Tekninen suunnitelma

Ohjelman rakennesuunnitelma

Ohjelmassa on neljä eri luokkaa Card, Player, Round ja Game. Game kuvaa peliä ylimmällä tasolla ja sisältää listan pelaajista ja korteista ja hoitaa pelin aloittamiseen ja lopettamiseen sisältyvät toiminnot. Se myös tallentaa pelaajien pisteet tekstitiedostoon. Round kuvaa yksittäistä pelikierrosta, pitää huolen pakan sekoittamisesta, korttien jakamisesta, pelivuoron vaihtamisesta, korttien paikkojen vaihtamisesta ja pisteiden laskemisesta. Player sisältää tiedot pelaajasta ja pelaajan korteista. Card on luokka korteille, jossa on kortin nimi, ja kortin pistearvot pöydässä ja kädessä.

Game

- players (lista Player objekteja)
- cards (lista Card objekteja)
- next_dealer (osoittaa pelaajaan joka on jakaja seuraavalla kierroksella

- set_players
    - Pyytää käyttäjältä tietoa pelaajista ja tallentaa tiedot players listaan Player objekteina.
- check_end
    - Tarkistaa jokaisen kierroksen jälkeen mikäli jollain pelaajalla on 16 pistettä.
- save_game
    - Tallentaa pelaajien pisteet tekstitiedostoon
- load_game
    - lataa pelaajien pisteet tekstitiedostosta
- change_dealer
    - Laittaa jokaisen kierroksen jälkeen next_dealer muuttujan osoittamaan seuraavaan pelaajaan.
- initialize_cards
    - Luo listan cards joka sisältää Card objekteja
- final_results
    - Tulostaa lopputulokset

Round

- table_cards (lista pöydässä olevista korteista)
- deck (lista pakassa olevista korteista)
- next_player (pelivuorossa oleva pelaaja)

- deal_cards
    - Sekoittaa korttipakan ja jakaa jokaiselle pelaajalle 4 korttia, lisää table_cards listaan 4 korttia ja lisää loput kortit deck listaan.
- change_player

    - Mikäli kierros on juuri alkanut osoittaa jakasta seuraavaan pelaajaan muuten siirtyy askeleen eteenpäin pelaajalistassa ja palauttaa pelaajan (eli pelivuorossa olevan pelaajan.
- the_play
    - Pyytää pelaajalta syötettä siitä mitä hän haluaa tehdä ja kutsuu check_validity metodia tarkistamaan onko liike laillinen. Mikäli liike on laillinen tehdään vaaditut muutokset listoissa jotka sisältävät kortteja.
    - the_play metodi tulee kutsumaan muita metodeja hoitamaan pelivuoron eri osia
- check_validity
    - Tarkistaa onko pelaajan liike laillinen.
- give_points
    - Jakaa pelaajille pisteet kierroksen päätyttyä tai antaa pelaajalle mökki-pisteen.
- check_end
    - Tarkistaa onko pelaajilla edelleen kortteja kädessä.
- check_deck
    - Tarkistaa onko pakka tyhjä.
- check_table
    - Tarkistaa onko pöytä tyhjä.

Player
- name (pelaajan nimi)
- points (pelaajan pisteet)
- cards_hand (lista pelaajan kädessä olevista korteista tietyllä hetkellä)
- cards_deck (lista pelaajan pakassa olevista korteista tietyllä hetkellä)

Card
- name (kortin nimi joka näkyy pelaajille, esimerkiksi Herttajätkä-11)
- table_value (kortin arvo pöydässä)
- hand_value (kortin arvo kädessä)

Luokat Player ja Card olivat selviä valintoja sillä ne koostuvat yhteen kuuluvista tiedoista. Luokat Game ja Round taas ovat sopivan kokoisia kokonaisuuksia tiettyjen ongelmien ratkaisemiseen ja ne muodostavat myös loogisesti yhteenkuuluvat kokonaisuudet. Mahdollisia luokkia olisivat myös olleet luokka “Board” jossa olisi ollut tiedot korttien sijainnista mutta mielestäni oli toimivampaa jakaa ne Player ja Round luokkiin ja luokka “Play” joka olisi kuvannut pelitilannetta jossa pelaaja tekee siirron mutta se olisi muodostanut suhteellisen pienen kokonaisuuden joten päätin sisällyttää vaaditut toiminnallisuudet luokan Round metodeihin.

Käyttötapauskuvaus

Käyttäjän käynnistäessä ohjelman main moduuli toivottaa käyttäjän tervetulleeksi ja informoi käyttäjää sallituista ja mahdollisita syötteistä. Tämän jälkeen luokan Game set_players

metodia kutsutaan joka pyytää käyttäjältä tietoa pelaajista ja tallentaa pelaajat listaan, asettaa ensimmäisen pelaajan jakajaksi ja luo listan korteista metodilla initialize_cards. Tämän jälkeen ensimmäinen kierros voi alkaa, luokan Roundin metodi deal_cards jakaa kortit ja metodi change_player asettaa jakajaa seuraavana olevan pelaajan pelivuoroon. Pelivuorossa oleva pelaaja näkee omat korttinsa ja pöydällä olevat kortit.
Molemmissa kortit ovat numeroitu järjestyksen mukaan esimerkiksi näin:
Pöytä:
1.Ruutu-7 2. Risti-7 3. Hertta-4 4. Risti-3

Käsi:
1. Ruutu-6 2.Herttajätkä-11 3. Risti-10 4.Risti-5

Roundin metodi the-play kysyy pelaajalta haluaako hän ottaa kortillaan pöydästä kortteja ja mikäli haluaa ohjeistetaan ensin antamaan syötteenä minkä kortin hän haluaa käyttää ja tämän jälkeen mitkä kortit hän haluaa pöydästä ottaa. Tämän jälkeen luokan Round check_validity metodi tarkistaa voiko hän ottaa haluamansa kortit. Tässä tapauksessa hän antaa esimerkiksi syötteen:
3
Jonka jälkeen kysytään mitkä kortit hän haluaa ottaa pöydästä ja hän antaa syötteen:
1 4
jolloin check_validity tarkistaa onko liike laillinen, tässä tapauksessa se on ja Ruutu-7 ja Risti-3 poistetaan pöydästä ja lisätään pelaajan pakkaan ja tarkistetaan metodilla check_end mikäli kierros jatkuu. Tämän jälkeen kutsutaan metodia change_player joka asettaa seuraavan pelaajan pelivuoroon jne. Tämä jatkuu siihen asti kunnes check_end palauttaa True ja kierros loppuu. Sen jälkeen pelaajille jaetaan pisteet metodilla give_points ja tämän jälkeen luokan Game metodi check_end tarkistaa jos jollain pelaajista on 16 pistettä. Mikäli on peli loppuu ja mikäli ei ole metodi change_dealer vaihtaa jakajaa ja uusi kierros alkaa. Kun peli loppuu metodi final_results tulostaa lopputulokset ja tämän jälkeen käyttäjältä kysytään haluaako hän pelata uuden pelin vai lopettaa. Mikäli hän valitsee lopettamisen ohjelmasta poistutaan.

Algoritmit

Pelin merkittävin algoritmi on algoritmi joka tarkistaa pöydästä otettujen korttien laillisuus. Jokaisella kortilla on oma pistearvonsa. Pelaajan pöydästä valitsemat kortit ovat listassa ja tämä listan kombinaatioita aletaan käydä läpi ja mikäli löytyy kombinaatio joka vastaa pistearvoltaan pelaajan valitsemaa korttia, poistetaan kombinaatio listasta ja käydään rekursiivisesti pienempää listaa läpi kunnes lista joko on tyhjä tai jokainen kombinaatio on käyty läpi. Mikäli lista on tyhjä pelaajan valitsemat kortit ovat sallittuja ja mikäli lista ei ole tyhjä ne eivät ole sallittuja.

Toinen vaihtoehtoinen toteutus olisi ollut tehdä algoritmi joka käy läpi kaikki pöydässä olevien korttien kombinaatiot, pelaajan valitseman kortin perusteella ja pelaaja voisi sitten valita näistä kombinaatiosta mitkä kortit hän haluaa ottaa. Eli sen sijaan että pelaaja kertoo ensin itse mitkä kortit hän haluaa pöydästä, hänelle kerrottaisiin mitkä kortit ovat otettavissa.

Tämä algoritmi olisi hieman monimutkaisempi ja mikäli kortteja olisi pöydässä paljon saattaisi se viedä hieman enemmän aikaa.

Muita haastavimpia algoritmeja ei ohjelmassa vaadita kun myös korttien sekoittamiseen löytyy Pythonilta valmis funktio random.shuffle() moduulista random.

Tietorakenteet

Tässä ohjelmassa Pythonin taulukkorakenne on riittävä tiedon varastoimiseen. Pythonilla on taulukkorakenteeseen laajat sisäänrakennetut funktiot, taulukosta hakemiseen, lisäämiseen ja poistamiseen, mikä helpottaa koodaamista ja lyhentää tarvitun koodin määrää. Mahdollinen vaihtoehto olisi ollut linkitetyt listat mutta tässä tapauksessa taulukot riittävät. Varastoitavan tiedon määrä on myös sen verran pieni että taulukot ajavat asiaansa.

Aikataulu

1-15.3 Saada ensimmäinen versio luokista valmiiksi.
16-31.3 Saada ensimmäinen toimiva versio ohjelmasta valmiiksi.
1-30.4 Testausta, ohjelman parantelua ja dokumentaation kirjoittaminen.

Arvio ajankäytöstä:
10-15h Ensimmäinen versio luokista
10h Saada ohjelma toimimaan halutun lailla.
10h Testausta ja testien kirjoittamista
5h Dokumentaatio

Aloitan luultavasti ohjelman keskeisten ominaisuuksien tekemisestä ja haastavimmista osista. Tämä tarkoittaa korttien tarkistusalgoritmin kirjoittamista ja eri luokkien perusominaisuuksien lisäämistä kuten pelaajien lisäämistä, korttien alustamista, nostamista ja pöytään laittamista. Aina kun lisään uuden ominaisuuden testaan että se toimii. Kun luokat ovat kirjoitettu valmiiksi kirjoitan main moduulin valmiiksi ja tämän jälkeen pääsen testaamaan kokonaista ohjelmaa.

Yksikkötestaussuunnitelma

Ohjelman keskeisintä metodia check_validity voidaan esimerkiksi testata syötteillä check_validity(Herttakuningatar(12), [Herttajätkä (11), Pataässä (1]) jonka pitäisi palauttaa True sillä sekä annetun kortin eli Herttakuningatar(12) ja pöydästä otettujen korttien pistearvo on sama eli 12. Voidaan myös testata syötteillä check_validity(Herttakuningatar(12), [Ruutu-2, Patajätkä (11)]) jonka pitäisi palauttaa False sillä pisterarvot ovat eri, eli 12 ja 13. Lopuksi testataan syötteillä check_validity(Herttakuningatar(12), [Herttajätkä (11), Pataässä (1), Ruutu-6, Risti-5]) jonka

pitäisi palauttaa False. Tässä voidaan pöydässä olevat kortit jakaa kahteen osaan eli Herttajätkä (11) + Pataässä (1) ja Ruutu-6 + Risti-5 + Pataässä (1), joiden molempien pistearvo on 11 eli sama kuin Herttajätkän(11) mutta Pataässä (1) löytyy molemmista mikä ei ole sallittua ja check_validity metodin pitäisi huomata tämä.

Metodia the_play:ta testaan eri syötteillä jossa pelaaja joko valitsee vaihtoehdon nostaa pöydästä kortteja, laittaa kortin pöytään, joutuu laittamaan kortin pöytään sillä ei pysty nostamaan kortteja ja joutuu laittamaan kortin pöytään sillä pöytä on tyhjä.

Kirjallisuusviitteet ja linkit

https://docs.python.org
https://www.tutorialspoint.com/python/index.htm
https://www.programiz.com/python-programming/examples/shuffle-card
https://stackoverflow.com/