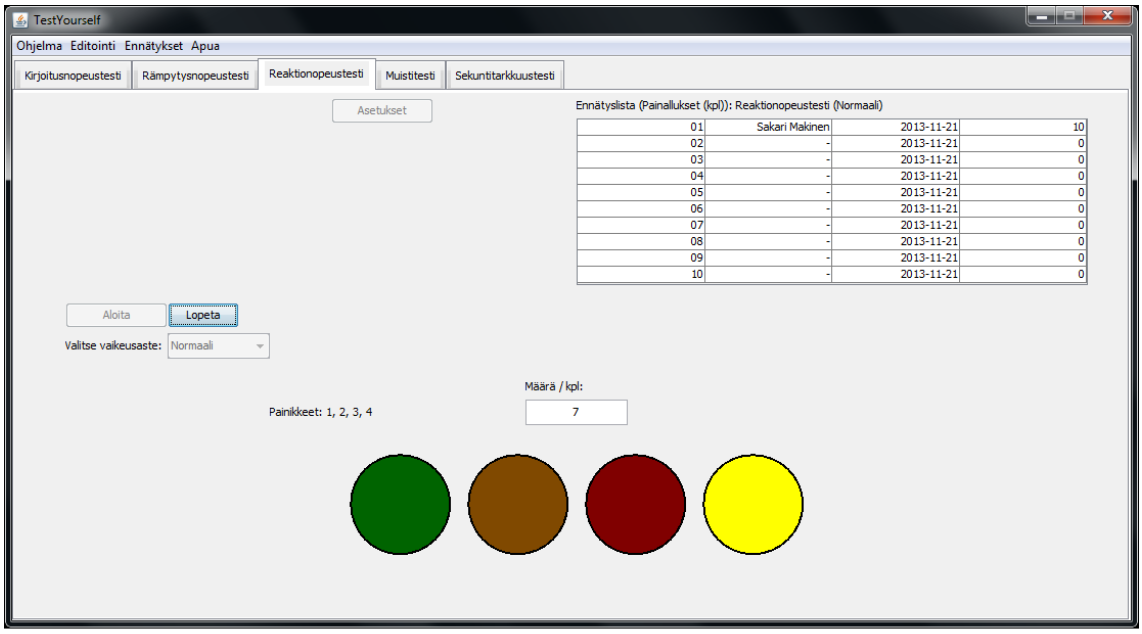


Kuva 9. Rämpytysnopeustesti-välilehti: Asetukset-paneeli

Testin alettua ohjelma näyttää ensimmäisen klikkauksen ja testin alun välisen reaktioajan. Sen edetessä ohjelma laskee ja näyttää klikkausten lukumäärän. Lisäksi käyttäjä voi seurata rämpytyksen tasaisuutta testin aikana reaaliaikaisesti päivittyvästä rämpytysnopeuden keskiarvokäyrästä. Testin lopuksi ohjelma laskee ja näyttää rämpytysnopeuden keskihajonnan. Testi lopetetaan, kun ajanotto on nolla.

6.2.4 Reaktionopeustesti-välilehti

Klikattaessa Reaktionopeustesti-välilehteä avautuu kuvan 10 mukainen käyttöliittymä.



Kuva 10. Reaktionopeustesti-välilehti

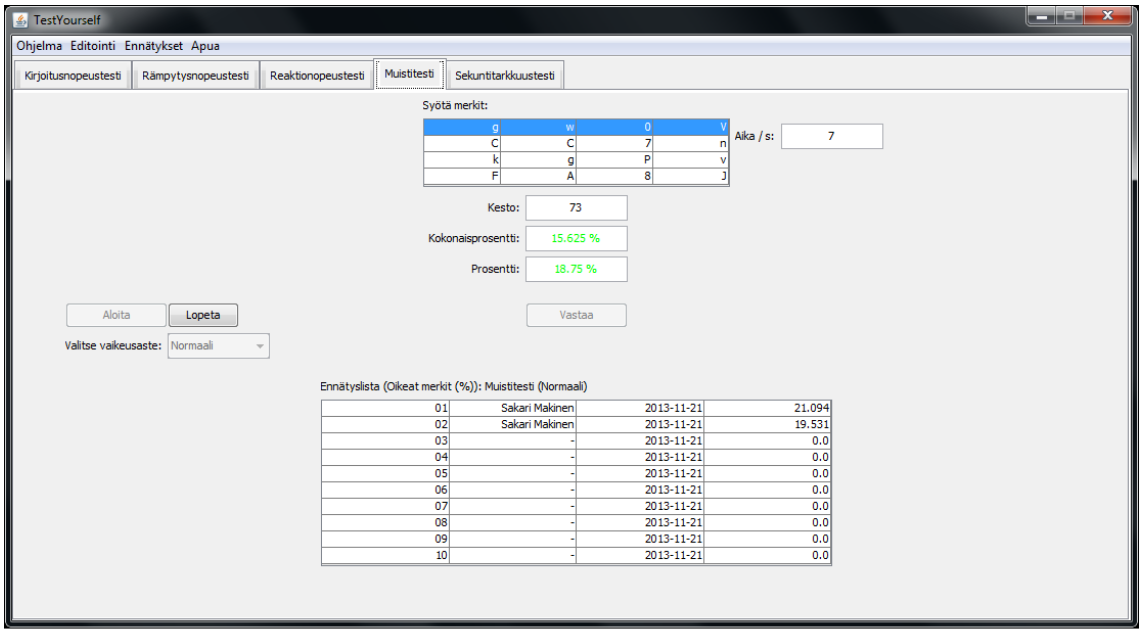
Reaktionopeustesti-välilehdellä käyttäjä voi testata esimerkiksi reagointi- ja vastaamisnopeutta painikkeiden valaisemiseen koko ajan nopeutuvassa tahdissa. Ennen testin alkua valitaan testin vaikeusasteeksi Normaali tai Helppo ja klikataan Aloita-painiketta ja testi alkaa. Testissä painetaan neljää eriväristä painiketta sitä mukaa, kun niitä valaistaan jatkuvasti nopeutuvassa tahdissa. Ohjelma valaisee painikkeita vaikeusasteen mukaisella nopeudella. Painikkeita klikataan joko hiirellä tai klikataan vastaavia kuvassa 11 esitetyn Asetukset-paneelin kautta valittuja näppäimistön painikkeita.

Kuva 11. Reaktionopeustesti-välilehti: Asetukset-paneeli

Testin edetessä ohjelma laskee ja näyttää klikkausten lukumäärän. Testin lopuksi ohjelma valaisee jokaisen painikkeen samanaikaisesti määritellyn lukumäärän. Testi lopetetaan, kun valaistuja painikkeita klikataan liian hitaasti, klikataan väärää painiketta tai ei klikata ollenkaan painikkeita.

6.2.5 Muistitesti-välilehti

Klikattaessa Muistitesti-välilehteä avautuu kuvan 12 mukainen käyttöliittymä.

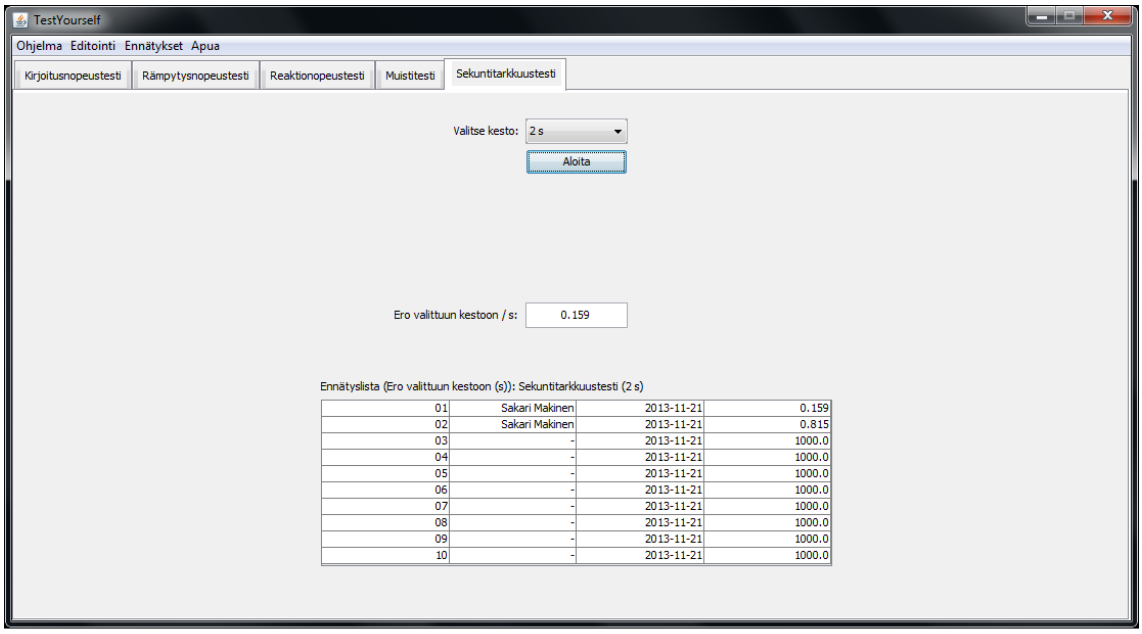


Kuva 12. Muistitesti-välilehti

Muistitesti-välilehdellä käyttäjä voi testata esimerkiksi lyhytkestoista muistia alfanumeerisilla merkeillä. Ennen testin alkua valitaan testin vaikeusasteeksi Normaali tai Helppo ja klikataan Aloita-painiketta ja testi alkaa. Testissä keskustellaan ohjelman kanssa: Ohjelma näyttää noin 10 – 2 sekuntia joko 8 tai 16 merkkiä vaikeusasteen mukaisesti kuvassa 12 esitetyssä ruudukossa nopeutuen noin sekunnilla joka kierroksella. Kun merkit on näytetty, kirjoitetaan noin 30 sekunnin aikana näytetyt merkit samassa järjestyksessä samaiseen ruudukkoon ja klikataan Vastaa-painiketta. Testin edetessä ohjelma laskee ja näyttää testin ja kierroksen oikeiden vastausten lukumäärät prosentteina joka kierroksella. Prosenttien väheneminen ilmaistaan punaisella ja kasvaminen vihreällä värillä. Testin kesto vähenee jokaisella kierroksella väärien vastausten lukumäärän. Testi lopetetaan, kun kesto on nolla.

6.2.6 Sekuntitarkkuustesti-välilehti

Klikattaessa Sekuntitarkkuustesti-välilehteä avautuu kuvan 13 mukainen käyttöliittymä.

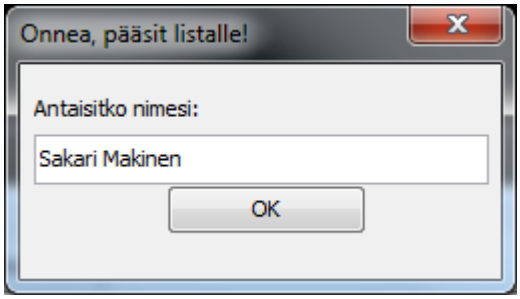


Kuva 13. Sekuntitarkkuustesti-välilehti

Sekuntitarkkuustesti-välilehdellä käyttäjä voi testata esimerkiksi ajantuntemusta. Ennen testin alkua valitaan testin kesto ja klikataan Aloita-painiketta ja testi alkaa. Testissä Aloita-painikkeen jälkeen lasketaan sekunteja päässä ja klikataan Lopeta-painiketta. Testin lopuksi näytetään valitun keston ja käyttäjän aika-arvion välinen erotus sekunteina. Testi lopetetaan, kun painiketta painetaan sekuntien laskemisen jälkeen.

6.2.7 Ennätyslistan nimikysely

Jokaisella testillä on ennätyslistansa. Ennätyslistalle päästessä avautuu kuvan 14 mukainen modaalinen dialogi.

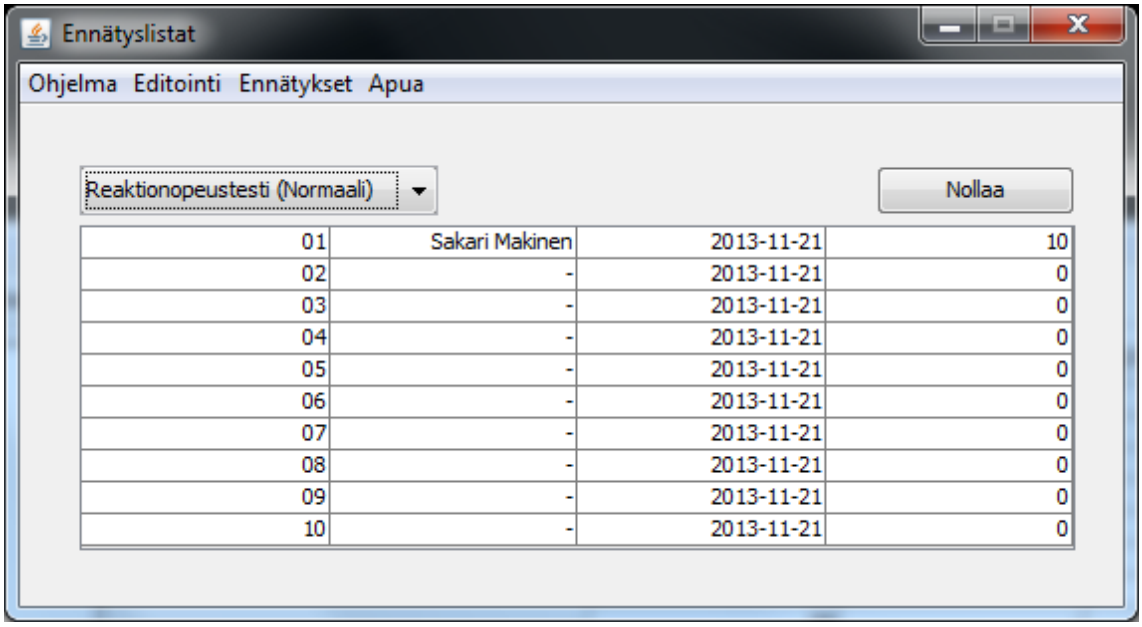


Kuva 14. Ennätyslistan nimikysely -dialogi

Dialogissa ohjelma onnittelee käyttäjää ja pyytää antamaan nimensä, joka tallennetaan testin tuloksen ja silloisen päivämäärän mukana tietokannan ennätyslistaan. Dialogin modaalisuus estää etenemästä ohjelmassa ennen nimikyselyn sulkemista [21, s. 360].

6.2.8 Ennätyslistat-ikkuna

Valittaessa ohjelman valikkoriviltä Ennätyslistat avautuu kuvan 15 mukainen käyttöliittymä.

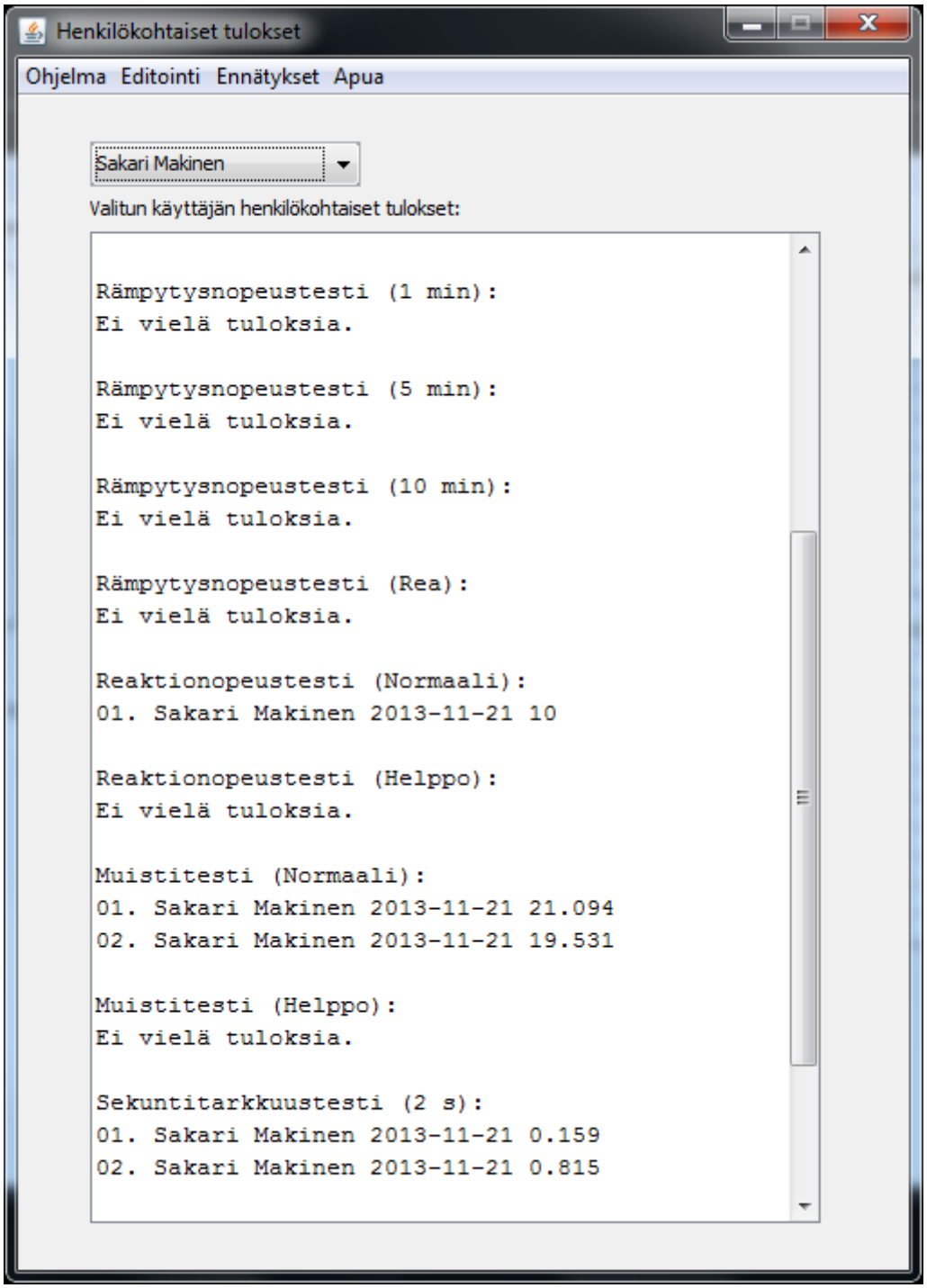


Kuva 15. Ennätyslistat-ikkuna

Ennätyslistat-ikkunassa käyttäjä voi valita haluamansa testin ennätyslistan yhdistelmäruudusta näytettäväksi ruudukossa. Ennätyslistan on myös mahdollista nollata klikkaamalla kuvassa 15 esitettyä Nollaa-painiketta, minkä jälkeen valittu ennätyslista näytetään päivitettynä.

6.2.9 Henkilökohtaiset tulokset -ikkuna

Valittaessa ohjelman valikkoriviltä Henkilökohtaiset tulokset avautuu kuvan 16 mukainen käyttöliittymä.

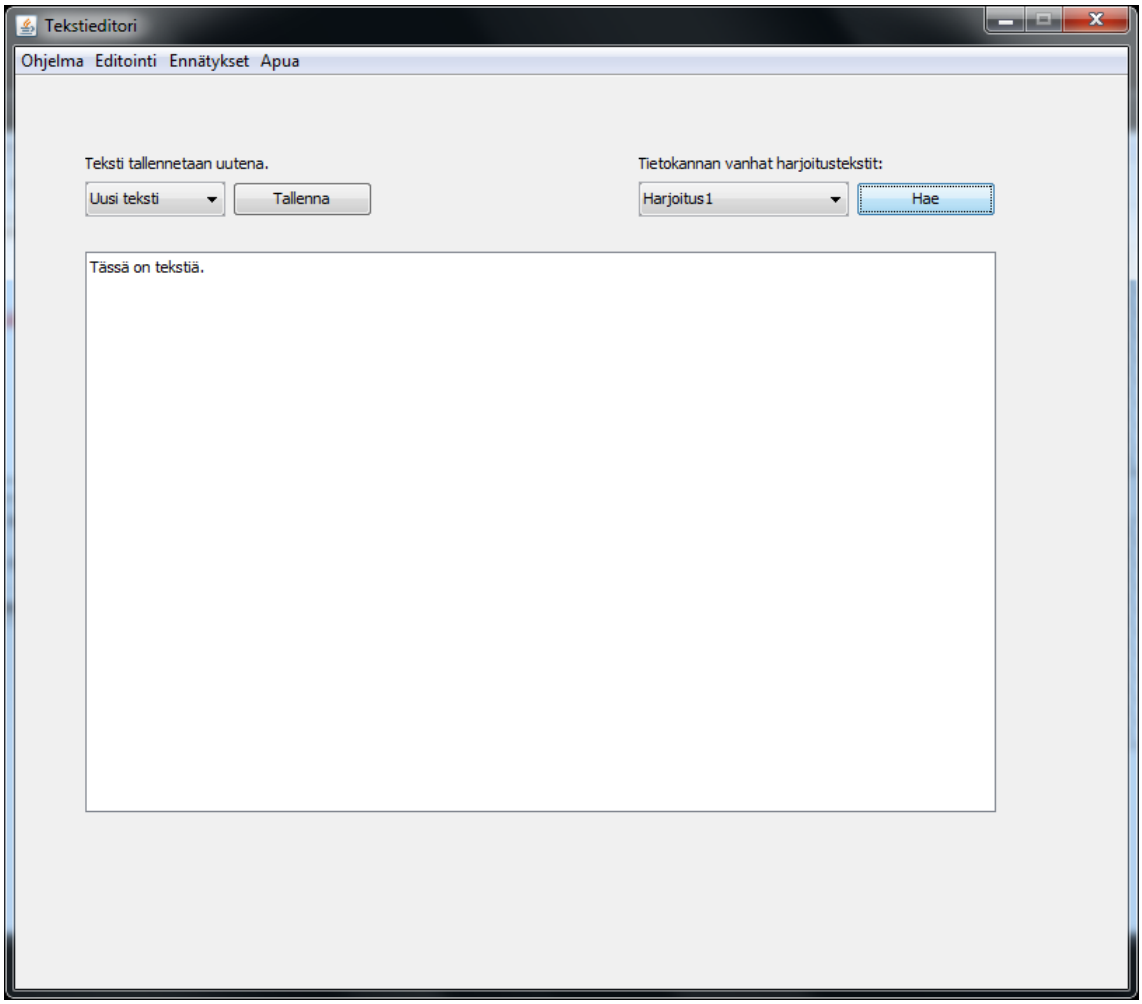


Kuva 16. Henkilökohtaiset tulokset -ikkuna

Henkilökohtaiset tulokset -ikkunassa käyttäjä voi valita nimen yhdistelmäruudusta. Ohjelma näyttää valitun nimen henkilökohtaiset tulokset kaikissa ohjelman testeissä kuvassa 16 esitetyllä tekstialueella.

6.2.10 Tekstieditori-ikkuna

Valittaessa ohjelman valikkoriviltä Tekstieditori avautuu kuvan 17 mukainen käyttöliittymä.



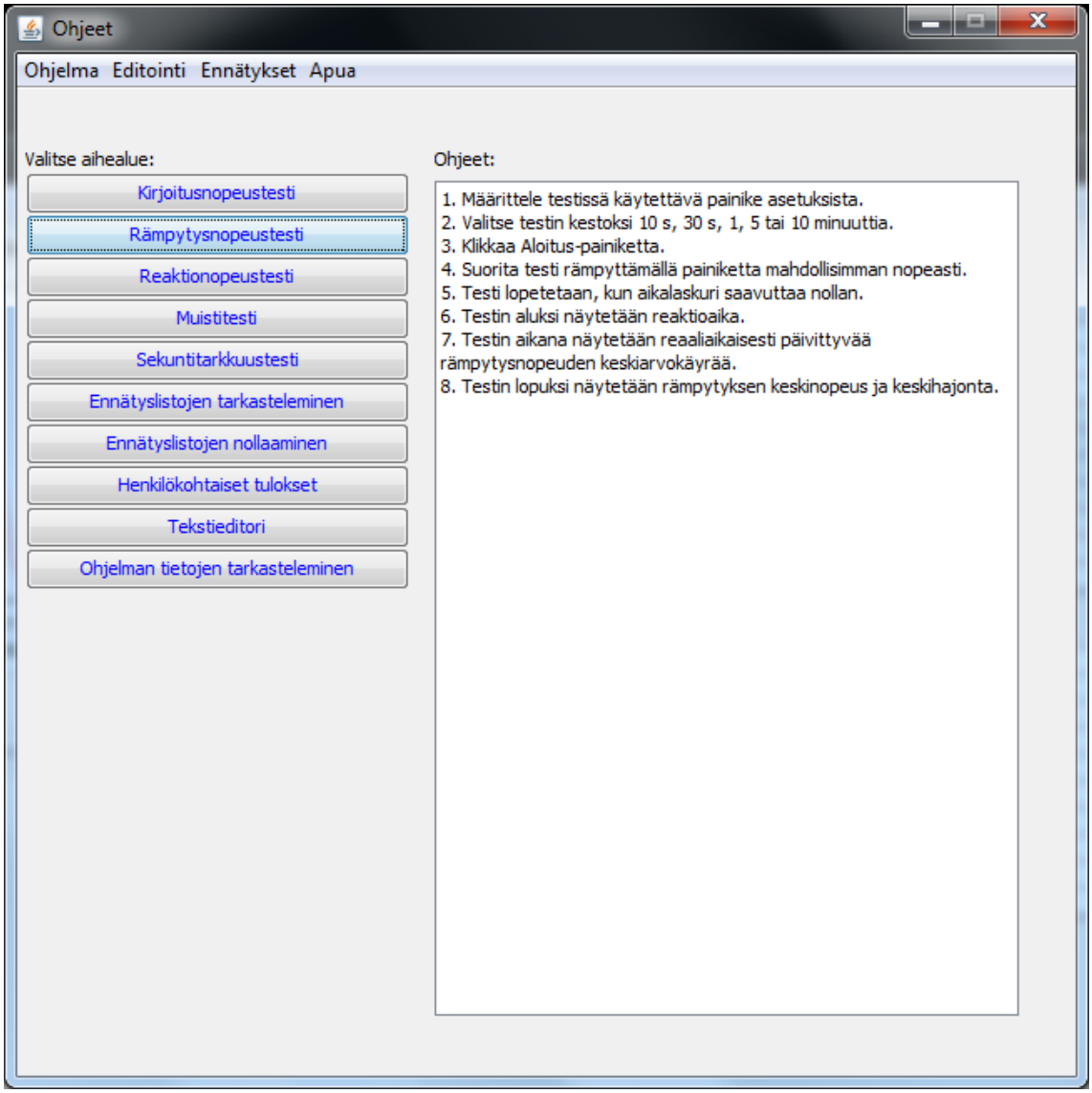
Kuva 17. Tekstieditori-ikkuna

Tekstieditori-ikkunassa käyttäjä voi kirjoittaa uuden kirjoitusnopeustestin harjoitusmoodin harjoitustekstin tekstieditorin tekstikenttään ja tallentaa tekstin uutena tietokantaan valitsemalla kuvassa 17 esitetystä yhdistelmäruudusta Uusi teksti ja klikkaamalla Tallenna-painiketta. Ohjelma nimeää tekstin automaattisesti. Olemassa olevan harjoitustekstin voi myös valita yhdistelmäruudusta ja hakea tietokannasta muokattavaksi tekstieditorin tekstikentässä. Kyseisen tekstin voi tallentaa muutosten jälkeen ylikirjoittaen vanhan harjoitustekstin tietokantaan valitsemalla yhdistelmäruudusta Vanha teksti, valitsemalla olemassa olevan harjoitustekstin nimen ja klikkaamalla Tallenna-

painiketta. Tekstiä ei voida tallentaa tyhjänä eikä tekstissä saa olla yli 300 merkkisiä sanoja.

6.2.11 Ohjeet-ikkuna

Valittaessa ohjelman valikkoriviltä Ohjeet avautuu kuvan 18 mukainen käyttöliittymä.



Kuva 18. Ohjeet-ikkuna

Ohjeet-ikkunassa käyttäjä voi klikata haluamansa aihealueen ohjeet näytettäväksi tekstialueella. Ohjeet näytetään järjestysnumeroin kuvan 18 mukaisesti.

6.2.12 Tiedot-ikkuna

Valittaessa ohjelman valikkoriviltä Ohjelman tiedot avautuu kuvan 19 mukainen käyttöliittymä.

Kuva 19. Tiedot-ikkuna

Tiedot-ikkunassa käyttäjälle näytetään ohjelman tiedot. Näytettäviä tietoja ovat ohjelman nimi, versio, toteutuspäivämäärä ja tekijä kuvan 19 mukaisesti.

7 Testaus

Tässä luvussa kerrotaan lyhyesti testauksesta. Luvun aluksi kerrotaan vähän, mitä testaus on ja esitellään testaustapoja. Sen jälkeen kerrotaan lyhyesti ohjelman testaustavoista. Testaus määritellään suunnitelmalliseksi virheiden etsimiseksi ohjelmaa suorittaessa [22]. Sen päätavoitteena on havaita ohjelman häiriöt, jotta virheet voitaisiin korjata [23].

7.1 Yleistä

Testaus tarjoaa tiedon testattavan tuotteen laadusta. Ohjelmaa voidaan testata missä tahansa kehitysvaiheessa testaustavasta riippuen. Testaus suoritetaan kuitenkin usein pääasiassa, kun toiminnalliset vaatimukset on määritelty ja ohjelma on ohjelmoitu valmiiksi. Erilaisia testaustapoja on musta-, harmaa- ja lasilaatikkotestaus. Mustalaatikkotestauksessa testaajalla ei ole lähdekoodia saatavilla, vaan ohjelmaa testataan esimer-

kiksi käyttöliittymän kautta. Harmaalaatikkotestauksessa testaaja voi halutessaan tarkastella algoritmeja ja tietorakenteita suunnittelun tasolla, mutta testaus on mustalaatikkotestausta. Lasilaatikkotestauksessa testaaja voi halutessaan tarkastella algoritmeja, tietorakenteita ja lähdekoodia testauksen aikana. [23.]

7.2 Ohjelman testaus

Testaus toteutettiin musta- ja lasilaatikkotestauksella koko ohjelmointivaiheen ajan kokeilemalla käyttöliittymän kaikki ominaisuudet yksi kerrallaan myös virheellisillä syötteillä ja raja-arvoilla sitä mukaa, kun ominaisuuksia lisättiin ohjelmaan. Testaajalla oli koko ajan kuitenkin mahdollisuus halutessaan tarkastella ohjelman lähdekoodia, jotta testattavien virheellisten syötteiden ja raja-arvojen keksiminen olisi helpompaa. Mahdollisia ohjelmointivirheitä korjattiin sitä mukaa, kun niitä ilmeni. Ohjelmointivaiheen loppupuolella käyttöliittymän jokainen ominaisuus testattiin vielä uudestaan läpi kaikenlaisilla syötteillä. Ominaisuudet suoritettiin järjestelmällisesti uudestaan mahdollisten uusien ohjelmointivirheiden löytämiseksi ja korjaamiseksi.

Testaus toteutettiin myös moduulitestauksena. Siinä jokainen pienempi ohjelmamoduuli testattiin erikseen, jotta varmistuttiin toteutuksen onnistumisesta. Lisäksi testaus toteutettiin järjestelmätestauksena, jossa integroitu ohjelma testattiin kokonaan. Tällöin varmistuttiin, että luvussa 2.1 olevat toiminnalliset vaatimukset täyttyvät. [23.]

8 Pohdintaa

Tämä luku sisältää vain pohdintaa. Luvun aluksi arvioidaan toteutettua ohjelmaa. Seuraavaksi esitellään joitakin toteutusvaiheen ongelmia. Lopuksi vielä esitellään kaksi ehdotusta, miten ohjelmaa voisi kehittää eteenpäin ja käyttää hyödyksi.

8.1 Toteutetun ohjelman arviointia

Insinöörityön tuloksena syntyi Java-ohjelmointikielellä toteutettu TestYourself-ohjelma, jossa on viisi testiä. Kirjoitus-, rämpytys- ja reaktionopeustestejä on Internetissä monia, mutta löydetyistä jokaisessa oli enemmän tai vähemmän eroavaisuuksia. Muistitestejä on myös monia, mutta ne olivat melko erilaisia. Sekuntitarkkuustestejä taas ei ollut ol-

lenkaan. Testit olivat erillisiä, eikä monia edellä mainittuja testejä sisältäviä yksittäisiä ohjelmistoja ollut. Java-ohjelmointikieli valittiin, koska se tarjoaa Swing-kirjaston, jonka valmiilla rajapinnoilla, luokilla ja komponenteilla käyttöliittymän rakentaminen on mielestäni kätevää ja suoraviivaista. Lisäksi Java on alustariippumaton ohjelmointikieli ja sillä kirjoitettu lähdekoodi on selkeää, mitkä ovat jatkokehityksen kannalta hyviä asioita. Ohjelman lähdekoodi pyrittiin kirjoittamaan mahdollisimman selkeäksi ja kommentoiduksi, jotta sen jatkokehitys olisi helpompaa. Jotkin ominaisuudet olisi luultavasti kirjoitettavissa lyhyemmin ja yksinkertaisemmin, mutta mielestäni lyhyt esimerkiksi yhden rivin pitkälle optimoitu toteutustapa ei aina ole helpoimmin ymmärrettävissä. Ohjelman tietokanta taas toteutettiin Microsoft Access -ohjelmistolla, koska se tarjoaa komentokehotteen sijaan graafisen näkymän tietokantataulujen ja niiden välisten yhteyksien luomiseen. Graafisen näkymän ansiosta myös tietokannan sisältämien tietojen myöhempi tarkastelu ja muokkaaminen on mutkattomampaa kuin komentokehotteen kautta.

Työn alkuvaiheessa koottiin luettelo toiminnallisista vaatimuksista, jotka toteutettavalla ohjelmalla tulee voida suorittaa. Toteutettu ohjelma täyttää nämä vaatimukset ja on siltä osin hyvin onnistunut. Seuraavaksi arvioidaan ohjelman osia tarkemmin.

8.1.1 Pääikkuna

Toteutettu ohjelma käynnistyy melko hitaasti, mihin on syynä mielestäni Java-virtuaalikoneen käynnistyminen ja ohjelman kääntäminen sillä. Käynnistyttyään ohjelma kuitenkin suorittaa toimintonsa nopeasti, eikä edes tietokannan käyttämisessä ole havaittavissa hitautta. Ensimmäisenä avautuva pääikkuna on mielestäni sopivan kokoinen, kun tarkastellaan testien paneelien täyttöastetta. Kirjoitus-, rämpytys- ja reaktionopeustestien käyttöliittymien komponentit täyttävät paneelit hyvin ja tyhjää tilaa on sopivasti. Muisti- ja sekuntitarkkuustestit taas sisältävät vähemmän tai pienempiä komponentteja ja siksi tyhjää tilaa on enemmän. Näiden testien komponentit on sijoitettu selkeästi paneelien keskelle, kuten esimerkiksi Internet-sivujen sisällöt usein, ja näin on saatu jaettua tyhjää tilaa tasaisesti komponenttien molemmille puolille. Vaihtoehtona olisi ollut toteuttaa testit kukin omaan erikokoiseen ikkunaansa, mutta silloin ikkunoita olisi ollut liikaa ja ohjelmassa eteneminen olisi ollut sekavampaa ja vaikeampaa. Tästä syystä ohjelman pääosa-alueiden eli testien sijoittaminen välilehtiin oli moderni ja hyvä valinta.

Kirjoitusnopeustestin käyttöliittymän tärkeimmät komponentit eli tekstialueet ovat mielestäni selkeästi esillä ja sopivan kokoiset. Niiden väliin on sijoitettu muita komponentteja, jotta ne eivät olisi liian lähellä toisiaan. Myös ajanotto ja oikeiden sanojen laskuri ovat tärkeitä, ja ne näytetään tekstialueiden välissä ja alemman tekstialueen vieressä, jolloin niitä on helppo seurata tekstiä kirjoittaessa kyseiseen tekstialueeseen. Muut komponentit eivät ole testin suorittamisen kannalta tärkeitä, joten esimerkiksi kirjoitusnopeuden keskiarvokäyrää piirretään oikealla reunassa.

Rämpytysnopeustestin käyttöliittymässä nopeuden keskiarvokäyrä on testin suorittamisen kannalta tärkeä, koska siitä ilmenee rämpytyksen tasaisuus. Tästä syystä se on sijoitettu mielestäni selkeästi esille. Rämpytysten lukumäärän, ajanoton, valitun painikkeen, reaktioajan ja nopeuden keskihajonnan näyttäminen on myös tärkeää, ja ne esitetäänkin siksi käyrän alla. Rämpytys- ja reaktionopeustestien Asetukset-paneeleita ei ole tarvetta näyttää jatkuvasti, ja tästä syystä ne ovatkin enimmäkseen piilossa.

Reaktionopeustestin suorittamisen kannalta käyttöliittymän tärkeimmät komponentit ovat värilliset painikkeet ja painallusten lukumäärän näyttävä tekstikenttä. Tästä syystä ne on sijoitettu esille samalla tavalla kuin reaalimaailman vastaavassa pelissä eli käyttöliittymän alaosaan keskelle allekkain. Valittujen painikkeiden näyttäminen on myös tärkeää, ja ne esitetäänkin lähellä värillisiä painikkeita. Vaikeusasteen valitseminen reaktionopeus- ja muistitestissä ei ole testin suorittamisen kannalta niin tärkeää, joten sen mahdollistavat komponentit on sijoitettu molemmissa käyttöliittymän vasempaan reunaan. Kirjoitus-, rämpytys- ja reaktionopeustesteissä ennätyslistat on sijoitettu käyttöliittymässä oikealle suuremman paneelien täyttöasteen takia. Muisti- ja sekuntitarkkuustestissä ne näytetään taas alimpana. Testien ennätyslistat esitetään selkeästi ruudukoissa, eikä esimerkiksi tekstialueilla, joissa tekstien tasaaminen on työläämpää.

Muistitestin suorittamisen kannalta tärkeimmät käyttöliittymän komponentit ovat merkkitaulukko, ajanoton näyttävä tekstikenttä ja painike vastaamiseen. Tästä syystä ne on sijoitettu mielestäni selkeästi käyttöliittymän yläosaan. Sekuntitarkkuustestin suorittamisen kannalta tärkeimmät komponentit taas ovat painike testin aloittamiseen ja lopettamiseen sekä käyttöliittymässä oleva tekstikenttä. Tästä syystä ne on sijoitettu myös vastaavasti kuin muistitestissä.

Kirjoitusnopeus-, rämpytysnopeus- ja sekuntitarkkuustesteissä on mielestäni sopiva lukumäärä kestoja valittavana, mitkä lisäävät testien monipuolisuutta. Reaktionopeus-

testin vaikeusasteet ovat mielestäni sopivan tasoisia painikkeiden valaisemisnopeuden kasvaessa. Muistitestissä taas vaikeusasteet osoittautuivat testauksessa melko vaikeiksi, mutta sopivasti haastava on mielestäni parempi kuin liian helppo. Siinä kysyttävien merkkien lukumäärät olivat alun perin vaikeusasteen mukaan 12 ja 20. Merkkejä oli kuitenkin liikaa, joten lukumäärät muutettiin 8:aan ja 16:een, jotta testi olisi sopivan haastava.

Sekuntitarkkuustestiä lukuun ottamatta jokaisessa testissä on painike testin keskeyttämiseen. Se on pyritty sijoittamaan niin, ettei sitä voisi vahingossa klikata testin suorituksen aikana. Jokainen testi keskeytetään myös välilehtien vaihtuessa. Tällä on pyritty nopeuttamaan ohjelman toimintaa, kun esimerkiksi ohjelman säikeitä ei suoriteta päällekkäin. Tällöin prosessoriaikaa jää enemmän kullakin hetkellä suoritettavalle testille.

8.1.2 Muut ikkunat

Ohjelman muut osa-alueet ovat valittavissa jokaisen käyttöliittymän yhtenevältä valikkoriviltä, ja ne avautuvat omiin erikokoisiin ikkunoihinsa. Ikkunoihin päädyttiin välilehtien sijaan, jotta vain ohjelman pääosa-alueet eli testit on sijoitettu selkeästi pääikkunaan. Lisäksi, jos välilehtiä olisi käytetty, niitä olisi ollut liikaa ja esimerkiksi haluttaessa siirtyä suorittamaan jonkin testin saattaisi päätyä ensin vahingossa tekstieditoriin tai muihin osa-alueisiin.

Tekstieditori-ikkunan käyttöliittymän tekstialue on mielestäni sopivan kokoinen, jotta pitkäkin teksti näkyisi kokonaisuudessaan ilman vieritysruutua. Harjoitustekstien haku- ja tallennuspainikkeet taas on sijoitettu erilleen toisistaan, jotta väärää painiketta ei voisi klikata vahingossa.

Ennätyslistat-ikkunan käyttöliittymän yhdistelmäruudussa ennätyslistat on lueteltu samassa järjestyksessä, kuin testit on sijoitettu pääikkunan välilehtiin vasemmalta oikealle. Tämä on loogisempi toteutustapa, kuin jos testit olisivat esimerkiksi aakkosjärjestyksessä. Lisäksi ennätyslistojen nollauspainike on sijoitettu erilleen yhdistelmäruudusta, jota sitä ei klikattaisi vahingossa. Henkilökohtaiset tulokset -ikkunan tekstialueella ennätyslistat on myös lueteltu loogisesti vastaavasti kuin Ennätyslistat-ikkunassa. Tekstialueeseen päädyttiin, koska siitä tulosten tarkastelu on nopeampaa ja vaivattomampaa kuin esimerkiksi valittaessa yhdistelmäruudusta jokaisen ennätyslistan erikseen.

Ohjeet-ikkunan käyttöliittymän painikkeet näyttävät erilaisilta kuin muut ohjelman painikkeet. Tähän päädyttiin, jotta ohjelman käyttöliittymä sisältäisi monipuolisemmin erilaisia komponentteja. Ohjeet näytetään tekstialueella järjestysnumeroin, jotta niitä seuraten osa-alueiden suorittaminen olisi helpompaa askel askeleelta. Tiedot-ikkunan koko on mielestäni sopiva ja siinä esitettävät tiedot selkeästi lueteltuna allekkain.

Ohjelman testausvaiheessa testikäyttäjä suoritti ohjelman kaikki ominaisuudet järjestelmällisesti. Testikäyttäjän mielestä ohjelman käyttöliittymät olivat selkeitä, mutta käytettävyyttä pitäisi parantaa. Tällä tarkoitettiin joidenkin ohjelman testien vaiheita, jotka osoittautuivat epäselviksi. Tämän takia käyttöliittymää muutettiin selkeämmäksi mukavamman käyttökokemuksen takaamiseksi.

8.2 Toteutusongelmia

Ohjelman toteutuksessa oli paljon erilaisia ongelmia. Tässä luvussa esitellään joitakin niistä.

Alkuperäisenä tarkoituksena oli, että ohjelma hyödyntäisi Internetin pilvipalveluja tiedon tallennustilana. Tällaisen ohjelman toteuttaminen osoittautui kuitenkin liian haastavaksi ja aikaa vieväksi, joten ohjelma päädyttiin rajaamaan käyttämään tietokoneen kovalevyn tietokantaa tietovarastona. Pilvipalvelujen lisääminen ohjelmaan on yhtenä jatkokehitysehdotuksena.

Tietokannan ohjelmaan liittämisessä ja sen hyödyntämisessä oli paljon haasteita. Etenkin DAO-suunnittelumallin mukainen tiedonkäsittelykerroksen eristäminen muusta ohjelmasta oli haastavaa. Lisäksi oikeanlaisten SQL-kyselyjen kirjoittaminen oli työlästä ja vaativaa varsinkin, kun nimien hakemiseen Nimitaulu-taulusta käytettiin luonnollista liitosta ennätystaulujen ja Nimitaulu-taulun välillä.

MVC-suunnittelumallin mukaisen ohjelman toteuttaminen oli myös haastavaa. Ohjainkerros oli toteutusvaiheessa pitkään käyttöliittymän komponenttien tapahtumankuuntelijana ja toteutti muun muassa ActionListener-rajapinnan tapahtumankäsittelymetodin. Näin toteutettaessa näkymäkerros oli kuitenkin liian sidoksissa ohjainkerrokseen ja vaikeammin vaihdettavissa tarvittaessa. Tästä syystä edellä mainittu tapahtumankäsittelymetodi poistettiin ohjainkerroksesta. Lisäksi käyttöliittymäluokkiin lisättiin anonyymit

sisäluokat kuuntelemaan komponenttien tapahtumia ja kutsumaan ohjainkerroksen tavallisia metodeja.

Keskiarvokäyrän piirtäminen ja ajanotto tietyin väliajoin toteutettiin Observer-suunnittelumallin mukaisesti. Lisäksi käyrän piirtämiseen käytettiin kaksoispuskurointia ja ajan laskemiseen säiettä. Nämä asiat olivat vaativia ja niiden toteutuksessa oli monia ongelmia, mutta useiden yritysten jälkeen ne kuitenkin saatiin toimimaan yhdessä.

Rämpytysnopeustestiin lisättiin nopeuden keskihajonnan laskeminen, jotta voitiin esittää testin aikaisen rämpytyksen tasaisuus numeroarvona. Laskeminen oli kuitenkin vaikea toteuttaa, koska jo keskihajonnan yhtälö on mielestäni monimutkainen ja haastava ymmärtää.

8.3 Ohjelman jatkokehitys

Insinöörityön tarkoituksena oli luoda ohjelma, jolla nuoriso ja muut kiinnostuneet voisivat testata sekä harjoittaa tietoteknisiä taitojaan, muistiaan ja tarkkuuttaan erilaisilla testeillä. Luvussa esitellään kaksi ehdotusta, miten ohjelmaa voisi kehittää eteenpäin ja käyttää hyödyksi.

8.3.1 Hoitoalalle toteutettu ohjelma

Ohjelmaa voitaisiin kehittää eteenpäin esimerkiksi iäkkäiden ihmisten hoitoalalle. Ohjelma voisi olla hyödyllinen esimerkiksi sormien jäykkyyttä, reaktiokyvyn hitautta, muistin huononemista tai ajantuntemuksen epätarkentumista ehkäisevässä hoidossa. Ohjelmaan voitaisiin lisätä ominaisuudeksi testien tavoitteiden asettamisen ja niiden täyttyessä tulos tallennettaisiin. Tämän jälkeen ohjelmassa voitaisiin siirtyä seuraavalle tavoitetasolle.

Ohjelma myös voisi tallentaa kaikki testeissä saadut tulokset tietokantaan, eikä vain kymmentä parasta tulosta. Näin voitaisiin esimerkiksi tarkastella käyttäjän kehittymistä kuukausittain ja vertailla eri ajanjaksojen tuloksia keskenään, mikä helpottaisi testien tavoitteiden asettamista. Lisäksi tietokannasta voitaisiin tulostaa esimerkiksi käyttäjien kuukausien tuloksia raportteina, ja vertailla yhden käyttäjän tuloksia kaikkien tulosten keskiarvoon.

8.3.2 Internetin pilvipalveluja hyödyntävä ohjelma

Ohjelman oli alun perin tarkoitus hyödyntää Internetin pilvipalveluja tietojen tallennustilana. Tällaisen ohjelman toteuttaminen osoittautui kuitenkin liian haastavaksi ja aikaa vieväksi, joten pilvipalvelujen lisääminen ohjelmaan tietokannan rinnalle olisi hyvä ja haasteellinen jatkokehitysehdotus. Ohjelmaan voitaisiin lisätä ominaisuudeksi keskustelualue, jota kautta käyttäjät voisivat esimerkiksi vertailla tuloksiaan ja kilpailla keskenään. Lisäksi katsomo olisi mielenkiintoinen ominaisuus ohjelmassa, missä muut käyttäjät voisivat seurata yhden käyttäjän suoritusta testeissä. Ohjelman hyödyntämät pilvipalvelut mahdollistaisivat sen, että eri tietokoneiden käyttäjillä olisi käytössään samat ohjelman käyttämät tiedot eli testien ennätyslistat ja tekstit, mikä lisäisi testien tavoitteita sekä ohjelman kiinnostavuutta ja käyttöikää entisestään.

9 Yhteenveto

Tässä luvussa kerrataan dokumentin sisältöä. Luvun aluksi kerrataan insinöörityön tavoite. Sen jälkeen kerrataan vielä insinöörityön vaiheet ja saadut tulokset. Insinöörityön tavoitteena oli luoda ohjelma, jolla voidaan testata sekä harjoittaa lukunopeutta, kymmensormijärjestelmää, rämpytysnopeutta, rämpytyksen tasaisuutta, reaktioaikaa, lyhytkestoista muistia ja ajantuntemusta erilaisilla testeillä. Ohjelma sisältää viisi pääosa-aluetta: kirjoitusnopeustestin, rämpytysnopeustestin, reaktionopeustestin, muistitestin ja sekuntitarkkuustestin. Muut ohjelman osat liittyvät edellä mainittuihin testeihin. Tekstieditori-ikkunassa voidaan muokata kirjoitusnopeustestin harjoitusmoodin harjoitustekstejä. Ennätyslistat-ikkunassa voidaan tarkastella ja nollata testien tuloksia. Henkilökohtaiset tulokset -ikkunassa voidaan tarkastella ohjelman käyttäjien tuloksia. Ohjeet-ikkunassa voidaan tarkastella ohjelman osien käyttöohjeita. Tiedot-ikkunassa voidaan tarkistaa tiedot eli ohjelman nimen, version, toteutuspäivämäärän ja tekijän. Tavoitteena oli myös, että ohjelma hyödyntää tietojen tallennuksessa tietokoneen kovalevyn tietokantaa, joka mahdollistaa sen, että tietokoneen eri käyttäjillä on käytössään samat tietokantaan keskitetysti tallennetut tiedot.

Insinöörityön aluksi valittiin aihe, pohdittiin työn tarkoitusta ja asetettiin tavoitteeksi luoda edellä mainittu ohjelma. Seuraavaksi kirjoitettiin ohjelman toiminnallinen määrittely. Tämä aloitettiin kokoamalla ohjelman toiminnalliset vaatimukset eli ominaisuudet, jotka siinä tulisi olla. Niiden pohjalta kirjoitettiin ohjelman käyttäjän tyypillisiä tilanneku-

vauksia ja laadittiin käyttötapauskaavioita, mitkä ovat liitteinä. Sitten määriteltiin ohjelman ominaisuudet eli mietittiin, mitä ohjelma tekee. Tämän jälkeen seuraavana vaiheena kirjoitettiin tekninen määrittely eli suunnitelma, jossa mietittiin, miten ohjelma tekee toimintonsa. Tämä aloitettiin valitsemalla ohjelmointikielet ja ohjelmointiympäristöt, joilla ohjelma toteutetaan. Seuraavaksi mietittiin, mitä valmiita Javan kirjastoja ohjelmoinnissa käytetään. Sitten suunniteltiin, millä menetelmillä ohjelma toteutetaan, sekä mitä valmiita rajapintoja ja luokkia ohjelmassa käytetään. Määrittelyn ja suunnittelun jälkeen seuraavaksi oli ohjelman toteutusvaihe.

Ohjelman toteuttamisen yhteydessä pohdittiin tarkemmin jokaisen edellä mainitun testin tarkoitusta ja tuloksia. Ohjelman toteutusvaiheessa pohdittiin myös, miten tietokanta yhdistettäisiin lähdekoodiin, miten se eristettäisiin mahdollisimman irralliseksi osaksi ohjelmaa ja mitä tietokantatauluja tarvittaisiin. Toteutetusta ohjelmasta piirrettiin luokkakaavio, jossa esitetään ohjelman luokat ja niiden väliset yhteydet. Sen jälkeen esiteltiin ohjelman käyttöliittymä kokonaisuudessaan ja kerrottiin ohjelman ominaisuudet. Lisäksi kerrottiin lyhyesti ohjelman testauksesta toteutusvaiheessa ja sen jälkeen. Lopuksi vielä arvioitiin insinöörityön tulosta eli toteutettua ohjelmaa ja kerrottiin joistakin toteutusvaiheen ongelmista. Lisäksi luvussa 8.3 esiteltiin kaksi ehdotusta, miten tulokseksi saatua ohjelmaa voitaisiin kehittää eteenpäin ja käyttää hyödyksi tulevaisuudessa.

Insinöörityön tulokseksi saatiin Java-ohjelmointikielellä toteutettu testausohjelma, jossa on kaikki toiminnalliset vaatimukset ominaisuuksina. Ohjelma käyttää tietojen pysyvään tallennukseen tietokoneen kovalevyn tietokantaa. Ohjelma toteutettiin MVC-suunnittelumallin mukaisesti, jolloin käyttöliittymäluokat, ohjain, malliluokat ja ohjelman käyttämä tietokanta ovat mahdollisimman erillään toisistaan ja siten helposti tarvittaessa vaihdettavissa. Toisena tuloksena saatiin dokumentti, joka sisältää kaikki insinöörityön vaiheet taulukoineen ja kuvineen.

Lähteet

1 Motoriikka. 2013. Verkkodokumentti. Havaintomotoriikka. <http://fi.wikipedia.org/wiki/Motoriikka>. Luettu 11.11.2013.

2 Siivonen, Marko. Yleistä sensomotoriikasta. Verkkodokumentti. <http://www.sensomotoriikka.fi/>. Luettu 11.11.2013.

3 Julkisen hallinnon tietohallinnon neuvottelukunta Juhta. 2007. Tietojärjestelmän vaatimusten määrittely osana järjestelmän hankintaa. Verkkodokumentti. <http://docs.jhs-suositukset.fi/jhs-suositukset/JHS165/JHS165.html>. Luettu 11.11.2013.

4 Luukkainen, Matti. Laine, Harri. 2010. Ohjelmistojen mallintaminen. Verkkodokumentti. <http://www.cs.helsinki.fi/u/mluukkai/ohmas10/ohma.pdf>. Luettu 11.11.2013.

5 Alfanumeerinen. 2013. Verkkodokumentti. <http://fi.wiktionary.org/wiki/alfanumeerinen>. Luettu 11.11.2013.

6 Alustariippumattomuus. 2013. Verkkodokumentti. <http://fi.wikipedia.org/wiki/Alustariippumattomuus>. Luettu 11.11.2013.

7 Java. 2013. Verkkodokumentti. <http://fi.wikipedia.org/wiki/Java>. Luettu 11.11.2013.

8 SQL. 2013. Verkkodokumentti. <http://fi.wikipedia.org/wiki/SQL>. Luettu 11.11.2013.

9 Eclipse (IDE). 2013. Verkkodokumentti. <http://fi.wikipedia.org/wiki/Eclipse_(IDE)>. Luettu 11.11.2013.

10 Microsoft Access. 2013. Verkkodokumentti. <http://en.wikipedia.org/wiki/Microsoft_Access>. Luettu 11.11.2013.

11 Silander, S., Ollikainen, V. & Peltomäki, J. 2010. Java. Jyväskylä: Docendo.

12 Swing (Java). 2013. Verkkodokumentti. <http://fi.wikipedia.org/wiki/Swing_(Java)>. Luettu 11.11.2013.

13 Abstract Window Toolkit. 2013. Verkkodokumentti. <http://fi.wikipedia.org/wiki/Abstract_Window_Toolkit>. Luettu 11.11.2013.

14 Gamma, E., Helm, R., Johnson, R. & Vlissides, J. 2001. Design Patterns, Olio-ohjelmointi, Suunnittelumallit. Helsinki: Oy Edita Ab.

15 Prosessi (tietotekniikka). 2013. Verkkodokumentti. Säie. <http://fi.wikipedia.org/wiki/Säie_(tietotekniikka)>. Luettu 11.11.2013.

16 Words per minute. 2013. Verkkodokumentti. <http://en.wikipedia.org/wiki/Words_per_minute>. Luettu 11.11.2013.

17 Reaction Time Statistics. 2013. Verkkodokumentti. <http://www.humanbenchmark.com/tests/reactiontime/stats.php>. Luettu 11.11.2013.

18 Hajontaluku. 2013. Verkkodokumentti. <http://fi.wikipedia.org/wiki/Hajontaluku>. Luettu 11.11.2013.

19 Tietokanta. 2013. Verkkodokumentti. <http://fi.wikipedia.org/wiki/Tietokanta>. Luettu 11.11.2013.

20 Käsitteellinen mallintaminen (Entity-Relationship -kaaviot) –Luento 2. 2011. Verkkodokumentti. <http://appro.mit.jyu.fi/tiedonhallinta/luennot/luento2/>. Luettu 11.11.2013.

21 Peltomäki, J. & Silander, S. 2003. Java 2, Ohjelmoinnin peruskirja. Jyväskylä: Docendo.

22 Testauksen tavoitteet. Verkkodokumentti. <http://www.oamk.fi/sbc/testaus/testauksen_tavoitteet.htm>. Luettu 11.11.2013.

23 Ohjelmiston testaaminen. 2013. Verkkodokumentti. <http://fi.wikipedia.org/wiki/Ohjelmiston_testaaminen>. Luettu 11.11.2013.

Tilannekuvaukset

Testin valitseminen

Tilannekuvauksen esiehtona on, että ohjelman käyttäjä haluaa valita suoritettavan testin taulukon 8 kuvauksen mukaisesti.

Taulukko 8. Testin valitseminen

Kuvaus	1. Käyttäjä käynnistää ohjelman. 2. Hän valitsee avautuvasta käyttöliittymästä testin: kirjoitusnopeustesti, rämpytysnopeustesti, reaktionopeustesti, muistitesti tai sekuntitarkkuustesti.
Poikkeukset	-
Lopputulos	Käyttäjä on valinnut haluamansa testin.

Kirjoitusnopeustestin suorittaminen

Tilannekuvauksen esiehtona on, että käyttäjä on valinnut kirjoitusnopeustestin ja haluaa suorittaa sen taulukon 9 kuvauksen mukaisesti.

Taulukko 9. Kirjoitusnopeustestin suorittaminen

Kuvaus	1. Käyttäjä valitsee harjoitusmoodin tai kilpailumoodin. Harjoitusmoodissa testattava teksti valitaan tietokannasta. Kilpailumoodissa ohjelma arpoo testattavan tekstin tietokannasta (Poikkeus: Tietokannan käsittely ei onnistunut). 2. Hän valitsee yhdistelmäruudusta testin keston: 1, 5 tai 10 minuuttia. 3. Hän klikkaa Aloitus-painiketta. 4. Hän suorittaa testin kirjoittamalla mahdollisimman nopeasti ja virheettömästi. Mahdollisten kirjoitusvirheiden jälkeen hän korjaa punaiseksi värjätyt sanat. 5. Testi lopetetaan, kun ajanotto saavuttaa nollan tai teksti on kirjoitettu kokonaisuudessaan oikein.
Poikkeukset	Tietokannan käsittely ei onnistunut: Ohjelma

	ilmoittaa käyttäjälle tilanteesta ja hän yrittää myöhemmin uudelleen.
Lopputulos	Käyttäjä on suorittanut kirjoitusnopeustestin onnistuneesti ja on samalla tutkinut keskiarvokäyrästä kirjoituksen keskinopeutta sekä katsonut keskinopeuden.

Rämpytysnopeustestin suorittaminen

Tilannekuvauksen esiehtona on, että käyttäjä on valinnut rämpytysnopeustestin ja haluaa suorittaa sen taulukon 10 kuvauksen mukaisesti.

Taulukko 10. Rämpytysnopeustestin suorittaminen

Kuvaus	1. Käyttäjä määrittelee testissä käytettävän hiiren tai näppäimistön painikkeen asetuksista. 2. Hän valitsee yhdistelmäruudusta testin keston: 10 s, 30 s, 1, 5 tai 10 minuuttia. 3. Hän klikkaa Aloitus-painiketta. 4. Hän suorittaa testin rämpyttämällä painiketta mahdollisimman nopeasti. 5. Rämpytystä ei saa aloittaa liian aikaisin, jotta ei tule varaslähtöä. 6. Testi lopetetaan, kun ajanotto saavuttaa nollan.
Poikkeukset	-
Lopputulos	Käyttäjä on suorittanut rämpytysnopeustestin onnistuneesti ja on samalla tutkinut keskiarvokäyrästä rämpytyksen tasaisuutta sekä katsonut reaktioajan ja nopeuden keskihajonnan.

Reaktionopeustestin suorittaminen

Tilannekuvauksen esiehtona on, että käyttäjä on valinnut reaktionopeustestin ja haluaa suorittaa sen taulukon 11 kuvauksen mukaisesti.

Taulukko 11. Reaktionopeustestin suorittaminen

Kuvaus	1. Käyttäjä määrittelee testissä käytettävät hiiren tai näppäimistön painikkeet asetuksista. 2. Hän valitsee yhdistelmäruudusta testin vaikeusasteen.

	3. Hän klikkaa Aloitus-painiketta. 4. Hän suorittaa testin painamalla neljää eriväristä painiketta sitä mukaa, kun niitä valaistaan jatkuvasti nopeutuvassa tahdissa. Ohjelma valaisee painikkeita vaikeusasteen mukaisella nopeudella. 5. Testi lopetetaan, kun valaistuja painikkeita klikataan liian hitaasti, klikataan väärää painiketta tai ei klikata ollenkaan painikkeita.
Poikkeukset	-
Lopputulos	Käyttäjä on suorittanut reaktionopeustestin onnistuneesti.

Muistitestin suorittaminen

Tilannekuvauksen esiehtona on, että käyttäjä on valinnut muistitestin ja haluaa suorittaa sen taulukon 12 kuvauksen mukaisesti.

Taulukko 12. Muistitestin suorittaminen

Kuvaus	1. Käyttäjä valitsee yhdistelmäruudusta testin vaikeusasteen. 2. Hän klikkaa Aloitus-painiketta. 3. Hän suorittaa testin keskustelulla ohjelman kanssa: Ohjelma näyttää noin 10 – 2 sekuntia joko 8 tai 16 merkkiä vaikeusasteen mukaisesti käyttöliittymän ruudukossa nopeutuen sekunnilla joka kierroksella. Kun merkit on näytetty, hän kirjoittaa noin 30 sekunnin aikana näytetyt merkit samassa järjestyksessä kyseiseen ruudukkoon ja klikkaa Vastaus-painiketta. Testin kesto vähenee jokaisella kierroksella väärien vastausten lukumäärän. 4. Testi lopetetaan, kun kesto on nolla.
Poikkeukset	-
Lopputulos	Käyttäjä on suorittanut muistitestin onnistuneesti.

Sekuntitarkkuustestin suorittaminen

Tilannekuvauksen esiehtona on, että käyttäjä on valinnut sekuntitarkkuustestin ja haluaa suorittaa sen taulukon 13 kuvauksen mukaisesti.

Taulukko 13. Sekuntitarkkuustestin suorittaminen

Kuvaus	1. Käyttäjä valitsee yhdistelmäruudusta testin keston: 2 s, 10 s tai 10 minuuttia. 2. Hän klikkaa Aloitus-painiketta. 3. Hän suorittaa testin painamalla painiketta aloittaessaan ja lopettaessaan päässään sekuntien laskemisen. 4. Testi lopetetaan, kun painiketta painetaan sekuntien laskemisen jälkeen.
Poikkeukset	-
Lopputulos	Käyttäjä on suorittanut sekuntitarkkuustestin onnistuneesti.

Testien ennätyslistojen tarkasteleminen

Tilannekuvauksen esiehtona on, että käyttäjä haluaa tarkastella testien ennätyslistoja taulukon 14 kuvauksen mukaisesti.

Taulukko 14. Testien ennätyslistojen tarkasteleminen

Kuvaus	1. Käyttäjä käynnistää ohjelman. 2. Hän valitsee avautuvan käyttöliittymän valikkoriviltä Ennätyslistat. 3. Hän valitsee yhdistelmäruudusta ennätyslistan tarkasteltavaksi tietokannasta (Poikkeus: Tietokannan käsittely ei onnistunut).
Poikkeukset	Tietokannan käsittely ei onnistunut: Ohjelma ilmoittaa käyttäjälle tilanteesta ja hän yrittää myöhemmin uudelleen.
Lopputulos	Käyttäjä on tarkastellut testien ennätyslistoja onnistuneesti.

Testien ennätyslistojen nollaaminen

Tilannekuvauksen esiehtona on, että käyttäjä haluaa nollata testien ennätyslistoja taulukon 15 kuvauksen mukaisesti.

Taulukko 15. Testien ennätyslistojen nollaaminen

Kuvaus	1. Käyttäjä käynnistää ohjelman. 2. Hän valitsee avautuvan käyttöliittymän valikkoriviltä Ennätyslistat. 3. Hän valitsee yhdistelmäruudusta ennätyslistan tarkasteltavaksi tietokannasta (Poikkeus: Tietokannan käsittely ei onnistunut). 4. Hän nollaa valitsemansa testin ennätyslistan tietokannasta (Poikkeus: Tietokannan käsittely ei onnistunut).
Poikkeukset	Tietokannan käsittely ei onnistunut: Ohjelma ilmoittaa käyttäjälle tilanteesta ja hän yrittää myöhemmin uudelleen.
Lopputulos	Käyttäjä on nollannut testien ennätyslistoja onnistuneesti.

Henkilökohtaisten tulosten tarkasteleminen

Tilannekuvauksen esiehtona on, että käyttäjä haluaa tarkastella henkilökohtaisia tuloksia taulukon 16 kuvauksen mukaisesti.

Taulukko 16. Henkilökohtaisten tulosten tarkasteleminen

Kuvaus	1. Käyttäjä käynnistää ohjelman. 2. Hän valitsee avautuvan käyttöliittymän valikkoriviltä Henkilökohtaiset tulokset. 3. Hän valitsee yhdistelmäruudusta käyttäjän nimen perusteella henkilökohtaisia tuloksia tarkasteltavaksi tietokannasta (Poikkeus: Tietokannan käsittely ei onnistunut).
Poikkeukset	Tietokannan käsittely ei onnistunut: Ohjelma ilmoittaa käyttäjälle tilanteesta ja hän yrittää myöhemmin uudelleen.
Lopputulos	Käyttäjä on tarkastellut henkilökohtaisia tuloksia onnistuneesti.

Kirjoitusnopeustestin harjoitustekstin muokkaaminen tekstieditorilla

Tilannekuvauksen esiehtona on, että käyttäjä haluaa kirjoittaa uuden kirjoitusnopeustestin harjoitusmoodin harjoitustekstin ja muokata olemassa olevaa harjoitustekstiä tekstieditorilla taulukon 17 kuvauksen mukaisesti.

Taulukko 17. Harjoitustekstin muokkaaminen tekstieditorilla

Kuvaus	1. Käyttäjä käynnistää ohjelman. 2. Hän valitsee avautuvan käyttöliittymän valikkoriviltä Tekstieditori. 3. Hän kirjoittaa uuden kirjoitusnopeustestin harjoitusmoodin harjoitustekstin tekstieditorin tekstikenttään ja tallentaa tekstin uutena tietokantaan (Poikkeus: Tietokannan käsittely ei onnistunut). 4. Hän valitsee yhdistelmäruudusta ja hakee olemassa olevan harjoitustekstin tietokannasta muokattavaksi tekstieditorin tekstikentässä ja tallentaa ylikirjoittaen vanhan harjoitustekstin muutosten jälkeen tietokantaan (Poikkeus: Tietokannan käsittely ei onnistunut).
Poikkeukset	Tietokannan käsittely ei onnistunut: Ohjelma ilmoittaa käyttäjälle tilanteesta ja hän yrittää myöhemmin uudelleen.
Lopputulos	Käyttäjä on muokannut kirjoitusnopeustestin harjoitusmoodin harjoitustekstejä tekstieditorilla onnistuneesti.

Ohjelman ohjeiden tarkasteleminen

Tilannekuvauksen esiehtona on, että käyttäjä haluaa tarkastella ohjelman ohjeita taulukon 18 kuvauksen mukaisesti.

Taulukko 18. Ohjelman ohjeiden tarkasteleminen

Kuvaus	1. Käyttäjä käynnistää ohjelman. 2. Hän valitsee avautuvan käyttöliittymän valikkoriviltä Ohjeet. 3. Hän klikkaa painikkeesta ohjeen tarkasteltavaksi tietokannasta. Ohjeissa neuvotaan ohjelman eri osa-alueiden käyttäminen (Poikkeus: Tietokannan käsittely ei onnistunut).

Poikkeukset	Tietokannan käsittely ei onnistunut: Ohjelma ilmoittaa käyttäjälle tilanteesta ja hän yrittää myöhemmin uudelleen.
Lopputulos	Käyttäjä on tarkastellut ohjelman ohjeita onnistuneesti.

Ohjelman tietojen tarkasteleminen

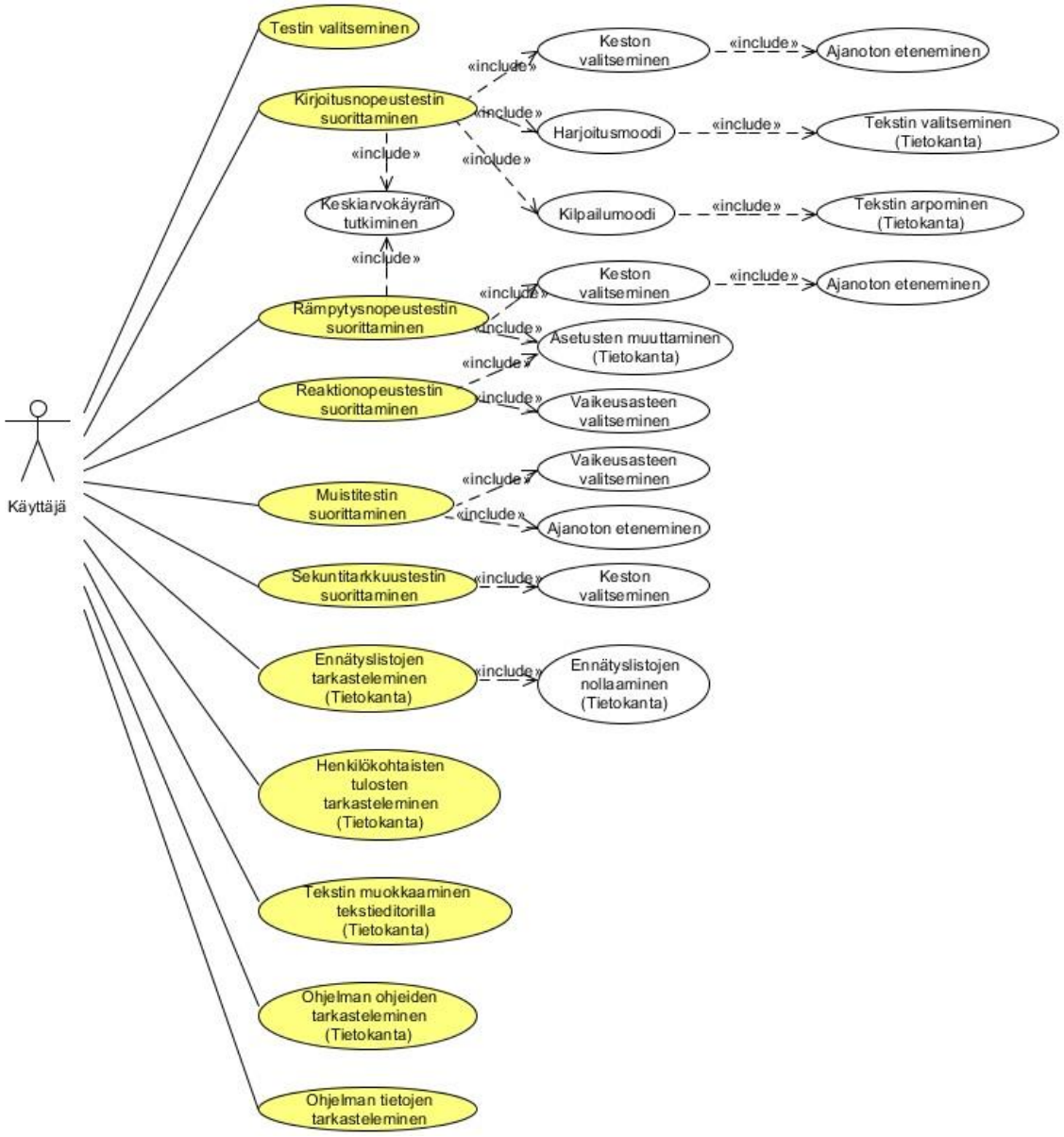
Tilannekuvauksen esiehtona on, että käyttäjä haluaa tarkastella ohjelman tietoja taulukon 19 kuvauksen mukaisesti.

Taulukko 19. Ohjelman tietojen tarkasteleminen

Kuvaus	1. Käyttäjä käynnistää ohjelman. 2. Hän valitsee avautuvan käyttöliittymän valikkoriviltä Ohjelman tiedot. 3. Hän tarkastelee ohjelman tietoja. Tietoja ovat ohjelman nimi, versio, toteutuspäivämäärä ja tekijä.
Poikkeukset	-
Lopputulos	Käyttäjä on tarkastellut ohjelman tietoja onnistuneesti.

Käyttötapaukset

Käyttötapauskaavio sisältää käyttötavat, käyttäjän ja toteutettavan ohjelman. Insinöörityössä toteutettavan ohjelman käyttötapauskaavio on kuvassa 20.

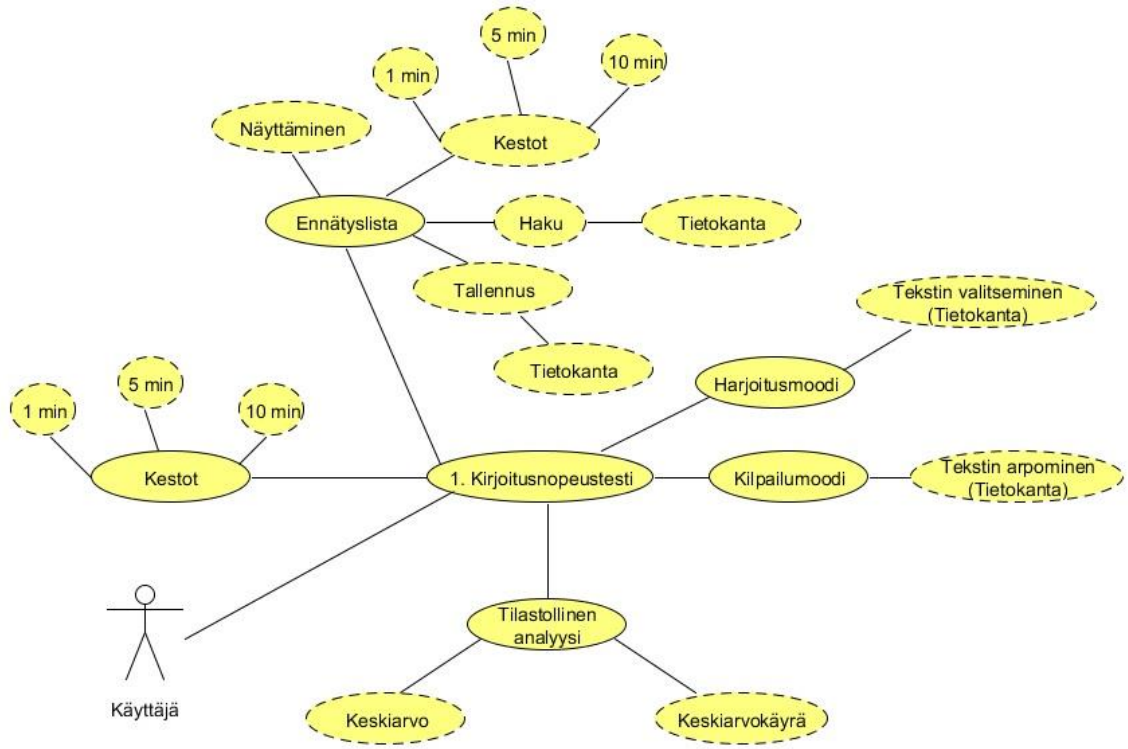


Kuva 20. Ohjelman käyttötapauskaavio

Seuraavissa alaluvuissa on esitelty kokonaiskäyttötapauskaavion käyttötapaukset tarkemmin.

Kirjoitusnopeustesti

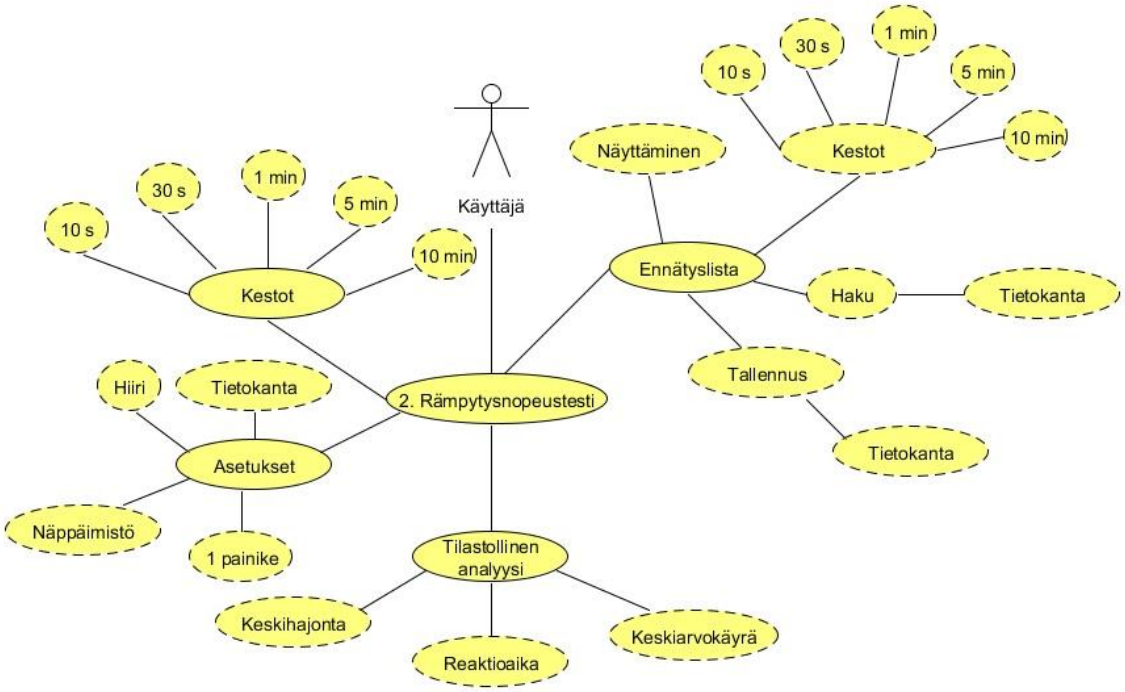
Kirjoitusnopeustestin käyttötapauskaavio on esitetty kuvassa 21.



Kuva 21. Kirjoitusnopeustestin käyttötapauskaavio

Rämpytysnopeustesti

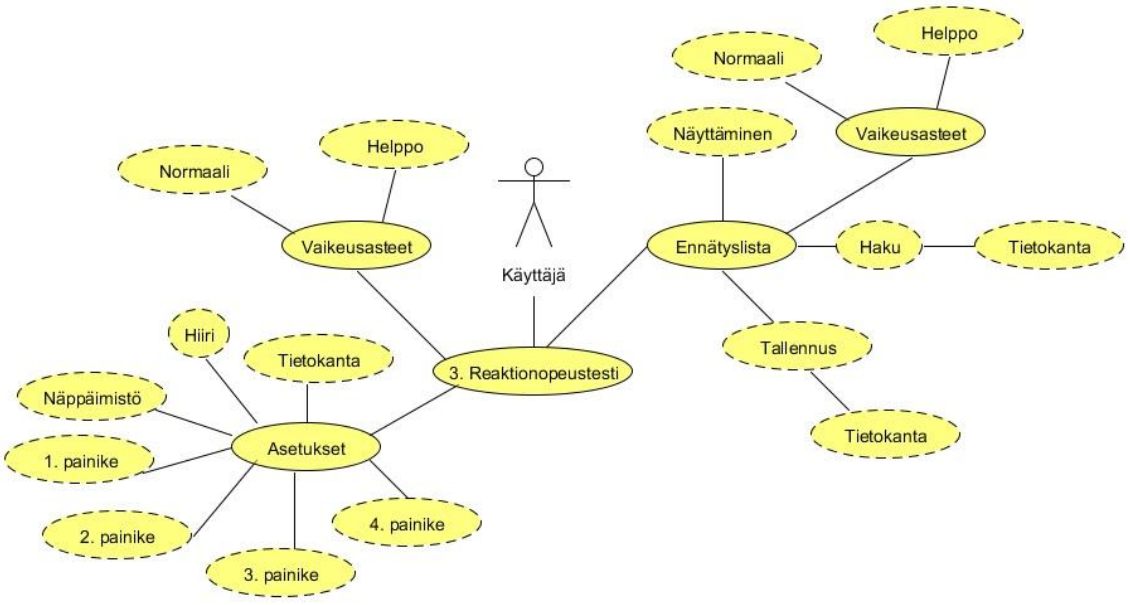
Rämpytysnopeustestin käyttötapauskaavio on esitetty kuvassa 22.



Kuva 22. Rämpytysnopeustestin käyttötapauskaavio

Reaktionopeustesti

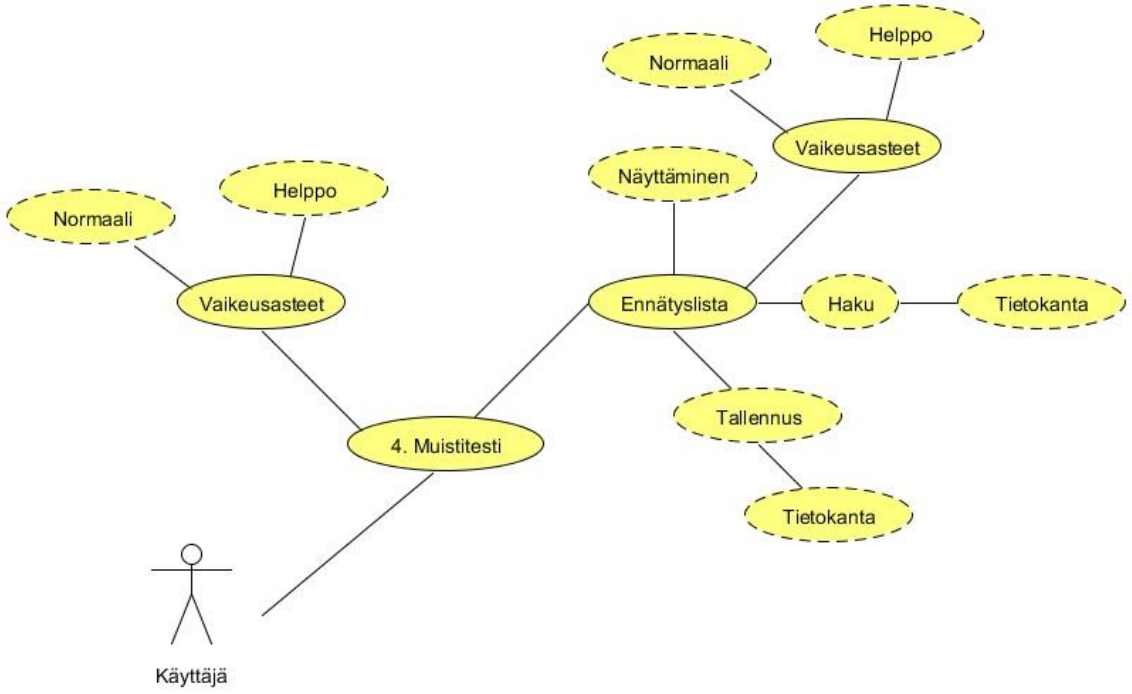
Reaktionopeustestin käyttötapauskaavio on esitetty kuvassa 23.



Kuva 23. Reaktionopeustestin käyttötapauskaavio

Muistitesti

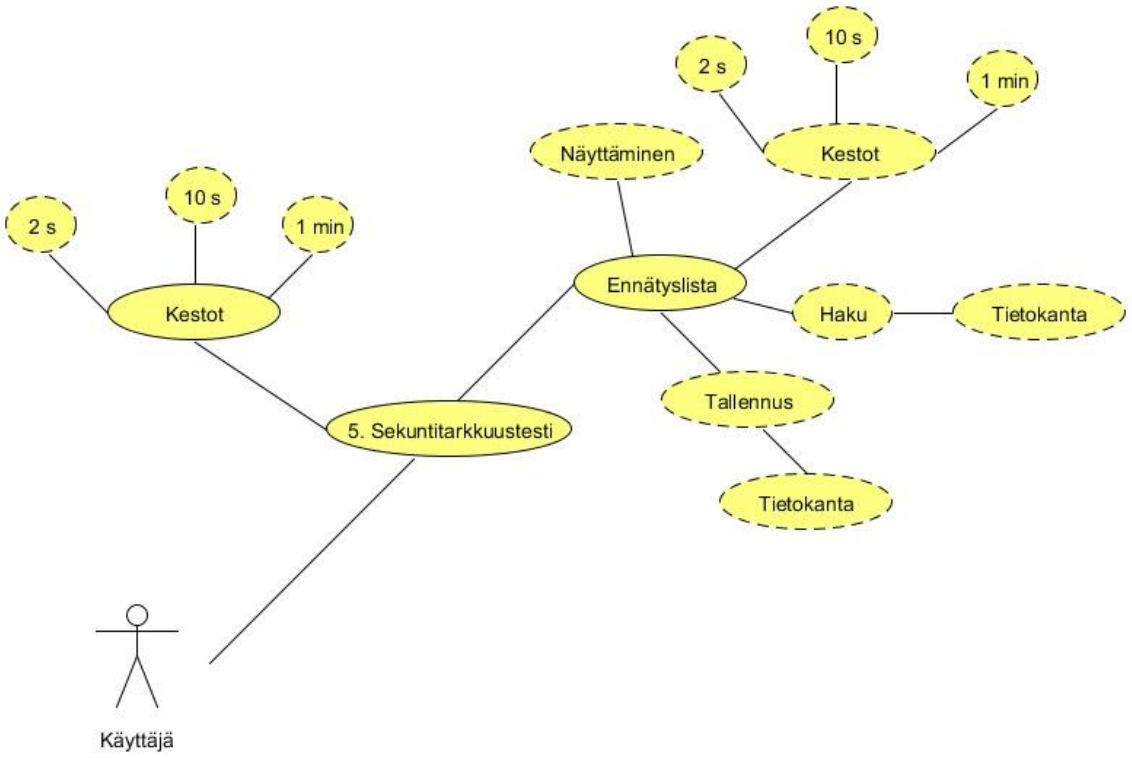
Muistitestin käyttötapauskaavio on esitetty kuvassa 24.



Kuva 24. Muistitestin käyttötapauskaavio

Sekuntitarkkuustesti

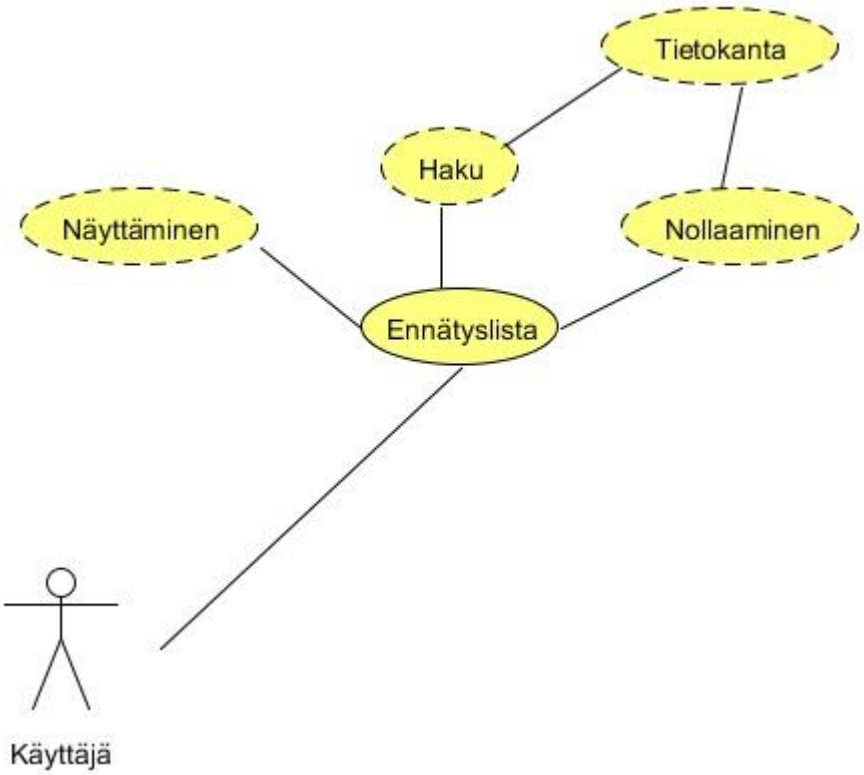
Sekuntitarkkuustestin käyttötapauskaavio on esitetty kuvassa 25.



Kuva 25. Sekuntitarkkuustestin käyttötapauskaavio

Ennätyslistojen näyttäminen ja nollaaminen

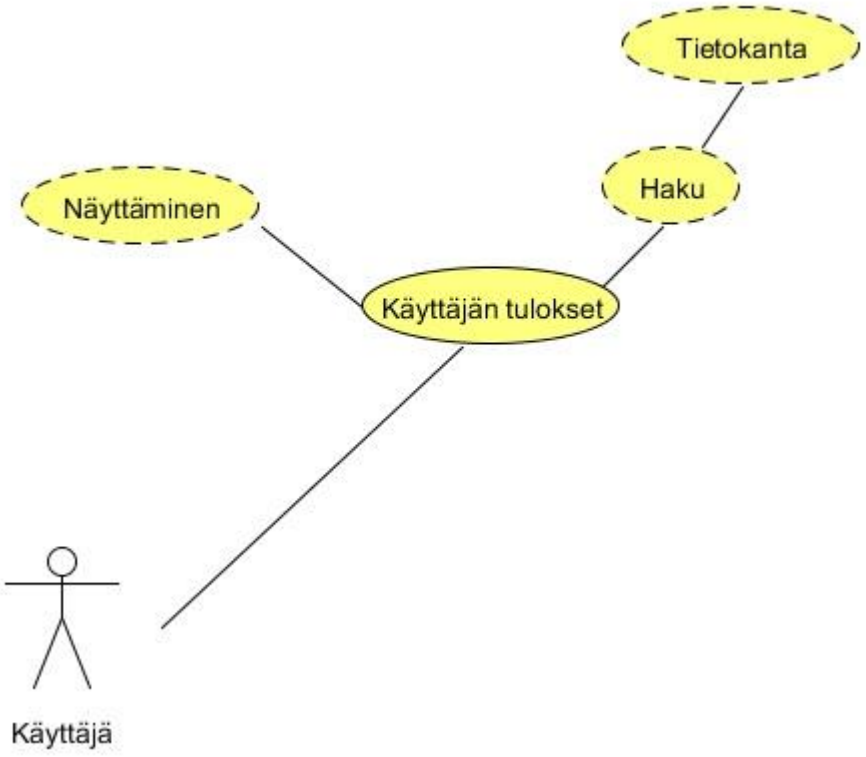
Ennätyslistojen näyttämisen ja nollaamisen sisältävä käyttötapauskaavio on esitetty kuvassa 26.



Kuva 26. Ennätyslistojen näyttämisen ja nollaamisen sisältävä käyttötapauskaavio

Henkilökohtaiset tulokset

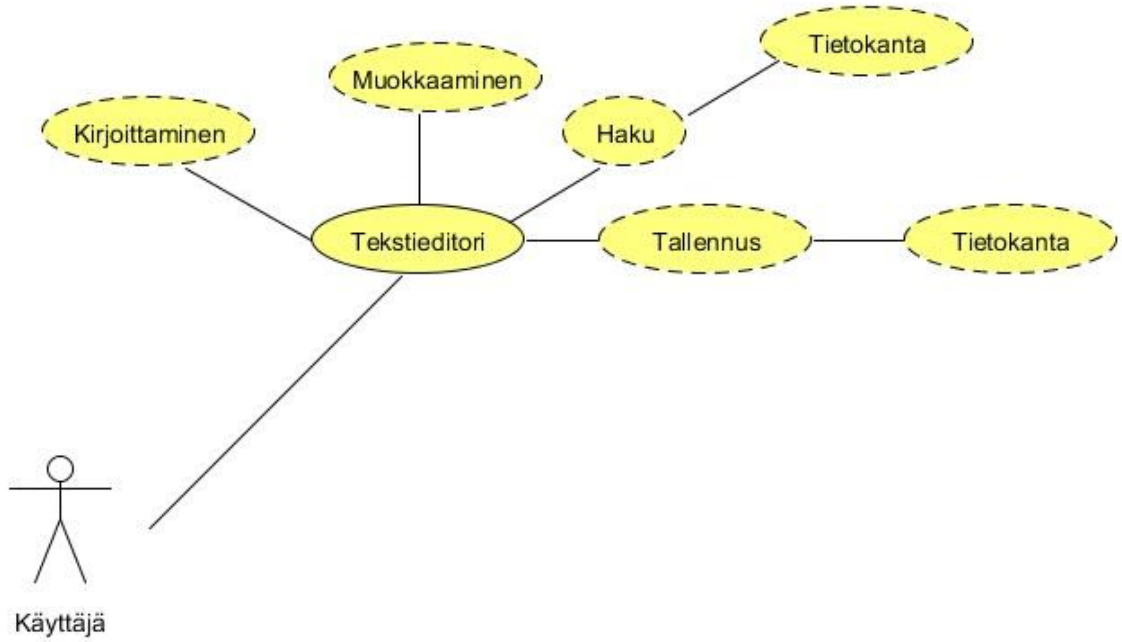
Henkilökohtaiset tulokset sisältävä käyttötapauskaavio on esitetty kuvassa 27.



Kuva 27. Henkilökohtaiset tulokset sisältävä käyttötapauskaavio

Tekstieditori

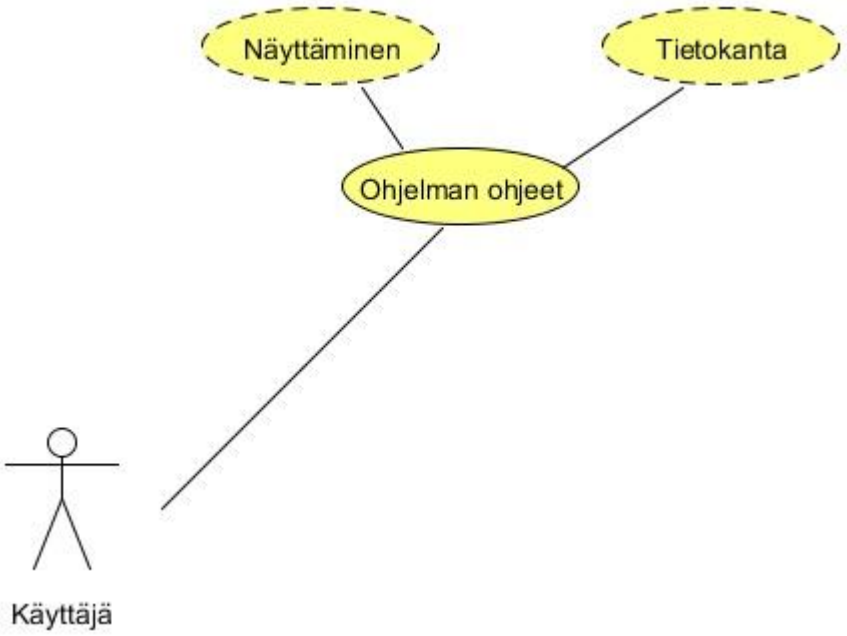
Tekstieditorin käyttötapauskaavio on esitetty kuvassa 28.



Kuva 28. Tekstieditorin käyttötapauskaavio

Ohjelman ohjeet

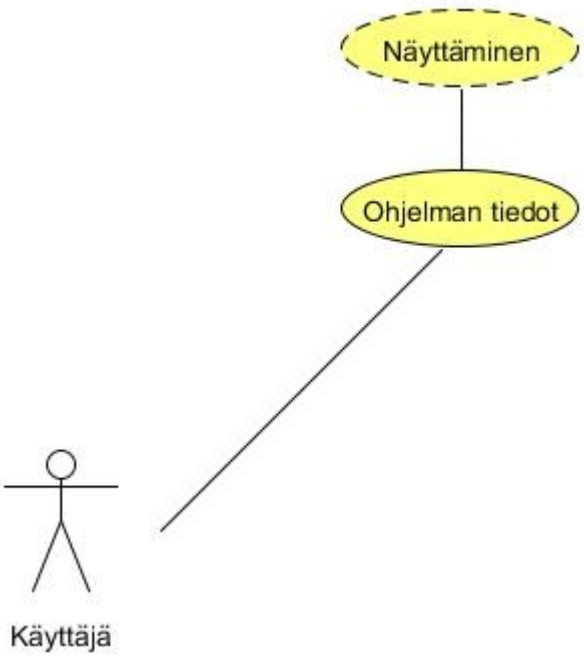
Ohjelman ohjeiden käyttötapauskaavio on esitetty kuvassa 29.



Kuva 29. Ohjelman ohjeiden käyttötapauskaavio

Ohjelman tiedot

Ohjelman tietojen käyttötapauskaavio on esitetty kuvassa 30.



Kuva 30. Ohjelman tietojen käyttötapauskaavio