Eero Saarro 710222 Bioinformaatioteknologia (2. Vuosi) 19.02.2021

2)

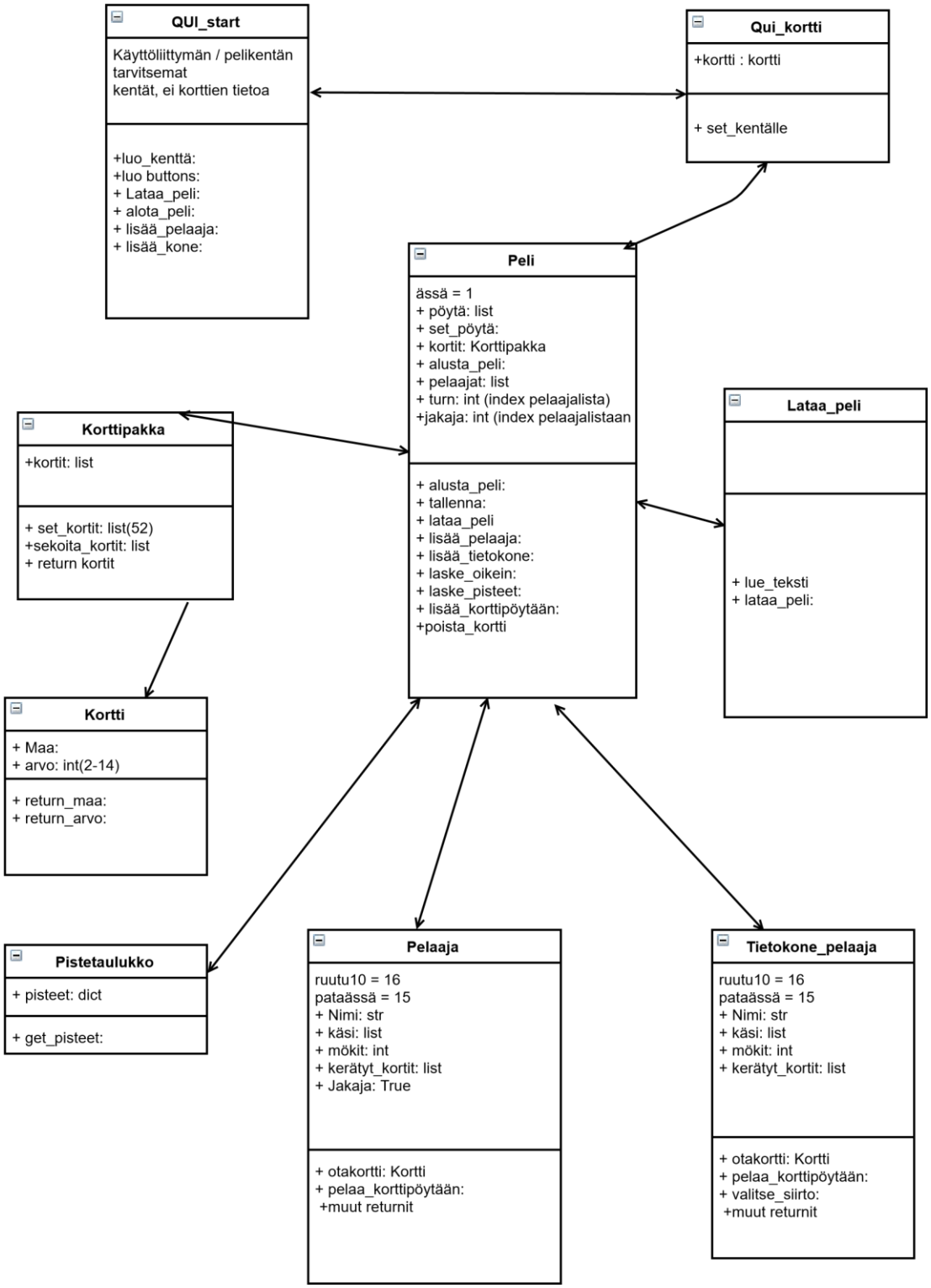
Tehtävänä on toteuttaa pakka-kasino korttipeli graafisena versiona. Pelissä kerätään pisteitä sääntöjen mukaisesti, kunnes joku pelaajista kerää 16 pistettä, jolloin peli loppuu. Peliä pystyy pelaamaan 2-N pelaajaa ja mahdollisesti myös 0-N tietokonepelaajaa. Pelissä pystytään myös tallentamaan sen hetkinen tilanne eli toisin sanoen säilyttämään pelitilanne samanlaisena ja lataamaan tallennettu peliversio.
Tavoitteenani on suorittaa vähintään keskivaikea versio, jossa vaaditaan graafinen käyttöliittymä sekä täsmällinen pelitilanteen tallennus sekä lataus mahdollisuus.
Tavoitteenani on kuitenkin pyrkiä suorittamaan myös vaikea versio, jossa vaaditaan mahdollisuutta pelata peliä 0-N tietokonevastustajaa vastaan, jotka pystyvät jossain määrin taktikoimaan omaa peli strategiaansa eikä vain lätkiä kortteja pöydälle.

3)

Ohjelma kommunikoi käyttäjän kanssa graafisen käyttöliittymän kanssa. Graafisessa käyttöliittymässä on aluksi kaksi painiketta aloita sekä lataa, joista toisella voidaan aloittaa täysin uusi pelikerta ja toisella lataa tallennettu peliversio ja jatkaa siitä. Uuden pelin aloittaminen lisää halutun määrän pelaajia, sekoittaa korttipakan sekä alustaa pelin jakamalla kortit. Itse pelissä näkyy ruudulla ainakin pöydällä olevat kortit, mitkä näkyvät kaikille. Tämän lisäksi pöydällä on piilotettuna/väärinpäin kortit, jotka kuuluvat pelaajalle. Vain pelaaja, jolla on pelivuoro sillä hetkellä, pystyy näkemään omat korttinsa. Peliä pelatessa pelaaja/tietokone pystyy omien korttien avulla valitsemaan pöydästä sääntöjen puitteissa kortteja, joiden oikeellisuuden algoritmi pelissä tarkistaa. Pelissä on myös tallennus painike, jota painamalla käyttäjä voi tallentaa sen hetkisen pelitilanteen, ohjelman sulkemis- painike, mikä lopettaa ohjelman suorittamisen ja menettää tallentamattomat tiedot sekä uuden pelikierroksen eli pakan sekoittamis-painike, joka käynnistää uuden pelikierroksen. Pelikentän viereen voisi olla sopivaa luoda pistetaulukko näyttämään pelaajien pistetilanne.

4)

Ohjelmakokonaisuus on helpointa luoda olioiden ympärille. Qui olio vastaa käyttöliittymästä ja pelikentän mallintamisesta sekä käyttäjän eri valintojen toteutuksesta. Tämän päälle voisi luoda Qui_kortti olion, joka vastaa pelikentällä olevien korttien graafisesta mallinnuksesta sekä luo kortit. Käyttöliittymä vuorovaikuttaa Peli olion kanssa, mikä vastaa pelikentän toteutuksesta. Pelikentällä näkyy pöydän kortit ja luodessa uuden pelikenttäolion se sekoittaa korttipakan ja jakaa kortit. Korttipakka olio koostuu 52 erilaisesta Kortti-oliosta, joilla on maa sekä arvo väliltä 2-14. Peli-olion avulla jaetaan pelaajille lisää kortteja. Pelaajat (tietokone/käyttäjä) on helpointa hahmoittaa omina olioina, joita kutsutaan Peli-oliosta. Pelaajilla on käsi-lista, jossa on pelaajan sen hetkiset kortit kädessä. Tämän lisäksi Pelaaja olioon tallennetaan tiedot pelaajan saamista korteista pöydältä, mikä helpottaa pistelaskua sekä saadut mökit. Tämän lisäksi pelissä on pistetaulukko-olio, johon tallennetaan aina pelikierroksen lopussa pelaajien pisteet.
Alustava UML-kaavio eri luokista ja niiden metodeista:


QUI_start
Käyttöliittymän / pelikentän tarvitsemat
kentät, ei korttien tietoa
+luo_kenttä:
+luo buttons:
+ Lataa_peli:
+ alota_peli:
+ lisää_pelaaja:
+ lisää_kone:
Qui_kortti
+kortti : kortti
+ set_kentälle
Peli
ässä = 1
+ pöytä: list
+ set_pöytä:
+ kortit: Korttipakka
+ alusta_peli:
+ pelaajat: list
+ turn: int (index pelaajalista)
+jakaja: int (index pelaajalistaan
+ alusta_peli:
+ tallenna:
+ lataa_peli
+ lisää_pelaaja:
+ lisää_tietokone:
+ laske_oikein:
+ laske_pisteet:
+ lisää_korttipöytään:
+poista_kortti
Lataa_peli
+ lue_teksti
+ lataa_peli:
Korttipakka
+kortit: list
+ set_kortit: list(52)
+sekoita_kortit: list
+ return kortit
Kortti
+ Maa:
+ arvo: int(2-14)
+ return_maa:
+ return_arvo:
Pistetaulukko
+ pisteet: dict
+ get_pisteet:
Pelaaja
ruutu10 = 16
pataässä = 15
+ Nimi: str
+ käsi: list
+ mökit: int
+ kerätyt_kortit: list
+ Jakaja: True
+ otakortti: Kortti
+ pelaa_korttipöytään:
+muut returnit
Tietokone_pelaaja
ruutu10 = 16
pataässä = 15
+ Nimi: str
+ käsi: list
+ mökit: int
+ kerätyt_kortit: list
+ otakortti: Kortti
+ pelaa_korttipöytään:
+ valitse_siirto:
+muut returnit

5)
Pöydässä sekä kädessä olevia kortteja on helpoin käsitellä listoina. Listat ovat dynaamisia tietorakenteita ja sopivat siten hyvin esittämään pelissä pöydällä sekä kädessä olevia kortteja, sillä listoista on helppo poistaa sekä lisätä alkioita tietyltä paikalta. Kyseiset listat ovat myös hyvin lyhyitä keskimäärin alle 10 alkioita lyhyitä, joten ne ovat algometrianalyysisesti käteviä kyseiseen tilanteeseen. Myös korttipakka on helppo luoda listana, josta pelin edetessä poistetaan kortteja käsiin, kunnes pakka loppuu. Pisteiden laskua varten voisi hyödyntää sanakirjaa, jossa avaimena on pelaaja olio ja arvona pelaajan pisteet.
6)
Ohjelmassa täytyy pystyä tallentamaan sen hetkinen tilanne ja lataamaan se myöhemmin, jotta peliä voidaan jatkaa samasta tilanteesta. Tässä kätevintä on tallentaa pelitieto järjestelmällisesti tekstitiedostoksi. Tekstitiedostoon tallennetaan pelitilanne esimerkiksi muuntamalla ruutu10 kuvaamaan r10, joten se toimii hyvin samanlaisesti kuin shakkipeli koodissa kierroksella 5. Näin saadaan tallennettua kätevästi eri lohkot (pelaajat ja niiden kädet, pisteet, pöydän kortit, jäljellä oleva pakka…).

7)
Projektin tärkein algoritmi on tarkastaa ja sallia vain oikeat/hyväksytyt siirrot pelaajilta. Tässä toimii yksinkertainen plus/miinus lasku, jonka avulla voidaan katsoa, että pöydästä valittujen korttien summa täsmää kädessä olevan kortin arvoon. Tämän lisäksi algoritmin tulee estää käyttäjää käyttämästä samaa pöydässä olevaa korttia eri yhdistelmiin. Eli jos kädessä on ruutu10, jonka arvo kädessä on 16. täytyy käyttäjän pöydältä valitsemien korttien summa olla myös 16.
Tietokone-pelaajalle täytyy muodostaa myös algoritmi, jonka avulla tietokone valitsee pöydästä kortteja. Tähän voidaan käyttää samanlaista lähestymis- tapaa kuin käyttäjän tapauksessa. Pöydällä sekä kädessä olevien korttien pitäminen lyhyinä listana helpottaa tätä ongelmaa, sillä listojen avulla on yksinkertaista käydä eri kombinaatiot lävitse. Tietokone laskee järjestelmällisesti eri mahdolliset vaihtoehdot pöydältä, joiden summa on sama kuin kortin arvo ja valitsee siten yhdistelmän, joka tuottaa tietokonepelaajalle eniten pisteitä. Tämän lisäksi tietokoneen pitää katsoa, mitä kortteja pöytään jää jäljelle, jotta pöydälle ei jää liian helppoja mökkejä. Tämä on erityisen tärkeää, tietokoneen joutuessa laittamaan kortin pöydälle, jotta pöydälle laitettu kortti avaa mahdollisimman vähän uusia yhdistelmiä ja siten mökki mahdollisuuksia. Esimerkiksi jos pöydällä on valmiiksi arvot 9 ja 2 niin tietokoneen tulee laittaa pöydälle kortti, jonka arvo on yli 3 tai 5, jotta toinen pelaaja ei saa helppoa mökkiä mitenkään eli pääse tyhjentämään pöytää.
Korttipakan sekoittaminen tapahtuu random algoritmilla, valitsemalla kortteja satunnaisesti järjestetystä korttipakka listasta uuteen korttipakkalistaan.

8)
Ohjelmaan täytyy luoda jo ohjelman tekovaiheessa yksikkötestejä testaamaan algoritmien oikea toiminta. Näin varmistutaan muun muassa siitä, että metodit estävät vääränlaisen pelaamisen sekä erityisesti testaavat tietokonepelaajan algoritmin toiminnan. Yksikkötestejä kannattaa luoda myös muille pelin keskeisille metodeille kuten korttipakan luomiselle sekä korttien liikkumiselle käden ja pöydän välissä.
Graafisen käyttöliittymän valmistuttua voidaan pelin toimintaa testata vielä lopullisesti pelaamalla itse peliä ja varmistumalla siten koko pelin oikea oppinen toiminta.

9)

Projektissa täytyy käyttää PyQT5 kirjastoa käyttöliittymän tekemiseen. Tämän lisäksi tarvitaan random-kirjaston rand() funktioita sekoittamaan pakka satunnaiseen järjestyksen.

10)

Aikataulutuksen kannalta eniten aikaa menee graafisen käyttöliittymän luomiseen vähintään 20 tuntia oletettavasti.

Ohjelmointi työn aloitan tekemällä pelin olennaisimmat oliot ensimmäisenä kuten pelikentän, pelaajan, korttipakan jne. jonka jälkeen alan ohjelmoimaan graafista käyttöliittymää. Viimeisimpänä teen tallennus sekä lataa metodit oikein, sillä nämä vaativat, että peli toimii muuten oikein. Myös tietokonepelaajan tekeminen on viimeisiä asioita, sillä se vaatii, että käyttäjä pystyy jo pelaamaan peliä.

11)

Kasino-pelistä löytyy paljon ohjeita netistä, mitkä helpottavat ohjelman tekemistä. Erityisesti korttipelin graafiseen toteutukseen saa paljon vinkkejä netistä. Tämän lisäksi täytyy hyödyntää PyQt5 kirjaston implementaatiota.