

1. Johdanto

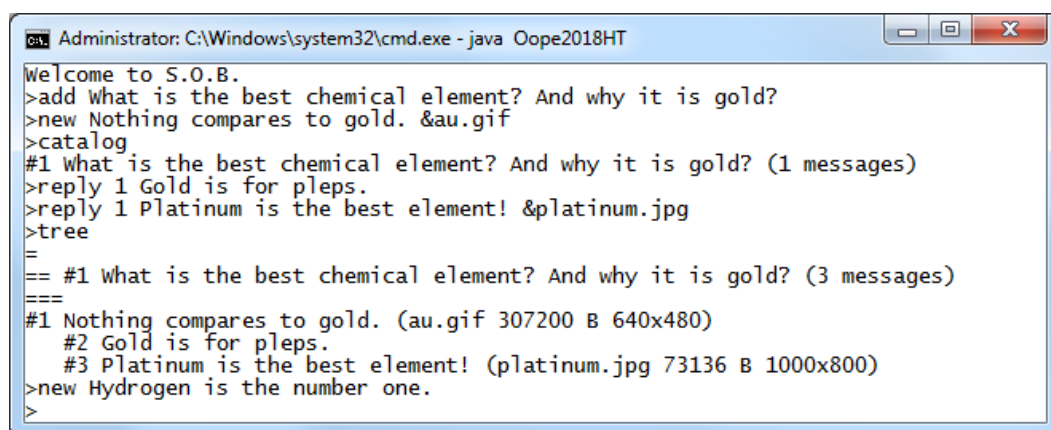
1.1. Tehtävä

Sähköisillä keskustelupalstoilla [1] vaihdetaan viestejä eri aiheisiin liittyen. Ensimmäiset palstat olivat uutisryhmiä (“nyyssit”) ja BBS-järjestelmien (“purkki-en”) palstoja. Sähköinen keskustelu on siirtynyt nykyisin pääosin Internet-selainpohjaiseksi. Vanhoista erillisellä asiakasohjelmalla käytettävistä keskustelutavoista on enää voimissaan lähinnä IRC.

Tiettyyn aiheeseen liittyvät sähköiset viestit muodostavat ketjun, joka tunnetaan myös muun muassa lankana. Ketjun aloittava viesti ei vastaa mihinkään viestiin, kun taas myöhemmät viestit voivat olla vastauksia toisiin viesteihin tai uusia keskustelunavauksia, jotka eivät vastaa aiempiin viesteihin. Yhdellä viestillä voidaan vastata yhteen tai useampaan aiempaan viestiin. Joillakin palstoilla viesteihin voi liittää tiedostoja. Keskustelupalsta on yleensä jaettu teemoittain keskustelualueisiin (huoneisiin), joilla on tyypillisesti kymmenittäin keskusteluketjuja. Suuremmilla palstoilla voi olla kymmeniä alueita.

Harjoitustyössä on tehtävänä simuloida olioperustaisesti pientä keskustelupalstaa Simple Oope Board (S.O.B.), jolla on vain yksi alue, jolloin keskustelupalsta- ja alue ovat käytännössä sama asia. Tehtävää on yksinkertaistettu monin tavoin, jotta se ei kävisi liian vaikeaksi. Tärkein rajoite on se, että ohjelman käyttäjä voi käytännössä keskustella vain itsensä kanssa. Tehtävää on yksinkertaistettu myös siten, että vastaus voi liittyä vain yhteen aiempaan viestiin. Viesteihin liitettäviä kuvia ja videoita vastaavat tekstitiedostot, joissa on kuvan tai videon keskeisimmät tiedot.

Ohjelman käyttö on kömpelöä, koska kaikki toiminnot suoritetaan komentopohjaisesti ja ketjut ja viestit yksilöidään tunnisteluvuilla. Kuvassa 1 on esitetty alkuaineisiin liittyvän keskustelun aloitus harjoitustyöohjelmassa.



```
Administrator: C:\Windows\system32\cmd.exe - java Oope2018HT
Welcome to S.O.B.
>add What is the best chemical element? And why it is gold?
>new Nothing compares to gold. &au.gif
>catalog
#1 What is the best chemical element? And why it is gold? (1 messages)
>reply 1 Gold is for plops.
>reply 1 Platinum is the best element! &platinum.jpg
>tree
=
== #1 What is the best chemical element? And why it is gold? (3 messages)
===
#1 Nothing compares to gold. (au.gif 307200 B 640x480)
  #2 Gold is for plops.
    #3 Platinum is the best element! (platinum.jpg 73136 B 1000x800)
>new Hydrogen is the number one.
>
```

Kuva 1: Keskustelu, jossa ohjelman käyttäjä väittelee itsensä kanssa parhaasta alkuaineesta. Aluksi luodaan ketju ja sitten sen ensimmäinen viesti, jossa kannatetaan kultaa. Viestiä painotetaan liittämällä siihen kulta kuva. Viestiin tulee kaksi vastausta ja viimeinen viesti on itsellinen kannatusääni vedylle. Keskustelun kulku voidaan visualisoida puumaisena rakenteena.

1.2. Työn organisointi

Harjoitustyö tehdään **itse ja lähinnä omalla ajalla**. Muiden kurssilaisten kanssa voi vaihtaa ideoita, mutta koodin kopiointi ei tietenkään ole sallittua. Samoin muualta kuin kurssin verkkosivulta löytyneen koodin kopiointi omaan ohjelmaan on kielletty.

1.3. Pakollisuus ja korvaavuudet

Harjoitustyö on pakollinen kurssin 10 opintopisteen laajuisena suorittaville opiskelijoille. Tällaisia kurssilaisia ovat erityisesti kaikki tietojenkäsittelytieteiden tutkinto-ohjelman omat opiskelijat. Ota yhteyttä tutkinto-ohjelmasi opinto-koordinaattoriin tai kurssin vastuuopettajaan, jos et tiedä tuleeko sinun suorittaa kurssi 5 vai 10 opintopisteen laajuisena.

Harjoitustyön voi korvata A) muiden oppilaitosten opinnoilla, B) edellisellä Olio-ohjelmoinnin perusteet -kurssilla hyväksytyllä harjoitustyöllä tai C) edellisten kohtien tapaisella painavalla syyllä. Kohdan A perusteella on annettu kaikki pyydetty korvaavuudet. Kohdan B perusteella annettavista harjoitustyökorvaavuuksista on tarkempia tietoja kurssin kotisivuilla. B-kohdan osalta on tärkeintä muistaa, että harjoitustyö korvautuu vain, jos ottaa yhteyttä kurssin vastuuopettajaan. Tähän mennessä tulleet yhteydenotot ja sopimukset on kirjattu ylös eikä uusia yhteydenottoja näiltä osin tarvita.

2. Ohjelman toiminnot

Ohjelman keskustelualueeseen liittyvillä toiminnoilla luodaan keskusteluketju, listataan alueen keskusteluketjut, valitaan aktiivinen keskusteluketju ja lopetetaan ohjelma. Muut toiminnot liittyvät viesteihin. Ohjelmassa voidaan luoda uusi viesti, vastata aiempaan viestiin, tulostaa ketjun viestit puu- ja listamuodossa, tulostaa listamuodossa tietty määrä vanhimpia tai uusimpia viestejä, tyhjentää viesti ja hakea tietyn merkkijonon sisältävät viestit. Seuraavassa esitellään ohjelman toiminnallisuutta pienten esimerkkien avulla. Laajempia esimerkkiajoja julkaistaan kurssin kotisivujen *Harjoitustyö*-kohdassa.

2.1. Ohjelman käynnistäminen

Ohjelma luo käynnistymisen yhteydessä tyhjän keskustelualueen.

2.2. Käyttöliittymä

Ohjelmalla on oikean komentotulkin tapainen tekstipohjainen käyttöliittymä. Ohjelman käynnistyessä tulostetaan kuvassa 2 esitetyllä tavalla ensin tervehdysteksti "Welcome to S.O.B." ja omalle rivilleen komentokehoteen merkki. Tämän jälkeen jäädään odottamaan käyttäjän syötettä samalle riville välittömästi kehotemerkin jälkeen.

```
Welcome to S.O.B.  
>
```

Kuva 2: Ohjelma käynnistyksen jälkeen.

Ohjelmaa käytetään komentoikkunan tapaan kirjoittamalla halutun toiminnon käynnistävä komento ja painamalla *Enter*-näppäintä. Komento ja sen mahdolliset parametrit erotetaan toisistaan yhdellä välilyönnillä. Kaikkien muiden kuin lope-tuskomennon jälkeen tulostetaan kehoite riviä vaihtamatta ja jäädään odottamaan uutta komentoa välittömästi kehotemerkin jälkeen, jos komento oli oikeellinen.

Ohjelma vastaa virheelliseen komenttoon tulostamalla omalle rivilleen "Error!" ennen uuden komennon lukemista. Virheeksi katsotaan tuntematon komento tai tunnettu komento virheellisellä parametrilla (kuva 3). Tunnetulla komennolla voi olla myös väärä määrä parametreja. Ylimääräiset välilyönnit ennen komentoa, komennon jälkeen, komennon ja sen parametrin välissä tai komennon parametrin välissä katsotaan virheeksi.

Viestiketjuun kohdistuvan komennon antaminen ennen ensimmäisen ketjun luomista on virhe.

```
>sage  
Error!  
>reply 666 There is a special place for sulfur.  
Error!
```

Kuva 3: Tuntemattoman komennon aiheuttama virhe ja virheellisen parametriarvon aiheuttama virhe. Vastaus ei onnistu, koska ketjussa ei ole viestiä tunnuksella 666.

Kaikkien **tulosteiden muoto on kiinnitetty**, jotta ohjelmien automaattinen, tulosteiden vertailuun perustuva tarkistus olisi mahdollista (luku 4). Tarkistusta automatisoimalla pyritään siihen, että kurssin opettajille jää enemmän aikaa työn rakenteen ja laadun arviointiin.

Komennot pitää lukea *In*-apuluokan avulla, jotta ohjelmien tehtävänannon mukainen käyttäytymisen olisi todennäköisempää. Harjoitustyössä käytetään *oo-pe2018ht.apulaiset*-nimiseen pakkaukseen (luku 3.5) sijoitettua versiota *In*-luokasta, joka on saatavilla kurssin kotisivujen *Koodit*-kohdassa. Tässä pakkauksessa ovat myös opettajan antamat rajapinnat ja annotaatiot.

2.3. Viestiketjun luominen

Keskustelualueelle luodaan uusi viestiketju *add*-komennolla. Ohjelma antaa uudelle ketjulle tunnusteen, joka on juokseva kokonaisluku. Ensimmäisen ketjun tunniste on luku yksi.

Komennon parametri on merkkijono, joka kuvaa viestiketjun aihetta. Huomaa, että merkkijono on viestien tekstin tapaan yleensä lause, jossa sanat erotetaan toisistaan välilyönnein. Merkkijonossa on oltava ainakin yksi merkki. Yksi välilyönti ei kelpaa keskustelun aiheeksi. Kuvassa 4 luodaan uusi alkuaineisiin liittyvä viestiketju.

```
> add What is the best chemical element? And why it is gold?
>
```

Kuva 4: Luodaan keskustelalueen ensimmäinen viestiketju.

2.4. Viestiketjujen listaaminen

Ketjut listataan luomisjärjestyksessä *catalog*-komennolla. Kunkin ketjun tiedot ovat omalla rivillään. Ketjun rivin alussa on ristikkomerkki ja ketjun tunniste. Alkua seuraa ketjun aihe, joka erotetaan alusta yhdellä välilyönnillä. Aihetta seuraa yksi välilyönti ja kaarisuljeparin sisällä annettu ketjuun kuuluvien viestien lukumäärä (kuva 5). Komento ei tulosta mitään, jos alueella ei ole vielä ketjuja.

```
>catalog
#1 What is the best chemical element? And why it is gold? (1 messages)
>
```

Kuva 5: Keskustelalueen ainoan ketjun tiedot. Ketjussa on vain sen aloittava viesti.

2.5. Viestiketjun valinta

Ketjujen välillä liikutaan *select*-komennolla. Komento aktivoi parametrilla yksilöidyn ketjun siten, että kaikki ketjuihin liittyvät toiminnot kohdistuvat valittuun ketjuun. Alue aktivoi automaattisesti ensimmäisen siihen luodun ketjun. Parametri on virheellinen, jos se ei ole jonkin luodun ketjun yksilöivä kokonaisluku. Kuvasssa 6 aktivoidaan alueen toinen ketju.

```
>select 2
>
```

Kuva 6: Aktivoidaan alueen toinen ketju.

2.6. Uuden viestin luominen

New-komento luo uuden viestin, joka ei ole vastaus mihinkään aiemmista viesteistä. Ohjelma antaa uudelle viestille tunnusteen, joka on juokseva kokonaisluku. Alueen ensimmäisen viestin tunniste on luku yksi. Huomaa, että uuden viestin tunnus ei riipu ketjusta, johon viesti kuuluu. Viestiketjun tunnukset eivät ole näin ollen peräkkäisiä kokonaislukuja, jos ketjuja vaihdetaan viestejä kirjoitettaessa. Jos esimerkiksi ensimmäiseen ketjuun kirjoitetaan viisi viestiä ja sitten kirjoitetaan kaksi viestiä toiseen ketjuun, ovat ensimmäisen ketjun viestien tunnukset 1–5 ja toisen ketjun viestien tunnukset 6 ja 7.

Komennon pakollinen parametri on merkkijono (tavallisesti lause), jossa on oltava vähintään yksi merkki, jonka ei saa olla välilyönti. Viestiin voidaan liittää valinnaisesti tiedosto siten, että merkkijonon perään kirjoitetaan yksi välilyönti, *et*-merkki ja tiedoston nimi. Liittäminen onnistuu, jos ohjelman ajohakemistossa on tiedosto annetulla nimellä. Ajohakemisto on hakemisto, jossa ohjelman ajoluokka käynnistetään (luku 3.7). Liitettävä tiedosto ei ole “oikea” kuva tai video, vaan tekstitiedosto, jossa on tiedosto-olion luomiseen tarvittavat tiedot (luku 3.6). Kuvasssa 7 luodaan uusi viesti ja liitetään siihen kuvatiedosto.

```
> new Nothing compares to gold. &au.gif  
>
```

Kuva 7: Luodaan uusi viesti. Kuvan liittäminen onnistuu, koska *au.gif*-tiedosto on ajohakemistossa.

2.7. Viestiin vastaaminen

Aiemmin lähetettyyn viestiin vastataan *reply*-komennolla. Vastaukselle annetaan uuden viestin tapaan automaattisesti yksilöivä kokonaisluku (luku 2.6). Komennon kolmesta parametrasta kaksi ensimmäistä ovat pakollisia. Komentoa seuraa siitä välilyönnillä erotettu tunniste, joka yksilöi vastattavan viestin. Tämä parametri on oikeellinen, jos se on kokonaisluku, joka yksilöi aktiiviseen viestiketjuun kuuluvan viestin. Toinen parametri on merkkijono ja kolmas vastaukseen mahdollisesti liitettävä tiedosto (luku 2.6). Kuvassa 8 vastataan kahdesti aktiivisen ketjun aiempaan viestiin.

```
>reply 1 Gold is for pleps.  
>reply 1 Platinum is the best element! &platinum.jpg  
>
```

Kuva 8: Kaksi vastausta viestiin, jonka tunniste on luku 1. Jälkimmäisessä vastauksessa on lauseen lisäksi kuva.

2.8. Ketjun tulostus puumuodossa

Tree-komento tulostaa aktiivisen viestiketjun viestit näytölle keskustelun kulkua visualisoivassa puumaisessa muodossa. Ensin tulostetaan keskusteluketjun aihe pitkälti samaan tapaan kuin *catalog*-komennossa (luku 2.4). Erona on se, että tuloste koristeltu kolmen rivin laajuiseksi (kuva 9).

Kunkin viestin merkkijonoesitys (luku 3.4) tulostetaan omalle rivilleen. Komento tulostaa vain viestiketjun aiheen, jos ketjussa ei ole vielä viestejä. Vastauksen rivi sisennetään kolmella välilyönnillä. Alkuperäiseen viestiin vastaavien viestien sisennyksen taso on kolme välilyöntiä, näihin vastauksiin vastaavat viestit sisennetään kuudella välilyönnillä ja niin edelleen. Viestit tulostetaan esijärjestyksessä (luku 3.5).

Kuvassa 9 esitetään keskustelualueen aktiivisen viestiketjun puumainen visualisointi. Tulosteen kolmelta ensimmäiseltä riviltä käy ilmi, että kyseessä on alueen ensimmäinen viestiketju, ketjun liittyy alkuaineiden paremmuuteen ja että ketjussa on viisi viestiä. Ensimmäinen ja neljäs viesti eivät ole vastauksia ja siten niitä ei sisennetä. Toinen ja kolmas viesti on sisennetty kolmella välilyönnillä, koska ne ovat vastauksia ensimmäiseen viestiin. Viides viesti on sisennetty kuudella välilyönnillä, koska se on vastaus ensimmäiseen vastaukseen.

```
>tree
=
== #1 What is the best chemical element? And why it is gold? (5 messages)
===
#1 Nothing compares to gold. (au.gif 307200 B 640x480)
  #2 Gold is for plebs.
    #5 You misspelled plebs. (picard_facepalm.webm 1048576 B 10.5 s)
      #3 Platinum is the best element! (platinum.jpg 73136 B 1000x800)
        #4 Hydrogen is the number one.
          >
```

Kuva 9: Aktiivisen viestiketjun esitys puumaisena rakenteena.

2.9. Ketjun tulostus listana

Aktiivisen ketjun kaikki viestit voidaan tulostaa myös listana, joka on järjestetty viestien tunnisteiden mukaiseen nousevaan järjestykseen, jolloin ketjun vanhin viesti tulostetaan ensimmäisenä ja uusin viesti viimeisenä. *List*-komento tulostaa ketjun aiheen kuten *tree*-komento (luku 2.8) ja myös viestien sisältö on sama. Listauksessa viestejä ei sisennetä ja ne tulostetaan luomisjärjestyksessä. Komento tulostaa vain viestiketjun aiheen, jos ketjussa ei ole vielä viestejä. Kuvassa 10 listataan keskustelualueen aktiivinen viestiketju.

```
>list
=
== #1 What is the best chemical element? And why it is gold? (5 messages)
===
#1 Nothing compares to gold. (au.gif 307200 B 640x480)
#2 Gold is for plebs.
#3 Platinum is the best element! (platinum.jpg 73136 B 1000x800)
#4 Hydrogen is the number one.
#5 You misspelled plebs. (picard_facepalm.webm 1048576 B 10.5 s)
>
```

Kuva 10: Aktiivisen viestiketjun esitys puumaisena listana.

2.10. Vanhimpien viestien listaaminen

Head-komento tulostaa parametrinaan saamansa määrän aktiivisen viestiketjun vanhimpia viestejä tunnuksen mukaan järjestettynä listana. Komento ei tulosta ketjun aihetta. Komennon parametri on virheellinen, jos se on pienempi kuin yksi tai suurempi kuin ketjun alkioden lukumäärä. Komennon kohdistaminen tyhjiin viestiketjuun on virhe. Kuvassa 11 listataan aktiivisen viestiketjun ensimmäinen viesti.

```
>head 1
#1 Nothing compares to gold. (au.gif 307200 B 640x480)
>
```

Kuva 11: Aktiivisen viestiketjun ensimmäinen viesti listattuna.

2.11. Uusimpien viestien listaaminen

Tail-komento tulostaa parametrinaan saamansa määrän aktiivisen viestiketjun uusimpia viestejä tunnuksen mukaan järjestettynä listana. Komento ei tulosta ketjun aihetta. Komennon parametri on virheellinen, jos se on pienempi kuin yksi tai suurempi kuin ketjun alkioden lukumäärä. Komennon kohdistaminen tyhjiin viestiketjuun on virhe. Kuvassa 12 listataan aktiivisen viestiketjun viimeinen viesti.

```
>tail 1
#5 You misspelled plebs. (picard_facepalm.webm 1048576 B 10.5 s)
>
```

Kuva 12: Aktiivisen viestiketjun viimeinen viesti listattuna.

2.12. Viestin tyhjentäminen

Viestejä ei voi poistaa, mutta niiden sisältö voidaan tyhjentää. *Empty*-komento korvaa parametrilla yksilöidyn viestin merkkijonon kymmenestä peräkkäisestä yhdysmerkestä (näppäimistöllä miinusmerkki) koostuvalla merkkijonolla ja poistaa viestiin mahdollisesti liitetyn ~~kuvan~~ tiedoston. Kuvassa 13 käyttäjä on nolostunut kirjoitusvirheestä ja päättää tyhjentää toisen viestin. Teksti korvataan, mutta tiedostolle ei ole tarpeen tehdä mitään, koska sellaista ei ole liitetty viestiin.

```
>empty 2
>list
=
== #1 What is the best chemical element? And why it is gold? (5 messages)
===
#1 Nothing compares to gold. (au.gif 307200 B 640x480)
#2 -----
#3 Platinum is the best element! (platinum.jpg 73136 B 1000x800)
#4 Hydrogen is the number one.
#5 You misspelled plebs. (picard_facepalm.webm 1048576 B 10.5 s)
>
```

Kuva 13: Tyhjennetään toisen viestin sisältö.

2.13. Tekstin hakeminen

Find-komento hakee parametrinaan saamaansa merkkijonoa kaikista aktiivisen ketjun viesteistä. Pienet ja suuret kirjaimet katsotaan eri merkeiksi. Viesteistä voidaan hakea myös yhdestä tai useammasta välilyönnistä koostuvaa merkkijonoa. Haetun merkkijonon sisältävät viestit listataan tunnisteensa mukaisessa kasvavassa järjestyksessä. Komento ei tulosta mitään, jos merkkijonon sisältäviä viestejä ei löydetä tai viestiketju on tyhjä. Kuvassa 14 aktiivisesta viestiketjusta haetaan merkkijonoa "gif".

```
>find gif
#1 Nothing compares to gold. (au.gif 307200 B 640x480)
>
```

Kuva 14: Haettu merkkijono löydetään vain ketjun ensimmäisestä viestistä.

2.14. Ohjelman lopetus

Ohjelmasta poistutaan parametrittomalla *exit*-komennolla (kuva 15). Komennon jälkeen tulostetaan riviä vaihtaen "Bye! See you soon."-teksti.

```
>exit  
Bye! See you soon.
```

Kuva 15: Ohjelman lopetus komennolla.

3. Ohjelman rakenne

3.1. Yleistä

Ohjelma on jaettava **kapseloituihin luokkiin**, joiden sisältö puolestaan on jaettava järjevän mittaisiin metodeihin. Luokissa on vältettävä saman koodin kopioimista eri paikkoihin, koska koodista voi kirjoittaa asianomaisista paikoista kutsutavan metodin. Metodien korvaaminen, kuormittaminen ja monimuotoisuus parametrien välityksessä vähentävät tarvetta toistaa koodia eri metodien sisällä.

Muistathan, että luokat rakennetaan tiedon – ei toimintojen – ympärille. Luokkametodeista koostuvat, esimerkiksi *Math*-luokan tapaiset apuluokat ovat poikkeus tähän sääntöön. Ohjelman toimintoja vastaavia luokkia ei saa tehdä, koska toiminnot eivät ole luokkia itsessään, vaan luokkien metodeja.

Olio-ohjelmoinnissa vain keskeisimmät käsitteet mallinnetaan. Luokkia tarvitaan vain sen verran, että ohjelma saadaan toimimaan olioperustaisesti edellä määritellyllä tavalla. Toisaalta kullakin luokalla tulisi olla yksikäsitteinen vastuu. Luokkien lukumäärää ei saa pienentää antamalla luokalle useita sille kuulumattomia vastuuta. Erityisesti käyttöliittymäluokan sisältöä on syytä pitää silmällä. Käyttöliittymä vastaa vain ihmisen ja koneen vuorovaikutuksesta ja näin ollen keskustelualueen logiikasta vastaavaa koodia ei saa kirjoittaa sinne.

Muista noudattaa hyvää ohjelmointitapaa: sisennä koodia johdonmukaisesti luettavuuden parantamiseksi, kommentoi riittävästi ja oikeissa paikoissa, liitä jokaiseen metodiin yleisluonteinen kommentti, nimeä vakiot, muuttujat ja operaatiot järjevästi, käytä vakioita, pidä rivit riittävän lyhyinä, käytä välejä lauseiden sisällä ja erota loogiset kokonaisuudet toisistaan väliriveillä, pidä operaatiot järjevän mittaisia – operaation tulisi mahtua yhdelle A4-kokoiselle sivulle ja varaudu metodeissa virheellisiin parametrien arvoihin [2, 3].

Erityisesti olio-ohjelmointiin liittyviä hyviä tapoja ovat: luokkiin liitetyt yleiset kommentit, attribuuttien kommentointi, attribuuttien harkittu käyttö [4] ja luokkien tai rajapintojen sijoittaminen omiin tiedostoihinsa.

Sisennä koodi välilyönnein. WETO ei hyväksy muilla tavoilla sisennettyä lähdekoodia, jotta koodi näkyisi opettajan editorissa täsmälleen opiskelijan aikomalla tavalla.

Lue tiedot näppäimistöltä *oope2018ht.apulaiset*-pakkauksen **In-luokalla** (luku 3.7).

3.2. Luokkahierarkia

Harjoitustyössä periytymistä käytetään erityisesti tiedostojen mallintamiseen. Ohjelmassa tulee olla abstrakti ylikuokka *Tiedosto* ja siitä perityt konkreettiset *Kuva*- ja *Video*-luokat.

Ylikuokassa on *String*-tyyppinen attribuutti tiedoston nimelle ja *int*-tyyppinen attribuutti tiedoston koolle tavuina. Luokalla on kaksiparametrinen rakentaja, jonka parametrien tyypit ovat järjestyksessä *String* ja *int* sekä aksessorit molemmille attribuuteille. Nimessä tulee olla vähintään yksi merkki ja koon tulee olla vähintään yksi tavua.

Kuvan omat tiedot ovat leveys ja korkeus pikseleinä (*int*). Kuvan mittojen on oltava vähintään 1×1-pikseliä. Tee luokalle neliparametrinen rakentaja, jolle välitetään järjestyksessä luetellen tiedot uuden kuvan nimestä, koosta, leveydestä ja korkeudesta. Kutsu rakentajassa ylikuokan kaksiparametrin rakentajaa. Tee attribuuteille aksessorit.

Videosta tunnetaan perittyjen tietojen lisäksi videon pituus sekunteina (*double*, > 0). Tee luokalle kolmiparametrinen rakentaja, jolle välitetään järjestyksessä luetellen tiedot uuden videon nimestä, koosta ja pituudesta. Kutsu rakentajassa ylikuokan kaksiparametrin rakentajaa. Tee attribuuteille aksessorit.

Kaikki luokkahierarkian rakentajat ja asettavat metodit heittävät *IllegalArgumentException*-poikkeuksen, jos parametri on virheellinen. Attribuutin tyyppi määrää lukevan metodin tyypin ja asettavan metodin parametrin tyypin. Asettavilla metodeilla on aina yksi parametri.

Leimaa lukevat metodit *@Getter*-annotaatiolla ja asettavat metodit *@Setter*-annotaatiolla. Annotaatiot mahdollistavat aksessoreiden vapaamman nimeämisen, koska lisätieto auttaa WETOA tunnistamaan luku- ja asetusmetodit ilman tarkempia vaatimuksia näiden metodien nimille. Edellä mainitut annotaatiot on eivätkä ole Javan omia, vaan ne on määritelty *oopen2018ht.apulaiset*-pakkauksessa (luku 3.7). Annotaatiot on otettava käyttöön *import*-lauseen avulla, jotta Java osaa kääntää niitä käyttävän koodin. Annotaatio kirjoitetaan välittömästi ennen metodin otsikkoa. Esimerkiksi:

```
/** Yrittää asettaa tiedoston koon.
 *
 * @param uusiKoko koko tavuina.
 * @throws IllegalArgumentException jos koko on pienempi kuin yksi.
 */
@Setter
public void koko(int uusiKoko) throws IllegalArgumentException ...
```

Voit kirjoittaa *Tiedosto*-luokalle ja sen aliluokille oletusrakentajat, jos katsot niistä olevan hyötyä ohjelman toteuttamisessa.

Olion merkkijonoesityksestä on paljon iloa tässäkin työssä. Korvaa siksi *Object*-luokan *toString*-metodi luokkahierarkian jokaisella tasolla. Tee korvaus “ketjuttaminen”: kutsu aliluokkien *toString*-metodin korvauksissa ylikuokan versiota *super*-attribuutin avulla juuriluokka pois lukien, koska *Object*-luokan *toString*-metodia

ei pidä kutsua. Kuvan ja videon merkkijonoesitykset alkavat tiedoston nimellä ja koolla, jotka erotetaan toisistaan yhdellä välilyönnillä. Merkkijonon loppuun lisätään vielä välilyönti ja suuri b-kirjain.

Kuva-luokassa korvattu *toString*-metodi palauttaa merkkijonon, jossa tiedostoille yhteistä merkkijonoa seuraa välilyönti, kuvan leveys, pieni x-kirjain ja kuvan korkeus. Jos tiedosto-olion tiedot ovat esimerkiksi "au.gif", 307200, 640 ja 480, on metodin paluuarvo "au.gif 307200 B 640x480".

Video-luokan *toString*-metodin palauttama merkkijono koostuu yhteisestä merkkijonoesityksestä, välilyönnistä, videon pituudesta ja merkkijonosta " s". Jos videon tiedot ovat "picard_facepalm.webm", 1048576 ja 10.5, on merkkijonoesitys "picard_facepalm.webm 1048576 B 10.5 s".

Kuva- ja *Video*-luokkien *toString*-metodeja kutsutaan, kun keskusteluketju tulostetaan eri tavoin. Sijoita *Tiedosto*-, *Kuva*- ja *Video*-luokat omaa pakkaukseensa (luku 3.7).

3.3. *OmaLista*-luokka ja muut tietorakenteet

Ohjelman pääasiallisena tietorakenteena käytetään *fi.uta.csjola.oope.lista*-pakkauksen *LinkitettyLista*-luokasta perittyä *OmaLista*-luokkaa. Luokan tärkein käyttökohde on *Viesti*-luokka, jossa on tarpeen säilöä viitteet viestiin vastaaviin viesteihin (luku 3.4). Omaa listaa tarvitaan myös ketjun viestien säilömiseen ja keskustelualan ketjujen säilömiseen.

Lista-pakkaus on saatavilla kurssin kotisivujen *Koodit*-kohdasta. Huomaa, että **pakkauksen sisältöä ei saa muuttaa** millään tavoin. Erityisesti *Solmu*-luokka pitää jättää entiselleen. **Javan API:n tietorakenneluokkia, kuten *ArrayList*-luokkaa, ei saa käyttää.** Kysy neuvoa harjoitustyön ohjaajalta, mikäli olet epävarma tietojen tietorakenteiden käyttöön liittyvissä kysymyksissä.

Harjoitustyössä tutustutaan linkitettyyn listarakenteeseen myös toteuttamalla *OmaLista*-luokassa opettajan antaman *Ooperoiva*-rajapinnan määrittelemät listaoperaatiot. Yksi rajapinnan operaatioista lisää alkion listalle siten, että alkio sijoituu kaikkien itseään pienempien tai yhtä suurien alkioden jälkeen ja ennen kaikkia itseään suurempia alkioita. Alkioita vertaillaan lisättäessä *Comparable*-rajapinnan *compareTo*-metodilla, jolloin lista järjestyy automaattisesti nousevaan järjestykseen. Voit halutessasi tehdä muitakin kuin rajapinnan määrittelemiä listaoperaatioita. Käytä toteuttamiasi *Ooperoiva*-rajapinnan operaatioita *Viesti*-luokan metodeissa (luku 3.4).

Operaatioiden tulee olla "puhtaita" listaoperaatioita, joita voitaisiin käyttää myös muissa tehtävissä. Näin *OmaLista*-luokassa ei saa esiintyä harjoitustyön luokkien tunnuksia. Erityisesti nimiä *Tiedosto*, *Kuva*, *Video* tai *Viesti* ei saa käyttää oman listaluokan operaatioissa.

Sijoita *OmaLista*-luokka omaa pakkaukseensa (luku 3.7).

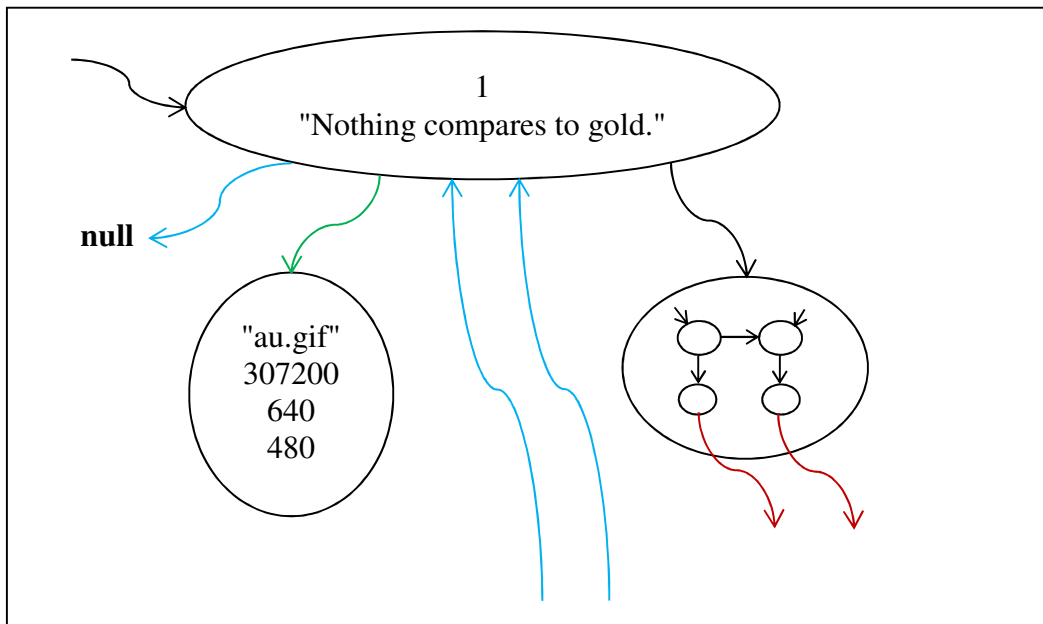
3.4. Viestit

Ohjelmassa tulee olla *Viesti*-luokka, jonka vastuulla on yksittäiseen viestiin liittyvien tietojen hallinnointi. Luokalla on attribuutit viestin tunnisteelle (**int**) ja merkkijonona ilmaisulle tekstille (*String*). Tunniste on nollaa suurempi kokonaisluku ja merkkijonossa on oltava vähintään yksi merkki.

Viesti voi vastata aiempaan viestiin. Tämä suhde mallinnetaan *Viesti*-tyyppinen attribuutin avulla, liittää viestiolion sen vastaamaan olioon. Attribuutin arvo on **null**, jos viesti on itsenäinen keskustelun avaus eikä siten vastaa toiseen viestiin. Luokan *Tiedosto*-tyyppinen attribuutti kiinnittää viestiin siihen mahdollisesti liittyvän tiedosto-olion. **Null**-arvo ilmaisee sen, että viestillä ei ole tiedostoa.

OmaLista-tyyppisen attribuutin tehtävänä on säilöä viitteet, jotka liittävät viestin siihen vastaaviin viesteihin. Vastausviestit ovat listalla tunnisteensa mukaisessa nousevassa järjestyksessä. Lista on tyhjä, kun vastauksia ei ole.

Kuvassa 16 on esitetty aiemman esimerkin (kuva 9) ensimmäisen viestiketjun ensimmäinen viestiolio osaolioineen ja viitteineen. Viesti ei vastaa aiempaan viestiin. Tästä syystä olion *Viesti*-tyyppinen viite (sininen) on **null**-arvoinen. Olio on liitetty *Tiedosto*-tyyppisen viitteen (vihreä) kautta *Kuva*-tyyppinen osaolio. Olion listalta on (punaiset) viitteet viestiin vastaaviin kahteen viestiin, joista on puolestaan viitteet (siniset) takaisin olioon.



Kuva 16: Viestiolio osaolioineen ja viitteineen. Olio on liitetty osina kuva ja lista. Listalla on viitteet kahteen vastaukseen, joista on paluuviihteet olioon. Olio itse ei vastaa aiempaan viestiin.

Kätke luokan attribuutit ja tee niille julkiset aksessorit. Leimaa aksessorit annotaatioilla, kuten luokkahierarkiassa (luku 3.2). Heitä tunnisteiden, tekstin ja listan asettavista metodeista *IllegalArgumentException*-poikkeus, jos parametri on virheellinen. Listaparametri ei saa olla **null**-arvoinen. Vastaus- ja tiedostoattribuuttien arvoksi voidaan asettaa **null**-arvo.

Tee luokalle ainakin neliparametrinen rakentaja, jonka parametrit ovat järjestyksessä luetellen viestin tunniste (**int**), teksti (*String*), viite vastattuun viestiin (*Viesti*) ja viite tiedosto-olioon (*Tiedosto*). Vastauksiin liittyvät viitteet säilövä *OmaLista*-olio luodaan rakentajassa ja liitetään siellä attribuuttiinsa. Heitä rakentajasta *IllegalArgumentException*-poikkeus, jos tunnisteeseen, tekstiin tai listaan liittyvä parametri on virheellinen.

Toteuta *Comparable*-rajapinnan *compareTo*-metodi *Viesti*-luokassa. Kiinnitä geneeriseksi tyyppiä *Viesti*. Olioita vertaillaan niiden tunnisteiden perusteella. Korvaa luokassa myös *Object*-luokan *equals*-metodi siten, että oliot katsotaan samoiksi, jos niiden tunnisteet samat. *CompareTo*- ja *equals*-metodeja tarvitaan uusissa listaoperaatioissa (luku 3.3).

Korvaa luokassa *Object*-luokan *toString*-metodi. Metodi palauttaa merkkijonon, jonka alussa on ristikkomerkki, viestin tunniste ja välilyönti. Alkua seuraa viestin merkkijono. Merkkijonon loppuosa koostuu välilyönnistä ja kaarisulkuparin sisään liitetystä tiedoston merkkijonoesityksestä (kuva 9).

Viesti-luokan on toteutettava *Komennettava*-rajapinta, jossa on ohjelman toteuttamiseen ja testaamiseen soveltuvia metodeja. ~~Kiinnitä geneeriseksi tyyppiä *Tiedosto* ja *OmaLista*.~~ Kiinnitä geneeriseksi tyyppiä *Viesti*.

Sijoita *Viesti*-luokka omaa pakkaukseensa (luku 3.7).

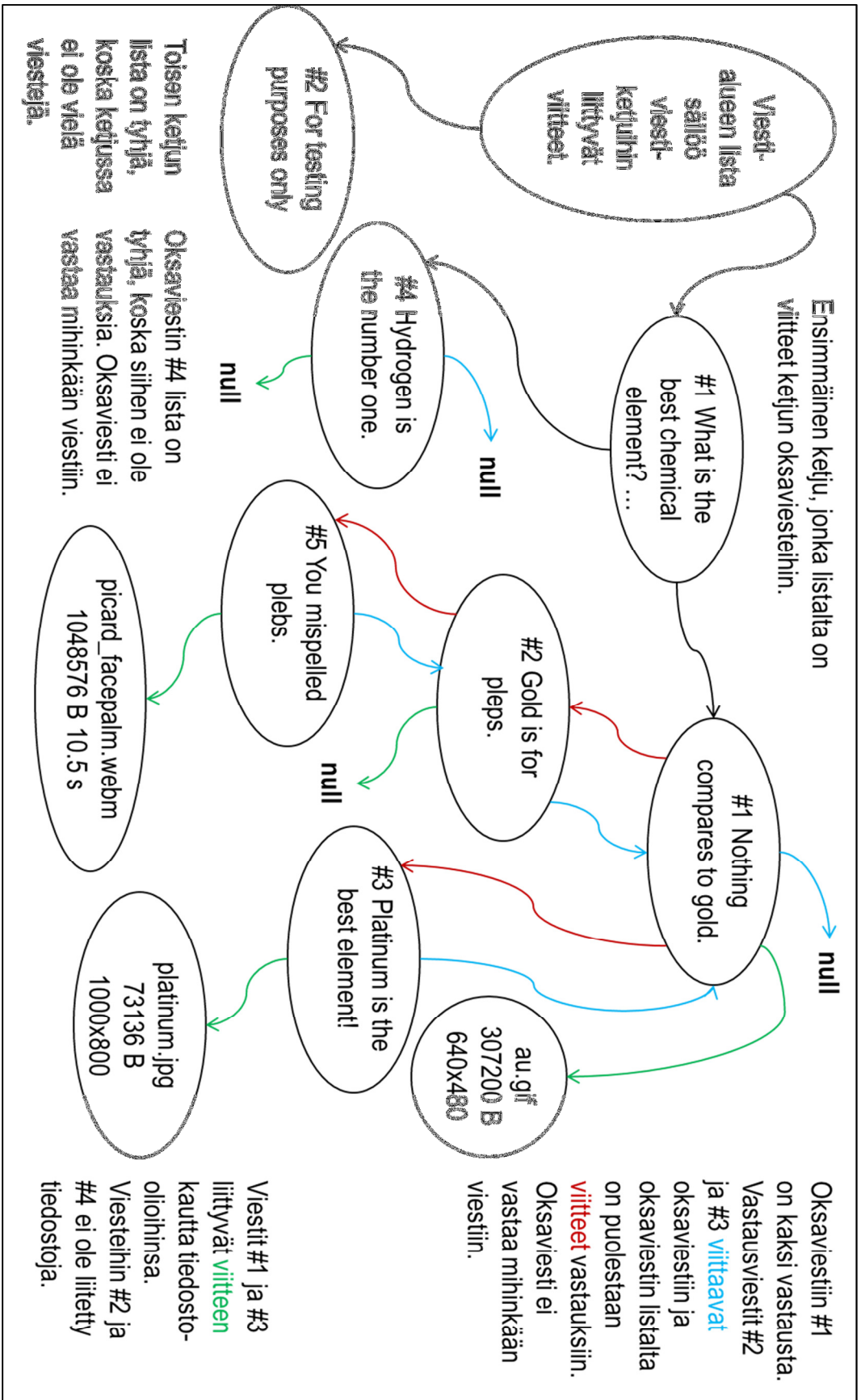
3.5. Ketjut ja viestialue

Viestit organisoidaan *Viesti*-luokan olioiden ketjuiksi siten, että kukin viesti, joka ei ole vastaus toiseen viestiin, muodostaa viestiketjuun “oksan”, johon kaikki kyseisen viestin vastaukset liitetään. Oksaan kuuluvat myös kaikki vastauksiin kirjoitetut vastaukset, vastausten vastauksiin kirjoitetut vastaukset ja niin edelleen.

Kuvassa 17 on esitetty alkuaineeseen liittyvän viestiketjun (kuva 9) rakenne *Viesti*-luokan olioina. Viesteihin liittyviä listaosioita ei ole piirretty tilasyistä näkyviin. Listan sisältämät viitteet (punaiset) lähtevät tästä syystä suoraan viestioliosista. Ketjussa on kaksi oksaa, joista ensimmäisessä kulta arvostetaan alkuaineista parhaaksi. Oksaviestiin liittyy kaksi vastausta, joista ensimmäiseen liittyy yksi vastaus. Toinen oksaviesti kannattaa määräsistä vetyä. Tähän viestiin ei ole vastauksia.

Kunkin viestiketjun oksaviestit ovat omalla listallaan tunnuksen määräämässä nousevassa järjestyksessä. Tälle listalle ei säilötä vastauksia, koska ne saavutetaan oksan kautta. Viestiketjut ovat puolestaan omalla listallaan tunnuksensa mukaisessa nousevassa järjestyksessä. Kaikki listaoliot luodaan *OmaLista*-luokasta.

Kuvasta 17 nähdään, että keskustelualueella on kaksi viestiketjua. Ensimmäisellä viestiketjulla on edellä käsitelty kaksi oksaa. Jälkimmäinen viestiketju on tyhjä. Siihen lisätään myöhemmin testiviestejä.



Kuva 17: Viestialue, kaksi viesti ketjua ja ketjujen viestit.

Harjoitustyössä on toiminto viestiketjun tulostamiseen puumuodossa (luku 2.8). Tulostus tapahtuu esijärjestyksessä (preorder), jolloin viesti tulostetaan ennen vastauksiaan. Oksaviesti vastauksineen on helpointa tulostaa rekursiivisesti. Alla on annettu kaksi algoritmia, joista käy ilmi tulostuksen keskeisin idea. Huomaa, että algoritmeista puuttuu muun muassa virheenkäsittely, tyyppimuunnoksia ja tulosteen tarkempi muotoilu. Algoritmit voi muuttaa paluuarvon avulla listauksen merkkijonona palauttavaan muotoon.

```
Algoritmi tulostaPuuna() {
    // Käydään läpi viestiketjun oksaviestit säilövä lista.
    i ← 0;
    while (i < viestit.koko()) {
        // Kutsutaan rekursiivista tulostusalgoritmia.
        tulostaPuuna(viestit.alkio(i), 0);
        i++;
    }
}

Algoritmi tulostaPuuna(Viesti viesti, int syvyys) {
    // Tulostetaan annetun syvyinen sisennys.
    ...
    // Tulostetaan viestin merkkijonoesitys.
    tulosta(viesti);
    // Asetetaan apuviite viestin vastaukset säilövään listaan.
    vastaukset ← viesti.vastaukset();
    // Tulostetaan vastaukset rekursiivisesti. Metodista palataan,
    // kun vastauslista on tyhjä.
    j ← 0;
    while (j < vastaukset.koko()) {
        tulostaPuuna(vastaukset.alkio(j), syvyys + TASONSYVYYS);
        j++;
    }
}
```

3.6. Tiedostot

Viesteihin liitettävissä tiedostoissa on kuva- ja videodatan asemasta yhdelle riville tallennettu merkkijono, joka koostuu luokan nimestä, välilyönnistä ja välilyönillä toisistaan erotetuista olion tietoista, jotka on annettu samassa järjestyksessä kuin merkkijonoesityksessä (luku 3.2). Esimerkiksi *au.gif*-tiedosto sisältää vain merkkijonon "Kuva 307200 640 480". Ohjelma luo tiedoston tietoja vastaavan olion, kun viestiin liitetään tiedosto antamalla tiedoston nimi. Tiedostojen sisällön oletetaan olevan aina kunnossa.

3.7. Koodin organisointi

Tee luokkahierarkian ja viestin luokkien lisäksi oma luokka käyttöliittymälle ja erillinen *main*-metodin sisältävä ajoluokka *Oope2018HT*. Viestiketjun ja keskustelualueen logiikka on suositeltavaa jakaa ketjusta ja alueesta vastaaviin luokkiin, vaikka näissä luokissa on hyvin samankaltaisia metodeja, koska ketjun ja alueen hallinnointi ovat selvästi erillisiä vastuita. Ohjelmaan voi lisätä näiden lisäksi muita omia luokkia.

Yksittäistä viestiketjua mallintavan luokan vastuulla on ensisijaisesti ketjun oksa-viestit sisältävän listan hallinnointi. Tässä luokassa tulisi olla siksi *OmaLista*-tyyppinen kätkeyty attribuutti ja viestiin läheisesti liittyviä metodeja. Muita ketju-luokkaan luontevasti sopivia piirteitä ovat ketjun tunniste ja aihe. Eräs mahdollisuus ketjun listamaisen esityksen (luku 2.9) tuottamiseen on pitää yllä ketjuluokassa oksalistan lisäksi järjestettyä listaa, jolla on viitteet ketjun kaikkiin viesteihin.

Alueesta vastaavalla luokalla on *OmaLista*-tyyppinen kätkeyty attribuutti, jolta on viitteet alueen viestiketjuille. Alueluokan tulee olla käyttöliittymästä erillinen. Alueluokan tehtävänä on toteuttaa ohjelman toiminnot yhteistyössä ketjuluokan ja viestiluokan kanssa. Joissain tapauksissa alueluokan metodi vain kutsuu toisen luokan metodia. Älä anna käyttäjän komentoa alueluokan metodille parametriksi sellaisenaan tai taulukoksi ositettuna, vaan osita komento erillisiksi arvoiksi ja tee tarvittavat tyypimuunnokset ennen metodin kutsua.

Kaikki käyttäjän antamat tiedot luetaan käyttöliittymässä ja käyttöliittymässä tehdään mahdollisimman suuri osa tulostuksista. *Viesti*-luokassa ja luokkahierarkian luokissa ei saa olla tulostuksia. Pääsilmut luonteva paikka on käyttöliittymän luokka. Pääsilmut kutsutaan komentoja vastaavia alueluokan metodeja joko suoraan tai apumetodien kautta.

Sijoita kaikki luokkasi *oope2018ht*-nimiseen pakkauksen alipakkauksiin. Pakkaukset ovat *oope2018ht*-hakemiston alihakemistoja. Ajoluokka ei kuulu pakkaukseen, vaan on pakkaushakemiston välittömässä ylihakemistossa, jossa on myös *fi.uta.csjola.oope.lista*-pakkaus sekä kuva- ja videotiedostot.

Tee *Tiedosto*-, *Kuva*- ja *Video*-luokille *oope2018ht.tiedostot*-niminen alipakkaus. *OmaLista*-luokka sijoitetaan *oope2018ht.omalista*-pakkaukseen ja *Viesti*-luokka *oope2018ht.viestit*-pakkaukseen, johon voi liittää myös ketju- ja alueluokat (kuva 18). Tee käyttöliittymälle oma alipakkaus. Voit tehdä halutessasi lisää alipakkauksia.

Harjoitustyössä käytetään *oope2018ht.apulaiset*-pakkaukseen sijoitettua *In*-luokkaa tietojen lukuun näppäimistöä. Tähän pakkaukseen on liitetty myös *@Getter*- ja *@Setter*-annotaatiot (luku 3.2) sekä *Komennettava*- ja *Ooperoiva*-rajapinnat (luvut 3.3 ja 3.4). Pakkausta ei saa muuttaa millään tavalla. Löydät *oope2018ht.apulaiset*-pakkauksen kurssisivujen *Harjoitustyö*-kohdasta.

Ohjelma kääntyy komentoikkunassa komennolla `javac Oope2018HT.java` ja on suoritettavissa komennolla `java Oope2018HT`, kun harjoitustyön koodi on organisoitu kuvassa 18 esitetyllä tavalla.

```
+---fi
|   +---uta
|       +---csjola
|           +---oope
+---Oope2018HT.java      +---lista
|                               |---AbstraktiLista.java
|                               |---LinkitettyLista.java
+---oope2018ht           |---Lista.java
|                               |---Solmu.java
|       +---tiedostot
|           |---Tiedosto.java
|           |---Kuva.java
|           |---Video.java
|       +---omalista
|           |---OmaLista.java
|       +---viestit
|           |---Viesti.java
|       ...
|       +---apulaiset
|           |---In.java
|           |---Getteri.java
|           |---Setteri.java
|           |---Komennettava.java
|           |---Ooperoiva.java
```

Kuva 18: Harjoitustyön oman pakkauksen rakenne ja valmiiden pakkausten paikat.

4. Palautus ja tarkistus

Harjoitustyössä on välipalautuspiste ennen lopullista palautusta. Välipalautuksen tavoitteena on saada kurssilaiset aloittamaan harjoitustyön teko ajoissa. Harjoitustyöstä täytyy palauttaa ~~keskiviikkoon 11.4.2018 klo 12.00~~ (keskipäivä) ~~klo 20.00~~ **perjantaihin 13.4. klo 16.00** mennessä Javalla toteutettu luokkahierarkia (luku 3.2), *OmaLista*-luokka (luku 3.3) ja *Viesti*-luokka (luku 3.4).

Valmis ohjelma sen dokumentaatio on palautettava viimeistään ~~perjantaina 27.4.2018 klo 12.00~~ (keskipäivä) **torstaina 3.5.2018 klo 23.55**.

Molemmat palautukset tehdään WETO-järjestelmään, joka tarkistaa automaattisesti ratkaisujen rakennetta ja toiminnallisuutta. WETO arvioi toiminnallisuutta pääosin vertailemalla mallivastauksen ja opiskelijoiden ratkaisujen tulosteita. Tästä syystä **edellä annettuja tulostemäärittelyjä on seurattava merkilleen**. Automaattinen vertailu vähentää rutiininomaista testaustyötä, jolloin opettajille jää enemmän aikaa mielekkäämpään työhön eli ohjelman rakenteen ja tyylin tutkimiseen. Opiskelijat hyötyvät tästä perusteellisempien kommenttien muodossa.

Automaattisessa tarkistuksessa käytettävät testit ja tarkemmat palautusohjeet julkaistaan kurssin verkkosivuilla.

Lisäaikaa työn tekoon voi saada muutaman päivän hyvästä syystä. Lisäajasta on sovittava ohjaajan kanssa ajoissa eli viimeistään päivää tai paria ennen palautuksen takarajaa. Lisäaika estää useimmissa tapauksissa hyvityspisteiden saamisen (luku 7). Harjoitustyön ohjaaja ei välttämättä ole tavattavissa henkilökohtaisesti lisäaikana.

Ennen palautusta on syytä varmistaa, että ohjelma toimii varmasti oikein viimeimpien muutosten jälkeen.

5. Dokumentointi

Harjoitustyöstä kirjoitetaan lopullisen palautuksen yhteydessä dokumentti, jossa tulee olla:

- Kansilehdellä tekijän nimi, opiskelijanumero ja sähköpostiosoite. Sivun keskellä tulisi olla suuremmalla fontilla dokumentin nimi. Kurssin kotisivuilla julkaistaan esimerkinomainen kansilehti.
- UML-luokkakaavio ja sen lyhyt sanallinen kuvaus. Itse tehtyjen luokkien luokkasymboleissa annetaan luokan kaikki vakiot, attribuutit ja metodit.
- Omia ajatuksia. Esimerkiksi: Oliko työ helppo, sopiva tai vaikea? Miksi koit työn helpoksi, sopivaksi tai vaikeaksi? Mitä uutta opit? Oliko työstä mitään hyötyä tekijälleen? Paljonko aikaa käytit työhön?

Dokumentin leipäteksti kirjoitetaan 12 pisteen fontilla ja yhdellä rivinvälillä. Valmiin tekstin lukeminen pariin otteeseen ei ole huonompi idea. Dokumentin kirjoitus tekstinkäsittelyohjelmalla sekä ohjelmasta löytyvän oikolukutoiminnon käyttäminen tekstin tarkistamiseen on myös suotavaa.

Luokkakaaviossa esiintyvien luokkien väliset suhteet mallinnetaan luokkien välisinä suhteina. Piirrä periytymis- ja toteuttamissuhteiden lisäksi näkyviin luokkien väliset **assosiaatiot**. Jos jossakin itse kirjoitetussa luokassa *A* on attribuutti, jonka tyyppi on toinen oma luokka *B*, niin tätä attribuuttia ei esitetä luokan *A* laatikossa, vaan *A*- ja *B*-luokkien symboleiden välisenä assosiaatioviivana. Anna kaikille assosiaatioille nimet, suunta sekä rooli ja kertautumiset.

Piirrä luokkakaavioon myös luokkien väliset riippuvuussuhteet. Riippuvuussuhde on assosiaatiota heikompi suhde luokkien ilmentymien välisen hetkellisen yhteistyön ilmaisemiseen. Jos oma luokka *C* hyödyntää toisen oman luokan *D* palveluja ilman attribuuttia, niin luokkalaatikoiden välille voi piirtää nuolen (varsi katkoviivaa) luokasta *C* luokkaan *D* riippuvuussuhteen merkiksi.

Kaavioiden piirto on helpompaa erityisesti UML-kaavioiden piirtoon tehtyjä ohjelmistoja käyttäen. Dia on ilmainen ohjelma, jolla voi piirtää vuo- ja ER-kaavioiden lisäksi myös UML-kaavioita. Jotkin ohjelmat osaavat muodostaa osan luokkakaaviosta automaattisesti lähdekoodin perusteella. Tällainen on UMLet-ohjelma, joista kerrotaan lisää kurssisivujen *Harjoitustyö | UML* -kohdassa. NetBeans- ja Eclipse-ohjelmistojen kaltaisiin kehitysympäristöihin on saatavilla takaisinmallinnuksen hallitsevia lisäosia.

Koodi dokumentoidaan lopullisessa palautuksessa kirjoittamalla lyhyet **Javadoc-kommentit** jokaiselle attribuutille, metodille ja luokalle. Rakentajia ja aksessoreita ei tarvitse kommentoida, mikäli ne ovat toiminnoiltaan tavanomaisia. Javadoc on Java-ympäristön tarjoama menetelmä koodin puoliautomaattiseen dokumentoimiseen. Javadoc-kommentoinnista annetaan erilliset ohjeet kurssin verkkosivujen *Harjoitustyö* | *Javadoc* -kohdassa. Javadoc-kommentit voi halutessaan kirjoittaa jo välipalautuksen yhteydessä.

Muista edelleen selittää **metodien sisällä** normaaliin tapaan metodien keskeiset ja vaikeaselkoiset osuudet **tavanomaisia kommentteja** käyttäen.

6. Ohjaus

Harjoitustyötä ohjataan myös ennen välipalautuspistettä. Ohjausten ajat ja paikat ilmoitetaan myöhemmin kurssin kotisivuilla. Ohjaukseen voi tulla omasta harjoitusryhmästä riippumatta: voit käydä ohjattavana aikana, joka sopii sinulle parhaiten. Luuppi tarjoaa ohjausta harjoitustyöhön koodauspajassaan. Vastuopettajaa voi käydä tapaamassa myös muina aikoina. Tapaamisaika kannattaa sopia etukäteen, jos haluaa varmistaa, että vastuopettaja on paikalla huoneessaan.

Ohjaukseen voi tuoda myös UML:llä tai muuten hahmotellun suunnitelman ohjelman rakenteesta ennen koodaamisen aloittamista.

Pienemmät kysymykset kannattaa esittää ohjaajille sähköpostitse. Harjoitustyön pääasiallinen sähköpostiohjaaja on mikroharjoitusryhmäsi vetäjä. Näet ryhmäsi WETO-järjestelmän *Students*-välilehdeltä. Kysy neuvoa sähköpostitse sopivimmaksi katsomaltasi opettajalta, jos et ole valinnut ryhmää.

Kysymyksiä ja niihin annettuja vastauksia kerätään kurssin verkkosivujen *Harjoitustyö*-kohtaan. Ennen kysymistä kannattaa siis tarkistaa kurssin verkkosivut – vastaus voi hyvinkin olla jo sivuilla. Monet ongelmat selviävät myös tehtävänantoa lukemalla ja tutustumalla verkkosivuilta löytyviin esimerkkiajoihin.

Kysyminen kannattaa erityisesti, kun on epävarma suuremman mittakaavan kysymyksissä. On pienempi vaiva selvittää epäselvä kohta ajoissa ohjaajan kanssa, kuin korjata valmista ohjelmaa jälkikäteen tehtävänantoa vastaavaksi.

7. Arvostelu

Arvostelu perustuu ensisijaisesti ohjelman oikeellisuuteen: ohjelman on toimittava testeissä tehtävänannossa määritellyllä tavalla. Harjoitustyö voidaan joko hylätä tai hyväksyä, jolloin työ arvostellaan asteikolla 0–4 pistettä. Pisteet lisätään harjoitushyvituspisteiden tapaan tenttipisteisiin vain, mikäli tentistä on saanut vähintään vaaditut 12 pistettä.

Harjoitustyöstä voi saada pisteitä vain, jos välipalautus läpäisee WETOn automaattisen testauksen ja työ on palautettu molemmilla palautuskerroilla ilman lisäaikaa. Aikavaatimuksen osalta voidaan joustaa äärimmäisen hyvästä syystä, jollaisia ovat esimerkiksi pitkä ulkomaan matka tai kokopäivätyöiden aiheuttamat ylitsepääsemättömät esteet.

Työn pisteet määräytyvät lopullisen palautuksen jälkeen tehtävissä testeissä. Testauksessa on julkinen ja salainen osuus. Huomaa, että salaisissa testeissä käytetään eri syötteitä kuin kotisivuilla annetuissa esimerkeissä. Ohjelman oikeellinen toiminta esimerkkien osalta ei siksi takaa salaisten testien läpäisyä.

Hylkäyksen perusteena voi olla tehtävänannon noudattamatta jättäminen, ohjelman toimimattomuus, harjoitustyön teossa kiellettyjen Javan ominaisuuksien käyttö, huono dokumentti, hyvän ohjelmointitavan noudattamatta jättäminen ja/tai plagiointi. Plagiointiin liittyy sanktio, joka koskee molempia opiskelijoita. Toiselta opiskelijalta tämän tietämättä kopioidun koodin käyttö johtaa kopioijan koko kurssisuorituksen hylkäämiseen.

Hylätyn työn voi palauttaa korjattuna pääsääntöisesti kolme kertaa. Välipisteessä hylätyn työn korjaukseen on aikaa muutama päivä. Lopullisen palautuksen korjausajaa on useimmiten viikko työn hylkäyksestä. Pisteytys perustuu aina ensimmäiseen palautukseen. Korjauskierroksen kautta ei voi korottaa pistemääräänsä.

Lähteet

1. Wikipedia-yhteisö, Internet forum,
https://en.wikipedia.org/wiki/Internet_forum (Luettu viimeksi 4.3.2018.)
2. J. Laurikkala, Lausekielinen ohjelmointi I -kurssin luentorunko, luku 14,
<https://coursepages.uta.fi/tiep1/syksy-2018/luennot/> (Luettu viimeksi 4.3.2018.)
3. J. Laurikkala, Lausekielinen ohjelmointi II -kurssin luentorunko, luku 7,
<https://coursepages.uta.fi/tiep5/syksy-2018/luennot/> (Luettu viimeksi 4.3.2018.)
4. J. Laurikkala, Olio-ohjelmoinnin perusteet -kurssin luentorunko, luku 4,
<https://coursepages.uta.fi/tiea2-1/kevat-2018/luennot/> (Luettu viimeksi 4.3.2018.)