Editorish

Dokumentaatio

Helsingin yliopisto Avoin yliopisto
AvoinTKTOHkv2014verkko 11.5.2014
Ohjelmoinnin harjoitustyö
Raul Becker (raul.becker@helsinki.fi)
Opiskelijanumero 014355291

Ohjaaja: Kai Korpimies

Tässä dokumentissa kuvataan ohjelmoinnin harjoitustyön ohjelmaa. Harjoitustyön ohjelma on eräänlainen tekstitiedoston lukuohjelma jossa voi hakea sanoja tiedostoista. Ohjelma skannaa valitun hakemiston ja sen alaiset hakemistot läpi ja laskee niistä sanat taustalla siten, että kaikki käyttöliittymän toiminnot ovat käytettävissä samanaikaisesti. Hakutuloksen tiedoston avaaminen toimii myös samanaikaisesti.

Sisällysluettelo

1 Ohjelman käyttöohjeet

Tässä luvussa kerrotaan miten ohjelmaa voi käyttää ja kerrotaan asioista joita ohjelmankäytössä on otettava huomioon.

1.1 Yleiskuvaus

Ohjelma on hakemistopolkuja rekursiivisesti läpikäyvä ja hakemiston tekstitiedostojen sanoja laskeva lukija. Ohjelmassa on hakutoiminto, jolla voidaan hakea yhtä tai useampaa sanaa kerralla. Haku on painotettu siten, että ensimmäinen sana saa painoarvon 1 ja sen jälkeiset kerrotaan 0,8 potenssiin monesko hakusana (sanojen lukumäärä * 0,8^monesko?). Tällöin samoja sanoja eri järjestyksessä kirjoitettuna saadaan eri hakutulokset. Graafinen käyttöliittymä tarjoaa käyttäjälle mahdollisuuden valita mistä hakemistosta tekstitiedostoja käydään lävitse. Käyttöliittymästä voidaan hakea tekstikenttään kirjoittaen sanoja tai oikeastaan merkkijonoja. Hakutoiminto näyttää hakutulokset käyttöliittymän perusvalikoiden alapuolella olevassa listauksesta. Ohjelma avaa tekstitiedoston tarkastelua varten hakutuloslistauksen alapuolella olevaan tekstikenttään, kun hakutuloslistasta klikataan jotakin listattua tiedostoa. Tekstitiedoston avauksen yhteydessä haetut sanat väritetään sattumanvaraisesti valitun värin mukaan per sana.
Kaikki toiminnot ovat sellaisia, joita ajetaan samanaikaisesti. Ohjelmassa pystyy hakemaan ja avaamaan hakutuloksista tiedostoja siihen mennessä skannatuista tiedostoista. Tiedostojen skannauksen edistymisestä näytetään tilannetietoja ohjelman alareunan palkissa.

1.2 Vaatimukset

Ohjelma on kehitetty Java-ohjelmointikielellä, minkä takia sen käyttöön vaaditaan tietokone ja käyttöjärjestelmä, johon saa Java-ohjelmien ajoympäristön (Java Runtime Environment). Ohjelma on kehitetty Javan versiolla 1.7, minkä vuoksi suositellaan, että ohjelmaa ajetaan tällä versiolla. Ohjelma toimii käyttöjärjestelmästä riippumatta ainakin Windowsissa ja Unixin kaltaisissa käyttöjärjestelmissä kuten Mac OS X ja Linux. Käyttöjärjestelmän on tarjottava graafinen käyttöliittymä.

Ohjelma käyttää paljon muistia, jonka takia tämä asettaa joitain rajoitteita mitä ohjelmalla kannattaa tehdä (esimerkiksi 973 tiedostoa, joiden yhteenlaskettu tilavuus on 800Mt vie muistia 4,5Gt).

1.3 Käyttöönotto

Varmista, että tietokoneessa, jossa ohjelmaa aiot suorittaa, on asennettuna Javan ajoympäristö. Ajoympäristö on asennettava, mikäli sitä ei ole asennettu [JRE-ASENNUS].
Sijoita ohjelmatiedosto sellaiseen hakemistoon, jossa on tiedostoja joita haluat tarkastella. Käynnistyksen yhteydessä ohjelma aloittaa skannauksen siitä hakemistosta, jossa se sijaitsee.

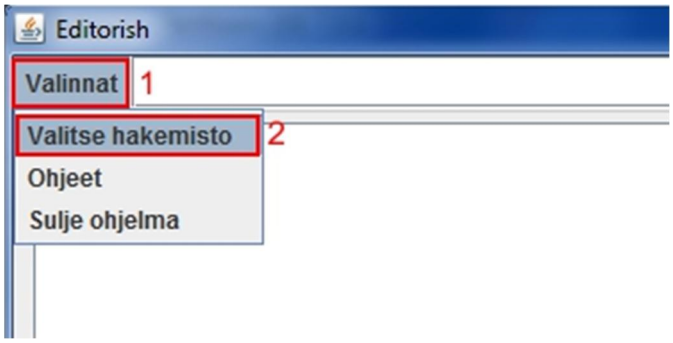
Käynnistääksesi ohjelman voit useimmissa käyttöjärjestelmissä tuplaklikata ohjelmatiedostoa Editorish.jar. Unixin kaltaisissa käyttöjärjestelmissä on varmistettava, että suoritusoikeudet ovat kunnossa. Vaihtoehtoisesti Windowsissa ohjelman voi käynnistää komentokehotteesta seuraan tapaisella komennolla:
X:\Polku\Kansioon\Java\bin\java –jar Y:\Polku\Ohjelman\Sijoituskansioon\Editorish.jar
Unixin kaltaisissa käyttöjärjestelmissä terminaalissa vastaavasti:

/Polku/Java/Kansioon/Java/Bin/java –jar /Polku/Ohjelman/Sijoituskansioon/Editorish.jar

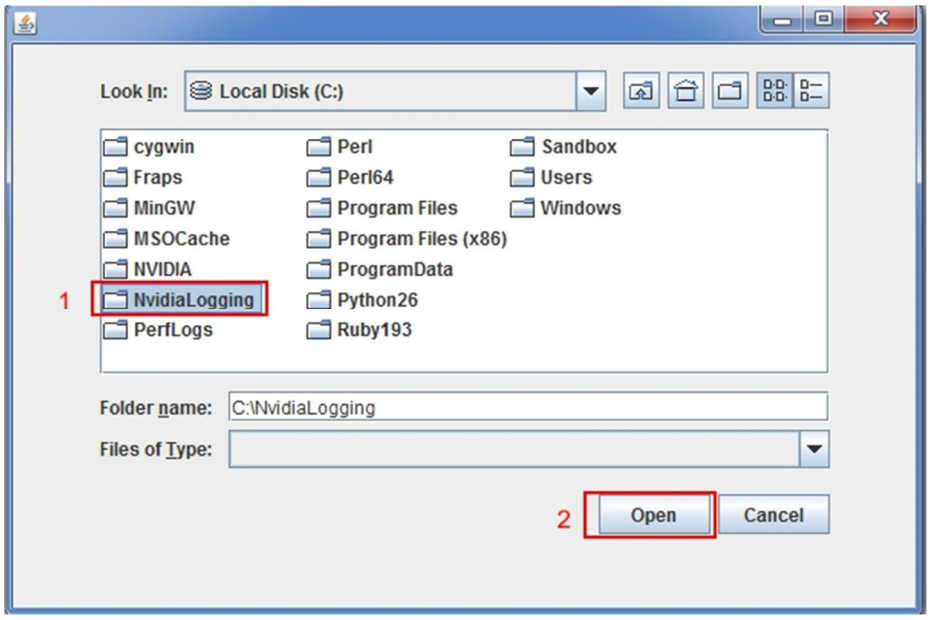
1.4 Käyttö

Ohjelman käynnistyttyä käyttäjä saa näkyviin ohjelman pääikkunan josta kaikki ohjelman toiminnot on tehtävissä. Seuraavassa käydään läpi ohjelman tärkeimmät toiminnot.

1.4.1 Hakemiston valinta

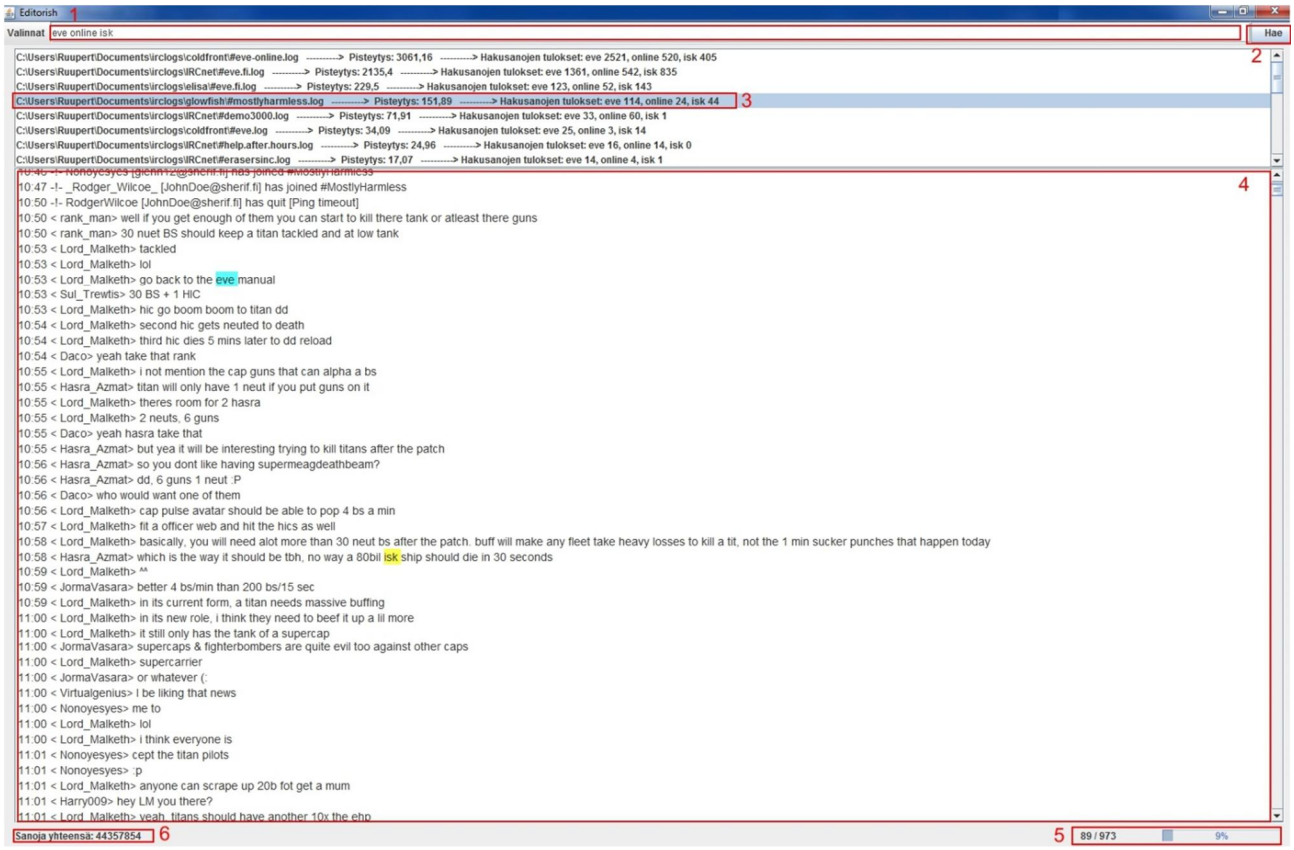


1) Avaa Valinnat valikko klikkaamalla sitä.
2) Klikkaa ”Valitse hakemisto”-toimintoa



1) Valitse hakemisto jonka haluat ohjelman käyvän läpi.
2) Klikkaa ”Open”-painiketta.

1.4.2 Sanojen haku ja hakuosumien avaaminen



1) Kirjoita hakusanasi tähän kenttään välilyönnein. Voit painaa enter-näppäintä toteuttaaksesi haun.
2) Hae –nappi toteuttaa myös haun.
3) Valitse avattava tiedosto hakutuloslistasta
4) Tähän tekstialueeseen avautuu valitsemasi tiedosto.
5) Näyttää taustalla ajettavan skannausprosessin edistymisen tilan esittämällä skannatut tiedostot ja niiden kokonaislukumäärän. Sama asia näytetään myös prosentein edistysmittarilla.
6) Näyttää kuinka monta sanaa on siihen mennessä käsitelty taustaprosessissa.

2 Kuvaus ohjelmiston rakenteesta ja toiminnasta

2.1 Tekninen yleiskuvaus

Kun ohjelma käynnistetään, se skannaa hakemiston ja kaikki sen alihakemistot, josta se käynnistettiin. Skannaus tapahtuu erillisessä säikeessä, joka mahdollistaa ohjelman muiden toimintojen samanaikaisen käytön. Ohjelmassa tiedostojen avaus on myös säikeistetty. Säikeistys on tehty SwingWorkeria hyödyntäen, sillä se tukee käyttöliittymä -elementtien sisältöjen päivittämistä. SwingWorkereille annetaan parametreinä ne käyttöliittymän elementit, joiden sisältöjä se tulee päivittämään.

Skannaksesta vastaava prosessi luo uuden FileSummary –ilmentymän jokaista skannattua tiedostoa kohden. Jokainen FileSummary sisältää tiedoston nimen ja HashMap-tyyppisen kirjaston, jossa avaimena on sana ja arvona FileWordSummary -tietotyyppi. Tiedoston rivejä skannaava metodi lukee 2000 riviä kerrallaan ja lisää ne tietorakenteeseen, minkä jälkeen metodin rivejä läpikäyvä silmukka odottaa 50 millisekuntia.

Tiedoston avaamisesta vastaa SwingWorkeria laajentava luokka FileOpener, joka saa konstruktorin parametreissä käyttöliittymän JTextPane –tekstialueen, hakutuloskirjaston, sekä sen indeksinumeron, mistä kohtaa avattava tiedosto löytyy sekä annetut hakusanat. Luokka luo niin monta sattumanvaraisesti valittua väriä kuin mitä hakusanoja on. Näin jokainen sana voidaan värittää erikseen.

2.2 Ohjelman rakenne

Ohjelman käyttöliittymäluokka EditoriGUI toimii ohjelman käynnistysluokkana ja käyttää Editorish- ja FileOpener luokkia. Editorish-luokkaa tarjoaa tiedostojen skannaus ja sanojen hakupalvelut. Editorish-luokka puolestaan käyttää tietotyyppiä FileSummary, mikä puolestaan sisältää kirjaston joka koostuu FileWordSummaryistä. FileOpener hoitaa tiedoston avaamisen ja sen sisällön tyylittelyn, sekä sen näyttämisen käyttöliittymän tekstialueella.

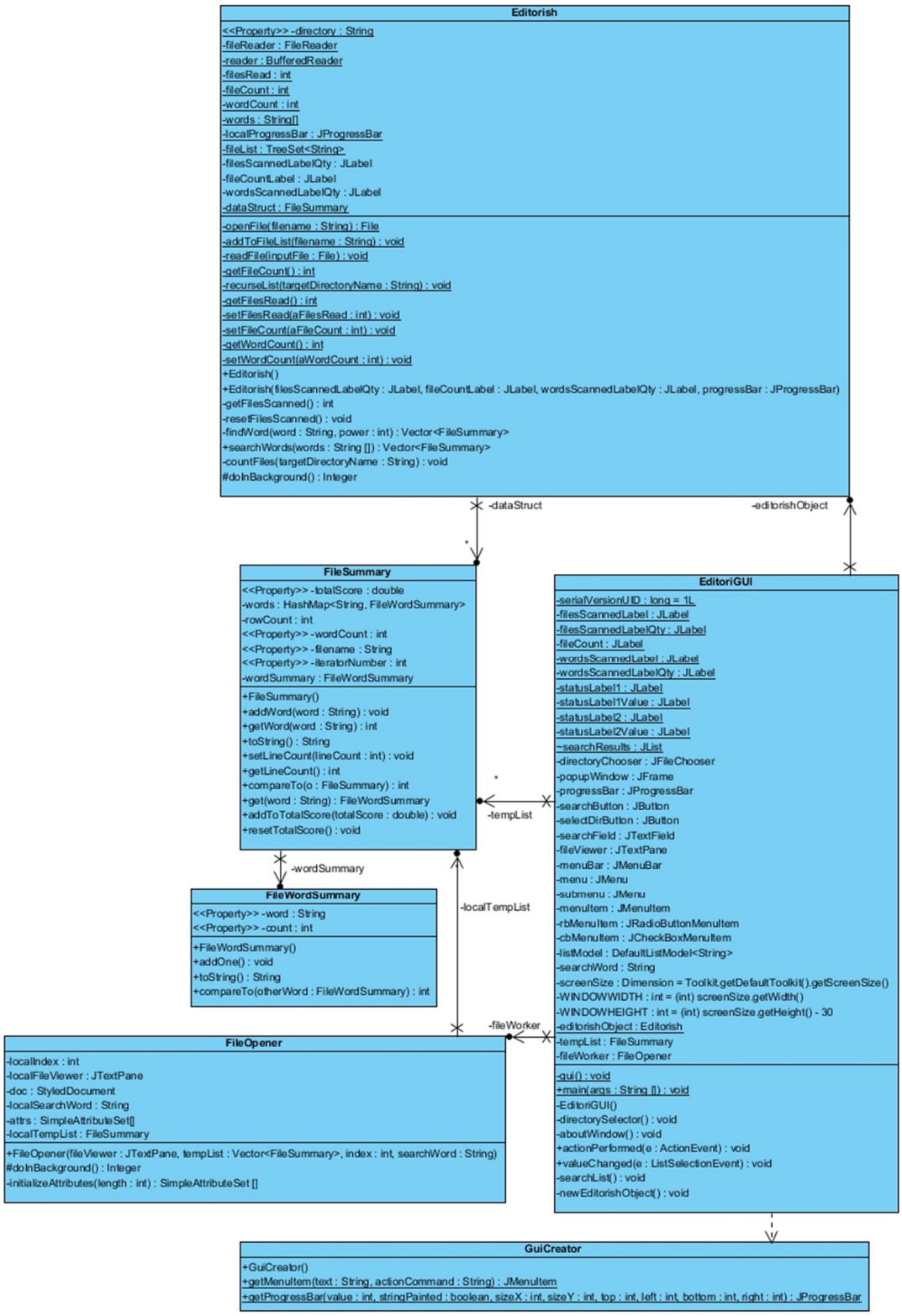
Editorish-luokan hakutoiminto ottaa vastaan merkkijonon ja jakaa sen välilyönneistä merkkijonotaulukoksi. Jokainen taulukon arvo haetaan erikseen tietorakenteesta. Haku tapahtuu painotetusti siten, että ensimmäinen sana saa painoarvon 1 eli sen sanojen lukumäärä on sama kuin mitä niitä tiedostossa esiintyy. Seuraavat sanat puolestaan saavat painoarvon 0,8. Siten seuraavasta sanojen lukumäärästä tulee lukumäärä * 0,8^1 ja sitä seuraava lukumäärä * 0,8^2 ja niin edelleen. Nämä luvut summataan ja palautetaan suuruusjärjestyksessä takaisin käyttöliittymälle. Käyttöliittymä hoitaa hakutuloksien näyttämisen käyttäjälle.

Käyttöliittymä luo uuden ilmentymän FileOpener-luokasta, kun käyttäjä valitsee hakutuloslistasta jonkin hakuosuman. FileOpener käy läpi silmukassa rivejä tiedostosta 200 riviä kerrallaan ja odottaa sen jälkeen 60 millisekuntia. Jokainen rivi pilkotaan merkkijonotaulukoksi ja jokainen taulukon sana verrataan hakusanataulukon sanoihin. Tyylitaulukosta valitaan hakusanataulukon indeksinumero silloin, kun verrattava sana löytyy hakutulostaulukosta. Vertailujen jälkeen sana tulostetaan tekstiosioon.

Sekä Editorish- että FileOpener-luokka käsittelee tiedostoja pienissä erissä ja sitten odottaa 50 tai 60 millisekuntia. Tämä hidastaa jonkin verran näiden toimintaa, mutta varmistaa sen ettei käyttöliittymä jää jumiin. Toisin sanoen käyttöliittymä pysyy käytettävänä vaikka samaan aikaan tehdään muita toimintoja.

2.2.1 Sekvenssikaavio

2.2.2 Luokkakaavio



2.2.3 Luokkakuvaukset

2.2.3.1 EditoriGUI.java

Ohjelma käynnistetään EditoriGUI-luokan main-metodilla. Tässä luokassa luodaan käyttöliittymä ja käsitellään toiminnot, joita käyttäjä tekee.
Ohjelman main–metodissa käynnistetään ainoastaan luokan jaettu metodi gui(). gui-metodi luo EditoriGUI–luokasta ilmentymän. Konstruktorissa määritellään käyttöliittymän elementit sekä alustetaan ja käynnistetään skannauksen toteuttava Editorish SwingWorker.

2.2.3.2 Editorish.java

Editorish laajentaa SwingWorker–luokkaa ja ottaa konstruktorissa parametreiksi käyttöliittymän JLabel–oliot joissa esitetään, kuinka monta tiedostoa on skannattu, tiedostojen lukumäärä ja laskettujen sanojen lukumäärän, sekä JProgressBarin, joka näyttää edistymisen prosenteissa.
Tämä luokka päivittää sille konstruktorissa annettujen käyttöliittymän elementtien sisältöjä tiedostojen skannauksen edistyessä.
Luokka tarjoaa aksessorin ja mutaattorin skannaushakemiston määrittelyyn, sekä sanojen hakumetodin, joka palauttaa FileSummary–olioita sisältävän Vectorin.

2.2.3.3 FileSummary.java

Luokka tarjoaa palvelut tiedoston perustietojen säilömiseen ja kirjaston joka säilöö avaimena sanaa ja sen arvona FileWordSummary olion. Näiden lisäksi tarjoaa hakutuloksien yhteispistemäärän kirjanpidon. Tätä luokkaa käytetään ohjelman tietorakenteessa ja siitä tehtyjen hakujen palautuksissa.

2.2.3.4 FileWordSummary.java

Luokka sisältää sanan ja sen lukumäärän. Tarjoaa aksessorit ja mutaattorit näiden arvojen hakemiseen ja asettamiseen.

2.2.3.5 FileOpener.java

Luokka toimii itsenäisenä SwingWorker–luokkaa jatkavana oliona, joka saa parametrikseen tiedoston ja JEditorPane–käyttöliittymäolion. Instanssi lukee tiedostosta 200 riviä kerrallaan ja lisää ne parametrinä saatuun JEditorPane–olioon.

2.2.3.6 GuiCreator.java

Luokka tarjoaa EditoriGUI-luokalle metodien palautuksina käyttöliittymäelementtejä. Sen tarkoitus on parantaa EditoriGUI-luokan konstruktorin koodin luettavuutta.

3 Testaus

Ohjelman toiminnallinen eli black-box –testaus on tehty Windowsissa, unixin kaltaisissa sekä Mac OS X käyttöjärjestelmissä. Ohjelman kaikki toiminnot on käyty läpi ja varmistettu haun toimivuus ja luotettavuus itse laskemalla README.txt tiedostossa esiintyvät sanat ja vertailemalla niitä haun tuloksiin.
Yksikkötesteissä (white box) on käytetty JUnit4 testejä. Niitä on tehty kaikille paitsi käyttöliittymäluokalle EditoriGUI.java.

3.1 EditorishTest.java

Testataan julkiset metodit setDirectory, getDirectory, doInBackground ja searchWords. SearchWords-metodissa haetaan hakusanoin “java jar” ja niiden haun ensimmäinen hakutulos on dist–hakemistossa oleva README.txt. Tämä tiedosto sisältää 229 sanaa, joten tätä arvoa vasten verrataan lopputulosta. Metodi doInBackground palauttaa arvon yksi, kun se on suoritettu loppuun. Testimetodissa käynnistetään tämä metodi ja odotetaan while-silmukassa sen päättymistä. Testi voi siis jäädä jumiin tähän while-silmukkaan silloin, kun tiedostojen skannaus jää ikuiseen silmukkaan tiedostorakenteessa.

3.2 FileSummaryTest.java

Testataan julkiset metodit addWord, toString, getWord, get, compareTo, setWordCount, setLineCount, setFilename, getWordCount, getLineCount, getFilename, getTotalScore, addToTotalScore, resetTotalScore, getIteratorNumber ja setIteratorNumber –metodit. Kaikki metodit ovat sellaisia joita voidaan aksessorein ja mutaattorein testata.

3.3 FileWordSummaryTest.java

Testaa setWord, addOne, setCount, getCount, toString ja compareTo –metodit. Metodille setWord ei ole omaa aksessoria, jonka takia testatessa asetetaan sana FileWordSummaryn ilmentymään ja sen jälkeen tehdään ilmentymästä reflektio. Tämä tarkoittaa sitä, että saadaan kopio luokan kaikista yksityisistä muuttujista ja niiden arvoista. Luokan yksityistä word-muuttujan arvoa verrataan setWord-metodin parametrinä annettuun arvoon.
Luokan muille metodeille löytyy mutaattori ja aksessori, jonka takia näiden testaaminen ei vaadi reflektiota.

3.4 FileOpenerTest.java

Testaa doInBackground-metodin siten, että luo uuden Editorish-ilmentymän joka annetaan skannata hakemistorakenne läpi. Hakemistorakenteesta löytyy aiemmin mainittu README.txt, jossa tiedetään esiintyvän sana ”java”.

3.5 GuiCreatorTest.java

Testaa metodit getMenuItem ja getProgressBar. Metodit palauttavat alustetut käyttöliittymäelementit. Testeissä luodaan ilmentymät näistä ja tarkistetaan, että korkeus, nimi ja tapahtumakomento vastaavat sitä, mitä metodin parametreissä annettiin.

4. liitteet

JRE-ASENNUS	http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html