Nippu-ohjelmistotuotantoprojekti
Testausdokumentti

Michael Forsström, Mikael Jokela, Ville V Nurmi, Ville A Nuutinen, Chen Zhao

Helsinki 31.8.2003

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Versiohistoria

Versio	päiväys	koostaja	kommentti
0.1	16.7.2003	Ville Nurmi	alustava runko
0.1.1	21.7.2003	Ville Nurmi	
0.2	22.7.2003	Mikael Jokela	mukana 1. iteraation testejä
0.3	19.8.2003	Mikael Jokela	mukana 2. iteraation testit
0.9	26.8.2003	Mikael Jokela	
0.99	30.8.2003	Mikael Jokela	toinen iteraatio meni läpi
1.0	31.8.2003	Mikael Jokela	lopullinen versio

Sisältö

1 Johdanto 1

2 Testauksen rajoitukset ja hallinnolliset seikat 2

2.1 Laitteistot ja ohjelmistot 2

2.2 Turvallisuus ja eristäminen 2

2.3 Apuvälineet 2

2.4 Vastuuhenkilöt 3

2.5 Aikataulu 3

3 Tehtäväjärjestys ja menetelmät 4

3.1 Tehtäväjärjestys 4

3.2 Testattavat osat 4

3.3 Testien nimeäminen 5

3.4 Menetelmät, tekniikat ja vaadittavat tulokset 5

3.5 Testaamatta jätettävät ominaisuudet 6

4 Ensimmäisen iteraation testit 7

4.1 Käyttöliittymä 7

4.2 Vastaanotto 8

4.3 Analysoija 8

4.4 Tulostushallinta 8

4.5 Tietoliikenne 9

4.6 Tietorakenteet 9

4.7 Kokonaisjärjestelmä 14

4.8 Suorituskyky 14

5 Toisen iteraation testit 15

5.1 Käyttöliittymä 15

5.2 Vastaanotto 21

5.3 Analysoija 23

5.4 Tulostushallinta 25

5.5 Tietoliikenne 32

5.6 Tietorakenteet 32
5.7 Kokonaisjärjestelmä 39
5.8 Suorituskyky 40
5.9 Järjestelmätestaus 41

Liitteet

1 Sanasto I

1 Johdanto

Tämä dokumentti sisältää **suunnitelman Nippu-tulostuspalvelinohjelmiston testauksesta sekä suoritettujen testien tulokset**. Testauksessa ohjelmistoa ajetaan tarkoituksena löytää suuri osa siinä olevista virheistä.

Testejä laaditaan riittävän kattavasti, jotta ohjelmiston **suurimmat virheet löytyvät** ja testien mentyä läpi voidaan todeta testauksen kohteena olleiden ominaisuuksien olevan riittävällä tavalla toteutettuja. Testien suoritus ja tulokset on kirjattu jokaisen testin määrittelyn jälkeen.

Tämä dokumentti toimii sekä testaussuunnitelmana että -dokumenttina, koska sisältöä päivitetään toteutusiteraatioiden ja testauksen edetessä. Dokumentin rakenne perustuu osittain Tampereen teknillisen korkeakoulun HYTT-laatukäsikirjan[1] testausdokumentointia koskevaan osuuteen.

Nipun keskeisin tarkoitus on **saavuttaa tehokkuutta suorittamalla samankaltaisia palvelupyyntöjä yhdessä**. Käytännössä ohjelmisto pyrkii esimerkiksi ryhmittelemään lähekkäin sijaitsevien tulostajien työt siten, että ne tulostuvat yhdessä nipussa jollain läheisellä, vapaalla tulostimella. Nippu on osa Helsingin yliopiston tietojenkäsittelytieteen laitoksen ohjelmistotuotantoprojekti-kurssia. Nippua on tarkoitus käyttää Helsingin yliopistossa koostavan ajoitusmenettelyn ja sen kannattavuuden tutkimisessa.

Liitteen 1 sanastossa selitetään tässä dokumentissa esiintyviä käsitteitä.

1 HYTT-laatukäsikirja on saatavilla verkko-osoitteessa http://www.cs.tut.fi/ohj/laatu/.

2 Testauksen rajoitukset ja hallinnolliset seikat

Tässä luvussa kuvataan Nipun testaukseen liittyvät rajoitukset sekä hallinnolliset seikat.

2.1 Laitteistot ja ohjelmistot

Testaus suoritetaan Helsingin yliopiston tietojenkäsittelytieteen laitoksen tietokoneissa Linux-käyttöjärjestelmässä. Tietokoneiden tulee olla kytketty samaan tietoliikenneverkkoon, jotta testauksessa tarvittava TCP-yhteys voidaan muodostaa. Java-ohjelmat suoritetaan Java 2 standard edition 1.4.1:n mukana toimitettavalla Java-virtuaalikoneella.

2.2 Turvallisuus ja eristäminen

Testauksessa on oleellista, että testauksen ulkopuoliset tahot eivät pääse häiritsemään testausta. Tätä ei kuitenkaan varmisteta, sillä se vaatisi kohtuuttoman suurta vaivannäköä (testaukseen käytettävien tietokoneiden irrottaminen Tietojenkäsittelytieteen laitoksen tietoliikenneverkosta). Integrointitestauksessa on sen takia otettava huomioon, että testausryhmän ulkopuoliset henkilöt voivat halutessaan ottaa yhteyden Nippu-järjestelmän komponentteihin ja lähettää niille käskyjä, jotka häiritsevät järjestelmän toimintaa tai estävät sen täysin. Lisäksi suorituskykytestauksessa on otettava huomioon, että testausryhmän ulkopuoliset käyttäjät voivat suorittaa prosesseja testikoneella, mikä vaikuttaa Nippu-järjestelmän suorituskykyyn.

2.3 Apuvälineet

Suorituskyvyn testauksessa voidaan käyttää Linux-käyttöjärjestelmän ajanmittauskomentoa **time** tai ulkopuolista ajanmittausmenetelmää, kuten sekuntikelloa. Muussa testauksessa käytetään etupäässä testauksen kannalta mielekkään tekstin tulostamista tekstimuotoiseen terminaalikonsoliin sekä käyttöliittymän tarjoamia visualisointimenetelmiä.

2.4 Vastuuhenkilöt

Kunkin Nipun komponentin vastuuhenkilöt vastaavat myös kyseisen komponentin testauksesta. Vastuuhenkilöt on lueteltu seuraavassa siten, että ensisijainen vastuuhenkilö on mainittu ensin:

käyttöliittymä	Chen Zhao, Ville A Nuutinen
vastaanotto	Michael Forsström
analysoija	Ville V Nurmi, Ville A Nuutinen
tulostushallinta	Michael Forsström
tietoliikenne	Ville V Nurmi
tietorakenteet	Ville A Nuutinen

Integrointitestauksesta vastaa koko Nippu-projektiryhmä yhdessä. Integrointitestaukseen liittyy Nipun sisäisten liittymien testaus.

2.5 Aikataulu

Testaus suoritetaan Nippu-ohjelmistotuotantoprojektin projektisuunnitelmassa kuvattujen iteraatioiden mukaan.

3 Tehtäväjärjestys ja menetelmät

Tässä luvussa kuvataan Nipun testaukseen liittyvät käytännöt ja periaatteet sekä yleisellä tasolla testaukselta odotettavat tulokset.

3.1 Tehtäväjärjestys

Nipun eri komponentteja testattiin samanaikaisesti. Integrointitestaus suoritettiin vasta, kun komponenttien todettiin toimivan riittävällä tavalla. Testaus suoritettiin kunkin Nipun projektisuunnitelman mukaisen iteraation päätteeksi. Jokaisessa iteraatiossa testattiin lähinnä samassa iteraatiossa toteutettuja ominaisuuksia. Testien läpi meneminen tarkoittaa iteraation päättymistä onnistuneesti.

Jokaisesta testistä on dokumentoitu seuraavat seikat:

- Mikä on testin nimi?
- Minkä asian toimivuutta testi mittaa?
- Miten testi suoritetaan?
- Montako kertaa testi suoritetaan?
- Mitä testitilanteessa pitäisi tapahtua?

CVS-versionhallintajärjestelmä liittää automaattisesti jokaiseen komponenttiin juoksevan versionumeroinnin. Vanhoja versioita voidaan tarvittaessa palauttaa sekä versionumeron että päivämäärän perusteella.

3.2 Testatut osat

Testejä on laadittu kullekin Nipun komponentille (käyttöliittymä, vastaanotto, analysoija, tulostushallinta ja tietoliikenne). Testejä on ainakin yksi komponentin jokaista keskeistä toimintokokonaisuutta kohti. Testejä on laadittu myös kokonaisjärjestelmälle sekä suorituskyvyn arviointiin.

3.3 Testien nimeäminen

Testit on nimetty tyyliin xx-nn, jossa xx on testattavan osan nimi ja nn on kyseiseen osaan liittyvä juokseva numero. Xx voi olla esimerkiksi jokin seuraavista:

kä	käyttöliittymä
va	vastaanotto
an	analysoija
tu	tulostushallinta
ti	tietoliikenne
ko	kokonaisjärjestelmä (tätä käytetään integrointitestauksessa)
mu	muu

Esimerkkejä testien nimistä ovat va-1 ja ko-27.

3.4 Menetelmät, tekniikat ja vaadittavat tulokset

Testauksessa löytyneet virheet luokitellaan suuriin ja pieniin virheisiin:

Pieni virhe on sellainen virhe, joka ei estä järjestelmän perustoimintaa. Pieniä virheitä ovat esimerkiksi tulosteiden kirjoitusvirheet, toimimaton grafiikka käyttöliittymän epäoleellisissa osissa, lievät suorituskykyongelmat sekä hyvin harvinaiset toimintahäiriöt.

Suuri virhe on sellainen virhe, joka jossain yleisessä käyttötapauksessa suuresti hankaloittaa järjestelmän käyttöä tai estää sen. Suuria virheitä ovat esimerkiksi järjestelmän kaatuminen kesken suorituksen, tulostustyön katoaminen tai sen tietojen muuttuminen työn ollessa Nippu-järjestelmässä. Suuriksi virheiksi katsotaan kaikki sellainen toiminnallisuus, joka on selvästi ristiriidassa vaatimusdokumentin sisällön kanssa.

Tietyn osan testauksen vastuuhenkilö huolehtii siitä, että kyseiselle osalle on laadittu **riittävän kattava testiaineisto**. Lisäksi hän varmistaa, että suunnitellut **testit on suoritettu ja dokumentoitu**. Käytännössä hän suorittaa ja dokumentoi ne itse.

Jokaisen iteraation testaus suoritetaan erikseen. Jokaista dokumentoitua testiä kohti on maininta sen suorittamisesta. Jos testin suorittaminen johti suuren virheen havaitsemiseen,

virhe on kirjattu. Jos **virheen syy** on tiedossa tai siitä on vahva epäilys, sekin on kirjattu tähän dokumenttiin.

Testi on mennyt läpi, jos sen suorittaminen ei paljastanut yhtään suurta virhettä. Jos jokin iteraatioon kuuluva testi ei mennyt läpi, sen paljastama **virhe on korjattu ja testi on suoritettu uudestaan**. Testin uusi suoritus on kirjattu tavalliseen tapaan testin edellisen suorituksen tilalle. Jos testi meni läpi, mutta se paljasti pieniä virheitä, virheitä ei ole välttämättä erikseen mainittu. Pieniä virheitä on voitu korjata testauksen tai myös seuraavan iteraation aikana.

3.5 Testaamatta jätetyt ominaisuudet

Terttu-järjestelmää ei testattu, sillä Tertun kehitysryhmä on sen jo testannut. Samasta syystä myöskään selvästi Nippuun kuulumattomia osia, kuten tietoliikenteen laitteistotason toimivuutta, ei ole testattu.

4 Ensimmäisen iteraation testit

Ensimmäisessä iteraatiossa toteutettiin vain joitakin Nippu-järjestelmän **välttämättömimpiä ominaisuuksia** eikä vielä edes kaikkea vaatimusdokumentin mukaista toiminnallisuutta.

Ensimmäinen iteraatio on suoritettu onnistuneesti, kun seuraava toiminta toteutuu: Käyttöliittymä ja analysoija saavat järjestelmän asetukset tulostushallinnan kautta. Käyttöliittymästä voidaan lähettää yksittäinen tulostustyö vastaanoton kautta ajoittajalle, josta työ siirtyy mahdollisesti täysin ilman järkevää niputusta tulostushallintaan. Lopuksi tulostushallinta ilmoittaa käyttöliittymälle, että työ on otettu vastaan ja käyttöliittymän näyttöön päivittyy tieto asiasta. Koska järjestelmältä ei ensimmäisessä iteraatiossa vaadita virheidenhallintaa, on testejä varsin niukasti.

4.1 Käyttöliittymä

Testi kä-1

Tehtävä: varmistutaan peruskäyttäjän käyttöliittymän toiminnasta.

Kuvaus: Luodaan peruskäyttäjän käyttöliittymä. Liittymä sisältää peruskäyttäjän kentän, jossa käyttäjätunnus on vaihtamatta. Liittymä sisältää myös visualisointinäkymän tulostuslokeineen.

Oletus: Käyttäjätunnus on vaihtamatta. Kun käyttäjä painaa tulosta-painiketta, tulostuslokiin ilmaantuu yksi rivi, joka sisältää käyttäjätunnuksen, työn nimen, sivumäärän ja pakotetun tulostimen nimen.

Tulos: testi meni odotetusti läpi 20.7.2003.

Testi kä-2

Tehtävä: varmistutaan pääkäyttäjän käyttöliittymän toiminnasta.

Kuvaus: Luodaan pääkäyttäjän käyttöliittymä käyttämällä p-komentorivi-parametria. Liittymä sisältää peruskäyttäjän kentän, jossa käyttäjätunnus on pudotusvalikko. Näkymä sisältää myös visualisointiliittymän tulostuslokeineen sekä pääkäyttäjän satunnaistöiden näkymän ja ajoittajan pysäytyspainikkeen.

Oletus: Käyttäjätunnus-valikko sisältää kaikki käyttäjätunnukset. Kun käyttäjä painaa tulosta-painiketta, tulostuslokiin ilmaantuu yksi rivi, joka sisältää käyttäjätunnuksen, työn nimen, sivumäärän ja pakotetun tulostimen nimen. Kun käyttäjä painaa "käynnistä satunnaistyöt" -painiketta, painikkeen nimeksi tulee "lopeta satunnaistyöt". Kun käyttäjä painaa "pysäytä ajoittaja" -painiketta, painikkeen nimeksi tulee "käynnistä ajoittaja".

Tulos: testi meni odotetusti läpi 20.7.2003.

4.2 Vastaanotto

Vastaanottoa ei testattu ensimmäisessä iteraatiossa, koska vastaanottoon komponenttina ei vielä liity mitään testattavaa toiminnallisuutta.

4.3 Analysoija

Ensimmäisen iteraation analysoija tarjosi palvelujaan vain Tertussa määritellyn, metodikutsuihin perustuvan rajapinnan kautta. Analysoijan viisi palvelua ovat *getCoordinates*, *cost*, *combinedCost*, *fusion* ja *toString*. Analysoijalla ei ole tilaa, joten palvelujen testaus ei riipu niiden suoritusjärjestyksestä. Lisäksi analysoijan toteutukset palveluille eivät sisällä minkäänlaista toimintalogiikkaa. Siksi analysoijaa ei ollut mielekästä testata ensimmäisessä iteraatiossa.

4.4 Tulostushallinta

Tulostushallintaa ei testattu ensimmäisessä iteraatiossa, koska tulostushallintaan komponenttina ei vielä liity mitään testattavaa toiminnallisuutta.

4.5 Tietoliikenne

Testi ti-1

Tehtävä: varmistutaan sekä asiakas- että palvelinpuolen tietoliikenne-ominaisuuksien keskeisestä toimivuudesta.

Kuvaus: Kirjoitetaan testiohjelma. Testiohjelma luo palvelimen ja asiakkaan, joilla on yksinkertainen viestintäprotokolla. Asiakas ottaa yhteyden palvelimeen. Asiakas lähettää palvelimelle viestin. Palvelin lähettää asiakkaalle vastausviestin.

Oletus: yhteys muodostuu, viestit kulkevat yhteyden yli ja kunkin viestin vastaanottaja osaa tulkita viestin oikein.

Tulos: testi meni odotetusti läpi 10.7.2003.

4.6 Tietorakenteet

Tässä luvussa käsitellään PrintJob- ja Bundle-luokkien testit. Testattavat versio olivat PrintJob-versio 1.6 ja Bundle-versio 1.10. Luokat sisältävät testipääohjelmametodit, joilla testit suoritettiin. Testeistä saa tarkempia tietoja tutustumalla testipääohjelmiin ja ajamalla luokat.

Testi PJ-1

Tehtävä: varmistetaan konstruktorin ja *toString*-metodin oikea toiminta.

Kuvaus: luodaan oliot molemmilla konstruktoreilla ja tulostetaan ne käyttäen *toString*-metodia.

Oletus: metodi tulostaa olioiden kenttien arvot oikein.

Tulos: testi meni odotetusti läpi 21.7.2003.

Testi PJ-2

Tehtävä: varmistetaan *equals*-metodin toiminta.

Kuvaus: Luodaan kaksi samanlaista PrintJob-oliota ja testataan metodia niihin. Sitten muutetaan toisessa oliossa joidenkin kenttien arvoja, jonka jälkeen testataan *equals*-metodia. Muutetut arvot palautetaan alkuperäisiksi, jonka jälkeen testataan jälleen *equals*-metodia.

Oletus: Metodi palauttaa aluksi true. Muutosten aikana metodi palauttaa false ja lopuksi jälleen true.

Tulos: testi meni odotetusti läpi 21.7.2003.

Testi PJ-3

Tehtävä: varmistetaan *serialize*- ja *deserialize*-metodien oikea toiminta.

Kuvaus: luodaan PrintJob-oliosta merkkijono metodilla *serialize* ja tulostetaan saatu merkkijono. Lopuksi merkkijono annetaan parametrina *deserialize*-metodille, jonka antama PrintJob-olio tulostetaan. Testaus suoritetaan sekä PrintJob-oliolle, jonka alkuarvot on asetettu, sekä ilman alkuarvojen asetusta.

Oletus: *Serialize*-metodi tulostaa oikein kaikkien kenttien arvot %-merkillä toisistaan erotettuina. *Deserialize*-metodi osaa luoda merkkijonosta alkuperäisen PrintJob-olion.

Tulos: testi meni odotetusti läpi 21.7.2003.

Testi PJ-4

Tehtävä: varmistetaan *set*-metodien oikea toiminta.

Kuvaus: Testataan *set*-metodit antamalla niille erilaisia arvoja. Metodeille annetaan sekä sopivia että sopimattomia arvoja. Kaikkien metodien palauttamat

arvot tulostetaan.

Oletus: metodit palauttavat oikeat arvot.

Tulos: testi meni odotetusti läpi 21.7.2003.

Testi Bu-1

Tehtävä: testataan peruskonstruktoria.

Kuvaus: luodaan Bundle-olio peruskonstruktorilla ja tulostetaan saatu olio.

Oletus: tiedot tulostuvat oikein.

Tulos: testi meni odotetusti läpi 21.7.2003.

Testi Bu-2

Tehtävä: *setOptimalPrinters*-metodin testaaminen.

Kuvaus: asetetaan tulostimet siten, että metodille annetaan parametrina liian iso taulukko, liian pitkä nimi taulukossa tai oikeanlainen taulukko.

Oletus: metodi palauttaa false kahdessa ensimmäisessä tapauksessa ja kolmannessa true.

Tulos: testi meni odotetusti läpi 21.7.2003.

Testi Bu-3

Tehtävä: *setJobs*-metodin toiminnan testaus.

Kuvaus: Asetetaan *setJobs*-metodilla Vector-olio, jossa on *Object*-olio. Toisena tapauksena asetetaan Vector-olio, jossa on vain PrintJob-olioita.

Oletus: Metodi palauttaa false, kun Vectorissa on *Object*-olio. Toisessa tapauksessa palautetaan true.

Tulos: testi meni odotetusti läpi 21.7.2003.

Testi Bu-4

Tehtävä: *union*-metodin toiminnan varmistaminen.

Kuvaus: Tehdään erilaisia Bundle-olioiden unioneja ja unioneja muun muassa itsensä kanssa sekä tulostetaan unionit. Katsotaan, ovatko tulosteessa kaikki kentät oikein.

Oletus: kaikki kentät tulostuvat oikein.

Tulos: testi meni odotetusti läpi 21.7.2003.

Testi Bu-5

Tehtävä: varmistetaan *serialize*- ja *deserialize*-metodien oikea toiminta.

Kuvaus: Luodaan Bundle-oliosta merkkijono metodilla *serialize* ja tulostetaan saatu merkkijono. Lopuksi merkkijono annetaan parametrina *deserialize*-metodille, jonka antama Bundle-olio tulostetaan. Testaus suoritetaan sekä siten, että Bundle-olion alkuarvot on asetettu, että alkuarvoja asettamatta.

Oletus: metodit toimivat halutulla tavalla.

Tulos: testi meni odotetusti läpi 21.7.2003.

Testi Bu-6

Tehtävä: *breakBundle*-metodin toiminnan varmistaminen.

Kuvaus: Käytetään *breakBundle*-metodia aikaisemmin tehtyyn yhdisteeseen.

Tulostetaan saadun taulukon sisältö.

Oletus: yhdiste-Bundle hajoaa erillisiksi nipuiksi halutulla tavalla.

Tulos: testi meni odotetusti läpi 21.7.2003.

Testi Bu-7

Tehtävä: varmistetaan *addOptimalPrinter*- ja *removeOptimalPrinter*-metodien oikea toiminta.

Kuvaus: Lisätään Bundle-olioon optimaaleja tulostimia *addOptimalPrinter*-metodilla. Lopuksi tulostetaan olio. Oliosta poistetaan tulostimia *remove-OptimalPrinter*-metodilla, jonka jälkeen olio tulostetaan.

Oletus: lisätyt tulostimet näkyvät Bundle-olion tulostuksessa, ja poistetut tulostimet tulevat poistetuiksi.

Tulos: testi meni odotetusti läpi 21.7.2003.

Testi Bu-8

Tehtävä: varmistetaan *createID*-metodin toiminta.

Kuvaus: Annetaan parametrina *createID*-metodille erilaisia kokonaislukupareja. Tulostetaan metodin palauttamat arvot.

Oletus: metodi palauttaa kokonaislukuja.

Tulos: testi meni odotetusti läpi 21.7.2003.

4.7 Kokonaisjärjestelmä

Testi ko-1

Tehtävä: varmistutaan eri komponenttien välisen viestinnän toimivuudesta ja siitä, että suunniteltu arkkitehtuuri on käytännössä mahdollinen.

Kuvaus: Kytketään käyttöliittymä, vastaanotto, ajoittaja, analysoija ja tulostushallinta toisiinsa tietoliikennekomponentin avulla. Lähetetään käyttöliittymästä yksi tulostustyö.

Oletus: Lähetetty työ kulkee komponenttien läpi päätyen tulostushallintaan. Tulostushallinta ilmoittaa käyttöliittymälle työn vastaanottamisesta. Käyttöliittymä näyttää käyttäjälle tekstimuotoisen viestin, joka kertoo työn onnistuneesta kulkemisesta järjestelmän läpi.

Tulos: Testi meni odotetusti läpi 28.7.2003. Testin jälkeen ohjelmiston CVS-version nimeksi nimeksi asetettiin **iteration_1_done**.

4.8 Suorituskyky

Ensimmäisessä iteraatiossa Nippu-järjestelmälle ei asetettu suorituskykyvaatimuksia. Testejä ei siis tarvittu.

5 Toisen iteraation testit

Toisessa iteraatiossa toteutettiin kaikki **vaatimusdokumentissa määritellyt ominaisuudet**. Iteraatio on suoritettu onnistuneesti, kun **järjestelmätestaus on mennyt läpi**.

5.1 Käyttöliittymä

Testi kä-3

Tehtävä: varmistutaan käyttäjätunnuksen tarkistuksesta ennen käyttöliittymän käynnistystä.

Kuvaus: Käyttäjä voi määritellä käyttäjätunnuksen käyttöliittymän käynnistyskomennossa *USER*-ympäristömuuttujalla. Käyttöliittymä ei käynnisty, jos kyseinen käyttäjätunnus puuttuu asetustiedostosta.

Oletus: Käyttöliittymä voidaan käynnistää joko siten, että käyttäjätunnus annetaan komentorivillä, tai siten, että käyttäjätunnus luetaan *USER*-ympäristömuuttujasta. Ensimmäisessä tapauksessa komentorivisyntaksi on
"java nippu/userinterface/Nippu käyttäjätunnus p" tai
"java nippu/userinterface/Nippu käyttäjätunnus". Jälkimmäisessä tapauksessa komentorivisyntaksi on
"java nippu/userinterface/Nippu p" tai
"java nippu/userinterface/Nippu".
Jos käyttäjätunnusta ei ole annettu komentorivillä eikä *USER*-ympäristömuuttujassa tai jos käyttäjätunnusta ei löydy asetustiedostosta, käyttöliittymä ei käynnisty vaan tulostaa virheilmoituksen ruudulle.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi kä-4

Tehtävä: varmistutaan peruskäyttäjän antamien tietojen tarkistuksesta.

Kuvaus: Käyttäjä syöttää tarpeellista tietoa peruskäyttäjän tulostustyön lähetystä varten. Työn nimi on oletuksena oletustiedosto.txt. Sivumäärä on oletuksena yksi. Sivumäärässä ei voi olla muita merkkejä kuin numeroita, ja sen tulee olla välillä [1, Integer.MAX_VALUE][2].

Oletus: Sivumäärän kentässä ei voi syöttää muita merkkejä kuin numeroita. Jos sivumääräksi annetaan nolla tai suurempi kuin Integer.MAX_VALUE, käyttöliittymä antaa virheilmoituksen tulosta-painiketta painettaessa eikä lähetä työtä.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi kä-5

Tehtävä: varmistutaan tietoliikenneyhteyden toiminnasta käyttöliittymän ja tulostushallinnan välillä (käyttöliittymän rekisteröinti, asetusten vastaanotto, ensimmäisen pääkäyttäjän tunnistus, visualisointi).

Kuvaus: Käyttöliittymä rekisteröityy tulostushallintaan kuuntelijaksi ja saa tulostushallinnasta kuuntelijan tunnisteen, asetustiedoston sekä ensimmäisen pääkäyttäjän lipun tulostushallinnasta. Visualisointinäkymä tulostuslokeineen näyttää tulostinten ja töiden tiloja.

Oletus: Kun käyttöliitymä käynnistyy, terminaalitulostuksesta näkyy käyttöliittymän rekisteröitymisen onnistuminen. Käyttäjätunnukset tulevat käyttäjätunnusvalikkoon ja tulostinten nimet "pakota tulostimelle" -valikkoon. Tulostuslokista näkyy kaikkien töiden tilat. Visualisointinäkymä näyttää myös tulostinkohtaiset jonot.

Tulos: testi meni odotetusti läpi 30.8.2003.

2 Tämän dokumentin useissa kohdissa käytetään arvovälien ilmaisussa matemaattista sulkeisiin perustuvaa merkintätapaa, koska se tekee ilmaisusta lyhyen ja yksikäsitteisen. Esimerkiksi [a, b] tarkoittaa arvoväliä a:sta b:hen siten, että välin päätepisteet a ja b ovat mukana. Normaali sulkumerkki hakasulkeen paikalla tarkoittaisi, että kyseinen päätepiste ei ole mukana.

Testi kä-6

Tehtävä: varmistutaan yhteyden toiminnasta käyttöliittymän ja vastaanoton välillä.

Kuvaus: käyttöliittymä lähettää tulostustöitä vastaanotolle.

Oletus: terminaalitulosteesta näkyy että tulostustöiden lähettäminen onnistuu.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi kä-7

Tehtävä: varmistutaan peruskäyttäjän tulostustöiden lähettämisen toimivuudesta.

Kuvaus: Kun tulosta-painiketta painetaan, käyttöliittymä lähettää yksittäisen tulostustyön vastanotolle. Tulostuslokiin lisätään rivi, jonka mukaan tulostustyö on otettu vastaan.

Oletus: Terminaalitulosteesta näkyy, että tulostustyön lähettäminen onnistui. Tulostuslokista näkyvät työn tiedot sekä tieto siitä, että työn on otettu vastaan.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi kä-8

Tehtävä: varmistutaan pääkäyttäjän antamien tietojen tarkistuksesta (sivumäärä, lähetystiheys).

Kuvaus: Pääkäyttäjä ei voi kirjoittaa muita merkkejä kuin numeroita sivumäärä- ja lähetystiheys-kentissä. Sivumäärä ei voi olla nolla.

Oletus: Käyttäjä voi kirjoittaa vain numeroita sivumäärä- ja lähetystiheys-kenttiin. Jos vähimmäis- tai enimmäissivumäärä on nolla, käyttöliittymä antaa virheilmoituksen eikä lähetä tulostustyötä.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi kä-9

Tehtävä: varmistutaan pääkäyttäjän satunnaistöiden lähettämisen toiminnasta (satunnaiset käyttäjätunnukset, sivumäärät, lähetystiheys).

Kuvaus: Käyttöliittymä arpoo tarvittaessa käyttäjätunnuksen, sivumäärän ja lähetystiheyden satunnaistöiden lähettämiseksi. Jokainen satunnaistyö on nimeltään testiN.txt, jossa N on kokonaisluku.

Oletus: Kun painetaan satunnaistöiden käynnistyspainiketta, vastaava tulostustyö nimeltään testiN.txt (jossa N on kokonaisluku) ilmaantuu tulostuslokiin ja painikkeen tekstiksi tulee "pysäytä". Kun painiketta painetaan, se ponnahtaa ylös ja tekstiksi tulee "käynnistä". Samalla satunnaistöiden lähetys loppuu.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi kä-10

Tehtävä: varmistutaan pääkäyttäjän satunnaistöiden parametrien muokkaamisen toiminnasta.

Kuvaus: Satunnaistöiden lähetyspainikkeen ulkonäkö ja teksti vaihtuvat, kun pääkäyttäjä muokkaa satunnaistöiden tietoja. Satunnaistöiden lähetys loppuu ja painike ponnahtaa automaattisesti ylös ja tekstiksi vaihtuu "käynnistä", kun pääkäyttäjä tekee muutokset sivumäärä- tai lähetystiheys-kenttiin tai valitsee vakiosivumäärän tai vakiotiheyden. Satunnaistöiden lähetys on siis käynnistettävä uudelleen, kun jotain muutoksia on tehty.

Oletus: Kun muutetaan jotain parametria, satunnaistöiden "pysäytä"-painike ponnahtaa automaattisesti ylös ja tekstiksi muuttuu "käynnistä". Samalla satunnaistöiden lähetys loppuu.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi kä-11

Tehtävä: Varmistutaan, että pääkäyttäjän satunnaistyöt ja peruskäyttäjän tulostustyöt eivät haittaa toisiaan. Jokaisella työllä on yksikäsitteinen tunniste.

Kuvaus: jokainen samasta käyttöliittymästä lähetetty työ, riippumatta siitä onko kyseessä peruskäyttäjän työ vai pääkäyttäjän satunnaistyö, saa juoksevan numeroinnin mukaisen tulostustyön tunnisteen.

Oletus: Kun satunnaistöiden lähetys on päällä, käyttäjä voi painaa peruskäyttäjän tulosta-painiketta. Jokainen tulostustyö saa juoksevan numeroinnin mukaisen tunnisteen, mikä voidaan havaita tulostuslokista.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi kä-12

Tehtävä: varmistutaan pääkäyttäjän toimintojen toiminnasta.

Kuvaus: Vain ensimmäinen pääkäyttäjä voi käyttää "maksimiviive palvelun saannille" -kenttää ja ajoittajanpysäytyspainiketta. Muut pääkäyttäjät voivat vain lukea niiden tiloja.

Oletus: Kun käynnistetään useita pääkäyttäjän liittymiä, vain ensimmäisessä pääkäyttäjän liittymässä voi käyttää "maksimiviive palvelun saannille" -kenttää ja ajoituksenpysäytyspainiketta. Muutoin nämä kaksi toimintoa eivät ole käytettävissä.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi kä-13

Tehtävä: varmistutaan tulostuslokin toiminnasta.

Kuvaus: Tulostuslokista näkyvät kaikkien tulostustöiden tilat koko Nippu-järjestelmässä. Jokaisella tulostustyöllä on neljä tilaa: otettu vastaan, jonossa, tulostumassa ja tulostettu.

Oletus: kun muut Nipun komponentit ovat käynnissä, jokaisesta tulostustyöstä kirjautuu tulostuslokiin työn tiloista kertovat neljä neljä.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi kä-14

Tehtävä: varmistutaan visualisointinäkymän toiminnasta.

Kuvaus: Visualisointinäkymä sisältää tulostinkohtaiset jonot. Tulostustyö näkyy visualisoinnissa, kun se on jonossa- tai tulostumassa-tilassa. Tietyn tulostimen jonossa olevat tulostustyöt (enintään viisi työtä) näkyvät tulostimen symbolin alapuolella. Tulostumassa oleva työ näkyy tulostimen symbolin yläpuolella. Tulostetut työt poistuvat visualisoinnista.

Oletus: Kun tulostuslokissa näkyy, että tietty tulostustyö on jonossa, se näkyy myös jononäytössä, jos tulostimen jonossa on vähemmän kun viisi työtä. Kun tulostuslokissa näkyy, että joku tulostustyö on tulostumassa, se näkyy visualisoinnissa tulostimen symbolin yläpuolella. Kun tulostustyö on tulostettu, se katoaa visualisoinnin jononäytöstä.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi kä-15

Tehtävä: varmistutaan tulostimen sekä tulostustyön lisätiedon näyttämisen toiminnasta.

Kuvaus: Visualisoinnin jononäkymä näyttää lisätietoja tulostimesta, kun hiiren osoittimen siirtää tulostimen symbolin päälle. Samoin kun osoittimen siirtää tulostustyön päälle, osoittimen alapuolella näkyy lisätietoja tulostustyöstä.

Oletus: käyttöliittymä toimii kuvauksen mukaan.

Tulos: testi meni odotetusti läpi 30.8.2003.

5.2 Vastaanotto

Vastaanoton tehtävä on viivyttää käyttöliittymän lähettämiä tulostustöitä tietyn ajan, jotta niputtamiskelpoiset työt saapuisivat ajoittajaan lähes samanaikaisesti. Lisäksi vastaanotto säilyttää töitä, jos ajoittaja ei hetkellisesti voi ottaa niitä vastaan. Testeissä hyödynnetään ajan säästämiseksi käyttöliittymää, analysoijaa ja ajoittajaa. Niiden toiminta oletetaan virheettömäksi.

Testi va-1

Tehtävä: varmistutaan työn vastaanottamisen ja lähettämisen toiminnasta kaikkein yksinkertaisimmassa tapauksessa.

Kuvaus: Lähetetään käyttöliittymällä yksi tulostustyö vastaanottoon. Odotetaan hetki ja katsotaan, siirtyykö työ ajoittajaan.

Oletus: Työn vastaanotto tapahtuu välittömästi ja työn lähettämisen tapahtuu muutaman sekunnin kuluttua siitä. Viive aiheutuu palvelunsaannin maksimiviiveestä.

Tulos: testi meni odotetusti läpi 18.7.2003.

Testi va-2

Tehtävä: varmistutaan siitä, että samankaltaiset työt lähtevät vastaanotosta samanaikaisesti, jos ne on lähetetty sinne lyhyen ajan sisällä.

Kuvaus: Lähetetään lyhyen ajan sisällä kolme saman käyttäjän työtä. Odotetaan, että työt lähetetään ajoittajalle. Lähetetään lyhyen ajan sisällä kolme eri käyttäjän

työtä. Odotetaan, että työt lähetetään ajoittajalle.

Oletus: Saman käyttäjän työt lähetetään lähes samanaikaisesti. Eri käyttäjien työt lähetetään hieman eri aikoihin.

Tulos: testi meni odotetusti läpi 18.7.2003.

Testi va-3

Tehtävä: varmistutaan siitä, että tulostustyöt kulkeutuvat lopulta ajoittajaan, vaikka ajoittaja hetkellisesti hylkäisikin työt.

Kuvaus: Pysäytetään ajoittaja. Lähetetään vastaanotolle kolme tulostustyötä. Odotetaan, että ne kaikki lähetetään ainakin kertaalleen ajoittajalle. Todetaan, että pysäytetty ajoittaja hylkää työt. Käynnistetään ajoittaja.

Oletus: Työt säilyvät vastaanotossa, kun ajoittaja hylkää ne. Vastaanotto yrittää uudelleenlähetystä tietyn ajan kuluttua. Kun ajoittaja on taas käynnissä, lähetys onnistuu, työt hyväksytään ajoittajaan ja ne poistetaan vastaanotosta.

Tulos: testi meni odotetusti läpi 18.7.2003.

Testi va-4

Tehtävä: varmistutaan palvelunsaannin maksimiviiveen muuttamisen toiminnasta.

Kuvaus: Lähetetään vastaanotolle uusi maksimiviiveaika. Lähetetään yksi tulostustyö ja mitataan sekuntikellolla aika, jonka työ viettää vastaanotossa. Suoritetaan testi maksimiviiveajoille 1 s, 60 s ja 0 s.

Oletus: Vastaanotolle lähetetty aika päivittyy käyttöliittymään välittömästi. Lähetetty työ viipyy vastaanotossa suunnilleen puolet maksimiviiveajasta.

Tulos: testi meni odotetusti läpi 20.8.2003.

Testi va-5

Tehtävä: varmistutaan maksimiviiveeseen liittyvän odotusajan puolittamisen toimivuudesta.

Kuvaus: Asetetaan maksimiviiveajaksi 128 sekuntia. Lähetetään identtisiä tulostustöitä vastaanottoon kolmen sekunnin välein. Lasketaan, montako työtä ehditään lähettää, ennen kuin vastaanotto lähettää työt ajoittajalle. Toistetaan sama testi, mutta töiden välillä pidetään kymmenen sekunnin tauko.

Oletus: töitä pitäisi ehtiä lähettää ensimmäisessä tapauksessa tasan kuusi kappaletta ja toisessa tasan neljä kappaletta.

Tulos: testi meni odotetusti läpi 23.8.2003.

5.3 Analysoija

Analysoijan viisi keskeistä metodia ovat *getCoordinates*, *cost*, *combinedCost*, *fusion* ja *toString*. Näistä *cost* ja *toString* ovat niin itsestään selviä, että niitä ei testattu. *Fusion*-metodin toiminta on suurelta osin toteutettu Bundle-luokassa, joten siitä testattiin vain optimaalisten tulostinten määrittäminen.

Testi an-1

Tehtävä: varmistutaan *getCoordinates*-metodin oikeellisuudesta.

Kuvaus: Luodaan neljä nippua, joissa kussakin on yksi tulostustyö. Tulostustöiden koordinaatit arvotaan. Viiteen työhön arvotaan vain käypiä koordinaatteja. Toisiin viiteen työhön arvotaan epäkelvot koordinaatit. Kutsutaan metodia *Analyzer*.*getCoordinates* yhden kerran kullakin nipulla. Tämä toistetaan kerran kullekin tavalle muodostaa tulostustyöavaruuden akselit (katso suunnitteludokumentti, luku 3.3.2, kohta COORDINATESTYLE).

Oletus: Kun akseleina on tulostajan sijaintiakselit, metodi palauttaa kaikissa tapauksissa samat koordinaatit kuin parametrina syötetyn nipun työssä oli. Muut tapaukset jätetään tarkan tarkastelun ulkopuolelle ajan säästämiseksi.

Tulos: Testi meni odotetusti läpi 18.7.2003. Myös tarkan tarkastelun ulkopuoliset tapaukset vaikuttivat pintapuolisesti järkeviltä.

Testi an-2

Tehtävä: varmistutaan *combinedCost*-metodin järkevästä toiminnasta.

Kuvaus: Luodaan nippupareja seuraavin tavoin: ensimmäisen nipun tulostajat ovat kaikki myös toisen nipun tulostajia, ensimmäisen nipun koordinaatit ovat lähellä toisen nipun koordinaatteja, ensimmäisen nipun koordinaatit ovat kaukana toisen nipun koordinaateista, nippujen yhteinen sivumäärä on hyvin suuri, nipuilla on sama pakotettu tulostin. Kutsutaan metodia *Analyzer.combinedCost* yhden kerran jokaisella nippuparille.

Oletus: Yhdisteen kustannus on pienehkö (ei paljon yhtä suurempi), kun niput ovat lähekkäin tai niissä on samojen tulostajien töitä. Kustannus on suurehko (ei paljon yhtä pienempi), kun työt ovat kaukana toisistaan tai niiden yhteinen sivumäärä on hyvin suuri.

Tulos: testi meni odotetusti läpi 18.7.2003.

Testi an-3

Tehtävä: varmistutaan *fusion*-metodissa tapahtuvan optimaalisten tulostinten määrittämisen järkevyydestä.

Kuvaus: Määritetään testiympäristöön yksi erittäin nopea tulostin ja muita hitaampia tulostimia. Luodaan kaksi nippua, joiden yhteinen sivumäärä on hyvin suuri ja jotka sijaitsevat kaukana nopeasta tulostimesta. Kutsutaan nipuille *fusion*-metodia. Tarkastellaan yhdistenipulle määrättyjä optimaalisia tulostimia.

Oletus: Optimaalisiin tulostimiin kuuluu nopea tulostin, vaikka se onkin kaukana. Hitaista tulostimista optimaalisia ovat vain hyvin lähellä olevat.

Tulos: testi meni odotetusti läpi 18.7.2003.

5.4 Tulostushallinta

Tulostushallinnan testit kohdistuivat *PrintingManagement*-luokkaan, *NippuPrinter-Simulator*-aliluokkaan ja *PMTimerTask*-aliluokkaan (lyhenteet PM, NS, PT).

Testi PM-1

Tehtävä: varmistetaan, että *PrintingManagement*-luokka lukee tulostushallinnan asetustiedoston oikein.

Kuvaus: luodaan tulostushallinnan asetustiedosto, joka on oikeassa muodossa, ja kokeillaan sen lukemista *PrintingManagement*-luokan konstruktorilla.

Oletus: ohjelma lukee asetustiedoston kentät oikein.

Tulos: testi meni odotetusti läpi 11.8.2003.

Testi PM-2

Tehtävä: varmistetaan, että *PrintingManagement*-luokka toimii oikein (eli käyttää oletusarvoja), jos tulostushallinnan asetustiedosto on tyhjä.

Kuvaus: luodaan tyhjä asetustiedosto ja kokeillaan sen lukemista *Printing-Management*-luokan konstruktorilla.

Oletus: ohjelma käyttää oletusarvojaan.

Tulos: ohjelma antoi *NullPointerException*-poikkeuksen.

Korjaustoimenpide: lisätään tarkistus null-arvon varalle.

Korjauksen tulos: testi meni läpi 11.8.2003.

Testi PM-3

Tehtävä: varmistetaan, että *PrintingManagement*-luokka toimii oikein (eli käyttää oletusarvoja), jos tulostushallinnan asetustiedostoa ei ole olemassa.

Kuvaus: poistetaan tulostushallinnan asetustiedosto ja kokeillaan sen lukemista *PrintingManagement*-luokan konstruktorilla.

Oletus: ohjelma käyttää oletusarvojaan.

Tulos: testi meni odotetusti läpi 11.8.2003.

Testi PM-4

Tehtävä: testataan, että luokan konstruktori lukee yhteisen asetustiedoston ja alustaa tulostimet (*NippuPrinterSimulator*-oliot) oikein.

Kuvaus: Luodaan yhteinen asetustiedosto, jossa tiedot ovat oikeassa muodossa. Tarkistetaan, että tiedot luettiin oikein ja että tulostimet alustettiin oikein.

Oletus: asetustiedoston luku onnistuu ja tulostimet alustetaan oikein.

Tulos: testi meni odotetusti läpi 12.8.2003.

Testi PM-5

Tehtävä: varmistetaan, että *PrintingManagement*-luokka toimii oikein, jos yhteinen asetustiedosto on tyhjä.

Kuvaus: luodaan tyhjä yhteinen asetustiedosto ja kokeillaan sen lukemista *PrintingManagement*-luokan konstruktorilla.

Oletus: ohjelma lukee tyhjän asetustiedoston, *configuration*-kenttä jää tyhjäksi ja yhtään tulostinta ei alusteta.

Tulos: testi meni odotetusti läpi 12.8.2003.

Testi PM-6

Tehtävä: varmistetaan, että *PrintingManagement*-luokka toimii oikein, jos yhteistä asetustiedostoa ei ole olemassa.

Kuvaus: poistetaan yhteinen asetustiedosto ja kokeillaan sen lukemista *Printing-Management*-luokan konstruktorilla.

Oletus: ohjelma kirjoittaa lokiin virheilmoituksen, *configuration*-kenttä jää tyhjäksi ja yhtään tulostinta ei alusteta.

Tulos: testi meni odotetusti läpi 12.8.2003.

Testi PM-7

Tehtävä: varmistetaan, että *execute*-metodi lähettää työn oikealle tulostimelle, jos käytetään pakotettua tulostusta.

Kuvaus: Luodaan *PrintingManagement*-ilmentymä ja kutsutaan sen *execute*-metodia. Parametrina annettava Bundle-olio sisältää yhden PrintJob-olion, jolla on pakotettu tulostin.

Oletus: e*xecute* siirtää työn tulostettavaksi pakotettua tulostinta vastaavalle *NippuPrinterSimulator*-ilmentymälle.

Tulos: testi meni odotetusti läpi 14.8.2003.

Testi PM-8

Tehtävä: varmistetaan, että *execute*-metodilla lähetetyt työt suoritetaan rinnakkaisesti.

Kuvaus: Luodaan *PrintingManagement*-ilmentymä ja kutsutaan sen *execute*-metodia kahdesti. Parametreina annettavat Bundle-olio sisältävät kumpikin yhden PrintJob-olion, jolla on pakotettu tulostin. PrintJob-olioilla on sama sivumäärä, ja valituilla pakotetuilla tulostimilla on sama nopeus.

Oletus: työt tulostuvat kumpikin omalla tulostimellaan rinnakkaisesti.

Tulos: testi meni odotetusti läpi 14.8.2003.

Testi PM-9

Tehtävä: Varmistetaan, että *execute*-metodi lähettää tulostettavat työt lähimmälle optimaaliselle tulostimelle. Testauksen kohteena on siis varsinaisesti *chooseOptimal* metodi.

Kuvaus: Luodaan *PrintingManagement*-ilmentymä ja kutsutaan sen *execute*-metodia. Parametrina annettava Bundle-olio sisältää yhden PrintJob-olion, jolla on asetettuna neljä optimaalista tulostinta.

Oletus: työ tulostuu sille tulostimelle, jonka koordinaatit ovat lähinnä työn tulostajan koordinaatteja.

Tulos: ohjelma ei valinnut parasta optimaalista tulostinta.

Korjaustoimenpide: muutettiin metodia *chooseOptimal* siten, että se laskee etäisyyden oikein.

Korjauksen tulos: testi meni läpi 14.8.2003.

Testi PM-10

Tehtävä: Varmistetaan, että *execute*-metodi lähettää tulostettavat työt lähimmälle tulostimelle, jos optimaalisia tai pakotettuja tulostimia ei ole asetettu. Testauksen kohteena on siis varsinaisesti *chooseAny*-metodi.

Kuvaus: Luodaan *PrintingManagement*-ilmentymä ja kutsutaan sen *execute*-metodia. Parametrina annettava Bundle-olio sisältää yhden PrintJob-olion, jolla ei ole asetettu pakotettua tai optimaalista tulostinta.

Oletus: työ tulostuu sille tulostimelle, jonka koordinaatit ovat lähinnä työn tulostajan koordinaatteja.

Tulos: testi meni odotetusti läpi 15.8.2003.

Testi PM-11

Tehtävä: varmistetaan, että nipun kaikki työt menevät samalle tulostimelle.

Kuvaus: Luodaan *PrintingManagement*-ilmentymä ja kutsutaan sen *execute*-metodia. Parametrina annettava Bundle-olio sisältää kolme PrintJob-oliota. Bundlelle on asetettu pakotettu tulostin.

Oletus: työt tulostuvat peräkkäin pakotetulle tulostimelle.

Tulos: testi meni odotetusti läpi 15.8.2003.

Testi PM-12

Tehtävä: varmistetaan, että *PrintingManagement*-luokka ei sisällä syntaksi-virheitä.

Kuvaus: kokeillaan kääntää luokka javac-komennolla.

Oletus: luokka kääntyy ilman virheilmoituksia.

Tulos: testi meni odotetusti läpi 16.8.2003.

Testi NS-1

Tehtävä: varmistetaan, että *NippuPrinterSimulator*-luokan konstruktori ja omat *get*-metodit toimivat oikein.

Kuvaus: luodaan uusi *NippuPrinterSimulator*-ilmentymä ja todetaan konstruktorin toiminta kokeilemalla ilmentymällä luokan ja yliluokan (*NippuPrinter*) *get*-metodeja.

Oletus: ilmentymän luonti onnistuu, ja *get*-metodit palauttavat siitä konstruktorille annettuja arvoja sekä oletusarvoja.

Tulos: testi meni odotetusti läpi 11.8.2003.

Testi NPS-2

Tehtävä: Varmistetaan, että töiden lisäys onnistuu luokassa. Samalla tulee testattua luokka *getNumberOfJobs.*

Kuvaus: Luodaan uusi *NippuPrinterSimulator*-ilmentymä ja annetaan sille kaksi työtä. Tarkistetaan debug-koodin avulla, että ensimmäinen työ menee ajastimelle ja että toinen menee jonoon. Tarkistetaan *getNumberOfJobs*-metodilla, että järjestelmässä on kaksi työtä. Lisäksi tarkistetaan, että odotusaika lasketaan oikein tulostusnopeudesta ja työn sivumäärästä.

Oletus: Töiden lisäys onnistuu, ja ne käsitellään oikein. G*etNumberOfJobs*-metodi palauttaa kaksi. Odotusaika on laskettu oikein.

Tulos: ohjelma toimi muuten oletetusti, mutta odotusaika oli 60 kertaa liian pieni.

Korjaus: muutettiin odotusajan laskukaavaa siten, että tulos kerrotaan 60:llä.

Korjaustulos: testi meni läpi 16.8.2003.

Testi NPS-3

Tehtävä: varmistetaan, että *NippuPrinterSimulator*-luokka ei sisällä syntaksi-virheitä.

Kuvaus: kokeillaan kääntää luokka javac-komennolla.

Oletus: luokka kääntyy ilman virheilmoituksia.

Tulos: testi meni odotetusti läpi 16.8.2003.

Testi PT-1

Tehtävä: Varmistetaan, että *run*-metodi toimii oikein, jos jonossa on muita töitä. Samalla tulee testattua myös *NippuPrinterSimulator*-luokan toimintaa.

Kuvaus: Luodaan uusi *NippuPrinterSimulator*-ilmentymä, johon lisätään kaksi tulostustyötä. NS-2-testistä tiedämme, että ensimmäinen työ menee ajastimelle ja toinen jonoon. Seurataan debug-koodin avulla, että metodi asettaa ajastimelle uuden työn ja että myös tämä työ pääsee *run*-metodiin.

Oletus: ensimmäinen työ on tulostunut odotusajan kuluttua, minkä jälkeen jonossa ollut työ siirtyy ajastimelle ja alkaa tulostua.

Tulos: Testistä NS-2 huolimatta *NippuPrinterSimulator*-luokassa oli ajastimen ajan laskemisessa virhe, jonka takia ajastimen ajaksi tuli aina 0. Lisäksi kerroin oli kertaluokkaa liian pieni. Muuten ohjelma toimi odotetusti.

Korjaus: muutettiin kaavaa, jolla ajastimen aika lasketaan.

Korjaustulos: testi meni läpi 12.8.2003.

Testi PT-2

Tehtävä: varmistetaan, että *run*-metodi toimii oikein myös useilla tulostustöillä ja että työt tulostuvat oikeassa järjestyksessä.

Kuvaus: Luodaan uusi *NippuPrinterSimulator*-ilmentymä, johon syötetään viisi tulostustyötä. Seurataan debug-koodin avulla töiden tulostumista.

Oletus: työt tulostuvat oikeassa järjestyksessä.

Tulos: testi meni odotetusti läpi 12.8.2003.

Testi PT-3

Tehtävä: varmistetaan, että *PMTimerTask*-luokka ei sisällä syntaksivirheitä.

Kuvaus: kokeillaan kääntää luokka javac-komennolla.

Oletus: luokka kääntyy ilman virheilmoituksia.

Tulos: testi meni odotetusti läpi 16.8.2003.

5.5 Tietoliikenne

Tietoliikenne tulee testattua kokonaisjärjestelmän ja suorituskyvyn testauksen yhteydessä.

5.6 Tietorakenteet

Tässä luvussa käsitellään PrintJob-, Bundle-, *NippuUser*-, *NippuPrinter*- ja *Configuration-Parser*-luokkien testit. Luokista käytettävät lyhenteet ovat PJ, Bu, NU, NP ja CP. Testattavat versiot ovat PrintJob 1.13, Bundle 1.24, *NippuUser* 1.4, *NippuPrinter* 1.8 ja *ConfigurationParser* 1.7. Luokat sisältävät testipääohjelmametodit, joilla testit suoritettiin. Testeistä saa tarkempia tietoja tutustumalla testipääohjelmiin ja ajamalla luokat. Useat näistä testeistä suoritettiin jo ensimmäisessä iteraatiossa, mutta ohjelmakoodin muuttumi-

sen vuoksi ne suoritettiin nyt uudelleen.

Testi PJ-1

Tehtävä: varmistetaan konstruktorin ja *toString*-metodin oikea toiminta.

Kuvaus: luodaan oliot molemmilla konstruktoreilla ja tulostetaan ne käyttäen *toString*-metodia.

Oletus: metodi tulostaa olioiden kenttien arvot oikein.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi PJ-2

Tehtävä: varmistetaan *equals*-metodin toiminta.

Kuvaus: Luodaan kaksi samanlaista PrintJob-oliota ja testataan metodia niihin. Sitten muutetaan toisessa oliossa joidenkin kenttien arvoja, minkä jälkeen testataan *equals*-metodia. Muutetut arvot palautetaan alkuperäisiksi, minkä jälkeen testataan jälleen *equals*-metodia.

Oletus: Metodi palauttaa aluksi true. Muutosten aikana metodi palauttaa false ja lopuksi jälleen true.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi PJ-3

Tehtävä: varmistetaan *serialize*- ja *deserialize*-metodien oikea toiminta.

Kuvaus: luodaan PrintJob-oliosta merkkijono metodilla *serialize* ja tulostetaan saatu merkkijono. Lopuksi merkkijono annetaan parametrina *deserialize*-metodille, jonka antama PrintJob-olio tulostetaan. Testaus suoritetaan sekä PrintJob-oliolle, jonka alkuarvot on asetettu että ilman alkuarvojen asetusta.

Oletus: *Serialize*-metodi tulostaa oikein kaikkien kenttien arvot %-merkillä toisistaan erotettuina. *Deserialize*-metodi osaa luoda merkkijonosta alkuperäisen PrintJob-olion.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi PJ-4

Tehtävä: varmistetaan set-metodien oikea toiminta.

Kuvaus: Testataan set-metodit antamalla niille erilaisia arvoja. Metodeille annetaan sekä sopivia että sopimattomia arvoja. Kaikkien metodien palauttamat arvot tulostetaan.

Oletus: metodit palauttavat oikeat arvot.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi PJ-5

Tehtävä: varmistetaan *hasForcedPrinter*-metodin oikea toiminta.

Kuvaus: Kokeillaan metodia sellaisiin PrintJob-olioihin, joilla joko on tai ei ole pakotettua tulostinta asetettuna. Katsotaan, mitä metodi palauttaa.

Oletus: metodi palauttaa false, kun pakotettua tulostinta ei ole, muulloin true.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi Bu-1

Tehtävä: testataan peruskonstruktoria.

Kuvaus: luodaan Bundle-olio peruskonstruktorilla ja tulostetaan saatu olio.

Oletus: tiedot tulostuvat oikein.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi Bu-2

Tehtävä: *setOptimalPrinters*-metodin testaaminen.

Kuvaus: Asetetaan metodilla sekä sallittuja että vääränlaisia tulostintaulukoita. Tulostetaan metodin palauttamat arvot.

Oletus: metodi palauttaa false, kun taulukko on sopimaton, muuten true.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi Bu-3

Tehtävä: *setJobs*-metodin toiminnan testaus.

Kuvaus: Asetetaan *setJobs*-metodilla Vector-olio, jossa on *Object*-olio. Toisena tapauksena asetetaan Vector-olio, jossa on vain PrintJob-olioita.

Oletus: Metodi palauttaa false, kun Vectorissa on *Object*-olio. Toisessa tapauksessa palautetaan true.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi Bu-4

Tehtävä: *union*-metodin toiminnan varmistaminen.

Kuvaus: Tehdään erilaisia Bundle-olioiden unioneja ja unioneja muun muassa itsensä kanssa sekä tulostetaan unionit. Katsotaan, ovatko tulosteessa kaikki kentät oikein.

Oletus: kaikki kentät tulostuvat oikein.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi Bu-5

Tehtävä: varmistetaan *serialize*- ja *deserialize*-metodien oikea toiminta.

Kuvaus: Luodaan Bundle-oliosta merkkijono metodilla *serialize* ja tulostetaan saatu merkkijono. Lopuksi merkkijono annetaan parametrina *deserialize*-metodille, jonka antama Bundle-olio tulostetaan. Testaus suoritetaan sekä siten, että Bundle-olion alkuarvot on asetettu, että alkuarvoja asettamatta.

Oletus: metodit toimivat halutulla tavalla.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi Bu-6

Tehtävä: *breakBundle*-metodin toiminnan varmistaminen.

Kuvaus: Käytetään *breakBundle*-metodia aikaisemmin tehtyyn yhdisteeseen. Tulostetaan saadun taulukon sisältö.

Oletus: yhdiste-Bundle hajoaa erillisiksi nipuiksi halutulla tavalla.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi Bu-7

Tehtävä: varmistetaan *addOptimalPrinter*- ja *removeOptimalPrinter*-metodien oikea toiminta.

Kuvaus: Lisätään Bundle-olioon optimaaleja tulostimia *addOptimalPrinter*-

metodilla. Lopuksi tulostetaan olio. Oliosta poistetaan tulostimia *removeOptimalPrinter*-metodilla, minkä jälkeen olio tulostetaan.

Oletus: lisätyt tulostimet näkyvät Bundle-olion tulostuksessa, ja poistetut tulostimet tulevat poistetuiksi.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi Bu-8

Tehtävä: varmistetaan *createID*-metodin toiminta.

Kuvaus: Annetaan parametreina *createID*-metodille erilaisia kokonaislukupareja. Tulostetaan metodin palauttamat arvot.

Oletus: metodi palauttaa kokonaislukuja.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi Bu-9

Tehtävä: varmistetaan *getNumberOfPages*-metodin oikea toiminta.

Kuvaus: testataan *getNumberOfPages*-metodia Bundle-olioon ja katsotaan, mitä metodi palauttaa.

Oletus: metodi palauttaa oikean sivumäärän.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi Bu-10

Tehtävä: varmistetaan *getForcedPrinter*-metodin oikea toiminta.

Kuvaus: tulostetaan metodin palauttama arvo.

Oletus: metodi palauttaa oikean pakotetun tulostimen.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi Bu-11

Tehtävä: varmistetaan *hasForcedPrinter*-metodin oikea toiminta.

Kuvaus: tulostetaan metodin palauttamat arvot tilanteissa, jossa Bundle-oliolla joko on tai ei ole pakotettua tulostinta.

Oletus: metodi palauttaa oikeat arvot.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi NU-1

Tehtävä: varmistetaan konstruktorin ja *toString*-metodin oikea toiminta.

Kuvaus: luodaan yksi *NippuUser*-olio ja tulostetaan se.

Oletus: tulostuksessa tulevat oikeat arvot.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi NP-1

Tehtävä: varmistetaan konstruktorin ja *toString*-metodin oikea toiminta.

Kuvaus: luodaan yksi *NippuPrinter*-olio ja tulostetaan se.

Oletus: tulostuksessa tulevat oikeat arvot.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi CP-1

Tehtävä: varmistetaan konstruktorin ja *toString*-metodin oikea toiminta.

Kuvaus: luodaan yksi *ConfigurationParser*-olio ja tulostetaan se.

Oletus: tulostuksessa tulevat oikeat arvot.

Tulos: testi meni odotetusti läpi 18.8.2003.

Testi CP-2

Tehtävä: varmistetaan *get*-metodien oikea toiminta.

Kuvaus: Kokeillaan kaikkia *get*-metodeita testiasetustiedostoon. Kokeillaan lukea mahdollisia ja mahdottomia arvoja.

Oletus: metodit palauttavat oikeat arvot.

Tulos: testi meni odotetusti läpi 18.8.2003.

5.7 Kokonaisjärjestelmä

Testi ko-1

Tehtävä: varmistutaan eri komponenttien välisen viestinnän toimivuudesta.

Kuvaus: Kytketään käyttöliittymä, vastaanotto, ajoittaja, analysoija ja tulostushallinta toisiinsa tietoliikennekomponentin avulla. Lähetetään käyttöliittymästä yksi tulostustyö.

Oletus: Lähetetty työ kulkee komponenttien läpi päätyen tulostushallintaan. Tulostushallinta ilmoittaa käyttöliittymälle työn vastaanottamisesta.

Käyttöliittymä näyttää käyttäjälle tekstimuotoisen viestin, joka kertoo työn onnistuneesta kulkemisesta järjestelmän läpi.

Tulos: testi meni odotetusti läpi 19.8.2003.

5.8 Suorituskyky

Testi SK-1

Tehtävä: testataan, paljonko töiden järjestelmä pystyy käsittelemään.

Kuvaus: Syötetään järjestelmään satunnaistöitä jatkuvalla syötöllä. Suoritetaan ajo monta kertaa ja tutkitaan, kuinka paljon töitä järjestelmä sietää.

Oletus: ainakin tuhannella työllä kaiken pitäisi toimia moitteettomasti.

Tulos: Järjestelmä toimi tuhannella työllä virheettömästi, mutta kun järjestelmässä on todella suuri kuormitus, toiminta hidastuu huomattavasti. Testi meni läpi 30.8.2003.

Testi SK-2

Tehtävä: varmistetaan, että järjestelmä toimii, kun käyttöliittymiä on useita.

Kuvaus: tutkitaan järjestelmän toimintaa, kun käytössä on samalla hetkellä 20 käyttöliittymää.

Oletus: kaikki käyttöliittymät saavat muodostettua yhteyden ja voivat lähettää töitä moitteettomasti.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi SK-3

Tehtävä: tutkitaan, miten järjestelmä toimii suurilla töillä.

Kuvaus: syötetään satunnaistöinä järjestelmään 10 kappaletta 1000 sivun töitä ja tarkastellaan järjestelmän toimintaa.

Oletus: Järjestelmän toiminta ei häiriinny. Tulostusajat ovat pitkiä.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi SK-4

Tehtävä: testataan järjestelmää pitkäaikaisessa, raskaassa käytössä.

Kuvaus: Avataan kaksi käyttöliittymää, joista lähetetään satunnaistöitä vuorokauden ajan. Tutkitaan järjestelmän toimintaa.

Oletus: järjestelmä toimii moitteettomasti.

Tulos: testi meni odotetusti läpi 30.8.2003.

5.9 Järjestelmätestaus

Testi jä-1

Tehtävä: varmistutaan, että järjestelmä toimii lukkiutumatta sadalla samanaikaisella tulostustyöllä.

Kuvaus: Syötetään järjestelmään niin paljon tulostustöitä, että järjestelmässä on ainakin sata tulostumatonta työtä jäljellä. Sen jälkeen odotetaan, kunnes kaikki työt ovat tulostuneet.

Oletus: kaikki työt tulostuvat.

Tulos: testi meni odotetusti läpi 28.8.2003.

Testi jä-2

Tehtävä: varmistetaan, että niputus onnistuu perustapauksessa.

Kuvaus: Yksi käyttäjä lähettää viisi samanlaista, alle kymmenen sivun tulostustyötä kahden sekunnin sisällä. Samanaikaisesti muut käyttäjät lähettävät yhteensä kymmenen satunnaista tulostustyötä.

Oletus: yhden käyttäjän kaikki viisi työtä tulostuvat peräkkäin samalla tulostimella.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi jä-3

Tehtävä: varmistetaan, että työn tulostus pakotetulle tulostimelle toimii.

Kuvaus: lähetetään sekä yksittäisiä että satunnaistöitä jollekin pakotetulle tulostimelle.

Oletus: kaikki työt tulostuvat valitulle pakotetulle tulostimelle.

Tulos: testi meni odotetusti läpi 30.8.2003.

Testi jä-4

Tehtävä: varmistetaan, että tulostinta läheltä tuleva tulostus menee oikealle tulostimelle.

Kuvaus: Asetetaan jonkun käyttäjän sijainti siten, että se on hyvin lähellä jotain tulostinta. Seuraavaksi kyseinen käyttäjä tulostaa sarjan töitä.

Oletus: työt menevät oikealle läheiselle tulostimelle.

Tulos: testi meni odotetusti läpi 30.8.2003.

Liite 1. Sanasto

Bundle on Nippu-ohjelmiston toteutuksessa keskeinen tietorakenne, joka sisältää yhtä tulostustyönippua koskevat tiedot (toteutusdokumentin luku 3.1.2). Bundle on englantia ja suomeksi nippu.

Integer.MAX_VALUE-niminen vakio on Javan enimmäisarvo kokonaisluvulle. Vakion arvo on 2 147 483 647.

Java on yleiskäyttöinen, olioperustainen ohjelmointikieli.

Luokka on Java-ohjelmoinnissa käytettävä tietorakenteiden ja aliohjelmien kokonaisuus.

Metodi on Java-ohjelmoinnissa käytettävä nimitys luokkaan sisältyvästä aliohjelmasta.

Pieni virhe on on sellainen virhe, joka ei estä tai vaikeuta järjestelmän keskeistä toimintaa.

PrintJob on Nippu-ohjelmiston toteutuksessa keskeinen tietorakenne, joka sisältää yhteen alkeistulostustyöhön liittyvät tiedot (toteutusdokumentin luku 3.1.1).

Suuri virhe on sellainen virhe, joka jossain yleisessä käyttötapauksessa suuresti hankaloittaa järjestelmän käyttöä tai estää sen. Suuriksi virheiksi katsotaan myös kaikki sellainen toiminnallisuus, joka on selvästi ristiriidassa vaatimusmäärittelyn kanssa.

TCP-yhteys tarkoittaa IP-verkossa TCP-yhteysprotokollalla (transmission control protocol) muodostettua tietoliikenneyhteyttä. TCP on UDP:n ohella tavallisin yhteysprotokolla IP-verkoissa. TCP-protokolla takaa sovelluksen lähettämien pakettien pääsemisen perille ja pakettien keskinäisen järjestyksen säilymisen.

Testaus tarkoittaa ohjelman suorittamista tarkoituksena löytää virheitä.

Testaussuunnitelma on kirjallinen suunnitelma siitä, millaisten testien avulla ohjelmistoa testataan ja millä perusteilla testit menevät läpi.

Testi on sellainen osa testaussuunnitelmaa, joka keskittyy ohjelmiston tiettyyn osaan; esimerkiksi johonkin luokkaan tai metodiin.

Testiajo on tapahtuma, jossa testejä suoritetaan.

Tietoliikenne on tiedon välitystä eri ohjelmistokomponenttien välillä käyttäen erityisesti TCP/IP-verkkoyhteyksiä.

Tietoliikenneprotokolla on säännöstö, jota verkkoasemien käyttöjärjestelmien ja sovellusten on noudatettava, jotta niiden välinen yhteys olisi mahdollinen. Esimerkkejä tietoliikenneprotokollista ovat Nipun sovellustason protokolla (toteutusdokumentin luku 3.3.3), yhteystason TCP-protokolla ja verkkotason IP-protokolla.

Tietoliikenneyhteys tarkoittaa kahden Nipun komponentin välistä tietoliikennettä erityisesti niin, että jos esimerkiksi vastaanotto on yhteydessä kahteen eri käyttöliittymään, on tällöin kyseessä kaksi eri tietoliikenneyhteyttä. Nipussa käytetään TCP-protokollan mukaisia tietoliikenneyhteyksiä.

Tulostustyö on Nipun tai jonkin toimintaympäristön laitteen käsittelyssä oleva, tulostajan antama tulostuskomento. Tulostustyö voi olla yhdiste (eli nippu) kahdesta tai useammasta tulostustyöstä tai niiden yhdisteestä.

Virhe on ohjelman poikkeama sen vaatimusmäärittelystä. Katso "pieni virhe" ja "suuri virhe".

Ympäristömuuttuja on käyttöjärjestelmän asettama käyttöympäristön parametrin sisältävä muuttuja, jonka arvon ajettava ohjelma voi lukea. Esimerkiksi unix-tyyppisissä järjestelmissä *USER*-ympäristömuuttuja sisältää käyttäjän käyttäjätunnuksen.