Robin Kaulbars-Staudinger, 479466, AUT 2014, 05.05.2017

1. **Yleiskuvaus**

Kasino korttipeli

- Jokaisen pelikierroksen alussa pakka sekoitetaan ja jakaja jakaa jokaiselle pelaajalle 4 korttia (eivät näy muille) sekä 4 korttia pöytään (näkyvät kaikille). Loput kortit jätetään pöydälle pinoon ylösalaisin.
- Jakajasta seuraava aloittaa pelaamisen. Seuraavalla pelikierroksella hän on jakaja.
- 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 itselleen pinoon. Pinon sisällöstä lasketaan korttien loputtua pisteet.
- Pöydässä olevien korttien määrä voi vaihdella vapaasti. Jos joku vaikkapa 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 enää lisää vaan pelataan niin kauan kuin kenelläkään on kortteja kädessä. Tällöin siis tietenkin alle 4 korttia on mahdollinen tilanne.)
- Kierroksen loputtua mahdolliset pöydälle jääneet kortit annetaan viimeksi keränneelle pelaajalle
- Kortilla voi ottaa pöydästä yhden tai useampia samanarvoisia kortteja ja kortteja, joiden summa on yhtä suuri, kuin kortin jolla otetaan.
- Vaikeusaste: keskivaikean ja vaativan välillä, täyttää vaativan vaatimukset graafista käyttöliittymää lukuun ottamatta

2. **Käyttöohje**

   Ohjelma käynnistetään ajamalla ‘casino.py’ tiedosto pythonilla. Pelin käynnistettyä ruudulle esiintyy ohjeita pelaamiseen. Mikäli tallennus edellisestä pelistä on olemassa, peli kysyy pelaajalta, haluaako tämä jatkaa siitä mihin jäi. Kortteja käsitellään indeksien avulla, indeksit näkyvät vastaavien korttien alla, mikäli kortin

keräys ei ole mahdollista, saa pelaaja vaihtoehdon laittaa kortin pöytään painamalla ’0’. Kombinaatioita voi ottaa kirjoittamalla indeksien ’+’ ja kombinaatiot erotetaan ’,’ merkillä. Mikäli pelaaja haluaa poistua pelistä, voi hän kirjoittaa ’QUIT’ missä kohtaa tahansa. Tallentaakseen pelin, pelaaja voi kirjoittaa ’SAVE’ omalla vuorollaan pelin aloitettua.

3. **Ohjelman rakenne**

   Casino:

   Luo Game-olion, joka aloittaa pelin.

   Game:

   Pelin tärkein luokka. Sisältää pelin logiikan ja pyörittää peliä.

   - addPlayer(self): kysyy pelaajan nimeä ja luo pelaajalle Player-olion lopuksi metodi lisää pelaajan pelaajalistaan.
   - addBots(self): kysyy vastustajien määrää, antaa niille nimen sekä luo niille Player-olion. Metodi lisää vastustajat pelaajalistaan ja antaa pelaajille satunnaiset istumapaikat.
   - newRound(self): Kutsuu initDeck metodia jonka jälkeen jakaja jakaa kortit. Lopuksi metodi kutsuu gamePlay-metodia, joka huolehtii pelin kulusta.
   - initDeck(self): Luo Card-olion jokaiselle kortille, lisää ne pakkaan ja sekoittaa pakan.
   - gamePlay(self): Pyörittää peliä, selvittää kenen vuoro on ja kutsuu vastaavat metodit. Kutsuu newRound-metodia, kun kierros on loppu
   - collect(self, player, handindex): Tarkistaa ja suorittaa pelaajan siirron. Metodi kutsutaan playerTurn-metodista.
   - command(self, string): tarkistaa onko syöte komento sulkea tai tallentaa peli.

   Player:

   Luo pelaaja-olion jolla on nimi, käsi, pino, tyyppi ja pisteet.

HardBot:

Tekoäly-luokka, muokkaa Player-oliota.

- arrange(self, cards): järjestää kortit tärkeysjärjestykseen.
- collect(self): etsii mahdollisia nostoja ja antaa ne parametrina stack-metodille.
- stack(self.cards): kerää kortit pöydästä ja lisää ne omaan pinoon. Mikäli collect on antanut ainoastaan yhden kortin, laittaa metodi kortin pöydälle.

Card:

Luo kortti-olion jolla on numero maa ja mahdollisesti erikoisarvo.

SetOfCards:

Yliluokka luokille Table ja Deck, jotka sisältävät pöydän ja pakan kortit.

ScoreKeeper:

Laskee pisteet kierroksen loputtua. Päättää pelin, mikäli voittaja on löytynyt.

SaveLoad:

Pelin tallennus ja lataus.

- save(self): kirjoittaa tiedoston kaikella tarpeellisella tiedolla pelistä, jotta se voidaan myöhemmin ladata.
- load(self): lukee tallennetun tiedoston ja alustaa pelin näillä tiedoilla.

4. **Algoritmit**

   Pelaajan valitseman korttiyhdistelmän laillisuus:

   Algoritmi vertaa pöydästä valittujen kombinaatioiden arvoa kädessä olevan kortin arvoon.

   Tekoäly:

Vertaa korttien pistearvoa ja järjestää kortit laskevaan järjestykseen. Mikäli korteilla on sama pistearvo, saa kortti jonka maa on pata korkeamman arvon. Jos pistearvo kahden valinnan välillä on yhtä suuri, valitsee tekoäly sen, missä on enemmän kortteja.

5. **Tietorakenteet**

   Projektissa käytössä on listoja, listat pitävät huolen niin pelaajista kuin korteista, listat ovat muuttuvatilaisia.

   Mahdollinen muu toteutus voisi olla sanakirja pelaajien ja korttien välillä.

6. **Tiedostot**

   Ohjelma käyttää tekstitiedostoa pelin tallentamiseen, tiedostossa tietojen erittelyyn käytetään rivinvaihtoa. Tiedosto luodaan pelistä, antamalla komento ’SAVE’.

7. **Testaus**

   Testaus tehtiin lähinnä kokeilemalla eri virhearvoja ohjelmaa pyörittäessä.

   Ohjelma läpäisee suunnitelmassa esitetyt testit.

   Testaaminen olisi voinut sujua paremmin jonkun IDE:n debuggerin avulla, koen että ohjelmalle on vaikea tehdä yksikkötestauksia yksikkötestaustaidoillani.

8. **Ohjelman tunnetut puutteet ja viat**

   Ei tunnettuja vikoja.

9. **3 parasta ja 3 heikointa kohtaa**

   + Tekoäly: valitsee hyvin kortit

   - Tekoäly: ei pysty ottamaan monta eri kombinaatiota. Enemmällä ajalla olisin varmaan voinut tämän ratkaista, ajatushan on sama kuten pelaajan kerätessä.

   + yleisesti siisti koodi, käyttänyt kommentteja, ulkoasu pelatessa on selkeä.

   +- pelaajan pitää itse kertoa millaisia kombinaation ottaa, mielestäni tämä on selkeämpää pelaajalle tekstipohjaisella käyttöliittymällä, mutta graafisella olisi parempi vain valita kortit.

10. **Poikkeamat suunnitelmasta**

    Graafinen käyttöliittymä jäi tekemättä, aikaa käytin suunnilleen sen verran mitä suunnittelin. Työn aloitus viivästyi parilla viikolla kandidaatintyön johdosta. Graafinen käyttöliittymä jäi tekemättä ajanpuutteen ja puutteellisten taitojen vuoksi; en ehtinyt opettelemaan.

11. **Toteutunut työjärjestys ja aikataulu**

    Työjärjestys oli kuin suunniteltu, työt alkoi pari viikkoa myöhässä suunnittelusta.

    1.4: ensimmäinen sessio

    4.4. peli toimii ilman tekoälyä, ei virhetilannekäsittelyä

    21.4 virhetilannekäsittely lisätty

    27.4 korjauksia virhetilannekäsittelyyn ja pelin kulkuun, tekoälyn ohjelmointi alkoi

    28.4 tekoäly toimii, tallennus lisätty

    2.5 pelin lataus lisätty

    5.5 loppuviimeistely

12. **Arvio lopputuloksesta**

    Olen tyytyväinen ohjelmaan, paria asiaa lukuun ottamatta se toimii niin kuin suunniteltu. Vielä paranneltavaa: Graafinen käyttöliittymä, jonka mukana tulisi korttien poimiminen ilman kombinaatioiden erottelua, sekä tekoälyn parannus jotta se voisi poimia monta kombinaatiota pöydästä.

    Ohjelma on pitkälti rakennettu niin että sitä voi jatkaa ja muuttaa olio-ohjelmoinnin avulla. Koodia on kommentoitu ja se on mielestäni hyvin jaoteltu. Printit voisi mahdollisesti laittaa omaan luokkaan.

13. **Viitteet**

    https://github.com

    https://www.python.org/doc/

http://stackoverflow.com/

https://www.tutorialspoint.com/