Komentorivipohjainen yatzy-peli

Olio Ohjelmoinnin Peruskurssin harjoitustyö

Ohjaaja: Vilho Kivihalme

Tekijät:

Onnimanni Hannonen
Erillisopiskelija, lukio yhteistyö
ohhann@utu.fi
613354

Jesse Lappalainen
Erillisopiskelija, lukio yhteistyö
jemlap@utu.fi
613352

Aku Paananen
Erillisopiskelija, lukio yhteistyö
akpaan@utu.fi
613348

Juho Torkkeli
Erillisopiskelija, lukio yhteistyö
jutato@utu.fi
613347

2. Tehtävän kuvaus ja analysointi
Teimme yatzy-pelin yksinkertaisella komentorivipohjaisella käyttöliittymällä. Peli tallentaa kaikki suoritukset tiedostoon, eli niitä voi vertailla aikaisempiin tuloksiin. Pelissä pelaaja pelaa yksin koittaen saada mahdollisimman suuren tuloksen.

Pelaaja saa siis heittää noppia ja laittaa pisteet haluamaansa paikkaan, pelimuodosta riippuen.

Mahdollisia laajennuksia peliin: Kunnollinen graafinen käyttöliittymä, online tulostaulu ja multiplayer.

Jouduimme jonkun verran muokkaamaan alkuperäistä suunnitelmaa, kun puutteita ilmeni. Päätimme tehdä metodeita muutamille koodinpätkille jotka toistuivat monessa paikassa. Tajusimme myös, että tietty toiminnallisuus piti tehdä erikseen eri pelimuotoja varten.

Aluksi annoimme kaikille omat luokat tehtäväksi, mutta loppua kohden kaikki auttoivat toisiaan toiminnallisuuksien yhdistämisessä, ongelmien ratkaisussa ja bugien korjailussa. Alkuperäinen luokka-jako oli seuraavanlainen:
Jesse:
- Pöytäkirja luokka

Aku:
- Tulostaulu luokka

Juho:
- Noppa-luokka
- Molemmat pelimuodot
- NoppienHeitto-luokka

Onnimanni:
- Rivi-luokka
- Suorittaja-luokka

3. Ratkaisuperiaate
Palautuskansiossa on liitteenä UML-kaavio projektiin.

4. Ohjelman ja sen osien kuvaaminen (lähinnä moduulien kuvaus kohdasta 3.5)

Abstrakti luokka `Yatzy` esittelee `gameController()` metodin, mutta ei toteuta sitä

`Suorittaja`-luokka
-sisältää metodin `aloita()`, joka kysyy käyttäjältä haluaako hän katsoa aikaisempien pelien tuloksia vai pelata, ja toimii sen mukaisesti
-sisältää metodin `pelaa()`, joka kysyy käyttäjältä mitä pelimuotoa halutaan pelata, ja toimii sen mukaisesti
-sisältää metodin `tulokset()`, joka kysyy minkä pelimuodon tilannetta pelaaja haluaa katsoa, ja toimii sen mukaisesti

`TavallinenPelimuoto`-luokka
-perii Yatzy luokan ja toteuttaa sen `gameController()` metodin siten, että pelaaja saa itse päättää rivien heittojärjestyksen

`PakkoyatzyPelimuoto`-luokka
-perii Yatzy luokan ja toteuttaa sen `gameController()` metodin siten, että pelaajan tulee heittää kaikki rivit järjestyksessä

`Noppa`-luokka
-mallintaa Yatzy-pelin noppaa
-sisältää muuttujan silmaluku, johon tallennetaan viimeisin heitto kokonaislukuna
-sisältää metodin `heitaNoppaa()`, joka arpoo silmaluku-muuttujalle uuden arvon (1-6)
-asetus- ja havainnointimetodi silmaluku-muuttujalle
-`equals()`-metodi kahden noppa-olion vertailemiseen

`NoppienHeitto`-luokka
-sisältää metodin `heitäNopat()`
-metodi hoitaa noppien heittämisen (noppia voi heittää halutessaan kahdesti uudelleen)
-palauttaa listan nopista, joka on noppien heiton jälkeinen noppien tila

`Rivi`-luokka
-mallintaa yhtä pöytäkirjan riviä
-sisältää muuttujan nimi, joka on rivin nimi
-sisältää muuttujan arvo, joka on rivin arvo
-sisältää metodin `annaRivi()`, joka palauttaa rivin luettavassa muodossa, esim “Pieni suora: 15”
-asetus- ja havainnointimetodi arvolle

`Pöytäkirja`-luokka:
-pitää kirjaa pelaajien pisteistä
-sisältää listan **Rivi** olioita, joiden avulla pisteistä pidetään kirjaa.
-konstruktori alustaa uuden tyhjän pöytäkirjan.
-metodi `luePoytakirja()` lukee pöytäkirjan tiedostosta ja asettaa nykyisen pöytäkirja olion arvot tiedostoa vastaavaksi.
-metodi `tallennaPoytakirja()` (ja `tallennaPoytakirjaJ()`), joka tallentaa pöytäkirjan tiedostoon.
-metodi `tulostaPoytakirja()`, joka tulostaa pelaajan pöytäkirjan numeroituna.
-metodi `asetaTuloksetHaluttuunPaikkaan()`, tulostaa pöytäkirjan ja lukee syötteenä minne pelaaja haluaa tuloksensa laittaa (tai vetää yli).

-metodi `asetaTuloksetSeuraavaanRiviin()`, jos pelimuoto on “järjestyksessä”, asettaa noppien tuloksen seuraavalle riville.
- metodi `kokonaisPisteet()` palauttaa kokonaislukuna koko pöytäkirjan yhteiset pisteet Yatzyn sääntöjä noudattaen.

`Tulostaulu`-luokka:
-hoitaa tulostaulun tiedostoon kirjoittamisen
-hashmap, jossa avaimena pistemäärä ja arvona pelaajan nimi.
-metodi `lisaaTulostauluun()`, joka lisää uuden tuloksen.
-metodi `tulostaParhaat()`, joka ottaa parametrina kuinka monta parasta halutaan tulostaa, ja tulostaa niin monta nimeä toplistalta.
-metodi `tallennaTulostaulu()`, joka järjestää ja tallentaa tulostaulun tiedostoon.
-metodi `lueTulostaulu()`, joka lukee tulostaulun tiedostosta ja tallentaa hashmappiin.
-metodi `lisaaTulostauluunJarjestyksessa()`, joka lisää uuden tuloksen (pakkoyatzy pelimuodossa).
-metodi `tulostaParhaatJarjestyksessa()`, joka ottaa parametrina kuinka monta parasta halutaan tulostaa, ja tulostaa niin monta nimeä toplistalta (pakkoyatzy pelimuodossa).
-metodi `tallennaTulostauluJarjestyksessa()`, joka järjestää ja tallentaa tulostaulun tiedostoon (pakkoyatzy pelimuodossa).
-metodi `lueTulostauluJarjestyksessa()`, joka lukee tulostaulun tiedostosta ja tallentaa hashmappiin (pakkoyatzy pelimuodossa).

5. Testausjärjestely
Projektin testaus tehtiin yksilöllisesti. Loimme siis omia testaus luokkia olioiden kokeiluun ja eri tilanteisiin asettamiseen. Kun suorittaja luokka valmistui, siirryimme testaamaan sen avulla ja ratkaisemaan ongelmia ja bugeja eri palasten välille.

Yritimme tehdä unit-testauksia, mutta se ei vaan meidän projektillemme tuntunut tarpeeksi laajalta. Bugit joita havaitsimme eivät olleet juurikaan loogisia virheitä (esim. nopan silmäluku voisi olla 10), vaan bugit johtuivat lähinnä eri luokkien yhdistämiseen liittyvistä ongelmista. Unit-testaukset ovat hyviä juuri loogisten virheiden minimointiin, joten emme kokeneet niitä tarpeellisiksi.

Ison määrän ongelmia meille tuotti Scanner-luokan käyttö, ja etenkin sen nextLine()-metodi. Jouduimme monesti tilanteeseen jossa käyttäjä antoi vain yhden syötteen, ja monet scannerit tulkitsivat että se on tarkoitettu heille. Näin ei scannerin documentaationkaan mukaan olisi kuulunut käydä. Ratkaisimme ongelman luomalla vain yhden scanner olion ja antamalla sen aina eteenpäin seuraaville metodeille jotka sitä tarvitsivat.

6. Liitteet:

- **o alkuperäinen tehtävänanto**
- **o yksityiskohtaisesti kommentoitu ohjelmalistaus: Git**
- **o UML-kaavio**

7. Käyttöohjeet:

Pura pakattu tiedostokansio haluamaasi paikkaan. Siinäpä se. Avaa purkamasi kansio ja klikkaa Yatzy.exe. Tämä avaa pelin komentokehotteeseen.

Jar file ja sen luomat tiedostot ovat Src kansiossa. Source koodi gitissä.