1. Tehtävän määrittely

(Tässä on aika paljon turhanpäiväistä höpinää, esimerkiksi käyttötapauskuvailuja, jotka on kirjoitettu enimmäkseen suunnittelutyötä varten. Niitä kannattaa hyppiä yli).

Aihe: Kortisto

Käyttäjät: Sinänsä irrelevanttia. Tehdään yhdelle henkilölle, eli ei varauduta mitenkään useaan käyttäjään.

Ohjelman toiminta: Tallennetaan tietoja kirjoista (lehdistä, artikkeleista...). Käyttökertojen välillä tiedot tallennetaan tekstitiedostoon. Kortistoon voi lisätä kortteja, sieltä voi hakea kortteja hakusanoin ja niiden tietoja voi muuttaa.

Käyttötapauksia:

1. Kortin lisääminen kortistoon
2. Kortin etsiminen kortistosta
3. Kortiston selaaminen
4. Kortin tietojen muuttaminen
5. Kortin tuhoaminen
6. Kortiston tallentaminen (tekstitiedostoon)
7. Kortiston lataaminen (tekstitiedostosta).
8. Aakkostaminen (eri kenttien mukaan?)

Korttiin tallennettavia tietoja:

1.	Kirjan nimi	id	
2.	Kirjan tekijä	id	Jos nämä ovat samat kuin aiemmassa, varoitetaan
3.	Sijainti		
4.	Vuosi		
5.	ISBN	ID	Jos tämä on sama kuin aiemmassa, varoitetaan (estetään?)
6.	Muut tekijät		
7.	Asiasanat		
8.	Muuta	(Tämä kenttä ei sisälly oletushakuun).	

(Tallennetaan vain oleellisena pidettyjä tietoja. Kustantaja yms. ovat kuitenkin aika samantekeviä).

Muuta: Mahdollisuuksien mukaan toteutus pidetään avoimena muutoksille tai yleisemmälle käyttötavalle**.**

2. Käyttötapaukset

Tässä on jätetty kirjoittamatta sellaiset itsestäänselvyydet, joista on vain haittaa.

Kortin lisääminen kortistoon.

Esiehto: jokin kortisto on ohjelman käytössä (tyhjä tai epätyhjä).
Jälkiehto: Kortistossa on kortti käyttäjän antamalla sisällöllä.

1. Käyttäjä klikkaa “lisää kortti” (TAI: valitsee “haku”-välilehden...)
2. Käyttäjälle avautuu ikkuna. (...jossa on kentät valmiina
3. Käyttäjä täyttää vaaditut kentät.
4. Painaa OK.
5. Järjestelmä tarkistaa, onko kortti jo kortistossa (varoitus, jos näyttää olevan)
6. Jos kortti on OK, se lisätään kortistoon.

Poikkeuksellinen toiminta: Tarpeeksi lisättävän kaltainen kortti on jo kortistossa. Käyttäjä ei täytä mitään tekstikenttää.

Kortin etsiminen kortistosta hakusanalla:
Esiehto: jokin kortisto (tyhjä tai epätyhjä) on ohjelman käytössä.
Jälkiehto: käyttäjälle on näkyvillä lista hakukriteerit täyttävistä korteista.

1. Käyttäjä klikkaa haku
2. valitsee ja täyttää hakusanakentän(t).
3. Klikkaa OK.
4. Ohjelma näyttää korttien nimet, jotka täyttävät hakuehdot. (Klikkaamalla voi tarkastella korttia)

Poikkeuksellinen toiminta: Hakusanaa ei ole kirjoitettu, tai hakukenttää ei ole valittu.

Kortiston selaaminen
Tavoite: katsella kortiston sisältöä ilman erikoistunutta hakukriteeriä
Esiehto: jokin kortisto on ohjelman käytössä (tyhjä tai epätyhjä).
Jälkiehtio: Käyttäjän näkyvillä on kortiston nimikkeet tai osa niistä.

1. Käyttäjä klikkaa “selaa”
2. Ohjelma näyttää sopivan määrän korttien nimikkeitä kortiston alusta.
3. Halutessaan käyttäjä klikkaa “seuraava”, ja ohjelma näyttää lisää kortteja.

Aakkostaminen
Käyttäjä: Ohjelma aina, kun kortti on lisätty?
Tavoite: laittaa kortisto järjestykseen, ehkä valitun hakusanan perusteella
Esiehto: ohjelman käytössä on kortisto
Jälkiehto: Kortiston järjestys on halutunlainen

1. Käyttäjä klikkaa “aakkosta”. (Klikkaa sopivaa akkostuskriteeriä)
2. (Kone kysyy, minkä kentän mukaan)

Poikkeuksellinen toiminta: kortisto on tyhjä

Kortin tietojen muuttaminen
Esiehto: Ohjelmalla on käytössään kortisto, jossa on kortti, jota halutaan muuttaa. Jokin osa kortista on näkyvillä: nimeke tai koko kortti.

Jälkiehto: kortin sisältö vastaa käyttäjän antamia uusia tietoja.

1. Käyttäjä klikkaa muuta
2. Näytölle ilmestyy kortti, jonka tekstikenttää klikkaamalla käyttäjä voi muuttaa kyseistä kenttää.
3. Käyttäjä muuttaa halumiaan kenttiä
4. Käyttäjä lopettaa muokkauksen.

Poikkeuksellinen toiminta: Kortin tiedot ovat muokkaamisen jälkeen tyhjät. Kortin tiedot ovat muokkaamisen jälkeen liian samankaltaiset kuin toisen.

Kortin tuhoaminen
Esiehto: ohjelman käytössä on kortisto, joka sisältää tuhottavan kortin.
Jälkiehto: kortisto ei enää sisällä tuhottua korttia.

1. Käyttäjä valitsee (joko kortista tai listasta) “tuhoa”
2. Ohjelma kysyy “Oletko varma”
3. Käyttäjä klikkaa “OK”.
4. Kortti poistetaan muistista.

Kortiston tallentaminen
Esiehto: Ohjelman muistissa on kortisto
Jälkiehto: kovalevyllä on tekstitiedosto, josta kortisto voidaan ladata uudelleen ohjelman muistiin.

1. Käyttäjä klikkaa “Tallenna”
2. Ohjelma kysyy tallennuspaikkaa ja tiedoston nimeä
3. Käyttäjä valitsee ja painaa OK.
4. Ohjelma kysyy tarvittaessa, korvataanko vanha samanniminen tiedosto (vain “save as”:ssä!).

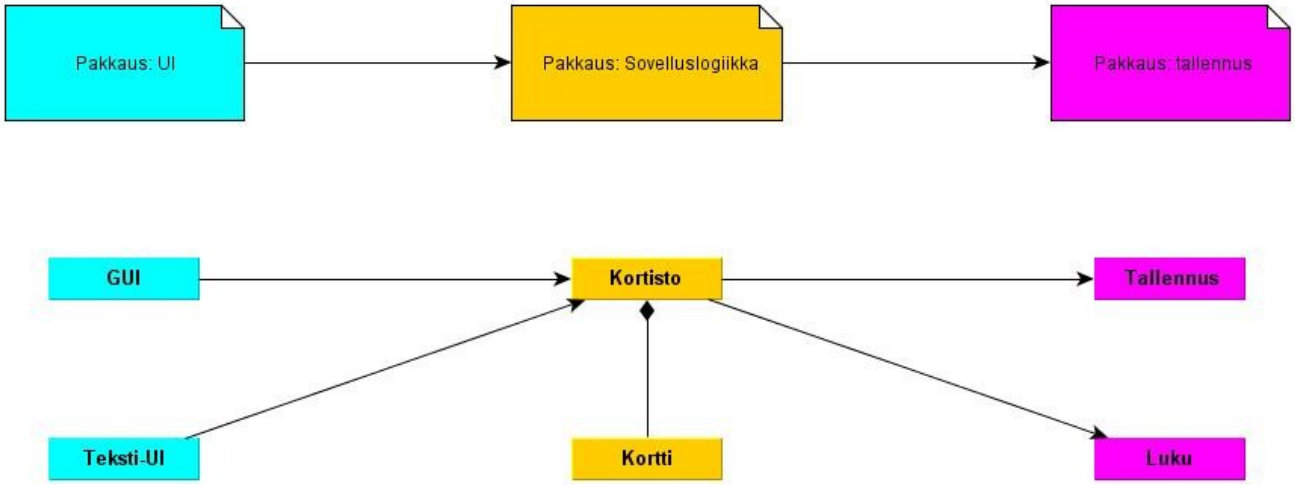
Poikkeuksellinen toiminta: Kortisto on tyhjä. Käyttäjä antaa epäkelvon tiedostonnimen.

Kortiston lataaminen
Esiehto: Kovalevyllä on tekstitiedosto, josta kortistoa yritetään ladata.
Jälkiehto: Tiedoston kortisto on ohjelman muistissa ja käyttäjän käytettävissä

1. Käyttäjä klikkaa “avaa kortisto”
2. Kone kysyy, mistä tiedostosta kortisto yritetään ladata
3. Käyttäjä valitsee
4. Kone lataa kortiston muistiinsa

Poikkeuksellinen toiminta: Käyttäjä valitsee epäkelvon tiedoston.

Ensimmäinen luokkakaavio:



3. Toteutus

Toteutus ja käyttövaatimukset muuttuivat iteraatioiden myötä. Samalla ohjelman rakenne monimutkaistui osiltaan, ja olisi saanut monimutkaistua toisilta osiltaan. Joistakin luokista tuli ehkä liian pitkiä.

Ensimmäinen iteraatio:

Korttien luominen, kortiston lataaminen ja tallentaminen toteutettu.

Lisäksi kokeiltu, miten nopeasti lataaminen tapahtuu. Tuhannella kortilla ~130 ms ja kymmenellä tuhannella ~330 ms. Kokeiltu myös korteista hakua siten, että hakusanaa etsitään kustakin kortin tekstikentästä. Tuhannella kortilla tähän kului ~8 ms ja kymmenellä tuhannella ~18 ms. Etsinnässä, jossa jokainen kortin tekstikenttä muutettiin ensin pieneksi, kului ~50 ms/10 000 korttia.

Toteutus lienee siis käyttötarkoitukseen tarpeeksi tehokas.

Kysymyksiä: onko parempi liikutella operaation onnistumisen tietoa booleaneina vai exceptioneina?

Toinen iteraatio:

Kortistosta etsiminen, lajittelu valitun kentän perusteella, kortin tietojen muuttaminen ja tuhoaminen.

Testasin haun nopeutta hakemalla yhtä kirjainta ensin kortin ensimmäisestä ja sitten viimeisestä kentästä (enemmän vertailuja). Nopeudet olivat ~40 ms ja ~60 ms. Lajittelussa käytetty comparator ei noudata equals-suositusta. Tarvitseeko sen oikeasti? Pitäisikö tekstikentät trimmata ennen lisäämistä kortistoon.

Kolmas iteraatio

Lisätty tekstikäyttöliittymä ja rekisteriin mahdollisuus valita merkitä kortteja valituiksi.
Tekstikäyttöliittymä on jokseenkin kevloton, mutta se toimii ainakin tyypillisissä tapauksissa, joten lisäsin sen main-branchiin. Tarkoitus on nyt lisätä dokumentaatiota ja testejä, ja usemapi haara gitillä lienee hankalaa. Mylhemmin koko tekstikäyttöliittymä voidaan korvata uudella tarvittaessa ilman, että siitä syntyisi ongelmia muussa ohjelmassa.

Neljäs iteraatio

Lisätään graafinen käyttöliittymä. Toimivaa ei ole saatu aikaiseksi, joten pyritään toteuttamana se seuraavalla tavalla:
Käyttöliittymän ja sovelluslogiikan välissä on välittäjäpakkaus, joka sisältää rajapinnat käyttöliittymälle ja sovelluslogiikalle sekä näiden kautta operoivan ohjausolion.

Jos halutaan pitää mukana mahdollisuus tietokannan käytöstä, ei riitä että tiedostojen lukeminen ja kirjoittaminen ovat rajapinnan takana, sillä jos varsinainen toiminta (etsiminen jne) eivät ole, ei tietokannoista saada juuri mitään hyötyä lisää.

Viides iteraatio

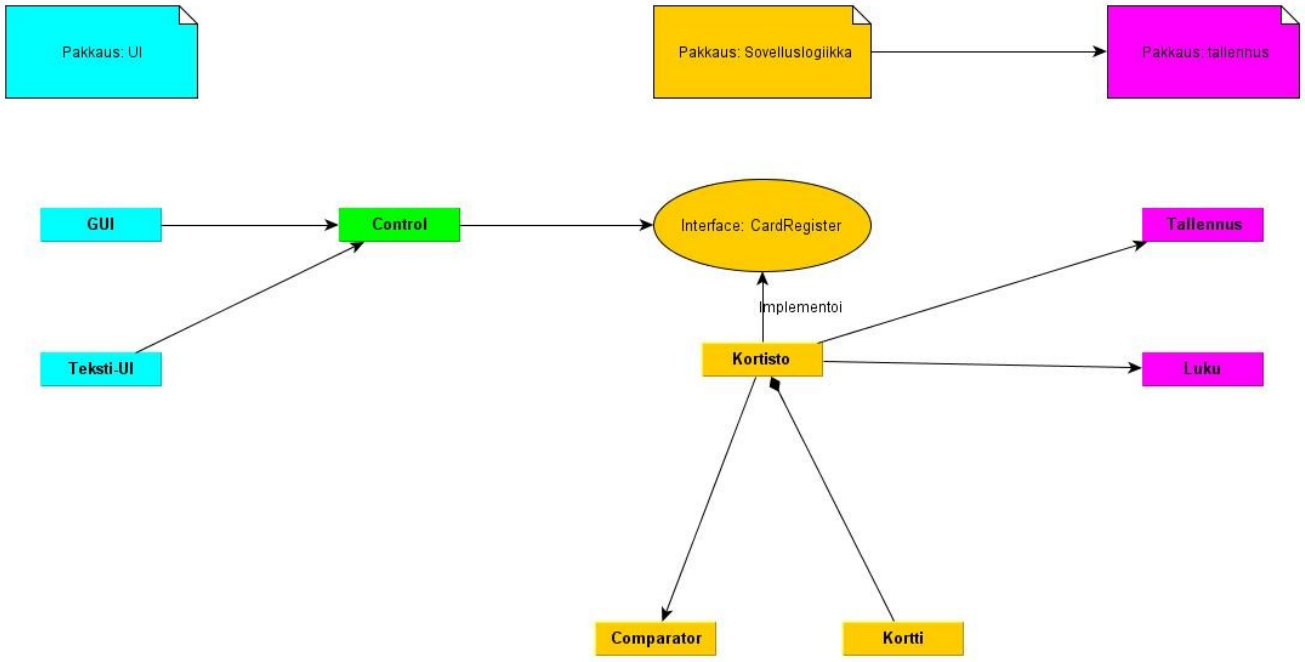
Uusi käyttöliittymä. Laitetaan se ensin jotenkuten toimivaan kuntoon. Nyt käyttöliittymän ja sovelluslogiikan välissä on Control-olio, jolle yritetään antaa minimaaliset tiedot kummastakin.

Harkitsin Controlin tekemistä Observeriksi ja sovelluslogiikkaa Observableksi, mutta jätän sen myöhemmäksi mahdollisuudeksi, koska se vattii kikkailua, ja toisekseen, sovelluslogiikassa ei tapahdu mitään ilman käyttäjän initaatiota. Control-olio siis tietää, milloin jokin muutos tapahtuu, ja voi reagoida sen mukaisesti.

Gui-komponenteissa on jonkin verran toistoa, jota voisi yrittää karsia lisäämällä niihin periytymistä. Mielestäni on kuitenkin parempi tehdä perityminen vasta, kun kaikki komponentit ovat valmiit, jolloin on paremmin selvillä, mitä yliluokilta edellytetään. Teen komponentit katseenkestäviksi vasta, kun kaikki toiminallisuus on valmista.

Toinen luokkakaavio:

Tästä on jätetty paljon turhaa pois, kuten GUI:n eri komponentit, jotka ovat aika itsetäänselviä, ja joiden lisääminen ei selvennä rakennetta yhtään. Teksitkäyttöliittymä tällaisessa ohjelmassa on jokseenkin turha, sillä nimenomaan käyttöliittymän graafisuus tekee sovelluksesta edes jollakin tavalla hyödyllisen. Comparator-luokka on olemassa vain tekstikäyttöliittymää varten, joten käytännössä se on turha.



Kuudes iteraatio

Käyttöliittymän sisäinen toiminta on nyt kunnossa, mutta se ei vielä täysin kommunikoi sovelluslogiikan kanssa.

TODO:

1. Tällä hetkellä “muokkaa” ja “tuhoa” saavat aikaiseksi vain tekstiä konsoliin. **Korjattu.**
2. Ohjeen sisältö on siansaksaa. **Korjattu.**
3. Sovelluslogiikan pitää kysyä varmistusta liian samankaltaisne kortin lisäämiseksi (esimerkiksi sama tekijä ja sama nimi). **Korjattu.**
4. Pitää miettiä, miten sovelluslogiikan equals metodi toteutetaan. **Ei tarvitse enää.**
5. Pitää lisätä varoituksia kriittisten toimintojen epäonnistumisesta (lataus, tallennus). **Lisätty.**
6. Pitää tarkistaa pop-uppien käyttäjäystävällisyys ja etsiä keinoa tehdä esimerkiksi etsimistoiminnosta alas putoava valikko **Jätetty sikseen ajan loppumisen takia.**
7. Pitää siivota koodia kautta linjan. **Tehty ajan sallimissa rajoissa.**
8. Pitää tehdä uusi toimiva tekstikäyttöliittymä.
9. Pitää vähentää toistoa abstraktion avulla. **Jätetty sikseen ajan loppumisen takia.**
10. 10. GUIN tulee reagoida poikkeuksiin. **Lisätty.**

Seitsemäs iteraatio

Ohjelma toimii moitteettomassti alkuperäisen suunnitelman mukaisesti. Siihen on tosin tehty pieniä lisäyksiä: kontekstimenu korttitaulukkoon, kortin sisällön näkee pitämällä hiiren osoitinta sen päällä jne.

Kahdeksas iteraatio

Kortiston tallennus ja lukeminen tiedostosta muutettu siten, että kortin kentissä voi olla rivinvaihtoja, mutta ei erästä epätodennäköistä merkkijonoa (”>#>#>#:::”). Tämä on mielestäni parempi vaihtoehto, sillä jos kortistoa käytetään esimerkiksi tutkielman tekemisen apuvälineenä, muistiinpanojen osuudesta voi tulla suurikin, ja rivinvaihdot ovat paitsi todennäköisiä (jos käyttäjä ei tiedä rajoitteesta), myös toivottavia käyttökelpoisuuden kannalta.

Kirjoitettu myös JavaDocit loppuun kaikkiin luokkiin ja siivottu koodia. Koodi on lopullisessa muodossa, paitsi jos virheitä ilmenee tai jää aikaa sen paranteluun.

Yhdeksäs iteraatio

Kirjoitettu testejä uusiksi sekä testailtu ohjelmaa. Käyttöliittymää on testattu manuaalisesti siitä asti kuin sellainen on ollut olemassa, joten hiirenliikutusrobotin luominen tähän tarkoitukseen tuskin on järkevää.

Kymmenes iteraatio

Kirjoitettu kokonaan uusi tekstikäyttöliittymä. Tehty paljon pieniä korjauksia.

4. Loppumietteitä

Lopulliseksi jääneessä versiossa olisi vielä paljon parantelemisen varaa, mutta aika ei riitä suurien muutosten tekemiseen, ja jotkin niistä olisivat ehkä liian suuria harjoitustyöhön. Ainakin seuraavia muutokset olisivat tehneet lopputuloksesta paremman:

1. Käyttöliittymien kommunikointi on kovakoodattu, olisi ollut parempi laittaa viestit omaan tiedostoonsa.
2. Monet GUI:n ikkunat toistavat samaa kaavaa (kortin lisäys, editointi ja haku). Olisi ehkä ollut parempi tehdä yksi perusikkuna, josta nämä olisivat perytyneet. Huomasin tämän kuitenkin vasta aika myöhään. Tätä olisi voinut pitää hyvänä esimerkkinä, kuinka toteutusvaiheessa pitää muuttaa ohjelman rakennetta.
3. GUI:ssa on liikaa pop-up ikkunoita. Alkuperäinen ajatukseni oli muuttaa myöhemmin ainakin hakuikkuna drop-down tyyliseksi. En kuitenkaan tiennyt, miten se olisi käytännössä pitänyt tehdä. Että tavallinen korttinäkymä on pop-up on kuitenkin perusteltavissa, sillä käyttäjä saattaa haluta pitää useita kortteja näkyvillä yhtäaikaa.
4. Tooltip korttitaulukossa on hieman ärsyttävä. Olisi pitänyt keksiä keino, jolla se ei muutu kokoajan uuteen korttiin siirryttäessä, vaan vasta pienen tauon jälkeen.
5. Joistakin asioista olisi pitänyt tehdä valinnaisia, kuten tooltipistä tai eri kenttien näyttämisestä korttitaulukossa.
6. Kommunikoinnin UI:n ja sovelluslogiikan välillä olisi voinut hoitaa Observer-Observable periaatteen mukaan. Se olisi ollut ehkä yleisempi ja parempi. Tässä tapauksessa tosin kaikki tapahtumat kortistossa ovat käyttäjän aikaansaamia, joten käytetty tapa riittänee.

7. Useampia asioita olisi voinut piilottaa rajapintojen taa. Esimerkiksi GUI:lle olisi hyvin voinut olla oma rajapintansa, jota Control käyttää.