# OOP 2. rühmatöö

*Autor: Mihkel Roomet*

## Programmi kirjeldus

Programmi kasutaja saab viia läbi NLHE Collect mängu, mis on pokkerivariant, mille rühmatöö käigus välja mõtlesin. Programmis saab vaadata ja uuendada mänguseisu ja lugeda mängureegleid, lisada ja kustutada mängijaid. Programm jätab mängijad akna sulgemisel meelde, kirjutades nende kohta info faili, ja järgmisel käivitamisel kuvab samad mängijad uuesti, lugedes nende info samast failist. Mängu lõppedes näitab programm infot võitja kohta ja lubab minna kas tagasi mänguseisu juurde või alustada uut mängu.

## Klasside eesmärgid ja olulisemad meetodid

### CardBox

Klass, mis aitab mugavamalt jõuda ühe kaardi kohta käivate graafiliste elementideni.

### NLHECollect

Peaklass, kus käivitatakse JavaFX rakendus. Siin on meetodid erinevate juurte loomiseks ja muude ettetulevate tegevuste läbiviimiseks.

### Player

Klass, kus hoitakse loogilisel kujul mängija mänguseisu. Salvestatud on mängija nimi, knocked raundide arv ja kogutud kaardid täisarvumassiivina.

### PlayerBox

Klass aitab mugavamalt ligi pääseda olulistele graafilistele elementidele, mille olekuid on vaja lugeda ja muuta. Üks PlayerBox vastab ühe mängija kohta käivatele graafilistele elementidele.

### Root

Klass, mis aitab vähendada virvendamist. Iga juure loomisel luuakse Root-objekt, mis sisaldab päris juurt ja selle peal olevat musta ristkülikut, mis vajadusel nähtavaks tehakse, et kasutaja eest peita akna suurendamisel tekkivat virvendust.

### Üld

Klassis on üks üldine abifunktsioon lõime magama panemiseks.

## Projekti tegemise protsessi kirjeldus

Esialgu võtsin aluseks 1. rühmatöö, mis oli tehtud erineva kaardimängu kohta, ja hakkasin seda muutma pokkerile sobivaks. Järgneb ajajoon.

**6. mai 1h** – Teen plaani. Selle järgi on tööetapid:

* Kirjutada valmis programm, millega saab tekstipõhiselt ühe vastase vastu mängida ehk nagu see, mis 1. rühmatöös oli, aga pokkeriga.
* Lisada sellele graafiline kasutajaliides, mis sisaldaks avamenüüd, kust saab navigeerida reeglite leheküljele ja mängima. Siin punktis on miinimumnõuded täidetud. Kui jääb aega, teha ka järgmistes punktides kirjeldatu.
* Teha *campaign* ehk saab järjest mängida erineva tasemega vastaste vastu - järgmise tasemeni jõudmiseks peab eelmise võitma. Tasemeid oleks 3 - esimene mängija vastu, kes teeb väga ebaoptimaalseid otsuseid, teine tavalise mängija vastu ja kolmas mängija vastu, kes teeb sohki (aga tal vb on mingi nõrkus, mille abil teda ikkagi konsistentselt võita saab).
* Lisada omatehtud muusika ja heliefektid.

**7. mai 1h** – Alustan tööga. Võtan projekti kirjelduse aluseks 1. rühmatöö projekti kirjelduse, eemaldades sealt ebasobiva. Nimetan koodis ümber mõned klassid ja muutujad ning kirjutan *help*-sektsiooni (kasutaja mängima õpetamise teksti) ümber.

**8. mai 2h** – Kirjutan ümber kõik, mis eelmisest mängust pärit ja pokkerisse ei sobi, v.a meetod *keepPlaying()*, kuhu on kirjutatud käe lõpetamise loogika, sellega tegelen hiljem. Nimetan klassi *Dealer* ümber *Player*iks. Lisan enum Position, kus hoitakse erinevaid võimalikke mängupositsioone. Kirjutan sinna meetodid suvalise positsiooni leidmiseks ja järgmise positsiooni leidmiseks ning teen nii, et positsioone saab mugavalt loetaval kujul sõnena esitada. Lisan meetodi *displayPot()* poti kohta info väljastamiseks, sest seda läheb ühe käe jooksul ilmselt korduvalt vaja. Mäng töötab nüüd kuni punktini, kus kaardid on kummalegi jagatud ja blindid mängijatelt võetud.

**10. mai 1h** – Lisan klassi Player isendiväljad käes hoitud kaartide jaoks ja praeguse panuse jaoks. Lisan meetodeid klassidesse *Player* ja *Stack*, sealhulgas sean klassi *Stack Comparable<Stack>* liidest realiseerivaks ja viin panuse suurendamise loogika kokku ühte klassi *Player* meetodisse *increaseBet()*. See kõik võimaldab lihtsustada käe alustamise protsessi väljanägemist klassis *NLHE*.

Alustan panustamisringi loogika kirjutamist meetodisse *bettingRound()*. See tundub sisemisest keerulisem, kui arvasin.

**16. mai 0.5h** – Ei ole jõudnud muude asjade kõrvalt projekti edasi teha, nii et pean ilmselt loobuma senisest ideest. Võin minna tagasi 1. rühmatöö juurde ja teha see graafiliseks. Vahepeal tekkis ka kolmas mõte, mille teostamine tundub tehtav - kirjutada programm, mis aitaks läbi viia ühte pokkerivarianti, mille just välja mõtlesin. Selles mängus võidab see, kes kogub mõtteliselt mingi arv ühte ja sama kaarti. Kaarte saab koguda võites käsi. Iga käe korral on kaart, mille võitja saab, *flop*i esimene kaart. Et mängijad ei läheks liialt agressiivseks, on *stack*i tühjaks jooksmisel ooteaeg kätes, mille jooksul tuleb mängust väljas istuda.

Programm näitaks iga mängija kohta, mis kaardid ta kogunud on. Saaks mugavalt lisada mängijaid ja uuendada seisu ning märkida väljas istumist. Programm teataks ka, kui mäng on lõppenud. Võiduks vaja minevate kaartide arv ja väljaistumispenalty kätes oleksid kohandatavad. Kui aega jääb, võiks lisada ka *undo*-mehhanismi kuni 50 eelneva seisuga.

Aega on siiski minimaalselt, nii et pean ilmselt piirduma võimalikult lihtsakoelise siiski nõudeid täitva projektiga.

**19. mai 2h** – Alustan uut projekti võttes aluseks eelnevalt kirjeldatud uusima idee, sest on imperatiivne saada töö võimalikult kiiresti valmis ja see tundub kõige lihtsamini realiseeritav.

Loon 8. koduülesannete *TTT* programmi eeskujul meetodid *getMainPane()* ja *getButtonPane()*.

Uurin erinevate paigutusvõimaluste kohta *JavaFX*-s. Kasutades *bind*-meetodit ja *ColumnConstraints* isendeid seon neljandiku akna suurusest parempoolse osaga, kümnendiku paremast osast nuppude vasaku ääre X-ga ja ülemise nupu ülemise ääre Y-ga ning üheksa kümnendikku paremast osast nuppude parema äärega.

**20. mai 6h** – Guugeldades leian, et tegelikult oleks õige kasutada paigutuse jaoks *FXML*-faili ja loogika jaoks *Controller*-faili. Saan ka teada, et *Scene Builder* abil on palju lihtsam projekte teha. Teen veel uurimistööd, tõmban *Scene Builder*i ja katsetan sellega ning *FXML*-faili käsitsi muutmisega.

Lisan *CSS*-faili ja kirjutan sellesse stiliseeringut. Siiski tundub, et tuleb veel palju õppida selle kohta, kuidas *Scene Builder*is kõike teha ja eri programmi osi omavahel integreerida.

**21. mai 7h** – Mulle tundub, et *Scene Builder*’i ja *FXML* kasutamine oleks siiski natuke imelik minu projekti puhul, sest ma peaks palju koodi *copy-paste*’ima, kuna ilma nendeta saaks kasutada tsüklit. Pealegi ei ole ma kindel, kas neid tööriistu on üldse lubatud kasutada, sest nendega tundub olevat palju lihtsam projekt valmis saada. Seega teen uue projekti ja püüan ilma läbi ajada.

Uurin, kuidas skaleerida teksti ja kuidas teha nii, et akna *ratio* oleks püsiv, aga akna suurust saaks ikkagi muuta. Leian, kui palju suurem on *stage* *scene*’ist, ja kasutan seda teadmist, et akna suuruse muutmine tööle saada. See töötab nüüd olukorras, kus *root* on *BorderPane*, millel on paar teist tühja *Pane*’i sees. Et neid eristada, uurin ka välja, kuidas aplikeerida *CSS*i objektidele ilma *FXML*’i kasutamata, ja teen *root*’iks oleva *Pane*’i sees olevad *Pane*’id eri värvi.

Saan teada, et kõiki arvutisse installitud fonte ei saa niisama *JavaFX*is kasutada – osade jaoks tuleb lisada viide fondifailile *CSS*-faili ja selle viite absoluuttees ei tohi olla tühikuid. See on masendav. Pean paljud kaustad ümber nimetama.

Tekitan nupud ja spinnerid *Label*itega paremale poole ning panen need ja nende teksti korralikult skaleeruma. Valin värvid, sealhulgas ka nuppude eri olekutele. Lisan funktsionaalsuse nuppudele *Full Screen* ja *Quit*.

**22. mai 4h** – Lisan funktsionaalsuse nupule *Help*. See vahetab stseeni juurt ja näitab *scroll*itavat juhendit *Back*-nupuga, mis vahetab stseeni juure tagasi.

**23. mai 1.5h** – Lisan vasakusse *Pane*’i *error message*’i.

**24. mai 2h** – Lisan mängijate jaoks *StackPane*’id ja optimeerin vasaku *Pane*’i struktuuri.

Lisan *knocked-message*’id.

**25. mai 6h** – Vahetan kasti sees oleva teksti näitamiseks mõeldud ühe alluvaga *StackPane*’id välja *TextField*ide vastu peale õppimist, kuidas nendega sellises olukorras töötada, et oleks loogilisem. Lisan vasakusse *Pane*’i kõik elemendid, mis enne puudu olid, ehk kõik, mis näitavad jooksvat infot mängija kohta, v.a *knocked* olemise kohta, ja ka nupud, millega mängija olekut muuta saab.

Koostan muid klasse ja teen juurde mõned isendiväljad, et oleks mugav ligi pääseda eri graafilistele elementidele. Teen ka eraldi klassi *Player* loogilisel kujul andmete salvestamiseks ja alustan selle sidumist graafiliste elementidega.

Teen nupud „*New Player*“ ja „X“ (kustutab mängija) töötavateks ja teen ka nii, et mängija nime saab muuta. Hakkan koostama meetodit, mis vasaku *Pane*’i värskendab.

**26. mai 4h** – Implementeerin loogika. Programm nüüd töötab. Lisan ka võidu näitamiseks eraldi stseeni juure.

## Rühmaliikmete panus ja ajakulu

Tegin kõik ise ja kulus 38h.

## Tegemise mured

Vähesed teadmised selle kohta, kuidas korralikult sellist projekti teha, alguses häirisid mind. Lõpupoole aga oli juba palju lihtsam.

## Hinnang töö lõpptulemusele

Läks hästi. Oleks võinud natuke paremini ja korralikumalt, kui oleks rohkem aega olnud. Näiteks oleksin tahtnud igasugusest virvendamisest lõplikult lahti saada.

## Kuidas testisin

Pidevalt programmi koostamise jooksul.