TKO\_2005 Olio-ohjelmoinnin perusteet 2018

Ohjaaja: Erkki Kaila

Harjoitustyödokumentaatio

Kuningaskunta-Simulaatio

Tommi Heikkinen, 517749, thheik@utu.fi

Santeri Loitomaa, 516587, sjmloi@utu.fi

Pasi Toivanen,

Sisällysluettelo

[1 Tehtävän kuvaus 1](#_Toc508622135)

[2 Ratkaisuperiaate 2](#_Toc508622136)

[2.1 Pelitilan kuvaaminen 2](#_Toc508622137)

[2.2 Kuninkaan vastaanoton kuvaaminen 3](#_Toc508622138)

[3 Ongelma ja sen osien kuvaaminen 4](#_Toc508622139)

[3.1 Kuningaskunta ja main-metodi 4](#_Toc508622140)

[3.2 Suku-luokka 4](#_Toc508622141)

[3.3 Kuningas ja pelitila 4](#_Toc508622142)

[3.3.1 Vuorokierto -metodi 5](#_Toc508622143)

[3.3.2 Pisteiden laskun metodit 5](#_Toc508622144)

[3.3.3 Sukuihin liittyvät metodit 5](#_Toc508622145)

[3.4 Ongelma-luokan päätökset, vaatimukset ja seuraukset 6](#_Toc508622146)

[3.5 TallennaLataaPisteet-luokka 7](#_Toc508622147)

[3.5.1 Pelitilan lataus ja tallennus (Santeri) 7](#_Toc508622148)

[3.5.2 Pisteet 7](#_Toc508622149)

[4 Testausjärjestely 8](#_Toc508622150)

[4.1 Sukujen ja kuninkaan testaaminen 8](#_Toc508622151)

[4.2 Vuorokierto ja ongelmat 8](#_Toc508622152)

[4.3 Pisteet ja pelin tallennus 8](#_Toc508622153)

[5 Liitteet 9](#_Toc508622154)

# 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

Kuninkaan vastaanotto on pelin ”päänäkymä”. Pelin alussa valitaan valtakauden pituus vuosina, ja joka vuosi tulee 12 ongelmaa (1 per kuukausi) joihin sinä kuninkaana vastaat. Jokaisesta ongelmasta tulee sinulle kertomaan jonkin suvun edustaja, yleensä edustajan suku on ongelman uhri tai muulla tavalla osallinen ongelmassa. Ongelmat vaihtelevat vaikutuksiltaan suuresti, ja koko peli koostuu näiden ongelmien ratkomisesta.

Kuninkaana sinulla on absoluuttinen valta päättää, miten reagoit ongelmaan, ja peli tarjoaa 1-5 vaihtoehtoa joista saat valita toimintosi. Osa vaihtoehdoista jää välillä näkymättömiksi, koska sinulla ei välttämättä ole tarpeellisia resursseja, tai muita vaatimuksia, jotta voisit ratkaista ongelman haluamallasi tavalla. Harva ratkaisu miellyttää kaikkia, ja yleensä voit olettaa ainakin jonkun suuttuvan tavasta jolla hoidit sukujen välisen kriisin. Tärkeää onkin löytää tasapaino hyvien suhteiden, resurssien ja tuotannon välillä, sillä nämä kaikki vaikuttavat lopullisiin pisteisiisi. Peli tulostaa aina ongelman ratkaisun jälkeen pienen selostuksen lopputuloksesta ja kertoo, kuka piti ratkaisusta ja kuka ei.

Kuukauden päätteeksi saat neuvonantajiltasi valtakunnan tilanneraportin, joka kertoo ruokasi, rahasi, niiden tuotot ja yleiset pisteesi, joka kuvastaa onnistumistasi johtajana.

# 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 ja pelitila

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.

### Vuorokierto -metodi

Vuorokiertoa kutsutaan (Kuningaskunta) main-metodissa, johon on kirjoitettu while –lausekkeen sisään kaikki muut pelin toiminnallisuudet. Vuorokierto alkaa tarkistamalla, että ongelmalista on ladattu täyteen ongelmia, jonka jälkeen sattumanvarainen ongelma valitaan vuorolle. Ongelma ja sen mahdolliset päätökset tulostetaan kutsumalla vuoroOngelma.tulosta(this), jossa this viittaa sen hetkiseen kuninkaaseen. Pelaaja syöttää päätösnumeron, jonka jälkeen lasketaan pisteet ja tarkistetaan ettei peli pääty. Jos peli jatkuu, pelaaja valitsee, haluaako jatka peliä vai lopettaako/tallentaako pelin seuraavalle kerralle.

Jos peliä jatketaan, ongelmalistasta poistetaan käytetty vuoro-ongelma, jotta sama ongelma ei toistu usein.

### 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%. Pelissä parhaimmat pisteet saa siis, kun valtakausi loppuu ennalta sovitusti, eikä kesken pelin.

### 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

Ongelmat luodaan pelissä meilläOnOngelmia() –metodilla vuorokierrossa. Yksi ongelma koostuu erilaisista päätöksistä, jotka kuvaavat yhtä mahdollista valintaa jonka pelaaja voi ongelman kohdatessa tehdä. Jokaiselle päätökselle on ainakin yksi vaatimus tai se voidaan jättää tyhjäksi kirjoittamalla:

new Vaatimus(Tyyppi.NULL, 0)

Jokaisella päätöksellä on useita seurauksia. Seurauslistan jokainen jäsen toteutuu, kun päätös on valittu. (Kts kuva)



Kuva 2: Kaavio ongelman rakenteesta

Ongelman luominen näyttä meilläOnOngelmia() –metodissa tällaiselta:

kunkku.ongelmat.add(

new Ongelma("Barbaarihyökkäys", randomRaakalaisnimi + " on hyökännyt " + x.annaNimi() + " suvun farmille!", x, paatokset)

);

Jossa x on jokin kohdesuku. paatokset –lista muodostetaan seuraavalla tavalla:

paatokset.add(

new Paatos(new Vaatimus[] {new

Vaatimus(Tyyppi.SUKUSUHDE,3,kunkku.etsiSukuTyypit(false,true,false,false,

false).get(0)) },

             new Seuraus[]{

                  new Seuraus(Tyyppi.RAHA, 10),

                  new Seuraus(Tyyppi.SUKUSUHDE, +25, kunkku.etsiSukuTyypit(false, false,

false, true, false))

             },

"Sotilaat ottavat raakalaiset kiinni ja antavat heidät kauppiaille myytäväksi orjakaupassa",

"Orjien myynnistä saadaan paljon rahaa ja kauppiaat hierovat käsiään tyytyväisenä."

            )

        );

Päätökseen merkataan, millainen vaatimus päätökselle on. Jos vaatimus ei täyty, niin päätöstä ei ole mahdollista tehdä kyseisellä pelivuorolla. Seuraukset ja vaatimukset tarkistetaan riippuen sen tyypistä, mikä tallennetaan enum muuttujana. Seurauksiin listataan esimerkiksi tässä tapauksessa, että kuninkaan rahat lisääntyvät kymmenellä ja sukusuhde kauppiaisiin paranee.

Kaikki erilaiset ongelmat on listattu kuningaskunnan lopussa meillaOnOngelmia() –metodissa.

## TallennaLataaPisteet-luokka

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

### Pelitilan lataus ja tallennus

Pelitilan tallennus toimii siten, että Kuningas-oliosta kutsutaan TallennaLataaPisteet luokan tallenna(Kuningas kunkku)-metodia arvolla this. Tila tallennetaan tiedostoon tallennus.tal FileOutputStream- ja ObjectOutputStream-olioita käyttäen. Lataus taas toimii päinvastaisella tavalla käyttäen FileInputStream- ja ObjectInputStream-olioita. Peli tallentaa automaattisesti vuoron lopussa ja pelin vanha tila on ladattavissa päävalikosta.

### Pisteet

Pelissä saadut pisteet lasketaan metodilla:

annaPisteet()

Suurin osa pisteistä tulee kansasi tyytyväisyydestä, mutta myös resursseilla ja erityisesti tuotoilla on väliä. Pisteet lasketaan kaavalla; Ruoka ja raha \*3 + näiden tuotot \*20 + aatelissukujen mielipide kuninkaasta \*5 + sukujen mielipide kuninkaasta, jos negatiivinen on vähennys puolet. Lopuksi kerrotaan lukema suhteellisella väkimäärällä, eli jos kuningaskuntasi on isompi kuin alussa saat bonusta ja jos se on pienentynyt, vähenee pisteesi. Jos peli päättyy ennen aikojaan johonkin katkaisuun kuten resurssipulaloppuun yms. menetät 80% pisteistäsi.

Koska sukusuhteet voivat olla välillä -100 & 100, on negatiivisille laitettu kertoimeksi ½ sen aiheuttaman haitan vähentämiseksi, kaikkia ei voi kuitenkaan miellyttää. Positiivisille arvoille ei anneta kertoimia sillä välillä 0-100 voivat tyytyväisyydet nostaa pisteitäsi jo huimasti. Tästä poikkeus on aatelissuvut, jotka ovat olemassa lähinnä pisteiden nostamisen vuoksi. Heitä ei kannata suututtaa, sillä heidän vaikutusvaltansa voi laskea pisteitäsi paljon. Ruuan ja rahan kerroin on olemassa vain, jotta niille saisi enemmän painoarvoa, resurssien tuotot nostavat pisteitä huimasti, sillä tulevaisuuden kannalta on kuningaskunnalla syytä olla tuottoja. Populaation kasvu nähdään myös etuna ja siksi se toimii kertoimena. Valtakunnan hajotessa ei kuningaskunnasta jää kuin huhuja, joten pisteesi tippuvat romahtaen.

Pisteiden tallennus toimii samankaltaisesti kuin tallennus ja latauskin mutta Kuningas-olion sijaan käytetään Pisteet-oliota, joka pitää kirjaa kymmenen parhaan pelaajan nimistä ja pistemääristä kahdessa Arrayssa. Pisteitä lisätessä täytyy suorittaa pitkähkö algoritmi, jotta oikeat pisteet päätyvät listalle, pisteet pysyvät laskevassa järjestyksessä ja pelaajien nimet eivät mene sekaisin. Pisteet lisätään listalle valtakauden päätyttyä, mikäli ne sinne asti riittävät. Lista on nähtävissä valtakauden lopussa ja päävalikossa, josta sen voi myös nollata takaisin oletusarvoille.

# 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

Pisteiden lasku on testattu yksinkertaisesti pelaamalla paljon eri tyylisiä testipelejä ja katsomalla, ettei mitkään osat pelikerroista aiheuta kohtuuttomia muutoksia pisteisiin. Attribuuttien maksimi ja minimiarvot on laskettu aritmeettisesti ennen testausta. Pisteiden tallennus toimi alun perin HashMappinä mutta koska tämä johti ongelmiin, se muutettiin olioksi, jolla on 2 Arrayta.

Lopuksi, kun suurin osa ongelmista on valmiina, on pelin testaamista jatkettu yksinkertaisesti pelaamalla peliä paljon ja etsimällä epäloogisuuksia. Yksi suurimpia virheitä pelin rakenteessa on ollut liian negatiivisten ongelmien luonti. Esimerkiksi lohikäärmeen vaikutus pelimekaniikkaan on ollut alustavasti niin suuri, että peli on usein loppunut jokaisen lohikäärmeen hyökkäyksen jälkeen. Epäreilut tilanteet on pyritty tasapainottamaan keksimällä ongelmia, jotka vaikuttavat pelaajan pistetilaan positiivisemmin ja heikentämällä pisteitä vähentävien ongelmien seurauksia.

# Liitteet

1. <https://github.com/Santtinen/Kuningaskunta-Simulaatio/> (peli ja sen lähdekoodi)

B. Lista vastuualueista

Tommi, Sukujen hallinta, pisteenlasku ja pelin päättyminen, ongelmien luominen

Pasi, Sukujen haku, tietorakenne ja hallinta, ongelmien luominen, harjoitustyön dokumentin muotoilu

Santeri, Ajettavan version tekeminen, vuorokierto ja sen vaikutus pelitilaan, tallennus, lataus ja pisteet, virheenkorjaus, palautus