Kasino-projektin dokumentti

Henkilötiedot

Kasino-projektin dokumentti, Joni Jäkälä, 61479S, 6.5.2016.

Yleiskuvaus

Toteutetaan kasino-korttipeli (pakka-kasino) tekstipohjaisena versiona. Pelissä kerätään pisteitä, jotka lasketaan aina jokaisen pelikierroksen lopussa. Peli jatkuu, kunnes joku pelaajista saavuttaa 16 pistettä. Jokaisen pelikierroksen alussa pakka sekoitetaan ja jakaja jakaa jokaiselle pelaajalle 4 korttia, jotka eivät näy muille sekä 4 korttia pöytään, jotka näkyvät kaikille. Loput kortit jätetään pöydälle pinoon ylösalaisin. Jakajasta seuraava aloittaa pelaamisen. Seuraavalla pelikierroksella hän on jakaja. Jakaja jakaa kunkin kierroksen alussa pöytään 4 korttia, kunnes pakka on tyhjä.

Omalla vuorollaan pelaaja voi kerrallaan käyttää jonkin kädessään olevista korteista: joko ottaa sillä pöydästä kortteja tai laittaa kortin pöytään. Jos pelaaja ei voi ottaa mitään pöydästä, täytyy hänen laittaa jokin korteistaan pöytään. Jos pelaaja ottaa pöydästä kortteja, hän kerää ne itselle pinoon. Pinon sisällöstä lasketaan korttien loputtua pisteet. Pöydässä olevien korttien määrä voi vaihdella vapaasti. Jos joku ottaa kaikki kortit, täytyy seuraavan laittaa jokin korteistaan tyhjään pöytään. Aina käytettyään kortin, pelaaja ottaa käteensä pakasta uuden, niin että kädessä on aina 4 korttia. Kun pöydällä oleva pakka loppuu, ei oteta pakasta kortteja enää lisää, vaan pelataan niin kauan kunnes kenelläkään ei ole kortteja kädessä.

Kortilla voi ottaa pöydästä yhden tai useampia samanarvoisia kortteja ja kortteja, joiden summa on yhtä suuri, kuin kortin jolla kortit pöydästä otetaan. Jos joku saa kerralla pöydästä kaikki kortit, hän saa ns. mökin, joka merkitään muistiin. Pelissä on muutama kortti, joiden arvo kädessä on arvokkaampi kuin pöydässä, Ässät: kädessä 14, pöydässä 1; Ruutu-10: kädessä 16, pöydässä 10; Pata-2: kädessä 15, pöydässä 2.

Kun kaikilta loppuvat kädestä kortit, saa viimeksi pöydästä kortteja ottanut loput pöydässä olevat kortit. Tämän jälkeen lasketaan pisteet ja lisätään ne entisiin pisteisiin. Seuraavista asioista saa pisteitä: Jokaisesta mökistä saa yhden pisteen. Jokaisesta ässästä saa yhden pisteen. Eniten kortteja saanut saa yhden pisteen. Eniten patoja saanut saa 2 pistettä. Ruutu-10 kortin omistaja saa 2 pistettä sekä lisäksi Pata-2 kortin omistaja saa pisteen

Tässä projektissa suoritettiin keskivaikea vaatimustaso johon kuuluvat: merkkipohjainen käyttöliittymä, 2-4 pelaajaa; täydellisesti toimiva pöydästä otettujen korttien tarkistusalgoritmi.

Käyttöohje

Ohjelma käynnistetään ajamalla test2-metodi. Tämän jälkeen valitaan kuinka monta pelaajaa peliin osallistuu. Ohjelmassa voi valita pelaajamäärän väliltä 2-4. Seuraavaksi pelaajille annetaan nimet. Pelaajalla on vuorollaan käytettävänä neljä komentoa (liite1.). Komennot ovat h, k, a ja s, jotka viittaavat seuraaviin sanoihin h = hylkää, k = kaappaa. a = auta ja s = säännöt. Jos pelaaja valitsee h (h = hylkää), tulee hänen valita seuraavaksi käsikorteistaan pöydälle hylättävä kortti niin että ensiksi annetaan kortin arvo isolla

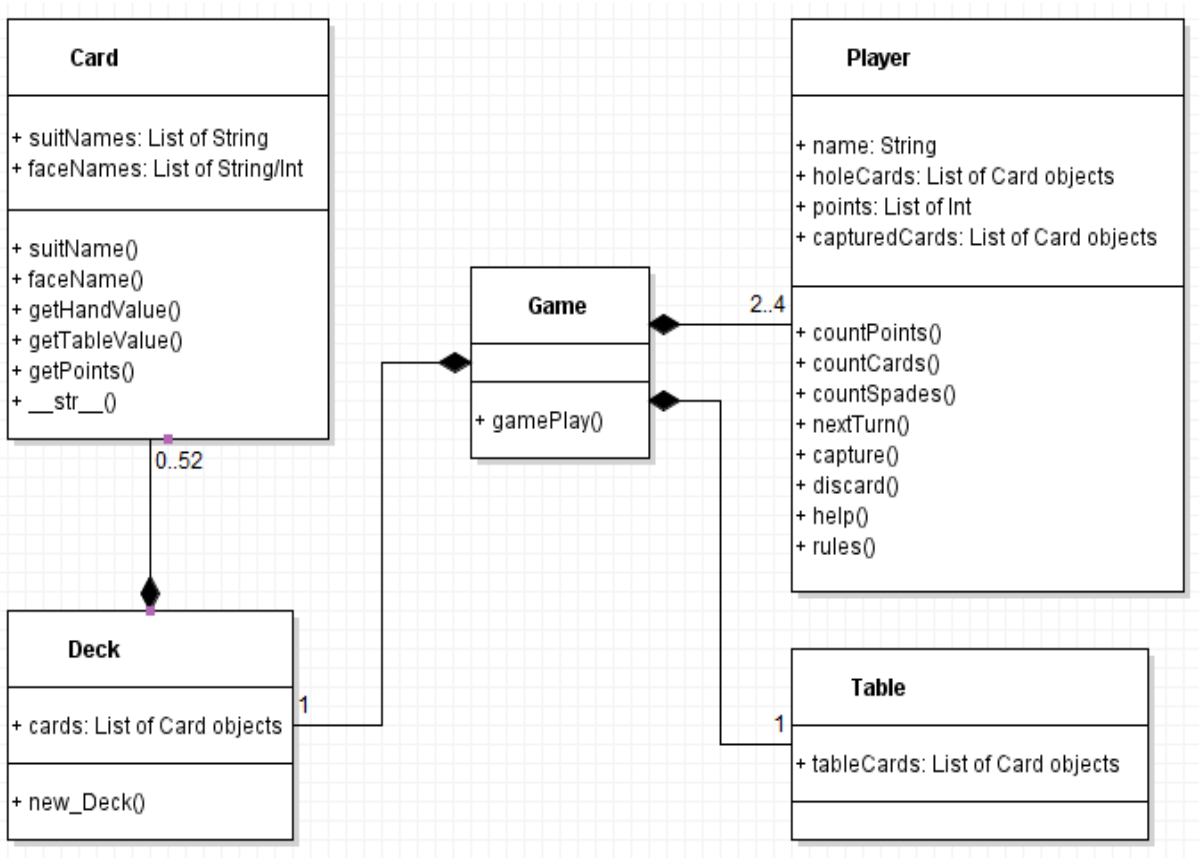
kirjoitettuna ja kortin maa pienellä yhteen kirjoitettuna (esim. Ad tai 3h). Komennon k (k = kaappaa) jälkeen pelaajan tulee antaa yksi käsikortti edellä mainitussa muodossa, jolla hän voi kaapata pöytäkortteja. Pöytäkortit tulee antaa ohjelmalle yksittäin. Pelkän enterin painaminen annettujen korttien jälkeen tarkoittaa, että ei haluta enää syöttää kaapattavia kortteja. Komennolla a saadaan hieman apua vuoron kulkuun. Komennolla s ruutuun tulee pelin säännöt.

Ohjelman rakenne

Ohjelman toteutuksessa luokat ovat jaettu viiteen ryhmään: Card, Deck, Game, Player ja Table. Card-luokassa luodaan lista korttien maista ja arvoista. SuitName ja faceName-metodit palauttavat listojen arvot ja __str__-methodi palauttaa arvon ihmiselle helposti luettavaan mutoon (esim. As tai 3h). Lisäksi getHandValue, getTableValue ja getPoints-methodit palauttavat kortin käsiarvon, pöytäarvon ja pisteet.

Luokka Deck on korttipakan toimintoja varten. Se sisältää listan, joka sisältää Card-luokkia. Luokalla on methodi new_deck, jossa korttipakka muodostetaan. Game-luokassa tapahtuu itse pelin eteneminen, joka suoritetaan gameplay-metodissa.

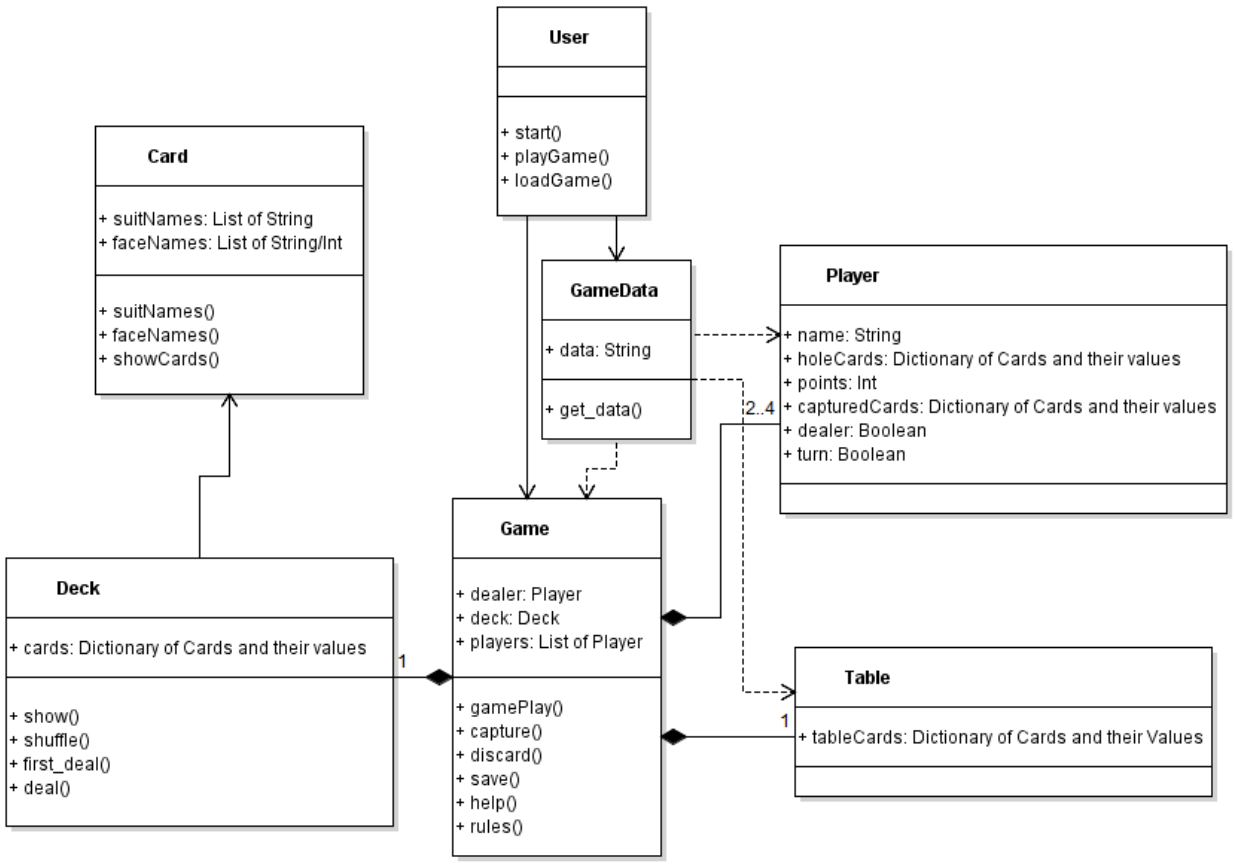
Table-luokka sisältää tiedot pöytäkorteista. Player-luokka sisältää pelaajasta keskeisimmät tiedot nimen, käsikortit, pisteet, kaapatut kortit. Player-luokassa käytetään lisäksi seuraavia metodeja: countPoints-, countCards-, countSpades-, nextTurn-, capture-, discard-, help_ - ja rules-metodit. Kolme ensiksi mainittua laskevat pelaajan pisteet, kaapatut kortit sekä niiden sisältämät padat. NextTurn-metodi suorittaa pelaajan vuoron. Capture-metodin avulla pelaaja voi kaapata kortteja pöydästä omaa käsikorttia käyttämällä. Discard-metodilla laitetaan yksi kortti pöydälle. Help_-metodi antaa hieman apua vuoron kulkuun. Rules-metodin avulla pelaaja voi kerrata pelin säännöt.



Kuva1. UML-kaavio toteutuneesta luokkajaosta

Seuraavaksi esitetään vaihtoehtoinen toteutustapa. Tämä toteutustapa oli alkuperäinen suunnitelma. Ohjelman toteutuksessa luokat ovat jaettu seitsemään ryhmään: Game, User, Card, Deck, GameData, Player, Table. Game-luokassa tapahtuu itse pelin eteneminen, joka suoritetaan gamePlay-metodissa. Luokassa käytetään lisäksi seuraavia metodeja: capture-, discard-, save-, help- ja rules-metodit. Capture-metodin avulla pelaaja voi kaapata kortteja pöydästä omia käsikortteja käyttämällä. Discard-metodilla laitetaan yksi kortti pöydälle. Save-metodilla pelitilanne voidaan tallentaa teksitiedostoksi. Help metodilla voidaan saada lisätietoa komennoista ja kuinka niitä käytetään. Rules-metodin avulla pelaaja voi kerrata pelin säännöt.

User-luokka sisältää metodit, joita käytetään ennen pelin alkua. Siinä on kolme metodia start, playGameja ja loadGame, joiden avulla käyttäjä joko ladata aiemmin tallennetun pelin tai aloittaa kokonaan uuden pelin. Card-luokassa luodaan lista korttien maista ja arvoista. SuitNames- ja faceNames-metodit palauttavat listojen arvot. Luokka Deck on korttipakan toimintoja varten. Show-metodi palauttaa pakan arvot, shuffle-metodi sekoittaa pakan satunnaiseen järjestykseen. First_deal-metodi tekee pelin ensimmäisen jaon. Deal-metodit tekee loput jaot. Table-luokka sisältää tiedot pöytäkorteista. Player-luokka sisältää pelaajasta keskeisimmät tiedot nimen, käsikortit, pisteet, kaapatut kortit sekä tiedon onko pelaaja jakaja. GameData-luokka sisältää tiedot tallennetuista peleistä. Get_data-metodilla haetaan keskeisimmät tiedot tallennettavaan tiedostoon.



Kuva2. UML-kaavio suunnitellusta luokkajaosta

Koska projektissa ei toteutettu lataus/tallennus-vaihtoehtoa, toteutuneesta projektista jäivät pois luokat User ja GameData. Korttipakka toteutettiin lista-muodossa, koska se sopi kätevämmin valittuun toteutustapaan, kuinka korttien pöytä- ja käsiarvot sekä pisteet lopulta saadaan haettua korteista. Nämä haettiin vertaamalla niitä korttien string-muotoiseen esitykseen ja siksi sanakirjaan ei tarvittu erikseen merkitä kyseisiä arvoja. Deck-luokassa ei tarvinnut käyttää erikseen muita metodeja kuin metodi pakan luomiseen, vaan gamePlay-metodissa esiintyvät suoraa nämä toiminnot. GamePlay-metodi olisi tullut yllä mainitulla toteutuksella todella pitkäksi, joten oli tarpeellista siirtää osa gamePlay-metodin koodista erillisiin metodeihin Player-luokkaan.

Algoritmit

Seuraavissa kappaleissa esitellään projektin kolme monimutkaisinta algoritmia, joilla toteutettiin korttien kaappaaminen (liiite1.), vuorojen vaihtuminen (liite2.) ja lopullinen pisteiden lasku (liite2.). Monimutkaisin algoritmi tässä projektissa oli toteuttaa korttien kaappaaminen oikein. Kortin kaappauksessa ensin katsottiin, että kaappaava sekä kaapattavat kortit ovat käsikorteissa (holeCards) tai pöytäkorteissa (tableCards). Korteista tehdään lista. Listan pituudesta riippuen kortteja siirretään pöytäkorteista kaapattuihin kortteihin (capturedCards). Korteista katsotaan myös ovatko pöytäkorttien yhteenlasketut pöytäarvot(*sumTableValue += lista[koko-1].getTableValue*()) jaollisia käsikortin käsiarvolla (*if sumTableValue % (lista[0].getHandValue())==0:)*. Tämä algoritmia korjattiin alkuperäisestä lyhemmäksi tekemällä lista kaikista korteista ja vertaamalla listan pituutta siihen kuinka kortteja siirretään eri luokkien ja listojen välillä. Alkuperäisessä suunnitelmassa jokaiselle eri kaapatulle korttimäärälle pöytäkorteista

tehtiin oma metodi. Tämä oli kuitenkin melko työlästä ja lisäksi koodista olisi tullut todella pitkä.

Vuorojen vaihtuminen oikein oli melko hankala toteuttaa. Pelaajista tehtiin lista pelaajaLista. Tästä listasta haetaan haluttu pelaaja hakemalla jakojäännös vuorosta ja pelaajamäärästä seuraavanlaisesti: *Pelaaja = pelaajaLista[vuoro%pelaajaMaara]*. Aina kun kierros tulee täyteen (*if (((vuoro)%pelaajaMaara==0) & (vuoro!=0)):)* pelaajalistan paikkoja vaihdetaan niin, että jakajasta seuraavasta tulee jakaja ja sitä seuraavasta kierroksen aloittaja. Tämä ratkaisu on lopulta yksinkertainen ymmärtää ja toteuttaa.

Pisteiden laskussa pelin lopussa algoritmi hakee jokaisen pelaajan pisteet. Pelaajien omista tiedoista listasta points saadaan pisteet aiemmin pelissä kirjatuista mökeistä. Ässistä sekä pata2 ja ruutu10-korteista lasketaan pisteet Card-luokan getPoints-metodin avulla. Loput pisteet saadaan laskemalla jokaisen pelaajan kaapattujen korttien (capturedCards) kokonaismäärä Player-luokan countCards-metodin avulla sekä patojen määrä Player-luokan getSpades-metodin avulla. Näiden arvojen kokonaismäärää verrataan muihin pelaajin. Eniten kaapattuja kortteja kerännyt pelaaja saa yhden lisäpisteen ja eniten patoja kerännyt pelaaja saa kaksi lisäpistettä. Tämän jälkeen pelaajien kokonaispisteitä verrataan keskenään ja niitä eniten kerännyt pelaaja (name) julistetaan voittajaksi. Tähän olisi voinut toteuttaa erillisen metodin pisteidenlaskua varten, mutta se onnistui hyvin myös käyttämällä vanhoja metodeja.

Tietorakenteet

Projektissa tietorakenteina korttien käsittelyyn sopivat parhaiten listat. Esimerkiksi taulukoita olisi voinut käyttää, mutta listojen käyttäminen on kätevämpää, sillä niiden kokoa pystytään muokkaamaan. Tämä on tärkeää, koska korttipakan korttien, käsikorttien, pöytäkorttien sekä kaapattujen korttien määrä muuttuu koko ajan pelin edetessä.

Korttilistat olisi voitu myös muuttaa sanakirjoiksi, jotta ne voisivaat sisältää myös korttien pelissä käytettävät arvot ja pisteet. Tätä ei kuitenkaan tarvinnut tehdä, koska korttien pöytä- ja käsiarvot sekä pisteet saadaan haettua korteista toisella tavalla. Korttien pöytä- ja käsiarvot sekä pisteet haetaan lopulta vertaamalla niitä korttien string-muotoiseen esitykseen ja siksi sanakirjaan ei tarvittu erikseen merkitä kyseisiä arvoja.

Jakajan ja vuorossa olevan pelaajan määrittämiseen olisi sopinut boolean-tyyppinen tieto parhaiten, koska toteutuksessa riittää tieto kyllä tai ei. Vuorojenvaihtoalgoritmi toimii kuitenkin tavalla, jossa nämä boolean-arvot olivat tarpeettomia.

Tiedostot

Ohjelma ei käsittele tiedostoja.

Testaus

Aivan aluksi testattiin muodostuuko kortit pakassa oikein. Myöhemmin ohjelmaa testattiin pääasiassa ajamalla sitä yleensä alusta lähtien jopa satoja kertoja. Aina kun ohjelma ei toiminut halutulla tavalla, pyrittiin tunnistamaan ongelman lähde ja muuttamaan koodia haluttuun suuntaan. Tämä toistettiin kunnes koodi ja ohjelma toimivat halutulla tavalla.

Alun perin ohjelmaa oli suunniteltu testattavan kolmella eri yksikkötestauksella, joissa olisi katsottu kulkeeko kortit oikein listojen välillä, lasketaanko pisteet aina oikein sekä toimiiko kaappaus oikein. Näitäkin asioita kuitenkin pystyi hyvin testaamaan ajamalla ohjelmaa lukuisia kertoja ja keskittymällä ja tutkimalla näitä yksityiskohtia huolella.

Ohjelman tunnetut puutteet ja viat

Ohjelmasta puuttuu tallennus/lataus-vaihtoehto. Lisäksi koodissa on jonkin verran toistoa, kun käsitellään eri pelaajamääriä. Toistoa olisi voinut hieman pienentää keksimällä parempia ja lyhempiä tapoja toteuttaa annetut toiminnot.

3 parasta ja 2 heikointa kohtaa

3 Parasta kohtaa:

1. Toimiva kaappausalgoritmi

   Kaappausalgoritmi toimii moitteetta ja pelaajalle varsin yksikertainen ymmärtää ja käyttää. Se tunnistaa onko kortit mahdollista kaapata ja kertoo pelaajalle jos näin ei ole.

2. Toimivat vuoronvaihdot

   Vuoronvaihdot ja kierroksenvaihdot saatiin toteutettua toimiviksi. Varsinkin vuoronvaihdot toteutettiin hyvin lyhyellä koodilla.

3. Ei virheilmoituksia

   Ohjelman testaukseen käytettiin paljon aikaa ja siksi virheilmoitukset saatiin kitkettyä ohjelmasta.

2 Heikointa kohtaa

1. Pelissä ei ole tallennus/lataus-vaihtoehtoa

   Tämä osio jätettiin pois, koska aikaa ja tietotaitoa ei ollut osion toteutukseen tarpeeksi.

2. Koodissa jonkin verran toistoa

   Varsinkin pelin loppuosassa, lopullisten pisteiden laskun yhteydessä on käytetty niin sanottua copy-paste-koodia.

Poikkeamat suunnitelmasta

Suunnittelema piti jokin verran paikkaansa. Itse pelinsuorittamismetodi gamePlay vei huomattavasti enemmän aikaa kuin oli suunniteltu. En alun perin ymmärtänyt, kuinka paljon koodaamista ja testausta gamePlay-metodi vielä vaatii toimiakseen kunnolla. Player-luokkaan tehtiin pelin kulkuun liittyviä metodeja suunniteltua enemmän, jotta gamePlay-metodista ja Game-luokasta ei tulisi liian pitkiä, joten Player-luokka vei myös huomattavasti enemmän aikaa kuin alun perin oli suunniteltu. Muiden vaiheiden teko vei yleensä hieman vähemmän aikaa kuin oli suunniteltu. Luokkia toteutettiin kaksi vähemmän alkuperäiseen suunnitelmaan verrattuna, koska peli ei sisällä

tallennus/lataus-vaihtoehtoa. Alla on esitetty ennen projektin tekoa tehty yksinkertaistettu karkea arvio käytetyistä kokonaistunneista ja eri työvaiheista.

1. tiedonhaku 10 h

2. korttiluokka, maat, numerot, kortinpalautusmetodi 5h

3. pakan luokka, sanakirja, sekoitus, ensijako, pelijako, pakantekometodi, testaus 7h

4. pöytäluokka, kortit, kortin otto, uusi jako, testaus 7h

5. käyttäjäluokka, aloitus, pelin aloitus, testaus 5h

6. pelaajaluokka, nimi, käsikortit, pisteet, kaapatut kortit, jakaja, vuoro, testaus 6h

7. pelimetodit, pelaajamäärä, pakan käyttö, pelin kulku, vuoronvaihto, testaus 10h

8. pelin lopetus, pisteiden lasku, uusi peli, tallennus, lataus 10h

9. testaus 5 h

yhteensä 65 h

Toteutunut työjärjestys ja aikataulu

Projektin suunnitelmasta poikettiin siinä, että projektia aloitettiin tekemään kunnolla ajanpuutteen vuoksi pari viikkoa suunniteltua myöhemmin. Lisäksi kaikki vaiheet kestivät hieman vähemmän lukuun ottamatta Game-luokkaa ja Player-luokkaa, joiden teko kesti huomattavasti pidempään.

1. tiedonhaku 8 h 20.2-30.3

2. korttiluokka, maat, numerot, kortinpalautusmetodi - 4h 1.3-18.3

3. pakan luokka, sanakirja, sekoitus, ensijako, pelijako, pakantekometodi, testaus - 6h 19.3-5.4

4. pöytäluokka, kortit, kortin otto, uusi jako, testaus - 5h 22.3-11.4

5. aloitus, pelin aloitus, testaus - 4h 22.3-11.4

6. pelaajaluokka, nimi, käsikortit, pisteet, kaapatut kortit, pelaajametodit, testaus - 10h 1.4-1.5

7. pelimetodi, pelaajamäärä, pakan käyttö, pelin kulku, vuoronvaihto, testaus- 20h 1.4-1.5

8. pelin lopetus, pisteiden lasku - 4h 1.5-6.5

9. testaus - 4h 1.5-6.5

yhteensä 65 h

Arvio lopputuloksesta

Kokonaisuudessaan projekti onnistui varsin hyvin. Ohjelma ei anna juurikaan virheilmoituksia. Erilaiset syötteet on otettu huomioon ohjelman toteutuksessa ja niihin reagoidaan sopivalla tavalla. Projektista puuttuu tallennus/lataus-vaihtoehto. Lisäksi koodissa on jonkin verran toistoa, joka olisi voitu korvata paremmilla algoritmeilla. Tietorakenteet ja luokkajaot ovat mielestäni toteutettu onnistuneesti, enkä näe muita vaihtoehtoja yhtään parempina. Ohjelman rakenne soveltuu muutosten tekemiseen hyvin, koska luokkajaoilla on selkeästi erotettu tärkeät toiminnot. Lisäksi Player-luokkaan on koottu onnistuneesti jokaista pelaajaa koskevat tiedot. Tällöin ne ovat esimerkiksi helpommin tallennettavissa tai muuten muokattavissa.

Projektin tekeminen oli haastavaa, mutta yllättävän antoisaa. Projektia tehdessä kehittyi koko ajan. Projektin loppupuolella monet asiat, kuten algoritmien suunnittelu tai testaus sujuivat huomattavasti helpommin kuin projektia aloittaessa, kun oli saanut jo rutiinia enemmän.

Viitteet

Ohjelmaa rakentaessa hyödynnettiin muutamia ohjelmointifoorumeita, joissa neuvotaan korttipelien ohjelmointia. Lisäksi projektissa hyödynnettiin Pythonin perusluokkakirjastojen API-kuvausta.

https://docs.python.org/3/library/index.html

http://codereview.stackexchange.com/

http://stackoverflow.com/

Liitteet

LIITE1. Pelaajien valinta- ja vuoroesimerkki

```
Kuinka monta pelaajaa peliin osallistuu (2-4)? 2
Anna pelaajan 1 nimi. Joni
Anna pelaajan 2 nimi. Pekka
Peli alkaa.
Valmistaudu vuoroosi Joni. Paina ENTER jatkaaksesi.
On vuorosi Joni. Käsikorttisi ovat:
Kh
10h
8s
Ac
Pöytäkortit ovat:
Kc
7s
7d
6c
Mitä teet? (h = hylkää, k = kaappaa, a = auta, s = säännöt) k
Anna käsikorteistasi kaappaava kortti.(esim. As) Kh
Anna kaapattavat kortit yksi kerrallaan.(esim. As) Paina ENTER lopettaaksesi. Kc
Anna kaapattavat kortit yksi kerrallaan.(esim. As) Paina ENTER lopettaaksesi.
Koitetaan kaapata annetuilla käsi- ja pöytäkorteilla.
Kh
Kc
Kortit kaapattu onnistuneesti!
Vuorosi päättyy Joni. Paina ENTER tyhjentääksesi ruudun.
```

LIITE2. Kierroksen ja pelin päättyminen

```
Seuraava kierros. Kierroksen aloittaja vaihtuu.
Pakassa kortteja jäljellä: 0
Pelaajien pisteet: Joni 2 pistettä, Pekka 5 pistettä
Kortit loppuivat pelaajilta.
Joni saa loput pöytäkortit.
Pelaajien lopulliset pisteet: Joni 2 pistettä, Pekka 8
Voittaja on Pekka!!!
Peli päättyy!
```