Patrik Ahvenainen / 013326292
patrik.ahvenainen@helsinki.fi
Karstulantie 8 C 43
00550 Helsinki

# Yleinen noppapeli

## Toteutusdokumentti

## Python 2.4

Helsingin yliopisto
Tietojenkäsittelytieteen laitos
Ohjelmoinnin harjoitustyö (python)
Ohjaaja: Joel Rybicki

Palautuspäivä: 04.05.2010

# Sisältö

# Käyttöohje

## Esittely

Ohjelma on viihdekäyttöön tarkoitettu peli yhdelle tai useammalle ihmispelaajalle. Pelin alussa pelaajat sopivat mitkä yhdistelmät peliin otetaan käyttöön, kuinka monta noppaa pelissä on, mikä on noppien suurin sallittu silmäluku sekä kuinka monta heittoa yhdellä vuorolla voi suorittaa. Pelaaminen tapahtuu joko hiirellä tai näppäimistöllä.

## Ajo-ohje

Pelin tiedostot ovat kahdessa paketissa eli kansiossa 'GUI' ja 'Game'. Pelissä käytettävät luokat ovat suurimmaksi osaksi samannimisten moduulien (tiedostojen) sisällä (Taulukko 1). Tiedostorakenteen tulisi olla esitetyn mukainen.

Pelin graafinen käyttöliittymä käynnistetään ajamalla Main.py -tiedosto komennolla

```
python Main.py
```

Komennon ”python” tulisi ajaa ohjelma python-tulkin (version 2.4) läpi.

**Taulukko 1. Graafisen käyttöliittymän ajamiseen tarvittavat paketit, moduulit ja luokat ja niiden keskinäinen rakenne.**

| Paketti | Moduuli | Luokka |
|---|---|---|
| Game | Combination | Combination |
| | Dice | Dice |
| | Game | Game |
| | HighScores | HighScores |
| | Logger | Logger |
| | Player | Player |
| | ScoreList | ScoreList |
| | Settings | Settings |
| | Throw | Throw |
| GUI | DiceRow | DiceRow |
| | GameState | GameState |
| | GUI | - |
| | Main | - |
| | ScoreBoard | ScoreBoard |
| | | ScoreBoardScore |
| | | TotalScore |
| | SimpleCallback | SimpleCallback |

## Laitteistovaatimukset

Ohjelma on suunniteltu toimimaan pythonin versiolla 2.4. Toimivuus muilla versioilla on mahdollista, mutta ei taattua. Ohjelman tulee kyetä luomaan kansioita siihen kansioon, mistä sitä ajetaan ja luomaan näihin

kansioihin tiedostoja. Tiedostot ovat kooltaan melko pieniä, eikä paljon kovalevytilaa siis tarvita. Ohjelman kokonaiskoko on alle 1 MB. Näytön suositusresoluutio on vähintään 1024x800.

Pelin ajamiseen tarvitaan lisäksi Tkinter-moduulia. Lisäksi käytetään muutamaan Tkinteriin liittyvää moduulia: `tkFileDialog`, `tkMessageBox`, `tkFont`. Lisäksi joissakin ohjelman luokissa käytetään seuraavia moduuleja: `os`, `logging`, `logging.handlers`, `random`, `pickle`, `datetime`, `unittest`, `time`, `sys`.

Nämä moduulit ja python kuuluvat nykyisin monen käyttöjärjestelmän perusasennuksiin. Jos niitä ei ole valmiiksi asennettuina, ne löytyvät Internetistä vapaasti asennettavina useammalle käyttöjärjestelmälle.

Peliä on testattu osittain seuraavilla kokoonpanoilla:

**Asus G2Pc kannettava tietokone**: ***Näyttö***: 17" WXGA+ (1440 x 900); ***Prosessori***: Intel Core 2 Duo T7200 (2.0Ghz); ***Keskusmuisti***: 2x1GB DDR2; ***Näytönohjain***: ATI Mobility Radeon X1700 512MB; ***Käyttöjärjestelmä***: Ubuntu 9.10; ***Python-versiot***: 2.4 ja 2.6.

**Jimm's Pro Gamer GTX 295 pöytäkone**: ***Näyttö***: 24" (1920 x 1200); ***Prosessori***: Intel Core i7 920 2.66Ghz; ***Keskusmuisti***: 3x2GB Elite, DDR3 1.3Ghz; ***Näytönohjain***: GeForce GXT 295 1792MB GDDR2; ***Käyttöjärjestelmä***: Windows Vista Home Premium 64-bit; ***Python-versiot***: 2.4, 2.5 ja 2.6.

Lisäksi pelin toimivuutta on testattu SSH-yhteydellä tietojenkäsittelytieteiden laitoksen melkki-palvelimella (python 2.5) ja tietojenkäsittelytieteiden laitoksen tietokoneluokassa (python 2.6).

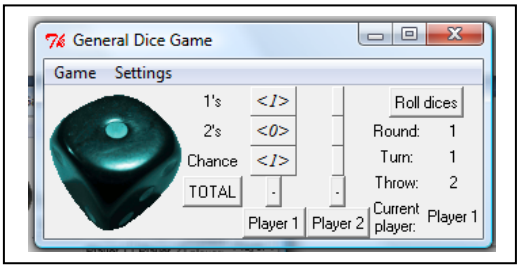
## Peli-idea

Kyseessä on noppapeli, jossa tarkoituksena on noppia heittämälle kerätä mahdollisimman ison kokonaispistemäärä. Jos pelaajia on useampia tarkoituksena on saada enemmän pisteitä kun muut kilpailijat. Yksinpelissä tarkoituksena on saada mahdollisimman paljon pisteitä ja päästä mahdollisimman korkealle kaikkien aikojen top-10 -listalla. Noppia voi heittää yhdellä heittovuorolla niin monta kerta kun pelin asetuksissa on määritetty (1-10). Ensimmäistä heittokertaa lukuun ottamatta pelaaja voi heittää vain osan nopista. Kaikkia heittovuoroja ei tarvitse käyttää. Kun pelaaja on saanut mieleisensä yhdistelmän tai heittovuorot ovat loppuneet, pelaaja merkitsee heittonsa jonkun yhdistelmän kohdalle ja saa kokonaispistesaldoonsa noppien silmälukujen oikeuttaman pistesaldon. Pelin voittaja on se, jolla on pelin lopussa suurin kokonaispistesaldo. Jos voittajan pistesaldo on suurempi kuin pelin asetuksia vastaavan top-10 -listan huonoin pistesaldo, voittajan pistesaldo lisätään top-10 -listalle.

## Ohjelman toiminnot

Pelissä esitellään seuraavat toiminnot:

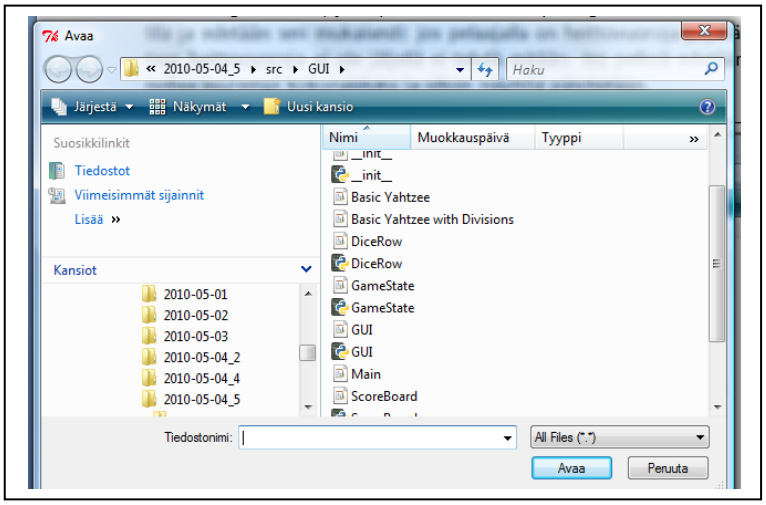
Nopan heittäminen ja uuden pelitiedoston avaaminen.




**Kuva 1. Nopan heittäminen. Kuvan noppa on lukittu.**

### *Nopan heittäminen*

Noppaa heitetään painamalla napista 'Roll dices', valitsemalla valikosta 'Game' kohta 'Roll dices' tai painamalla näppäintä r (Kuva 1). Nämä kaikki toiminnot kutsuvat GameState-luokan root.states-olion funktiota goForward, joka puolestaan kutsuu pelin goForward funktiota. Tässä funktiossa tarkastetaan pelin tila ja edetään sen mukaisesti: jos pelaajalla on heittovuoroja jäljellä heitetään lukitsemattomia noppia, jos taas heittovuoroja ei ole jäljellä ei tehdä mitään. Jos pelissä edetään palautuu pelin goForward-funktiosta nollaa suurempi kokonaisluku ja silloin näyttöä päivitetään.




**Kuva 2. Uuden pelitiedoston avaaminen (Windows Vista).**

### *Uuden pelitiedoston avaaminen*

Uusi pelitiedosto avataan valitsemalla 'Settings' valikosta kohta 'New Settings' (Kuva 1). Avautuvan valikon ulkonäkö riippuu käyttöjärjestelmästä (Kuva 2). Valikko tuotetaan tkFileDialog.askopenfilename-funktiolla, joka palauttaa tiedoston nimen. Pelin asetusten (game.settings) new_settings -funktiota kutsutaan tällä tiedostonnimellä. Tästä tiedostosta yritetään lukea yhdistelmiä. Jos yhtään yhdistelmää ei voida lukea tästä tiedostosta avataan oletustiedosto. Tämän jälkeen graafisen käyttöliittymän puolella tarkistetaan tuliko virheitä, ja jos tuli ne esitetään pelaajalle checkForErrors-funktion avulla tkMessagebox.showerror-funktiolla. Mahdolliset virheet on esitetty kohdassa virheilmoitukset (Taulukko 3, kolme ensimmäistä virheilmoitusta).

## Syöttötiedot

Suurin osa käyttäjän antamasta syöttötiedosta on "idioottivarmaa" eli käyttäjä ei voi kaataa peliä tai antaa (pelin toiminnan kannalta) virheellistä syötettä. Pelissä edetään nappuloita painamalla. Pelaajan heittotavat on esitelty aiemmin kohdassa 'Ohjelman toiminnot' - 'Nopan heittäminen'. Pelaaja valitsee yhdistelmän joko painamalla omaan tulostaulukon tulostaan hiirellä tai siirtymällä "TAB"-näppäimellä omien

käyttämättömien tulostaulukon tulosten välillä. Pelaaja voi käyttää myös seuraavia pikanäppäimiä pelissä etenemiseen: 1,..,n (lukitse noppa 1,..,n); r (heitä); Escape, control-x, control-q (lopeta); control-n (uusi peli); control-p (uudet pelaajat) ja control-h (Top-10 -lista).

## Tulostetiedot

Ohjelmaan kuuluu luokka Logger, jossa luodaan pelin lokitiedosto. Pelin lokitiedostoon kirjoitetaan joitakin pelin tärkeimpiä tapahtumia. Lisäksi on olemassa toinen lokitiedosto, joihin kirjoitetaan ainostaan peliin liittyvät virheilmoitukset ja varoitukset. Nämä kummatkin ovat kansiossa 'log'.

Lisäksi kansiossa 'Scores' sijaitsevat pelin top-10 -listat. Top-10 -listalle kirjoitetaan joka kerta kun peli päättyy. Pelin loputtua avataan kyseisen pelin top-10 -lista, jos sellainen on ja luetaan sen sisältö. Tämän jälkeen pelaajan pistesaldo lisätään listalle. Jos pelaajan tulos ei mahdu kymmenen parhaan joukkoon, poistetaan se listan lopusta. Lista tallennetaan pythonin pickle-moduulin avulla suoraan kirjastona (dict) ja luetaan kirjastona tiedostosta.

**Taulukko 2. Peliin asetettuja rajoituksia noppien lukumäärälle (n), noppien maksimisilmäluvulle (s) ja heittojen lukumäärälle (h) sekä näiden parametrien oletusarvot.**

| Parametrin nimi | Arvo | Tulkinta |
|---|---|---|
| **NMIN** | 1 | Pienin mahdollinen määrä noppia, joita pelissä voi esiintyä. |
| **NMAX** | 9 | Suurin mahdollinen määrä noppia, joita pelissä voi esiintyä. |
| **DEFAULT_N** | 5 | Jos pelin asetustiedostossa ei anneta noppien lukumäärää, tai annettu lukumäärä n ei ole välillä NMIN≤n≤NMAX, käytetään tätä noppien lukumäärää |
| **SMIN** | 2 | Pienin mahdollinen noppien maksimisilmäluku, joka pelissä voi esiintyä. |
| **SMAX** | 9 | Suurin mahdollinen noppien maksimisilmäluku, joka pelissä voi esiintyä. |
| **DEFAULT_S** | 6 | Jos pelin asetustiedostossa ei anneta noppien maksimisilmälukua, tai annettu lukumäärä s ei ole välillä SMIN≤s≤SMAX, käytetään tätä noppien maksimisilmälukua |
| **HMIN** | 1 | Pienin mahdollinen heittojen lukumäärä kierrosta kohti, joka pelissä voi esiintyä. |
| **HMAX** | 10 | Suurin mahdollinen heittojen lukumäärä kierrosta kohti, joka pelissä voi esiintyä. |
| **DEFAULT_H** | 3 | Jos pelin asetustiedostossa ei anneta heittojen lukumäärää kierrosta kohti, tai annettu lukumäärä h ei ole välillä HMIN≤h≤HMAX, käytetään tätä heittojen lukumäärää kierrosta kohti |

## Ohjelman rajoitukset

Pelille on annettu asetustiedostossa muutama parametri, jotka rajoittavat pelin toimintaa (Taulukko 2).

Tämän lisäksi pelille voidaan antaa parametrinä suurin mahdollinen yhdistelmien lukumäärä, joka on sallittu. Graafisen käyttöliittymän käynnistyksen yhteydessä luotavalla pelillä tämä arvo on 30 eli pelissä on korkeintaan 30 yhdistelmää. Tämä tarkoittaa, että loput yhdistelmistä jätetään suoraan pois.

Graafinen käyttöliittymä lisää rajoituksia myös pelaajien nimiin: pelaajien nimet eivät voi olla tyhjiä tai sisältää ainoastaan välimerkkejä (white space) ja niiden maksimipituus on 21 merkkiä.

**Taulukko 3. Tärkeimmät pelissä esiintyvät virheilmoitukset. Virheilmoitukset kirjoitetaan lokitiedostoon, tulostetaan päätteelle ja esitetään loppukäyttäjälle ponnahdusikkunoina.**

| Virheilmoitus | Selitys | Toiminta |
|---|---|---|
| `No dice combinations found from file '/home/pate/workspace/Yleinen noppapeli/src/GUI/DiceRow.pyc'! Loading default settings 'Basic Yahtzee.dat'.` | Pelille annetussa asetustiedostossa ei ole yhtään yhdistelmää. | Peli lataa automaattisesti oletusasetustiedoston. Jos haluat käyttää valitsemaasi tiedostoa, valitse uusi asetus-tiedosto tai lisää valitsemaasi asetustiedostoon yhdistelmiä. |
| `Given parameter for the number of dice (7e) is not a valid integer. Default value of 5 is used instead.` | Pelin asetustiedostossa oleva noppien lukumäärää olevaa parametria #n# seuraava merkkijono (7e) ei ole kään-nettävissä kokonaisluvuksi. | Korjaa asetustiedostossa olevan parametrin lukuarvoa. Tässä tapauksessa riittää poistaa e luvun 7 perästä. |
| `In the setting file '/home/pate/workspace/Yleinen noppapeli/src/GUI/Settings2.dat' there were no combinations that could be used with current settings. Loading default settings 'Basic Yahtzee.dat'.` | Pelille annetussa asetustiedostossa oli yhdistelmiä, mutta mitään niistä ei voitu ottaa käyttöön nykyisillä noppien luku-määrällä, maksimisilmä-luvulla ja heittojen luku-määrällä. | Peli lataa automaattisesti oletusasetustiedoston. Jos haluat käyttää valitsemaasi tiedostoa, poista tai muuta yhdistelmien perässä olevia rajoitteita. |
| *`No code Found for code '123123'! Zero is returned.`* | Valitun yhdistelmän koodia vastaavaa laskusääntöä ei löytynyt. | Tälle yhdistelmälle ei voi saada pisteitä ennen kuin koodia `123123` vastaava yhdistelmän laskusääntö lisätään Combinations-luokkaan. |
| *`You did not select any players!`* | Uusia pelaajia luotaessa käyttäjä ei ole valinnut (raksimalla) yhtään pelaajaa peliin tai kaikkien peliin valittujen pelaajien nimi on tyhjä. | Uusia pelaajia luotaessa, valitse pelaajille ei-tyhjä korkeintaan 21-merkkinen nimi (GUI:n rajoitus). Valitse vähintään yksi pelaaja raksimalla pelaajan nimen vieressä oleva laatikko. |

## Virheilmoitukset

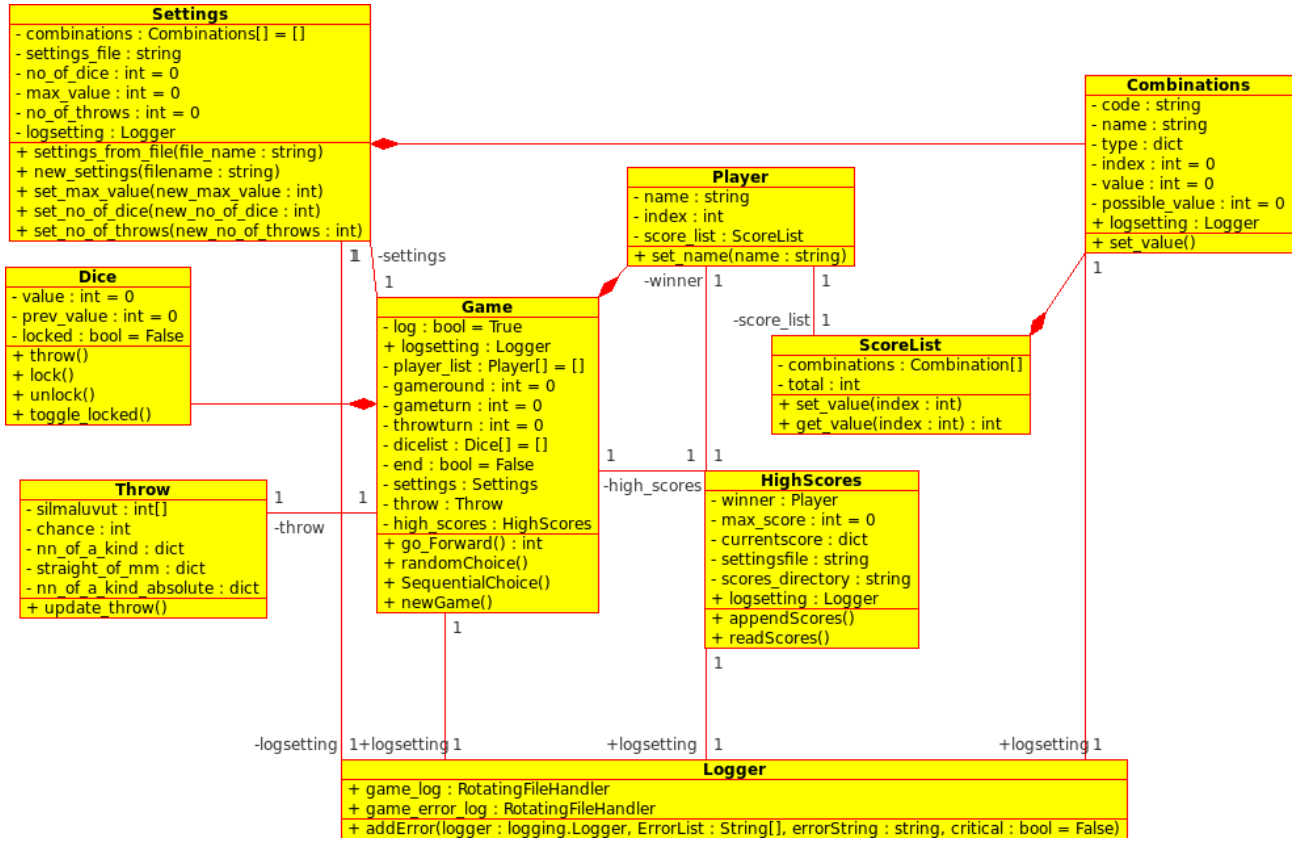
Suurin osa pelissä tapahtuvista virheistä korjataan valitsemalla huonolle parametrille parempi tai oletusarvo tai hylkäämällä huonot arvot. Pelin asetuksia voi muuttaa vain ennen ensimmäistä pelikierrosta. Näistä ”virheistä” toivutaan ja niihin ei kiinnitetä huomiota muuta kuin kirjataan pelin lokitiedostoon. Pelissä voi kuitenkin tapahtua muutama vakavampi virhe, joista on syytä ilmoittaa pelaajalle. Nämä virheilmoitukset kerätään peli-luokan muuttujaan `unNotifiedErrors`, josta ne siirretään ilmoituksen

jälkeen muuttujaan `unNotifiedErrors`. Ohessa on näiden virheiden englanninkieliset virheilmoitukset, virheilmoitusten selitykset vaadittu toiminta (Taulukko 3).

# Ohjelman toiminta ja rakenne

## Yleiskuvaus

## Luokkarakenne




**Kuva 3.** UML-kaavio yleisen noppapelin peruskomponenttien keskenäisistä suhteista. Luokkien attribuutit ovat merkitty yksityisiksi (-). Yksityisiksi merkittyihin attribuutteihin pääsee luokan ulkopuolelta käsiksi laittamalla alaviiva attribuutin nimen eteen. Kaikilla yksityisillä attribuuteilla on myös getterit (get_<attribuutin_nimi>), jotka on toteutettu pythonin property-ominaisuuden avulla. Vain "julkiseen" käyttöön eli toisten luokkien käyttöön tarkoitetut funktiot on esitetty. Sisäiseen käyttöön tarkoitettuja funktioita ei ole esitetty. Käännökset alkuperäisestä suomenkielisestä suunnitelmasta on esitetty liitteessä 1.

Pelin rakenteesta tuli melkolailla alkuperäisen suunnitteludokumentin suunnitelman mukainen, jos kielen vaihtoa ei oteta huomioon (Kuva 3). Uusia luokkia tuohon suunnitelmaan verrattuna ovat luokat HighScores (Top-10 -lista) ja Logger (loki). Esitetyssä luokkakaaviossa on vain luokkien julkiseen, eli toisten Pelin luokkien käyttöön tarkoitetut funktiot. Seuraavaksi luokittain lyhyt kuvaus luokkien välisistä suhteista ja luokkien tarkoituksista.

### *Game (Peli)*

Peli-luokan funktiot on tarkoitettu pelin ulkopuolisen käyttöliittymän käyttöön. Kaikki luokat ovat riippuvaisia Peli-luokasta, eli sen kautta pääsee käsiksi kaikkiin pelissä oleviin olioihin siltä osin kun niihin on tarkoitus päästä käsiksi. Peli-luokan kautta edetään pelissä.

### *Dice (Noppa)*
Noppa on yksinkertainen luokka, mutta erittän tärkeä, jossa on tieto nopan silmäluvusta ja siitä, onko noppa lukittu. Vain lukitsematonta noppaa voi heittää. Nopan silmälukua (value) voi vaihtaa vain heittämällä noppaa, eli arpomalla nopalle uuden silmäluvun yhdestä nopan maksimisilmälukuun.

### *Throw (Heitto)*
Heitto-luokka on ikään kuin apuluokka Noppa-luokalle. Siinä lasketaan noppien kaikkien silmälukujen yhteisominaisuuksia, kuten silmälukulista, sattuma (chance), nn samaa (nn_of_a_kind), tasan nn samaa (nn_of_a_kind_absolute) ja mm:n suora (straight_of_mm). Näitä voidaan sitten käyttää hyväksi laskiessa eri yhdistelmien pisteitä. Heitto-luokan olio luodaan joka kerta uudelleen kun noppaa heitetään.

### *Settings (Asetukset)*
Asetusluokan tehtävänä on käytännössä esittää pelin säännöt. Asetusluokan tärkeimmät attribuutit ovat noppien lukumäärä (no_of_dice, n), noppien maksimisilmäluku (max_value, s) ja heittojen maksimi-lukumäärä kierrosta kohti (no_of_throws, h). Nämä voidaan asettaa tämän luokan kautta. Kun niitä muutetaan tai asetukset luetaan ensimmäistä kertaa, luetaan pelin asetustiedostosta peliin edellä mainittujen attribuuttien sallimat yhdistelmät.

### *Combination (Yhdistelmä)*
Yhdistelmä on käytännössä sääntö, jonka mukaan yhden heiton noppien silmäluvuista saa jonkun määrän pisteitä. Nämä säännöt on määritetty jokaiselle eri yhdistelmäkoodille erikseen tässä luokassa eikä uusia yhdistelmiä voi lisätä peliin muuttamatta pelin lähdekoodia. Yhtä yhdistelmäkoodia vastaan voidaan luoda a) yksi yhdistelmä ('SINGLE'), joka voi olla joko ehdoitta pelissä ('ALL'), vain peleissä, joissa on parillinen määrä noppia ('EVEN') tai pariton määrä noppia ('ODD') tai b) useampi yhdistelmä ('ENUMERATE'), joissa yhdistelmien lukumäärä riippuu joko noppien lukumäärästä ('NO_OF_DICE'), maksimisilmäluvusta ('MAX_VALUE') tai kummastakin ('MAX_VALUE_AND_NO_OF_DICE '). Näitä kolmea viimeeksi mainittua parametria kutsutaan jatkossa luetteloparametreiksi. Jos mitään muuta parametria ei ole annettu yhdistelmälle luetaan yhdistelmiä peliin, siten että ensimmäiselle yhdistelmälle annetaan indeksiksi 1 ja seuraavalle sitä suurempi lukuarvo kunnes saavutetaan annetun luettoloparametrin minimiarvo. Lisäksi tämän yhdistelmien indeksoinneille voidaan antaa minimi/maksimiarvot, joita ei voi alittaa/ylittää.

Yhdistelmälle voidaan antaa rajoitteita siitä, milloin yhdistelmä on käytössä. Nämä rajoitukset ovat suurimpia mahdollisia arvoja asetusluokan kuvauksessa mainituille kolmelle parametrille. Nämä rajoitukset voidaan asettaa kaikille yhdistelmille ja tälläistä rajoitusta rikotaan, ei yhdistelmäkoodi esiinny pelissä lainkaan. Esimerkki asetustiedostosta, jota on käytetty testauksessa on liittessä 2, indeksointiin liittyvien kommenttejen kera.

### *Player (Pelaaja)*
Pelaaja on toistaiseksi hyvin yksinkertainen luokka, joka voitaisiin lähestulkoon jättään käyttämättä. Pelaajan luomisen yhteydessä tosin luodaan pelaajalla oma tulostaulukko. Pelaajalla on nimi ja järjestys-vuoro pelissä (index).

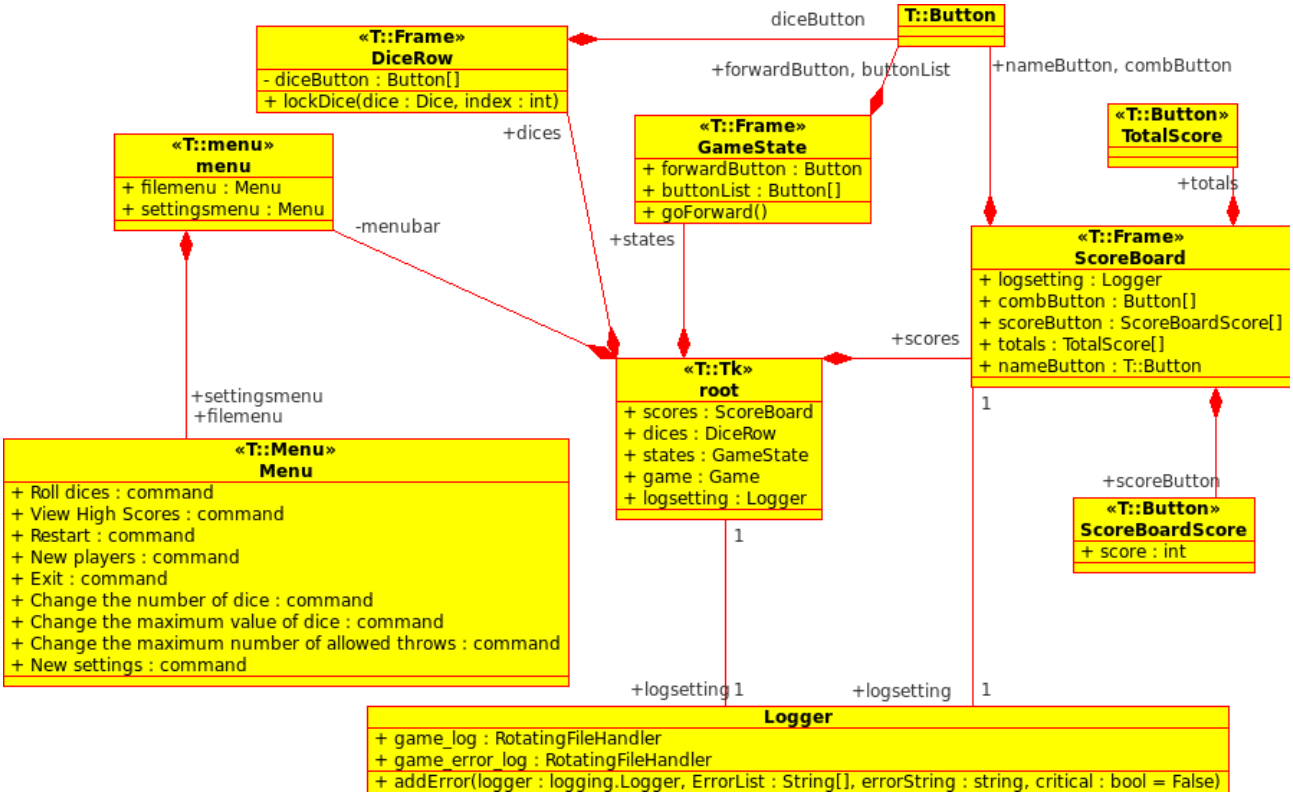
### *ScoreList (Tulostaulukko)*
Tulostaulukkoon kirjataan pelaajan pisteet. Jokainen tulostaulukon tulosarvo liittyy saman indeksin omaavaan yhdistelmään pelin asetuksissa. Pelaajan omassa yhdistelmälistassa pidetään kirjaa siitä, mitkä yhdistelmät ovat jo käytettyjä ja mikä on pelaajan kokonaissaldo (total).

### *HighScores (Top-10 -lista)*
Top-10 -listaa käytetään vain pelin lopussa kun tarkistetaan pääseekö pelin voittaja kaikkien aikojen top-10 -listalle.

### *Logger (Loki)*
Lokin avulla kirjataan pelin tärkeimpiä tapahtumia tiedostoon virheenkorjausta varten. Lisäksi lokiin tallennetaan tieto pelissä tapahtuvista mahdollisista virhetilanteista (warning), sekä varsinaisista virheistä (error, critical). Varsinaiset virheet lisätään myös pelin käsittelemättömien virheiden listaan

(unNotifiedErrors), josta käyttöliittymä voi tuoda niitä esiin (varoittaa käyttäjää) ja siirtää sitä mukaan käsiteltyjen listalle (NotifiedErrors).

## Graafinen käyttöliittymä




**Kuva 4.** Luokkakaavio graafisen käyttöliittymän luokkien suhteista. T viittaa Tkinter-moduliin.

Graafisen käyttöliittymän toteutus tehtiin pitkälti suunnitteludokumentissa esitetyllä tavalla. Kaikkien graafisten käyttöliittymän kohteiden pohjana käytettiin valmiita Tkinter moduleita (Kuva 4). Graafisen käyttöliittymän pohjana on Tkinter.Tk -ikkuna nimeltä root, jonka päällä on kolme Tkinter.Frame-oliota, jotka taas muodostuvat Tkinter.Button-olioista. Lisäksi root-olioon liittyy valikko (Tkinter.menu), joka on näytön ylälaidassa. Valikon uudet ikkunat on toteutettu Tkinter.Toplevel-olioina, joiden sisällä on muita Tkinter-olioita kuten Tkinter.Button, Tkinter.Entry, Tkinter.CheckButton, Tkinter.Spinbox, Tkinter.Label ja Tkinter.LabelFrame. Lisäksi asetustiedoston avaamiseen käytetään tkFileDialog.askopenfilename-funktiota ja virheistä tiedottamiseen ja toiminnan vahvistamiseen tkMessagebox.showerror ja tkMessageBox.askokcancel -moduuleita.

Graafisen käyttöliittymän toiminnot on hajoitettu pieniksi apufunktioiksi, jotka on kasattu moduliin GUI.

## Luokkien kuvaus

Luokkien ja luokkien funktioiden yksityiskohtaisempi kuvaus löytyy python pyDocista (englanniksi).

## Ohjeet ohjelman rajoitusten poistamiseksi

Pelissä ei oteta huomioon ikkunan mahtumista näyttön. Tästä syystä näytölle on annettu suosituskoko.

### Parannusehdotuksia

Pelissä voitaisiin ottaa huomioon näytön koko. Yhdistelmien pisteytystä olisi myös hyvä voida säätää siten, että pisteytys voitaisiin lukea suoraan asetustiedostosta. Asetustiedostossa olisi myös hyvä voida käyttää vapaasti pelin vapaita parametreja n, s ja h (noppien lukumäärä, noppien maksimisilmäluku ja heittojen maksimilukumäärä kierrosta kohti). Lisäksi perinteisessä Yatzissa esiintyy bonus, jonka voi saada kun yläosan pisteet ovat tarpeeksi suuria. Ei ole kuitenkaan kovin selvää, miten tätä voisi yleistää vapaisiin yhdistelmiin. Lisäksi joissakin Yatzin 'virallisissa' säännöissä mainitaan bonukset joita saa, jos heittää useamman kuin yhden Yatzin.

Lisäksi näytön ulkoasuan voisi vielä hiukan kaunistella. Noppien värin voisi jättää käyttäjän päätettäväksi. Lisäksi olisi hyvä, jos jokainen pelaaja voisi tallentaa omat asetuksensa, jolloin oletuspelinä voitaisiin ladata edellinen peli ja pelaajille voitaisiin kerätä omaa top-10 -listaa.

## Testauksen kuvaus

Testaus toteutettiin käyttämällä pythonin unittest-moduulia. Testauksessa toteutettiin kaksi eri testipakettia. Testaus voidaan suorittaa ajamalla paketissa Game oleva testiohjelma Unit_test_**2**.py. Testissä menee joitain kymmeniä sekunteja ja tulos tallennetaan tiedostoon `unit_test_2.txt`.

### TestNoSettingsFunctions (Vapaiden parametrien testaus)

Testauksessa haluttiin ensin testata voiko pelin vapaille parametreille n, s ja h (noppien lukumäärä, noppien maksimisilmäluku ja heittojen maksimilukumäärä kierrosta kohti) antaa vain oikeita arvoja. Tätä testattiin valitsemalla joukko arvoja, jotka käsittävät kaikki sallitut arvot sekä vääriä arvoja (`[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 'string', {'dict': 'value'}]`). Testauksen aluksi luotiin peli. Tulosteessa esitettiin arvoa, joka yritettiin vapaalle parametrille syöttää sekä syöttämisen tulos, eli mikä oli pelin parametrin arvon syötteen antamisen jälkeen. Jokainen parametri käytiin läpi erillisenä testi-tapauksena. Testissä varmistettiin vielä, että lopullinen arvo on sallitulla välillä unittestin `self.assertTrue(Settings.SMIN <= self.game.settings.max_value <= Settings.SMAX)`-komennolla. Mahdolliset virheet esitettiin kaikkien vapaiden parametrien käsittelyn jälkeen. Saaduista tuloksista (Liite 4) nähdään, että parametrit saavat oikean arvon: parametrin arvo muuttuu jos parametri on sallitulla välillä ja pysyy muuttumattomana, jos parametri on epäkelpo. Testiohjelma on toteutusdokumentin lopussa (Liite 3).

### TestGameCompletion (Pelin läpiajamistesti)

Tässä testissä testataan, saako pelin pelattua läpi kaikilla mahdollisilla parametrin arvoilla. Testauksessa käytetään saama arvojoukkoa parametreille kuin vapaiden parametrien testauksessakin. Testissä kokeillaan kaikkia mahdollisia eri vapaiden parametrien yhdistelmiä. Testin alussa luodaan uusi peli, muutetaan sen vapaita parametreja ja mennään yksi askel eteenpäin (aloitetaan peli). Tämän jälkeen kokeillaan unittestin `self.assertFalse(self.game.end, "Game end was reached before the game started")`-komennolla, että peli ei varmasti ole päättynyt. Tämän jälkeen käydään läpi kaikki heittovuorot käymällä

sisäkkäisissä loopeissa läpi kaikki pelaajat ja yhdistelmät. Tässä vaiheessa pelin pitäisi olla loppunut. Tämä tarkistetaan unittestin `self.assertTrue(self.game.end, "Game end was not reached")`-komennolla. Testin lopussa esitetetään vielä virheiden kokonaismäärä (0) ja testiin kulunut aika (60..70s). Lisäksi kun on ensin poistettu 'Scores'-kansiosta kaikki kansiot, nähdään, että peli kykenee luomaan peliasetuksille Scores-kansioon oman kansion ja luomaan sinne kaikille eri yhdistelmille top-10 -listat. Listojen kokonaismäärä oli testissä 720 kpl, joka on täsmälleen yhtä suuri kun kaikkien vapaiden parametrien arvoalueiden tulo, eli listoja luotiin oikea määrä ja kaikille parametriyhdistelmille. Lisäksi testitulosteesta (Liite 4) voidaan tarkastaa, että yhdistelmien lukumäärä vaihtelee parametrien (n ja s) mukaisesti.

# Liitteet

## Liite 1: Käännöksiä

**Taulukko 4. Yleisimpiä ja tärkeimpiä pelissä käytettyjä termejä ja niiden käännökset suomesta englanniksi ja päinvastoin.**

| Suomeksi | Englanniksi |
|---|---|
| **Arvo** | Value |
| **Asetukset** | Settings |
| **Heitto** | Throw |
| **Heittovuoro** | Throw Turn |
| **Graafinen käyttöliittymä** | Graphical User Interface (GUI) |
| **Indeksi** | Index |
| **Kokonaissaldo** | Total |
| **Koodi** | Code |
| **Loki** | Logger |
| **Lukitsematon** | Unlocked |
| **Lukossa** | Locked |
| **mm:n suora** | Straight of mm |
| **nn samaa** | nn of a Kind |
| **Nimi** | Name |
| **Noppa** | Dice |
| **Nopparivi** | Dice Row |
| **Pelaaja** | Player |
| **Peli** | Game |
| **Pelikierros** | Game Round |
| **Pelivuoro** | Game Turn |
| **Pelin tila** | Game State |
| **Sattuma** | Chance |
| **Silmäluku** | Value |
| **Top-10 -lista** | High Scores |
| **Tulostaulu (pelin)** | Score Board |
| **Tulostaulukko (pelaajan)** | Score List |
| **Tyyppi** | Type |
| **Yhdistelmä** | Combination |

| Englanniksi | Suomeksi |
|---|---|
| **Chance** | Sattuma |
| **Code** | Koodi |
| **Combination** | Yhdistelmä |
| **Dice** | Noppa |
| **Dice Row** | Nopparivi |
| **Game** | Peli |
| **Game Round** | Pelikierros |
| **Game State** | Pelin tila |
| **Game Turn** | Pelivuoro |
| **Graphical User Interface (GUI)** | Graafinen käyttöliittymä |
| **High Scores** | Top-10 -lista |
| **Index** | Indeksi |
| **Locked** | Lukossa |
| **Logger** | Loki |
| **Name** | Nimi |
| **nn of a Kind** | nn samaa |
| **Player** | Pelaaja |
| **Score Board** | Tulostaulu (pelin) |
| **Score List** | Tulostaulukko (pelaajan) |
| **Settings** | Asetukset |
| **Straight of mm** | mm:n suora |
| **Throw** | Heitto |
| **Throw Turn** | Heittovuoro |
| **Total** | Kokonaissaldo |
| **Type** | Tyyppi |
| **Unlocked** | Lukitsematon |
| **Value** | Arvo, Silmäluku |

## Liite 2: Esimerkki asetustiedostosta

```
--------------------------------------------------------------------------

The Basic Yahtzee Game

Rules by Patrik Ahvenainen

Last update May 3, 2010

--------------------------------------------------------------------------

#N#        5          'Number of dice'
#s#        6          'Maximum value of dice'
#H#        3          'Maximum number of allowed throws'

#y#        000100     ''s'
#y#        000200     'straight of' nMin 4  sMin 4 indexMin 4     indexMax 5 [1]
#y#        000300     'of a kind'   nMin 5         indexMin 2     indexMax 4 [2]
#y#        000400     'two pairs'                                 nMin 4 [3]
#y#        000700     'Yahtzee'     nMin 5  sMin 3 [4]
#y#        001000     'Full House'  nMin 5 [5]
#y#        000000     'Chance'
```

———

*Alla olevat kommentit eivät kuulu varsinaiseen tiedostoon (eivätkä yläindeksiviitteet)*

[1] Vain suorat neljästä viiteen ovat olemassa ja vain jos noppia on vähintään neljä ja noppien maksimisilmäluku on vähintään neljä.

[2] n samaa ovat olemassa kahdesta (pari) neljään samaan, kunhan noppia on vähinttään viisi. Indeksi on rajoitettu neljään, jotta tämä yhdistelmä ei mene Yatzin (Yahtzee) kanssa päällekkäin.

[3] Erityisyhdistelmä, jota ei voi saada jos on alle neljä noppaa. Siksi neljän rajoitus indeksille.

[4] Yatzi (Yahtzee) saadaan perinteisesti kun on viisi noppaa heittämällä kaikilla nopilla sama silmäluku. Tällä yhdistelmällä olisi ehkä mielekästä olla lisärajoituksena indeksille `nMax 5`, jolloin yli viidellä nopalla pelatessa ei saisi helposti Yatzia.

[5] Erityisyhdistelmä, jota ei voi saada jos on alle viisi noppaa. Siksi viiden rajoitus indeksille.

## Liite 3: Testiajokoodi

```
'''
Created on 2 May 2010
Last modified on 4 May 2010

@author: Patrik Ahvenainen
'''
```

```
import unittest
import Game
import Settings
import logging
import time
import sys


VALUE_TEST_LIST = [-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 'string', {'dict':
'value'}]
#VALUE_TEST_LIST = [3, 4, 5, 6, 7, 8, 9, 10, 11, 1000, 'string', {'dict':
'value'}]

SETTINGS_FILE = "Basic Yahtzee.dat"
WRITETOFILE = "unit_test_2.txt"
PLAYERS = "Player 1" , "Player 2"#, "Player 3", "Player 4"

WRITESTRING = ""

#class WriteStringClass(object):
#    def __init__(self):
#        self.WRITESTRING = ""



class TestGameCompletion(unittest.TestCase):

    def setUp(self):
        try: self.index1 == 0
        except: self.index1 = -1
        try: self.index2 == 0
        except: self.index2 = 0
        try: self.index3 == 0
        except: self.index3 = 0
        self.index1 += 1
        if self.index1 == len(VALUE_TEST_LIST):
            self.index2 += 1
            self.index1 = 0
        if self.index2 == len(VALUE_TEST_LIST):
            self.index3 += 1
            self.index2 = 0

        self.game = Game.Game(PLAYERS, SETTINGS_FILE, log=False)
        self.game.settings.max_value = VALUE_TEST_LIST[self.index1]
        self.game.settings.no_of_dice = VALUE_TEST_LIST[self.index2]
        self.game.settings.no_of_throws = VALUE_TEST_LIST[self.index3]
        self.game.goForward()

    def test_complete_game(self):
        self.assertFalse(self.game.end, "Game end was reached before the game
started")
        for no_of_players in range(len(self.game.player_list)):
            for i in range(0, len(self.game.settings.combinations)):
                self.game.SequentialChoice()
        if self.game.end:
            print "Game settings are: n= " + str(VALUE_TEST_LIST[self.index1])
+ "-->" + str(self.game.settings.no_of_dice) + \
                  ", s=" + str(VALUE_TEST_LIST[self.index2])  + "-->" +
str(self.game.settings.max_value) + \
                  ", h=" + str(VALUE_TEST_LIST[self.index3])  + "-->" +
str(self.game.settings.no_of_throws) + \
```

```
                  ", no of combinations=" +
str(len(self.game.settings.combinations))
        self.assertTrue(self.game.end, "Game end was not reached")

    def tearDown(self):
        logging.shutdown()

class TestNoSettingsFunctions(unittest.TestCase):

    def setUp(self):
        self.game = Game.Game(PLAYERS, SETTINGS_FILE, log=False)

    def test_set_max_value(self):
        print "\nTESTING 'MAX VALUE OF DICE' ASSIGNMENT"
        print "ACCEPTABLE VALUES ARE BETWEEN " + str(Settings.SMIN) + \
              " AND " + str(Settings.SMAX)
        for testItem in VALUE_TEST_LIST:
            self.game.settings.max_value = testItem
            print "Trying to set value '" + str(testItem) + \
                  "'. New value is = " + str(self.game.settings.max_value)
            logging.shutdown()
        self.assertTrue(Settings.SMIN <= self.game.settings.max_value <=
Settings.SMAX)

    def test_set_no_of_dice(self):
        print "\nTESTING 'NUMBER OF DICE' ASSIGNMENT"
        print "ACCEPTABLE VALUES ARE BETWEEN " + str(Settings.NMIN) + \
              " AND " + str(Settings.NMAX)
        for testItem in VALUE_TEST_LIST:
            self.game.settings.no_of_dice = testItem
            print "Trying to set value '" + str(testItem) + \
                  "'. New value is = " + str(self.game.settings.no_of_dice)
            logging.shutdown()
        self.assertTrue(Settings.NMIN <= self.game.settings.no_of_dice <=
Settings.NMAX)

    def test_set_no_of_throws(self):
        print "\nTESTING 'MAXIMUM NUMBER OF THROWS' ASSIGNMENT"
        print "ACCEPTABLE VALUES ARE BETWEEN " + str(Settings.HMIN) + \
              " AND " + str(Settings.HMAX)
        for testItem in VALUE_TEST_LIST:
            self.game.settings.no_of_throws = testItem
            print "Trying to set value '" + str(testItem) + \
                  "'. New value is = " + str(self.game.settings.no_of_throws)
            logging.shutdown()
        self.assertTrue(Settings.HMIN <= self.game.settings.no_of_throws <=
Settings.HMAX)

    def tearDown(self):
        logging.shutdown()

if __name__ == '__main__':
#    unittest.main()'
    no_of_total_errors = 0

    sys.stdout = open(WRITETOFILE, 'w')
    print ("Starting unit tests\n")

    totaltime = time.clock()

    test_suite = unittest.TestSuite()
```

```
    test_set_max_value_ = TestNoSettingsFunctions('test_set_max_value')
    test_set_no_of_dice_ = TestNoSettingsFunctions('test_set_no_of_dice')
    test_set_no_of_throws_ = TestNoSettingsFunctions('test_set_no_of_throws')
    test_suite.addTest(test_set_max_value_)
    test_suite.addTest(test_set_no_of_dice_)
    test_suite.addTest(test_set_no_of_throws_)
    testresults = unittest.TestResult()
    timeri = time.clock()
    test_suite.run(testresults)
    took = time.clock() - timeri
    if len(testresults.errors) > 0:
        no_of_total_errors += 1
    print ( "\nErrors: " + str(testresults.errors))
    if testresults.wasSuccessful():
        print ( "SUCCESS in " + str(took) + "s !" )

#    myObj = WriteStringClass()

    print "\nTESTING 'GAME COMPLETION' ASSIGNMENT USING RANDOM SELECTION OF
SCORES TO MARK, "
    print "TEST SETTINGS FILE '" + str(SETTINGS_FILE)
    timeri = time.clock()
    test_suite = unittest.TestSuite()
    test_complete_game = TestGameCompletion('test_complete_game')
    testresults = unittest.TestResult()
    for index, item1 in enumerate(VALUE_TEST_LIST):
        for item2 in VALUE_TEST_LIST:
            for item3 in VALUE_TEST_LIST:
                test_suite = unittest.TestSuite()
                test_suite.addTest(test_complete_game)
                test_suite.run(testresults)
                if len(testresults.errors) > 0:
                    no_of_total_errors += 1
                if len(testresults.errors) > 0:
                    print  ", Errors: " + str(testresults.errors)
                if testresults.wasSuccessful():
                    print  ", SUCCESS"
#        print "Game completion test %4.2f completed in %3.1fs" %
((index+1)/float(len(VALUE_TEST_LIST)), time.clock() - totaltime)

    print "\nIn total the test took " + str(time.clock() - totaltime) + "s."
    print "There were " + str(no_of_total_errors) + " errors in total"

    sys.stdout = sys.__stdout__
```

## Liite 4: Testiajon tulokset

Tiedostossa testiajon_tulokset suuren koon vuoksi.

Patrik Ahvenainen / 013326292
patrik.ahvenainen@helsinki.fi
Ohjaaja: Joel Rybicki

# Yleinen noppapeli

## Määrittelydokumentti

Ohjelmoinnin harjoitustyö
Tietojenkäsittelytieteen laitos
Helsingin yliopisto
Palautuspäivä: 21.03.2010

## 1 Ohjelman tarkoitus ja yleiskuvaus

Tehty sovellus on noppapeli, jonka sääntöjä voi helposti muokata. Peli on tarkoitettu viihdekäyttöön suomea puhuville käyttäjille. Peli toteutetaan käyttöjärjestelmäriippumattomaksi. Tämä toteutetaan

käyttämällä python-kieltä. Graafisen käyttöliittymän käyttöönotto asettaa jotain rajoituksia ohjelman siirrettävyydelle. Toteutuksessa käytetään Tkinter-moduulia, joka on nykyään yleensä asennettu Python-ohjelmointiympäristöjen kanssa. Pelissä koitetaan saada mahdollisimman paljon pisteitä heittämällä noppia ja koettamalla saada mahdollisimman suuria pistemääriä peliin peliasetusten mukaisilla yhdistelmillä. Yhdistelmä on sääntökokonaisuus, jonka mukaan pelaaja saa pisteitä kaikkien noppien silmälukujoukon perusteella. Pelaaja kerää eri yhdistelmien pisteitä kokonaispistesaldoona tulostaulukkoonsa.

Peliä voi pelata joko yksin tai useampaa ihmispelaajaa vastaan. Yksin pelattaessa pelaaja voi pelata useammalla tulostaulukolla, johon pelaaja kerää vuorotellen pisteitä lisäämällä heitettyjen noppiensa silmälukujen oikeuttaman pistearvon jonkin pelissä mukana olevan yhdistelmän kohdalle. Yksin pelatessa pelaaja voi yrittää tehdä piste-ennätystä kyseisille peliasetuksille. Parhaat kymmenen kokonaispistesaldoa ja niiden saaneiden pelaajien nimet tallennetaan listalle parhaista pistemääristä (top-10 -lista). Näitä listoja on yksi jokaista peliasetustiedostoa kohden. Lisäksi pelaaja voi muuttaa noppien lukumäärää, noppien maksimisilmälukua ja yhden heittovuoron heittokertojen lukumäärää vaihtamatta peliasetuksia. Näitä peli-sääntöjä vastaava top-10 -lista luodaan aina kun vastaavaa listaa ei ole valmiiksi olemassa. Jos tälläinen lista on jo valmiiksi olemassa voi pelaaja saada nimensä kyseiselle listalle, jos hänen pistemääränsä tähän oikeuttavat.

Peliä voi myös pelata moninpelinä, jossa pelaajat pelaavat toisiaan vastaan. Voittaja on se, jolla on peli-kerran lopuksi suurin kokonaispistesaldo. Voittaja, ja vain voittaja, voi saada nimensä top-10 –listalle samoin kuin yksinpelissä. Peliä pelataan vuorotellen samalla tietokoneella. Pelaajat voivat käyttö-järjestelmän ja käytettävissä olevan laitteiston niin salliessa käyttää useampaa hiirtä ja näppäimistöä, mutta pelaajien hiiren ja näppäimistön käyttöä ei rajoiteta koskemaan vain omaa vuoroaan, eli käytettävissä on vain yksi ”virtuaalinen” hiiri ja näppäimistö, jota tosin voidaan käyttää useammalla fyysisellä laitteella. Käyttöjärjestelmän vastuulle jätetään useamman ohjauslaitteen samanaikaisesta käytöstä aiheutuvat ongelmatilanteet.

# 2 Rajoitukset

Samaa peliä voi pelata vain yhdellä tietokoneella, eli peliin ei toteuteta minkäänlaisia verkkopeli-ominaisuuksia. Pelin ohjauslaitteita ovat näppäimistö ja hiiri, jota moninpelissä täytyy pelaajien käyttää vuorotellen. Pelilautoja toteutetaan vain yksi kappale.

Peliin voi osallistua vain neljällä tulostaulukolla. Peli ilmoittaa, minkä pelaajan vuoro on kyseessä, mutta ei estä ihmispelaajaa vahingossa pelaamasta toisen pelaajan vuorolla. Neljän tulostaulun rajoitus tarkoittaa myös sitä, että pelissä voi olla korkeintaan neljä pelaajaa kerrallaan.

Peliä voi pelata korkeintaan yhdeksällä nopalla. Pelissä täytyy olla vähintään yksi noppa. Tämä rajoitus on mahdollistaa myös pikanäppäinten käytön eri noppien toimintojen käsittelyn yhteydessä (lähinnä nopan x pito, x=1..9). Käytössä olevin noppien lukumäärää voi vaihtaa vain pelikertojen välissä.

Myös noppien maksimisilmäluvuksi voi valita korkeintaan yhdeksän. Tämä yläraja valittiin käytännön syistä; jos yhdistelmien maksimimäärää ei haluta rajoittaa suoraan, täytyy mahdollisten eri yhdistelmien lukumäärä pitää melko alhaisena, jotta koko tulostaulukko saataisiin näkymään kerralla. Ei-triviaali noppapeli edellyttää, että noppien silmäluvun maksimiarvon tarvitsee olla vähintään kaksi ja tämä asetetaan myös rajoitukseksi pelissä. Noppien maksimisilmälukua voi vaihtaa vain pelikertojen välissä.

Yhdellä heittovuorolla pelaajalle voidaan antaa yksi tai useampi heitto, jonka aikana pelaaja koittaa heittää noppiaan niin, että saa mahdollisimman paljon pisteitä jostain jäljellä olevasta yhdistelmästä. Jos pelaajalle annetaan vain yksi heitto korostuu pelissä lähinnä hyvä tuuri. Tästä syystä perinteisesti on tapana antaa pelaajalle mahdollisuus pitää tai lukita osa nopistaan ja heittää vain osaa nopistaan. Jos pelaaja on hyvin tyytymätön ensimmäiseen heittoonsa hän voi toki heittää kaikkia noppia uudelleen. Kerran pitoon otetun nopan voi poistaa pidosta ja heittää uudelleen, jos pelaaja päättää yrittää eri yhdistelmää ja heittovuoroja on vielä jäljellä. Heittokertoja yhdellä heittovuorolla voi olla korkeintaan yhdeksän ja niiden lukumäärää voi vaihtaa vain pelikertojen välissä. Yhdistelmän tuottaman pistesaldo ei saa riippua käytettyjen heittokertojen lukumäärästä.

Peliasetukseen täytyy kuulua vähintään yksi yhdistelmä. Yhdellä peliasetuksella voi olla korkeintaan 30 eri yhdistelmää. Tämä rajoitus toteutuu käyttämällä edellä olevia rajoituksia sekä Aiheen rajaus –dokumentissa esitettyjä yhdistelmiä (Liite 1).

Koska pelit ovat melko lyhyitä, ei pelitilannetta voi tallentaa. Pelaajille ei myöskään anneta mahdollisuutta perua ja uusia heittojaan, koska tämä mahdollistaisi huijaamisen. Myöskään oikeassa noppapelissä ei kirjattuja arvoja ole tapana muuttaa jälkeenpäin eikä noppien heittämistä voi enää perua valitsemalla heitetyille nopille niiden edellisen kierroksen arvo.

# 3 Käsiteltävän datan kuvaus

Pelissä käsiteltävään dataan kuuluu tietenkin oleellisesti noppien silmäluvut kyseisellä heittovuorolla. Lisäksi pelissä tulee pitää kirjaa siitä kuinka monta heittoa pelaajalla on vielä jäljellä kyseisellä heittovuorolla sekä tähän liittyen luonnolisesti tietoa siitä, kenen heittovuoro on. Lisäksi kaikkien pelaajien aikaisempien yhdistelmien tuottamat tulokset täytyy säilyttää kyseisen pelaajan tulostaulukossa. Peliasetuksissa tulee määritellä, mitkä yhdistelmät ovat käytössä pelikerran aikana. Yhden heittovuoron heittokertojen lukumäärä, noppien maksimisilmäluku sekä noppien lukumäärä voivat poiketa peliasetuksissa määritellyistä oletuisarvoista, joten niistä täytyy pitää jatkuvasti kirjaa. Pelaajien lukumäärä ja pelaajien nimet ovat myös tärkeitä tietoja pelin kannalta.

# 4 Tiedonkulun kuvaus

Pelin aluksi valitaan pelaajien lukumäärä ja pelaajien nimet. Käyttäjältä kysytään pelaajien nimiä lomakkeella, josta ruksitaan peliin osallistuvat pelaajat (1-4). Tämän jälkeen pelaajan tulee valita peliasetustiedosto. Kun peliasetustiedoston on ladattu onnistuneesti, näytetään pelaajille pelilauta, jossa on peliasetuksista luettu oletusmäärä noppia. Pelilaudalla on myös jokaisen pelaajan tulostaulukko, joka on aluksi tyhjä. Pelaajat siis näkevät tässä vaiheessa, mitä yhdistelmiä pelissä on käytössä. Pelaajat näkevät myös noppien maksimisilmäluvun sekä yhden heittovuoron heittokertojen lukumäärän. Jos pelaajat haluavat muuttaa näitä tai noppien lukumäärää, se onnistuu tässä vaiheessa asetukset-valikon kautta tai käyttämällä pikanäppäimiä.

Kun pelaajat päättävät aloittaa pelin, heittää ensimmäisenä vuorossa oleva pelaaja kaikkia noppia kerran painamalla ’heitä’ –nappulaa tai käyttämällä pikanäppäintä. Jos pelissä on useampia heittokertoja heittovuoroa kohden, pelaaja voi ensimmäisen heittonsa jälkeen valita osan nopista pitoon, jolloin ne näkyvät

pelilaudalla eri värisinä, eikä niitä voi heittää ennen kun ne poistetaan pidosta. Pelaaja voi siis kaikilla ensimmäistä heittokertaansa seuraavilla heittokerroilla heittää kaikkia noppiaan tai vain osaa nopistaan. Pelaaja voi oman heittovuoronsa aikaan milloin tahansa kirjata tuloksensa ylös tulostaulukkoonsa klikkaamalla hiirellä oikean arvon kohdalta. Pelaajan ei siis tarvitse käyttää kaikkia heittokertojaan. Kun viimeinenkin heittokerta on käytetty täytyy pelaajan hiirellä klikkaamalla valita jokin jäljellä oleva yhdistelmä tulostaulukostaan. Tämän yhdistelmän niillä noppien silmäluvuilla tuottama pistesaldo lisätään välittömästi pelaajan tulostaulukkoon yhdistelmän kohdalle sekä tämä pistesaldo lisätään pelaajan kokonaispistesaldoon. Pelaajalle näytetään jatkuvasti kuinka paljon pisteitä hän saa asettamalla pisteensä kyseisen yhdistelmän kohdalle tulostaulukkoonsa.

Kun ensimmäinen pelaaja on valinnut minkä yhdistelmän pistesaldon lisää tulostalukkoonsa, siirtyy vuoro seuraavalle pelaajalle, joka toistaa ensimmäisen pelaajan ensimmäisen heittovuoron suorituksen. Heittovuoroja on jokaisella pelaajalla sama määrä, eli yhtä monta kun on mahdollisia yhdistelmiä. Joka heittovuorolla täytyy pelaajan hiirellä klikkaamalla valita jokin jäljellä olevista yhdistelmistä. Pelaajan kokonaispistesaldo päivitetään siis jatkuvasti pelaajan tulostaulukon pisteiden mukaisesti.

Pelikerran loputtua ilmoitetaan pelaajille voittaneen pelaajan nimi. Jos voittanut pelaaja on päässyt top-10 –listalle, näytetään kyseinen top-10 –lista pelaajille. Tässä tapauksessa top-10 –lista päivitetään tiedostoon, josta se on luettu. Jos kyseisille peliasetuksille ei vielä ole olemassa kyseistä listaa luodaan uusi tiedosto, jonka ensimmäiselle paikalle lisätään voittajan nimi ja pistesaldo.

# 5 Toimintojen kuvaus

Pelin perustoimintoihin kuuluvat luonnollisesti uuden pelin aloittaminen ja pelin lopettaminen. Uuden pelin aloittamisessa käytetään oletusarvoina edellisen pelin peliasetuksia. Pelaajien nimiä kysytään käyttäjältä. Oletusarvoisesti tämä lomake on täytetty edellisen pelikerran pelaajien nimillä ja valittuina ovat edellisen kierroksen pelaajat. Pelin lopettamisen yhteydessä kysytään pelaajalta varmistusta.

Pelin tärkeä ominaisuus on myös peliasetusten muuttaminen, jonka voi tehdä pelikertojen välissä. Peliasetukset muutetaan valitsemalla haluttu peliasetustiedosto. Pelaajat voivat myös vaihtaa noppien maksimisilmälukua, heittovuoron heittokertojen lukumäärää ja noppien lukumäärä tässä vaiheessa Asetukset-valikon kautta.

Pelaaminen suoritetaan heittämällä noppia joko painamalla ruudussa olevaa nappula hiirellä tai käyttämällä pikanäppäintä. Heittovuoron suoritukseen kuuluu mahdollisesti myös pelaajien valitsema määrä uusintaheittoja, jossa pelaaja voi saman heittovuoron aikana heittää joko osaa tai kaikkia noppia uudestaan niin kauan kun hänellä on heittokertoja jäljellä. Ennen jokaista uusintaheittoa pelaaja voi valita osan nopista pitoon joko hiirellä tai pikanäppäimillä. Pidossa olevia noppia ei heitetä. Pelaaja voi myös poistaa nopan pidosta ennen uusintaheittoa.

Tärkein pelitapahtuman on ehkä kuitenkin pistesaldon kirjaaminen jollekin yhdistelmälle tulostaulukkoon pelivuoron päätteeksi. Pelivuoron voi lopettaa kirjaamalla pistesaldon jonkin käytössä olevan yhdistelmän kohdalle milloin tahansa pelin aikana. Kirjaus tapahtuu klikkaamalla hiirellä jotain pelaajan tulostaulukon riviä, jota ei ole vielä käytetty. Tämän jälkeen vuoro siirtyy seuraavalle pelaajalle, jos pelaajia on useampia. Kun viimeinenkin pelaaja on suorittanut ensimmäisen pelivuoronsa, alkaa toinen kierros, jonka aloittaa ensimmäinen pelaaja omalla toisella heittovuorollaan. Kierroksia on yhtä monta kuin on käytettävissä olevia yhdistelmiä.

Pelaaja voi myös milloin tahansa katsoa kyseisten peliasetusten top-10 –listaa valitsemalla Peli-valikosta kohdan top-10 –lista. Pelaaja ei voi tyhjentää top-10 –listaa pelin kautta.

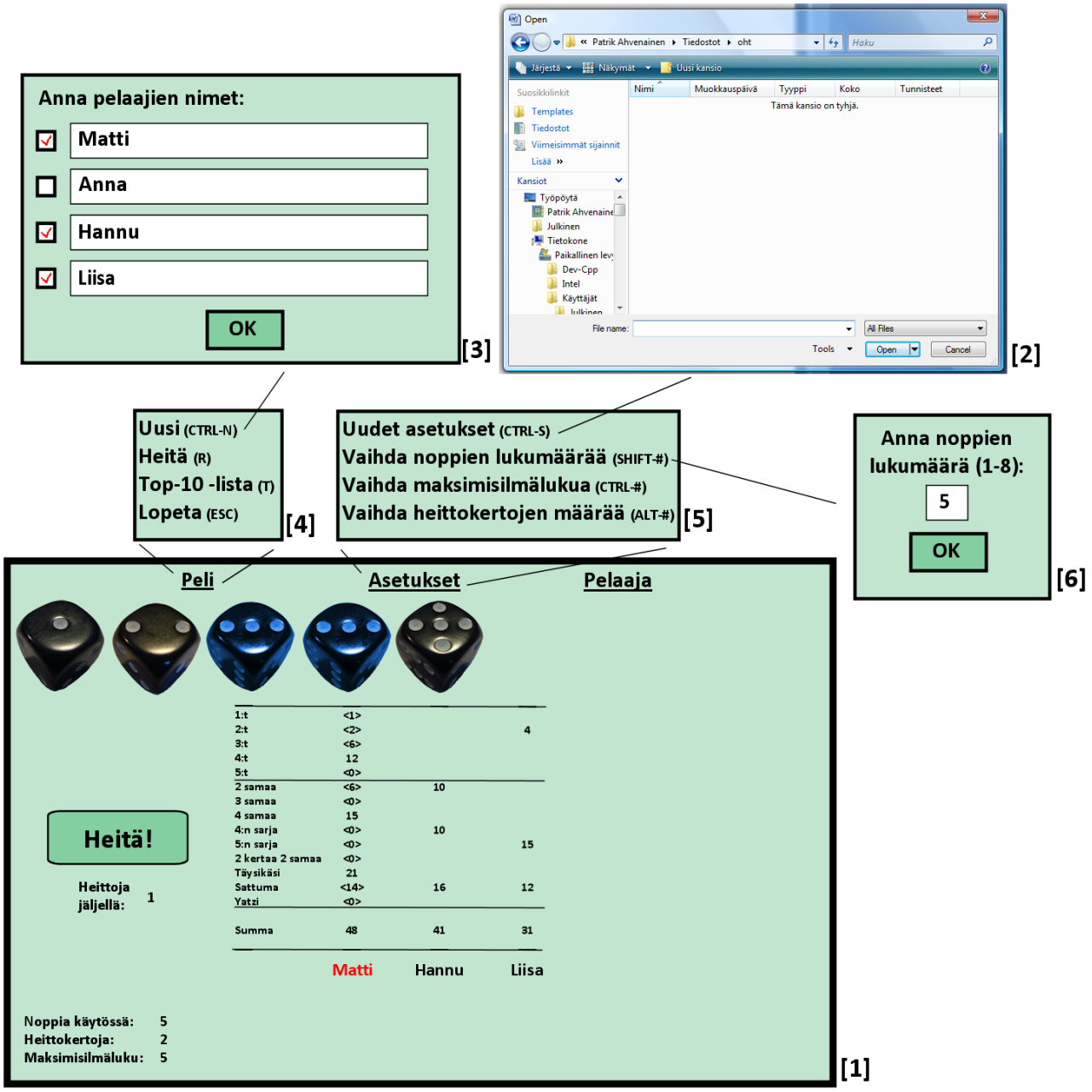
# 6 Käyttäytymisen kuvaus

Pelin käynnistyessä avautuu pelin pääikkuna [1] (Kuva 5). Käytössä on tällöin oletusasetustiedosto sekä oletusmäärä pelaajia. Pelaaminen tapahtuu painamalla 'Heitä'-nappulaa tai painamalla pikanäppäintä. Noppia voi laittaa pitoon joko pikanäppäimillä tai hiirellä klikkaamalla. Kun kaikki heittokerrat on käytetty tai pelaaja on jo saanut haluamansa yhdistelmän, hän valitsee hiirellä omasta mielestään parhaan yhdistelmän käyttämättömistä vaihtoehdoista. Valinnan helpoittamiseksi pelaajalle näytetään jatkuvasti nykyisten noppien tuottama pistemäärä kaikilla niillä yhdistelmillä, jotka pelaajalla on vielä käytössä.

Pelin pääikkunassa on kolme valikkoa, joista valikko 'Pelaaja' on tyhjä. Tämän valikon avulla on pelaajien luominen ja käsitteleminen laajennettavissa helposti.

Uuden pelin luominen tapahtuu 'Peli'-valikon avulla [4]. Nykyisellä toteutuksella tämä tarkoittaa käytännössä osallistuvien pelaajien valintaa [3]. Uuteen peliin on valittuna oletusarvoisesti kaikki edelliseen peliin osallistuneet pelaajat. Top-10 -lista on yksinkertainen ikkuna, jossa on listattuna kyseisten peli-asetusten kymmenen parasta tulosta sekä niiden saaneiden pelaajien nimet.

Asetusten muuttaminen tapahtuu 'Asetus'-valikon kautta [5]. Peliasetustiedoston valinta tapahtuu Tkinter-moduuliin rakennetulla tiedoston avaustoiminnolla. Tämän ulkonäkö ja toiminta saattaa siis vaihdella hiukan käyttöjärjestelmittäin [2]. Käyttäjällä on myös mahdollisuus vaihtaa noppien lukumäärää, maksimisilmälukua sekä heittovuoron heittokertojen määrää. Tämä voidaan tehdä joko pikanäppäimellä tai avattavalla valikolla [6].




**Kuva 5. Suunnitelma yleisen noppapelin graafiselle käyttöliittymälle. [1] Pelin päänäyttö. [2] Peliasetustiedoston valintaikkunan ulkonäkö riippuu käyttöjärjestelmästä. [3] Peliin osallistuvien pelaajien nimeäminen tapahtuu uuden pelin yhteydessä. [4] Peli-valikosta voi aloittaa uuden pelin ja lopettaa nykyisen sekä katsella top-10 listaa. [5] Asetukset-valikosta voi vaihtaa peliasetus-tiedostoa sekä noppien lukumäärää, maksimisilmälukua sekä heittovuoron heittokertojen määrää. Tämä onnistuu vain pelikertojen välissä. [6] Pikanäppäinten lisäksi Asetukset-valikon kolmea alinta asetusta voi muuttaa valintaikkunan kautta.**

# Liitteet

Liite 1 Yleisen noppapelin yhdistelmät (aiheen rajaus-dokumentista)

# Liite 1

| Yhdistelmän nimi | Ehto, kuvaus | Rajoitus |
|---|---|---|
| $x$:t[2] | Vähintään yksi noppa, jolle $sl = x$ | $x \leq p$ |
| $m$:n sarja[2,3] | $m$ kappaletta silmälukuja, jotka muodostavat peräkkäisten kokonaislukujen jonon | $2 \leq m \leq n, m \leq p$ |
| $t$ samaa silmälukua $sl$[2,3] | Vähintään $t$ kappaletta samaa silmälukua $sl$ | $2 \leq t \leq n, sl \leq p$ |
| $h$ kertaa $t$ samoja silmälukuja $sl_1, sl_2, \ldots sl_h$[1] | $h$ eri ryhmää, jossa jokaisessa on vähintään $t$ kappaletta samoja silmälukuja | $sl_1 \leq p,\ sl_2 \leq p,\ \ldots,$ $sl_h \leq p, h \times t \leq n,$ $h \times t \leq p, h \geq 2$ |
| $k$:lla jaolliset[1] | Kaikki silmäluvut ovat $k$:lla jaollisia ($k$ on joko 2 tai 3) | $k$=2 ∨ 3, $p \geq 2$ ∨ 3 |
| $k$:lla jaottomat[1] | Kaikki silmäluvut ovat $k$:lla jaottomia ($k$ on joko 2 tai 3) | $k$=2 ∨ 3, $p \geq 2$ ∨ 3 |
| Yatzi ($sl$) | $n$ kappaletta samaa silmälukua $sl$ | $sl \leq p$ |
| reilu jako | Nopat voidaan jakaa kahteen epätyhjään joukkoon siten, että molemmissa joukoissa on samoja silmälukuja yhtä paljon | $n$ parillinen |
| ahne jako | Nopat voidaan yhtä noppaa lukuun ottamatta jakaa kahteen epätyhjään joukkoon siten, että molemmissa joukoissa on samoja silmälukuja yhtä paljon. | $n$ pariton, $n \geq 3$ |
| täysikäsi[1] | $\frac{n-1}{2}$ kappaletta silmälukua $sl_1$ ja $\left(\frac{n-1}{2} + 1\right)$ kappaletta silmälukua $sl_2$ | $n$ pariton, $n \geq 3,$ $sl_1 < p,\ sl_2 < p$ |
| sattuma | Arvo on noppien silmälukujen summa | |

**Taulukko 5. Eri yhdistelmät, joita noppapelissä voi saada. Taulukossa $n$ on noppien lukumäärä ja $p$ suurin silmäluku.**

[1] Tämä ominaisuus on tarkoitus toteuttaa vain viidelle nopalle. Ominaisuus on yleistettävissä taulukossa osoitetulla tavalla.
[2] Tämä yhdistelmäjoukko sisältää (mahdollisesti) useampia yhdistelmiä. Yhdistelmien lukumäärä luetaan automaattisesti pelin asetuksista.
[3] $m/t$:lle voidaan asettaa minimiarvo, joka on suurempi kuin 2 muut rajoitukset huomioon ottaen.

Patrik Ahvenainen / 013326292
patrik.ahvenainen@helsinki.fi
Ohjaaja: Joel Rybicki

# Yleinen noppapeli

## Suunnitteludokumentti

Ohjelmoinnin harjoitustyö
Tietojenkäsittelytieteen laitos
Helsingin yliopisto
Palautuspäivä: 11.04.2010

# 1 Ohjelman yleisrakenne

## 1.1 Yleistä

Peli toteutetaan python sovelluksena. Käytetty kieliversio on 2.5, jotta toteutettu sovellus olisi helposti ajettavissa tietojenkäsittelytieteen laitoksen koneilla. Pelin suunnittelun lähtökohtana on suunnitella helposti muunneltava ja laajennettava ohjelma, jossa käyttöliittymä ja pelimoottori toteutetaan erillisinä osina. Peliin tulisi siis kyetä suunnittelemaan toinen, esimerkiksi tekstipohjainen käyttöliittymä. Luokkien suunnittelussa pyritään välttämään turhaa tiedon toistoa. Toisaalta pyritään tarjoamaan monipuoliset luokat, joiden avulla kyetään toteuttamaan hyvin toisistaan eroavia pelejä. Pelin sääntöjen rajoittamattomuuden vuoksi peliin ei toteuteta kurssisuorituksen yhteydessä tekoälyä.

Pelin luokkien attribuutit toteutetaan pythonin *@property*-ominaisuuksina, jotta niiden arvojen asettamiselle voidaan tarjota turvallisempi vaihtoehto. Pythonin 2.5 versiossa tämä ei välttämättä tee koodista yhtään luontevampaa ja edellyttää erillisten get, ja set -metodien luomista ja niiden yhdistämistä muotoon

attribuutti = property(get_attribuutti, set_attribuutti),

jossa sekä get_attribuutti ja set_attribuutti on määritelty aikaisemmin. Kaikki pelin luokat periytyvät pythonin object-luokasta tai toisesta, siitä jo perityneestä luokasta.

## 1.2 Luokkasuunnittelua: Peli-luokka

Peli-luokka on pelin pelimoottorin tärkein luokka. Sen luomiskutsulla luodaan myös kaikki pelin käyttämät luokat, joita tarvitaan peli käynnistyessä. Peli-luokka on oikeastaan lista peliin kuuluvista muilla luokista sekä sisältää tiedon pelikierroksesta ja pelivuorosta.

### Metodit

- `def __init__(self, pelaajat, asetustiedosto):`
  - Luo listan pelaajista. Lataa tämän jälkeen asetustiedostosta listan pelissä käytettävistä yhdistelmistä ja luon niiden perusteella pelaajille tulostaulukot. Luo peliin asetustiedostossa määritellyn määrän noppia.
- `def get_Pelaajalista(self):`
  `def get_Pelivuoro(self):`
  `def get_Pelikierros(self):`
  `def get_Yhdistetelmälista(self):`
  `def get_Heitto(self):`
  `def get_Noppalista(self):`
  - Palauttaa alaviivan jälkeisen attribuutin.
- `def Pelikierros_etene(self):`
  - Siirtää pelikierrosta yhdellä eteenpäin ja heittää noppaa ensimmäisen pelaajan kohdalla. Heitto-luokan ilmentymä luodaan aina kuin pelikierros tai pelivuoro etenee. Jos kyseessä on viimeinen kierros pelin loppumisesta ilmoitetaan ja pelin voittajan pisteet lisätään top-10 -listalle, jos pistemäärä on tarpeeksi suuri. Pelikierros tallennetaan Pelikierros-kokonaislukuun.

- `def Pelivuoro_etene(self):`
  - Yrittää edistää pelivuoroa siirtämällä pelivuoroa seuraavalle pelaajalle ja kasvattamalla Pelivuoro-muuttujaa. Jos kyseessä on viimeinen pelaaja, kutsutaan metodia Pelikierros_etene().

## 1.3 Luokkasuunnittelua: Asetukset-luokka

Peli-luokan jälkeen toiseksi tärkein luokka on varmaankin Asetukset-luokka. Tässä luokassa määritellään pelisäännöt. Peliasetukset luodaan lukemalla tiedostosta Peliasetukset. Peliasetuksiin luetaan eri yhdistelmien lisäksi oletusarvot seuraaville muuttujille: kierrosten lukumäärä, noppien maksimisilmäluku ja heittovuorojen lukumäärä yhdellä pelivuorolla. Näiden kolmen muuttujien arvoja voidaan muuttaa ennen pelin alkua (pelikierros 0).

### Metodit

- `def __init__(self, tiedoston_nimi, peli):`
  - Luo asetuksiin tarvittavat muuttujat. Muuttujien arvot luetaan peli-asetuksista kutsumalla metodia set_Yhdistelmalista_tiedostosta(...).
- `def get_Kierrosten lukumaara(self):`
  `def get_Moppien_maksimisl(self):`
  `def get_Heittovuorojen_lkm(self):`
  `def get_Asetuskoodi`
  - Palauttaa alaviivan jälkeisen attribuutin.
- `def set_Yhdistelmalista_tiedostosta(self, tiedoston_nimi):`
  - Lukee Peliasetukset-tiedostosta käytettävät yhdistelmät sekä kierrosten lukumäärän, noppien maksimisilmäluvun ja heittovuorojen lukumäärän yhdellä pelivuorolla.
- `def set_Kierrosten lukumaara(self, Uusi_kierrosten_lkm):`
  `def set_Noppien_maksimisl(self, Uusi_noppien_maksimisl):`
  `def set_Heittovuorojen_lkm(self, Uusi_heittovuorojen_lkm):`
  - Muuttaa alaviivan jälkeisen muuttujan arvoa jälkimmäisen parametrin (kokonaisluku) arvoksi, jos parametri on hyväksyttävällä välillä oleva kokonaisluku.

## 1.4 Luokkasuunnittelua: Noppa-luokka

Noppapelissä tärkeitä tekijöitä ovat tietenkin nopat. Pelin toteutuksessa nopalla ei itse asiassa ole montaa metodia ja attribuuttia. Nopalla on tieto vain neljästä asiasta: maksimisilmäluvustaan, silmäluvustaan, edellisestä silmäluvustaan, sekä lukitsemisstatuksestaan. Edellinen silmäluku-tieto nopalla on siksi, että se mahdollistaa edellisen siirron perumiseen, jos peliin halutaan toteuttaa "Peru"-toiminto. Tämän harjoitustyön toteutuksessa sitä toimintoa ei toteuteta.

### Metodit

- `def __init__(self):`
  - Pelin luomisen yhteydessä luodaan peliasetuksista luettu määrä noppia. Nopat luodaan myös muutettaessa noppien maksimisilmälukua tai noppien lukumäärää.
- `def get_Silmaluku(self):`
  `def get_Lukossa(self):`
  - Palauttaa alaviivan jälkeisen attribuutin. "Lukossa" on totuusarvo.

- `def Heita(self):`
  - Vaihtaa noppien silmalukua ja päivittää edellisen silmäluvun, jos noppa ei ole lukittu. Lukitun nopan heittäminen ei aiheuta mitään toimintoja.
- `def Lukitse(self):`
  - Muuttaa nopan lukossa-arvon todeksi (true).
- `def Avaa(self):`
  - Muuttaa nopan lukossa-arvon epätodeksi (false).
- `def toggle_Lukossa(self):`
  - Muuttaa nopan lukossa-arvon todeksi (true), jos se oli epätosi (false) ja epätodeksi (false), jos se oli tosi (true).

## 1.5 Luokkasuunnittelua: Heitto-luokka

Oleellisesti noppiin liittyy myös tieto kaikkien noppien silmälukujen ominaisuuksista. Yhdessä kaikkien noppien silmäluvut muodostavat heittovuoro-tiedon kanssa heiton. Heitto-luokan ilmentymä luodaan kun ensimmäinen pelaaja on heittänyt noppia. Heitto-luokka tarjoaa Yhdistelmät-luokalle jotain hyödyllisiä tietoja noppien silmäluvuista, kuten kirjaston (dict) noppien silmäluvuista, m:n nopan suorista sekä n:stä kappaleesta noppia, joilla on sama silmäluku (n samaa). Lisäksi Heitto-luokasta saadaan kaikkien noppien silmälukujen summa (sattuma).

### Metodit

- `def __init__(self, peli):`
  - Heittovuoro luodaan pelin ensimmäisen pelaajan heittäessä noppaa.
- `def get_Heittovuoro(self):`
  `def get_Sattuma(self):`
  `def get_Silmaluvut(self):`
  `def get_n_samaa(self):`
  `def get_m_sarja(self):`
  - Palauttaa alaviivan jälkeisen attribuutin.
- `def Uusi_Heittovuoro(self):`
  - Tätä metodia kutsutaan pelaajan heittäessä ensimmäisellä heitto-vuorollaan noppaa. Heiton attribuuteista Heittovuoron arvoksi asetetaan yksi ja muiden attribuuttien arvot luetaan noppien silmä-luvuista.
- `def Heittovuoro_etene(self):`
  - Heiton attribuuteista Heittovuoron arvoa kasvatetaan yhdellä ja muiden attribuuttien arvot luetaan noppien silmäluvuista. Jos kyseessä on viimeinen heittovuoro, ei tehdä mitään.

## 1.6 Luokkasuunnittelua: Yhdistelmä-luokka

Eri yhdistelmille voidaan asettaa erilaisia perusteita siitä, miten pisteet kyseiselle yhdistelmälle lasketaan. Jos yhdistelmä ei täytä kriteerejä, tulee yhdistelmän arvoksi nolla. Yhdistelmälle voidaan asettaa rajoituksiksi noppien maksimi/minimilukumäärä sekä noppien maksimi/minimisilmäluku. Perus-yhdistelmien pisteet lasketaan yhdistelmän koodia käyttäen.

### Metodit

- `def __init__(self, asetukset):`

- Yhdistelmät luodaan peliasetusten lataamisen yhteydessä ja listataan pelin yhdistelmälistaan.
- `def get_Yhdistelmannimi(self):`
  - Palauttaa Yhdistelmän nimen (string).
- `def Laske_arvo(self):`
  - Palauttaa yhdistelmän laskusääntöjen mukaisen pistearvon. Pistearvo voi riippua noppien lukumäärästä, yhdistemälle määritetyistä noppien maksimi/minimilukumäärästä ja maksimi/minimisilmäluvusta sekä noppien maksimisilmäluvusta.

## 1.7 Luokkasuunnittelua: Pelaaja-luokka

Pelaaja-luokka on luotu laajennettavuutta silmällä pitäen. Nykyinen Pelaaja-luokka sisältää pelaajan tiedoista ainoastaan nimen sekä tiedon pelaajan tulostaulukosta.

### Metodit

- `def __init__(self, pelaajan_nimi, peli):`
  - Pelaajat luodaan peliasetusten lataamisen jälkeen ja niille annetaan nimi sekä niille luodaan tulostaulukko.
- `def get_Nimi(self):`
  `def get_Tulostaulukko(self):`
  - Palauttaa alaviivan jälkeisen attribuutin.
- `def set_Nimi(self):`
  - Muuttaa pelaajan nimeä.

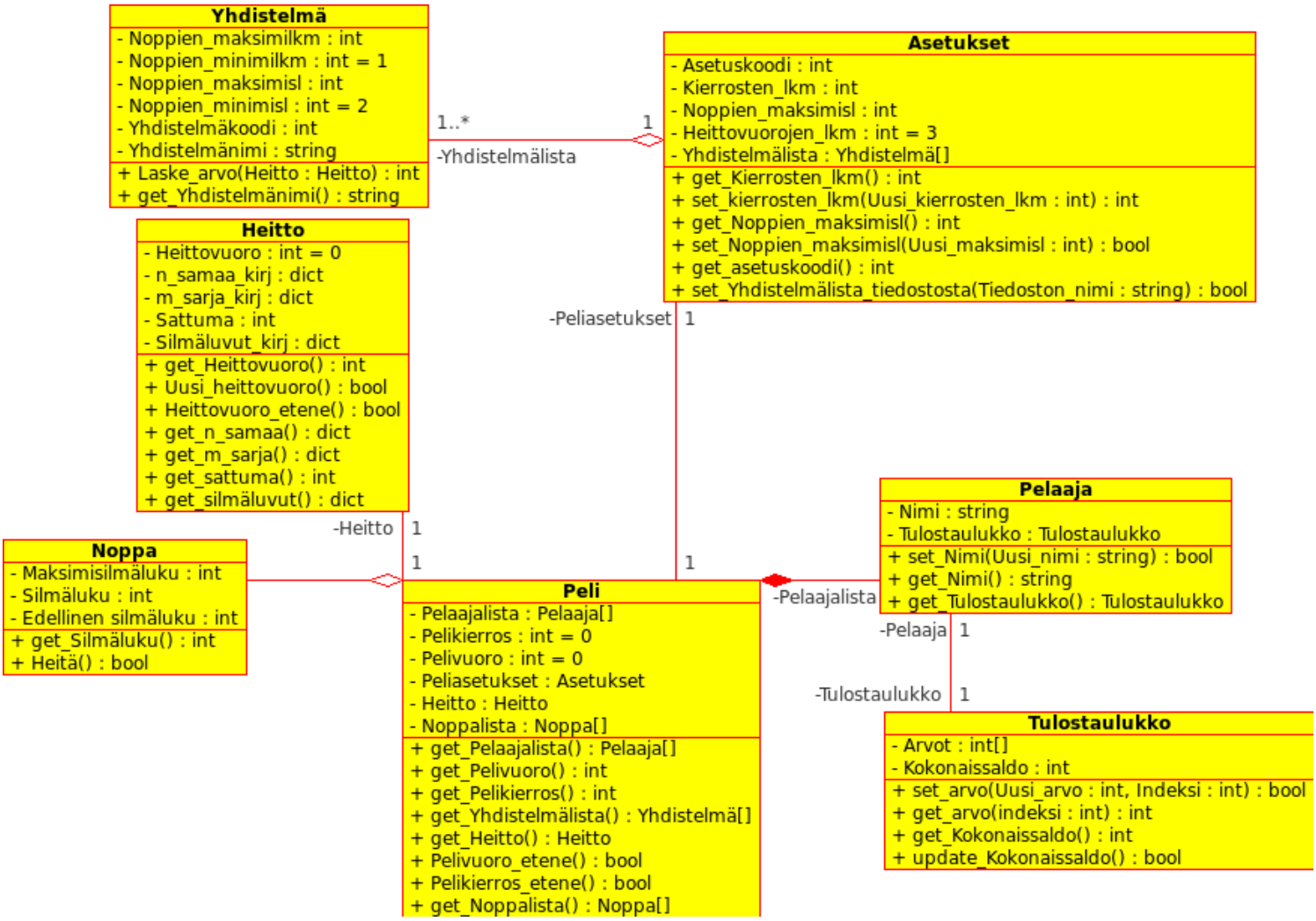
## 1.7 Luokkasuunnittelua: Tulostaulukko-luokka

Tulostaulukkoon kerätään pelaajien pisteet. Tulostaulukossa ei ole mitään tietoa siitä, mistä yhdistelmästä mikäkin pistemäärä tulee. Kokonaissaldo päivitetään aina muuttaessa yhtä arvoa.

### Metodit

- `def __init__(self, peli, pelaaja):`
  - Tulostaulukko pelaajalle luodaan aina uuden pelin aloittamisen yhteydessä sekä asetuksia muuttaessa.
- `def get_Arvo(self, indeksi):`
  `def get_Kokonaissaldo(self):`
  - Palauttaa alaviivan jälkeisen attribuutin.
- `def set_Arvo(self, indeksi, Uusi_arvo):`
  - Indeksin mukaisen arvo-muuttujan arvoa päivitetään uudella arvolla, jos vanha arvo on nolla. Jos vanha arvo on jo olemassa, mitään ei tehdä.

# 2 UML-kaavio pelimoottorista



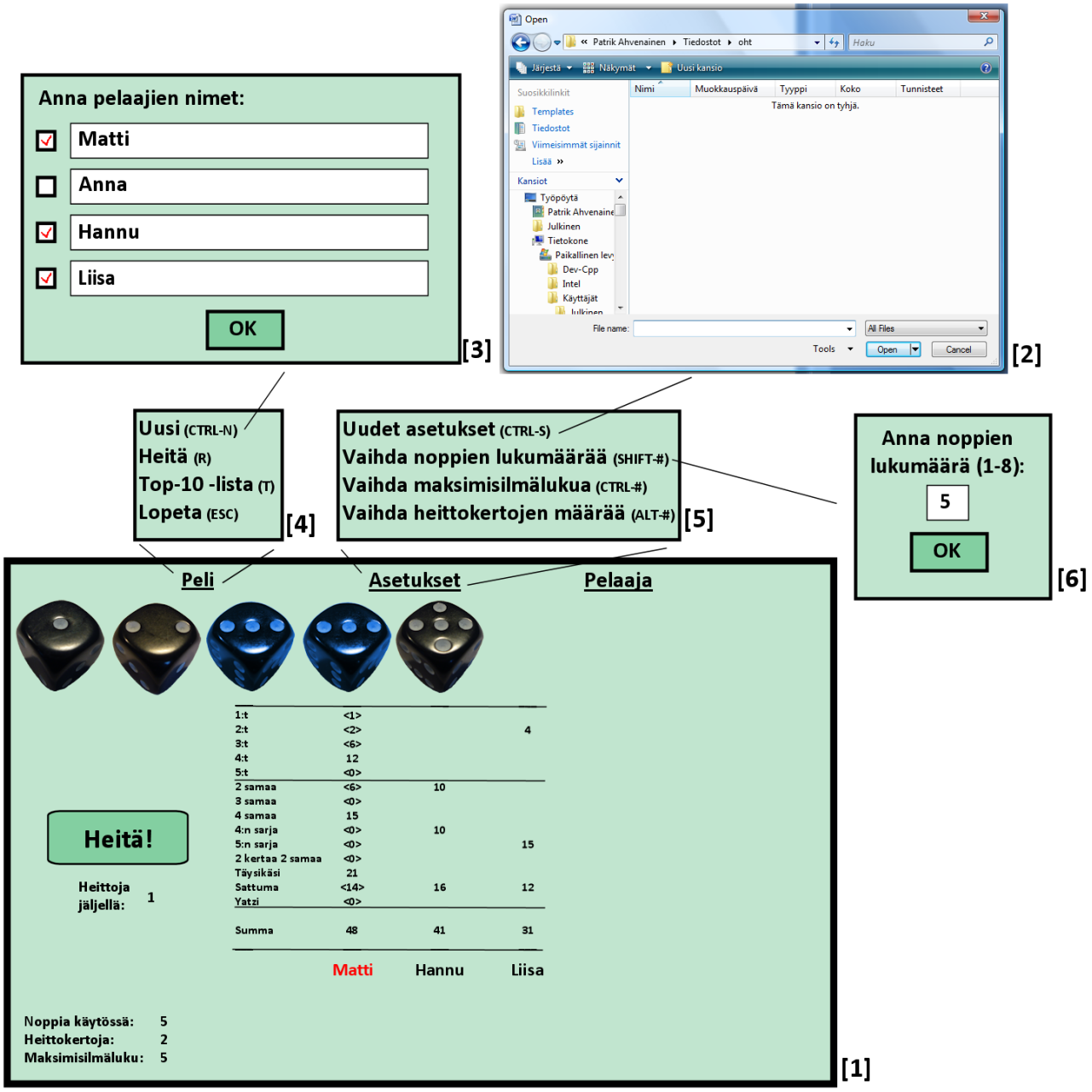

**Kuva 6. UML-kaavio yleisen noppapelin peruskomponenttien keskenäisistä suhteista. Luokkien attribuutit ovat merkitty yksityisiksi (-). Käytännössä tätä rajoitusta ei tulla toteuttamaan, mutta luokkien käyttöön toteutetaan "turvallisia" metodeja, joiden avulla on tarkoitus muokata luokkien attribuutteja.**

# 3 Käyttöliittymän suunnittelua

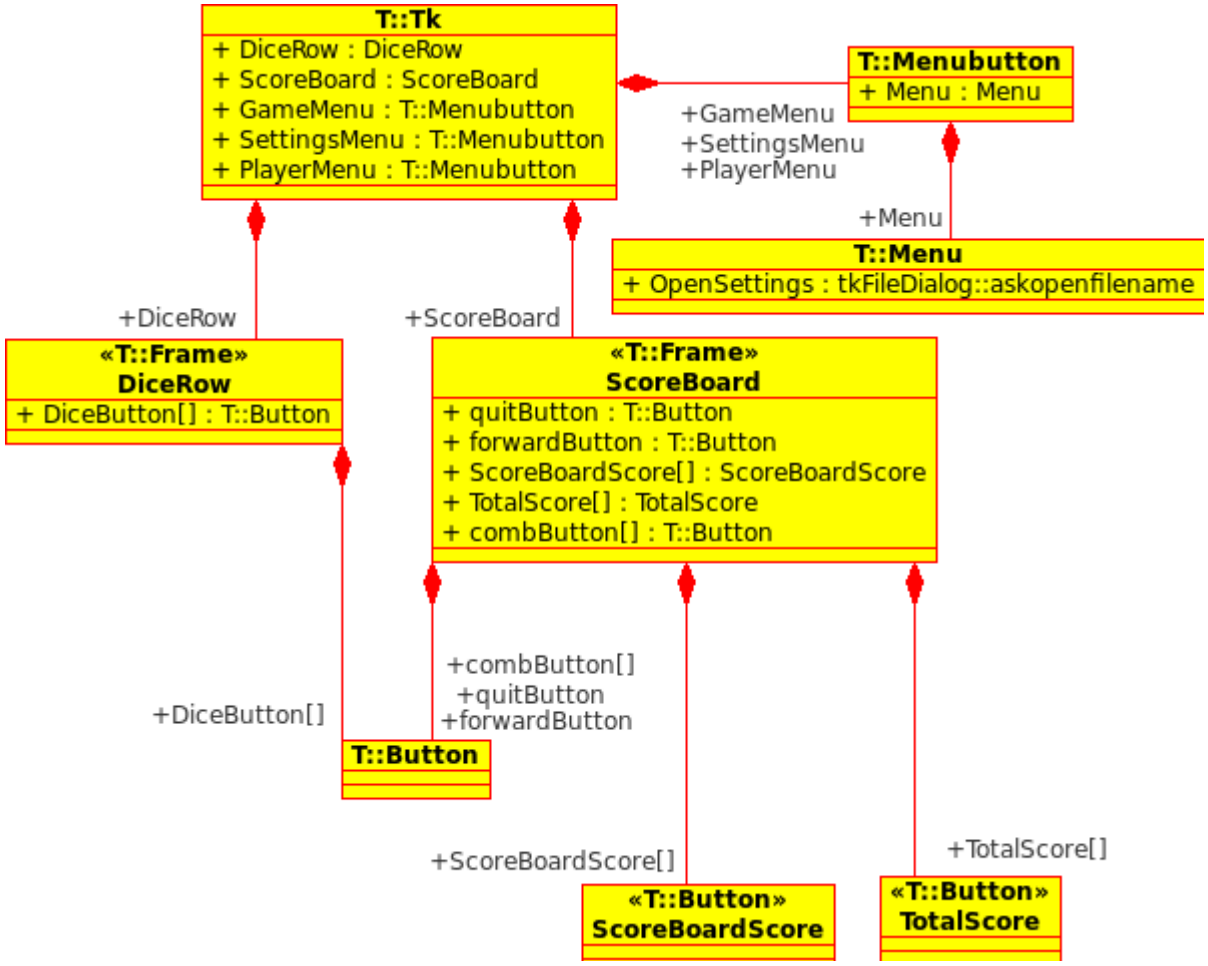
Käyttöliittymä toteutetaan pelimoottorista erillisenä komponenttina. Käyttöliittymä toteutetaan Tkinter-moduulin avulla. Käyttöliittymässä käytetään mahdollisimman paljon valmiita moduuleita, joiden toimivuus on testattu kaikissa käyttöjärjestelmissä.

Pelin päänäyttö [1] (Kuva 5) toteutetaan Frame-widgettinä. Siinä olevat nopat toteutetaan Button-widgettinä. Samoihin widgetteihin perustuvat myös Heitä-nappula, pelaajien tulostaulukkoiden pisteet yksittäisille yhdistelmille ja pika-asetusten lukumäärät (päänäytön vasen alanurkka). Jäljellä olevien heittojen lukumäärä sekä sekä kaikki muu teksti toteutetaan Label-widgettinä. Peli- ja Asetukset -valikot

[4][5] toteutetaan Menu ja MenuButton -widgettejen avulla. Pelaajien nimien kirjoitus-valikko [3] toteutetaan Entry-widgetin avulla. Noppien lukumäärän kyselyyn käytetään Spinbox-widgettiä. Peli-asetustiedoston avaamiseen [2] käytetään tkFileDialog.askopenfilename-ikkunaa.



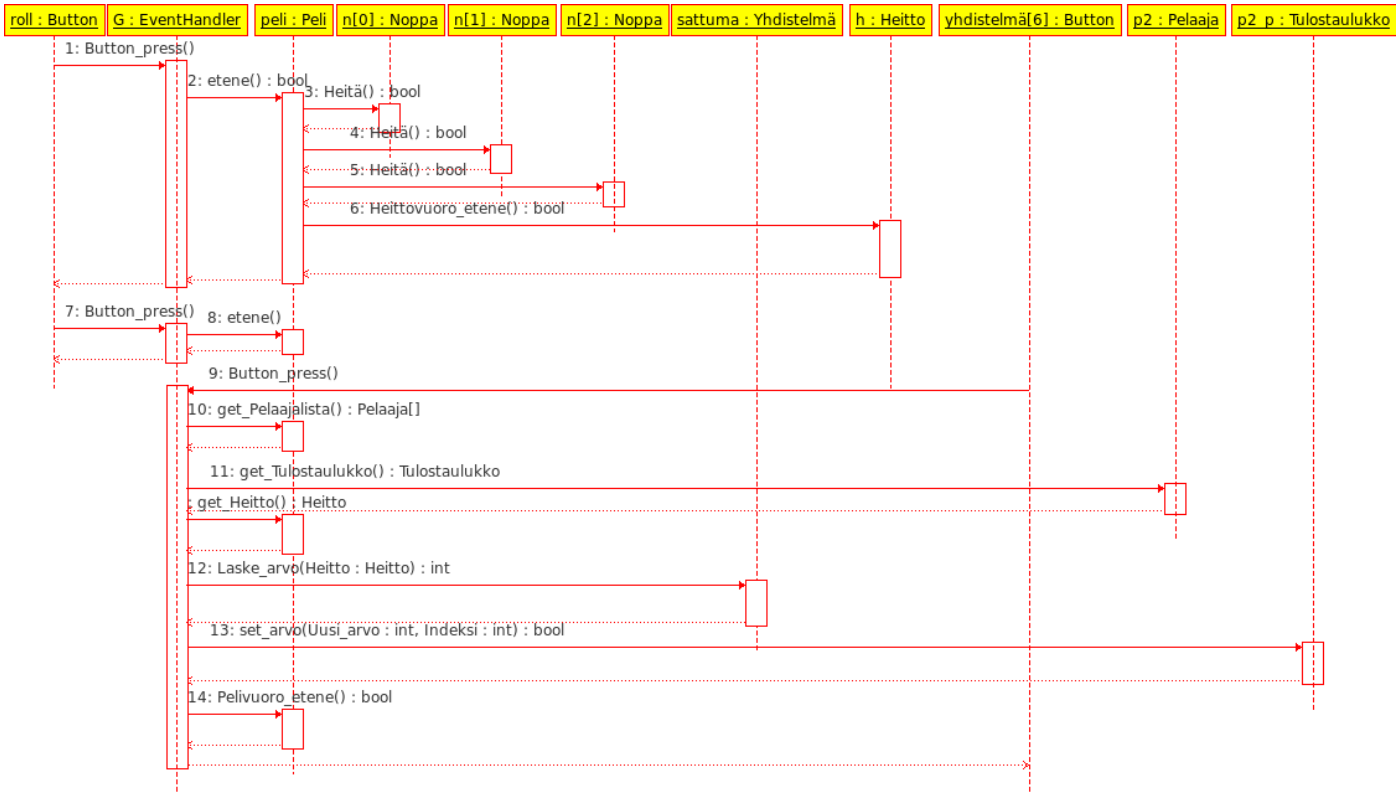

**Kuva 7. Suunnitelma yleisen noppapelin graafiselle käyttöliittymälle. [1] Pelin päänäyttö. [2] Peliasetustiedoston valintaikkunan ulkonäkö riippuu käyttöjärjestelmästä. [3] Peliin osallistuvien pelaajien nimeäminen tapahtuu uuden pelin yhteydessä. [4] Peli-valikosta voi aloittaa uuden pelin ja lopettaa nykyisen sekä katsella top-10 listaa. [5] Asetukset-valikosta voi vaihtaa peliasetustiedostoa sekä noppien lukumäärää, maksimisilmälukua sekä heitto-vuoron heittokertojen määrää. Tämä onnistuu vain pelikertojen välissä. [6] Pikanäppäinten lisäksi Asetukset-valikon kolmea alinta asetusta voi muuttaa valintaikkunan kautta.**




**Kuva 8. Luokkakaavio graafisen käyttöliittymän luokkien suhteista. T viittaa Tkinter-moduliin.**

Tarkemmin pelin graafisen käyttöliittymän luokkia on esitelty luokkakaaviossa (Kuva 8). Pelin pääikkunaan liittyy valikon Menubuttoneiden lisäksi kaksi Frame-objektia, jotka sisältävät pelin nopat ja pelin pistetaulukon. Valikon kautta avautuu uusia ikkunoita joidenkin valintojen tekemiseen (Kuva 5). Nämä ovat pääosin melko yksinkertaisia valikkoja, joiden toiminta palautuu suoraan pelimoottorin metodiksi.

# 4 Sekvenssikaavio pelivuoron etenemisestä




**Kuva 9. Sekvenssikaavio yhdestä pelivuorosta. Tapahtumakulku tässä sekvenssikaaviossa on seuraava. Pelaaja painaa "Heitä" -nappulaa (1). Pelin etene-metodi (2) heittää noppia (3-5) ja siirtää heittovuoroa eteenpäin (6). Tässä vaiheessa noppien silmäluvut vaihtuvat. Kyseisessä esimerkkipelissä ei ole kuin yksi heittokerta heittovuoroa kohti, joten kun pelaaja yrittää heittää uudestaan (7), ei tapahdu mitään (8). Kun pelaaja valitsee jonkin yhdistelmän painamalla tätä lukuarvonappulaa (9), päivitetään pelaajan (10) tulostaulukon (11) arvo heiton avulla lasketulla arvolla (12). Kun tämä lukuarvo on asetettu (13) siirretään pelivuoroa seuraavalle pelaajalle (14).**