TKO\_2005 Olio-ohjelmoinnin perusteet 2018

Ohjaaja: Erkki Kaila

Harjoitustyödokumentaatio

Kuningaskunta-Simulaatio

Tommi Heikkinen

Santeri Loitomaa

Pasi Toivanen

Sisällysluettelo

[1 Tehtävän kuvaus 1](#_Toc508374571)

[2 Ratkaisuperiaate 2](#_Toc508374572)

[2.1 Pelitilan kuvaaminen 2](#_Toc508374573)

[2.2 Kuninkaan vastaanoton kuvaaminen 2](#_Toc508374574)

[3 Ongelma ja sen osien kuvaaminen 3](#_Toc508374575)

[3.1 Kuningaskunta ja main-metodi 3](#_Toc508374576)

[3.2 Kuningas, pelitila ja vuorokierto 3](#_Toc508374577)

[3.3 Suku-luokka 3](#_Toc508374578)

[3.4 Ongelma-luokan päätökset, vaatimukset ja seuraukset 3](#_Toc508374579)

[3.5 TallennaLataaPisteet-luokka 3](#_Toc508374580)

[3.5.1 Pelitilan lataus ja tallennus 3](#_Toc508374581)

[3.5.2 Pisteet 3](#_Toc508374582)

[4 Testausjärjestely 4](#_Toc508374583)

[4.1 Sukujen ja kuninkaan testaaminen 4](#_Toc508374584)

[4.2 Vuorokierto ja ongelmat 4](#_Toc508374585)

[4.3 Pisteet ja pelin tallennus 4](#_Toc508374586)

[5 Liitteet 5](#_Toc508374587)

# Tehtävän kuvaus

Aiheemme on yksinkertainen komentorivi-peli toteutettuna Java-kielellä, jossa ohjelma voi tallentaa senhetkisen pelitilan ja parhaimmat saadut loppupisteet. Pelin etenemisessä syötteenä käytetään yksinkertaisia kokonaislukuja, joille on listattu jokin toiminto.

Pelin nimi on Kuningaskunta-simulaatio ja siinä tehdään kuninkaana päätöksiä, jotka vaikuttavat alamaisiin. Alamaisina on olemassa erilaisen tyyppisiä sukuja, kuten aatelisia, maagisia, uskonnollisia jne. joiden tyyppi vaikuttaa siihen, miten ne käyttäytyvät.

Jokaisella suvulla on suhde kaikkiin muihin sukuihin ja kuninkaaseen. Kuningaskunnassa on myös resursseja kuten ruoka ja raha, joita voidaan käyttää ratkaistaessa ongelmia. Ongelmien ratkaisuissa myös kuninkaan suhde eri sukuihin on yksi eräänlainen resurssi, joka mahdollistaa esimerkiksi, että sotilassuvulla voidaan käyttää sotilasvaltaa ongelmien ratkaisemiseksi.

Peli etenee niin, että eteen tulee erilaisia ongelmia, joihin pyydetään kuninkaalta ratkaisupäätöstä ja valittu päätös vaikuttaa kuningaskunnan sukuihin tai resursseihin. Peli loppuu, kun vapaavalintaisen pitkä valtakausi päättyy tai kun kuninkaana olo jostain muusta syystä päättyy. Pelin lopuksi lasketaan pisteet siitä, kuinka hyvässä kunnossa kuningaskunta on valtakauden päättyessä. Hyväksi kuningaskunnaksi katsotaan, kun sukujen suhteet kuninkaaseen on hyvät ja kansalaisia on vielä hengissä mahdollisimman paljon.

# Ratkaisuperiaate

Kuningaskuntaa pystytään mallintamaan koostumaan kuninkaasta ja kuningaskunnan kansalaisista, mutta jokaisen ihmisen vuorovaikutus kaikkiin muihin ihmisiin voisi olla liian monimutkaista kuvata ja läheskään kaikki ihmiset eivät pääse kuninkaan vastaanotolle. Lisäksi useamman ihmisen kohtaaminen kuninkaan kanssa samaan aikaan on vaikeaa kuvata tekstipohjaisesti.

## Pelitilan kuvaaminen

Pelimme muodostaa mallin kuningaskunnan kansalaisista niputtamalla nämä kaikki sukuihin. Suvun ajatellaan koostuvan usean perheen yhdistelmästä ja muodostaa kiinteän yhteisön, jonka suhdetta kuninkaaseen voidaan kuvailla yhdellä luvulla.



Kuva 1: Pelin entiteettien vuorovaikutus toisiinsa ER-kaaviona

Jotta yhteisö tuntuu konkreettisemmalta, kirjoitimme myös jokaiselle suvulle edustajan ja kuinka monta ihmistä kuuluu sukuun. Edustajan tarvitsemme, jos haluamme kuninkaan rankaisusta henkilökohtaisempia ja edustaja tulee aina kuninkaan vastaanotolle kertomaan oman sukunsa kohtaaman ongelman. Suvun väestön ammattikuntaa, asemaa ja koulutusta kuvaamme yksinkertaisesti sukutyypeillä:

1. Aatelinen
2. Maaginen
3. Sotilaallinen
4. Uskonnollinen
5. Kauppias
6. Maalainen

Halusimme tehdä joistain suvuista enemmän maagisia, kuin toiset suvut ja sukutyyppien yhdistelmistä mahdollisia, niin emme voineet toteuttaa sukutyyppiä yksinkertaisena enum-arvona. Päädyimme kuvaamaan suvun tyyppiä roolipeleistä tutulla tavalla asettamalla jokaiselle tyypille oman kokonaisluvun.

Pelaajalle kokonaislukujen esittäminen on kuitenkin melko huono pelikokemus, joten jokaiselle suvulle arvomme nimen, joka viittaa siihen mikä suvun tyyppi on kyseessä. Jotkut sukuyhdistelmät kuitenkin ovat mahdottomia, kuten esimerkiksi suku ei voi olla yhtä aikaa maalainen ja aatelinen, joten eristimme aateliset pois sukuyhdistelmistä ja jokainen peli silloin sisältää tasan kaksi aatelissukua, jotka eivät ole mitään muuta tyyppiä.

## Kuninkaan vastaanoton kuvaaminen

# Ongelma ja sen osien kuvaaminen

Olemme jakaneet pelin neljään pääluokkaan:

1. Kuningaskunta (main)
2. Kuningas
3. Suku
4. Ongelma
5. TallennaLataaPisteet

Seuraavaksi esitetään lyhyesti mitä kukin luokka tekee.

## Kuningaskunta ja main-metodi

Kuningaskunta-luokassa on main-metodi, jossa pelaajalta kysytään ensiksi mitä halutaan tehdä. Pelaaja voi valita uuden pelin, edellisen pelin, nähdä pisteet jne.

## Suku-luokka

Suku-luokkaan kuuluu private atribuutit:

int suhdeKuninkaaseen;

HashMap<Suku, Integer> suhteet

String nimi;

String edustaja;

int populaatio;

int aatelisuus;

int magia;

int sotilaallinen;

int uskonnollinen;

int kauppias;

int maalainen;

Joilla kuvataan suvun suhde kuninkaaseen, muihin sukuihin ja mitkä ovat suvun nimi, edustaja ja populaatio. Loput attribuutit kuvaavat sukutyyppiä. Suku()-konstruktori asettaa kaikkiin Integer-tyyppisiin attribuutteihin nollan ja arpoo etunimen edustajalle, muut arvot asetetaan suvulle Kuningas-luokassa metodissa lisaaSuku(), jotta suvut voidaan arpoa keskenään tasapuolisesti.

Suvuilla on getterit ja setterit muodossa anna<atribuutin nimi>() ja aseta<atribuutin nimi>(), joissa myös rajoitetaan samalla suurimmat mahdolliset arvot. Suvulla on myös toString() -metodi, joka tulostaa kuvauksen suvusta sanallisesti. Kuvaus on sanallinen, joten esimerkiksi tarkemman suhteen muihin sukuihin saa tulostaSuhteet() –metodilla joka tulostaa komentoriville suvun suhteet.

suhteet –hashmappiin tallennetaan suvun suhde kaikkiin muihin sukuihin. Tämä tehdään luokassa Kuningas metodissa generoiSukuSuhteet() .

## Kuningas, pelitila ja vuorokierto

Kuningasluokkaan kuuluu attribuutit:

private String nimi;

private int raha;

private int ruoka;

private int rahaTuotto;

private int ruokaTuotto;

private int vuorot;

private int sukujenLKM;

private boolean havitty = false;

transient private Scanner vastaus;

public ArrayList<Ongelma> ongelmat

public ArrayList<Suku> suvut

Joilla kuvataan kuninkaan nimi, resurssit, kuinka paljon sukuja kuuluu kuninkaan alaisiksi, onko kuningas syösty vallassa ja millaisia ongelmia ja sukuja kuninkaalla on. Näiden lisäksi kuninkaaseen on myös tallennettu kaikki mahdolliset sukunimet, koska kuninkaassa on metodit lisaaAatelisSuku() ja lisaaSuku(). Näitä tarkastellaan luvussa sukuihin liittyvät metodit.

Kuningas –konstruktorissa annetaan kuninkaalle nimi, vuorojen määrä, arvotaan kuinka paljon on resursseja, asetetaan sukujen lukumääräksi 25 ja lisätään tämän verran sukuja. Lopuksi kuninkaan suvuille generoidaan myös sukusuhteet.

### vuorokiertoon liittyvät metodit

### Pisteiden laskun metodit

Pelin pisteet näytetään pelaajalle jokaisen vuoron lopussa, ja lisätään pelin lopussa listaan jos ne ovat tarpeeksi hyvät. Pisteiden laskemiseen käytetään metodia:

annaPisteet()

Metodi huomioi pelin senhetkiset resurssit, niiden tuotot, aatelisten tyytyväisyyden, kaikkien sukujen tyytyväisyyden, populaation koon ja onko peli hävitty. Se käyttää hyväkseen kuninkaan metodeja:

annaRaha()

annaRahaTuotto()

annaRuoka()

annaRuokaTuotto()

annaSuhteellinenVakimaara()

Sekä suvun metodeja:

annaAatelisuus()

annaSuhdeKuninkaaseen()

Ennen pisteiden palautusta tarkistetaan vielä onko peli hävitty booleanilla havitty, jos näin on vähennetään pisteistä 80%.

### Sukuihin liittyvät metodit

Suvut lisätään kuninkaalle käyttäen tyhjää suvun konstruktoria, jotta sukujen ominaisuudet voidaan ripotella tasaisesti, eikä yhdellekään suvulle tule erittäin poikkeavia arvoja. Suvut lisätään käyttämällä metodeita:

lisaaSuku()

lisaaAatelisinSuku()

Aatelissuvut lisätään erikseen ja muiden sukujen kohdalla arvotaan mitä kahta tyyppiä halutaan kunkin suvun kohdalla luoda. Riippuen näistä kahdesta sukutyypistä muodostetaan suvun nimi käyttämällä hakua:

magianimet.get(r.nextInt(magianimet.size()));

Joka käytännössä ottaa sattumanvaraisen nimen (tässä tapauksessa magianimistä). Nyt sukujen nimet koostuvat kahdesta osasta, jolloin on todennäköistä, että jokaisella pelikerralla kuninkaalle lisätään uniikit, eli erinimiset, suvut.

Näiden lisäksi sukuja voidaan etsia metodeilla:

annaAatelisin()

etsiSukuTyypit(boolean,boolean,boolean,boolean,boolean)

etsiSukuKombo(boolean,boolean,boolean,boolean,boolean)

Näitä hakuja käytetään muodostettaessa ongelmia, jolloin tarvitaan suvut –listasta valikoida aina tietyntyyppisiä sukuja. Kaikki haut palauttavat ArrayList<Suku> -listan, joka on suosituimmuusjärjestyksessä. Eli jos haluttaisiin vaikkapa viitata **toiseksi suosituimpaan** aatelissukuun, niin se voitaisiin tehdä kutsumalla:

annaAatelisin().get(**1**)

Boolean –arvot parametreinä tarkoittavat, että mitä tyyppiä haetaan. Haettavan tyypin kohdalla arvoksi laitetaan true . Tyyppien järjestys on aina kaikkialla sama:

magia

sotilas

uskonnollinen

kauppias

maalainen

Esimerkiksi kaikki suvut joiden tyyppi on joko kauppias TAI maalainen haettaisiin näin:

etsiSukuTyypit(false, false, false, true, true)

jos halutaan etsiä suvut jotka ovat ainoastaan kauppiaita JA maalaisia, käytetään kombo-metodia:

etsiSukuKombo(false, false, false, true, true)

Sukujen väliset suhteet lisätään Kuningas –konstruktorissa käyttämällä generoiSukuSuhteet() –metodia, joka asettaa sattumanvaraisen sukusuhteen jokaiselle suvulle jokaista sukua kohti.

## Ongelma-luokan päätökset, vaatimukset ja seuraukset

Ongelmalla kuvataan...

## TallennaLataaPisteet-luokka

TallennaLataaPisteet-luokan vastuulla on tallentaa pelitila myöhempää käyttöä varten ja pitää kirjaa kaikista parhaimmista kuningaskuntien pisteistä.

### Pelitilan lataus ja tallennus

### Pisteet

# Testausjärjestely

Kuningaskunta-Simulaatiota testattiin pääsääntöisesti kolmessa vaiheessa. Ensimmäiseksi testattiin, että pelitila on olemassa ja se on luotu oikein käyttäen erilaisia satunnaisia elementtejä. Toiseksi testattiin, että vuorokierto pelin edetessä toimii ja pelissä voidaan kohdata ongelmia. Kolmanneksi testattiin, että pelin lopuksi voidaan laskea pisteitä ja pelitila voidaan tallentaa.

## Sukujen ja kuninkaan testaaminen

Kuninkaan ja sukujen luominen toteutettiin ennen vuorokierron toteuttamista. Sukujen ja kuninkaan sisältö tarkistettiin yksinkertaisesti tulostamalla .toString() –metodin avulla kaikki suvun tiedot. Suku sisältää paljon tietoa ja erityisesti suvun suhteet muihin sukuihin on 25 rivinen tulostus. Sukujen tulostusta on muutettu jälkeenpäin niin, että toString() -metodissa tulostetaan ainoastaan suvut joihin on olemassa huonoin ja parhain suhde. Tällaista tulostusta on myös kätevä käyttää, jos peliin jatkossa kirjoitetaan keino pelaajalle selvittää suvuista tietoa.

Sukujen luomisen jälkeen on helppo tarkistaa saako kuningas (pelaaja) sopivat arvot sisäänsä.

## Vuorokierto ja ongelmat

Vuorokierto on saatu toimimaan rakentamalla yksi ongelma, jossa on päätökset ja päätöksillä seuraukset. Päätöksillä voi olla myös vaatimuksia, jotka tulisi testata. Ensimmäinen testattava ongelma tulee sisältää siis mahdollisimman laajasti tarvittavia osia. Valitsimme ensimmäiseksi lisättäväksi ongelmaksi maanjäristyksen, joka vaikuttaa kuningaskunnassa laajasti ja vaatii monipuolisesti seurauksien ja vaatimusten eri osia.

## Pisteet ja pelin tallennus

# Liitteet