# Rühmatöö: Kes tahab saada miljonäriks?!

Autorid: Kärt Miidla ja Priit Pärn

Projektiks tegime mälumängu tuntud telesaate “Kes tahab saada miljonäriks?” järgi, kus program küsib kasutajalt erinevaid küsimusi ning kasutaja peab küsimustele vastama.

projekti põhjalik kirjeldus, kus on kirjas programmi eesmärk ja selgitus programmi üldisest tööst, vajadusel lühike kasutusjuhis;

Küsimused mõtlesime ise välja või valisime välja raamatust “Kes Tahab Saada Miljonäriks?”. Ühes mängus on kokku 15 küsimust ning meie koostasime 2x15 küsimuste *set*’i. Programm küsib kõigepealt mängija nime ning siis kumba set’i mängija soovib mängida. Mälumängu alguses on sissejuhatav tekst ning ka paar juhist (nt. Sisestage vastus, mitte vastuse ees olev number). Kui kasutaja on valmis vastama, kuvatakse ekraanile esimene küsimus 3 vastusevariandiga ning ka mängu lõpetamisvõimalusega. Vastusevariandid on suvalises järjekorras. Mängijal on 60 sekundit aega, et küsimusele vastata, muidu langeb ta automaatselt mängust välja (esineb ka hoiatus, et 10 s. on veel aega vastamiseks). Mängijal on võimalik 3 korda kasutada vihjevõimalust, kus ekraanile kuvatakse küsimusega seotud vihje. Kui mängija vasta õigesti, liigub ta automaatselt uue vooru juurde, kus küsimus on raskem. Kui mängija vastab kõigile 15. Küsimusele õigesti on ta võitnud.

iga klassi kohta eraldi selle eesmärk ja olulisemad meetodid;

Programmis on 3 klassi:

* KesTahabSaadaMiljonäriks – põhiklass, kus toimub „mängu mängimine“.
  + Meetod: küsimusteAndmed(failiteekond) loeb failist info ja salvestab selle.
  + Meetod: randomVastused(variandid) lööb vastusevariandid segi.
  + Meetod: setInterval() kasutatakse timeri loendamisel.
  + Meetod: lisaTekst(set) loeb teisest failist iga vastuse lisateksti sisse.
  + Meetod: väljastamine(mängija) toimub kõik vajalik mängu mängimiseks.
* Andmed – klass, kust saadakse andmed.
  + Konstruktor.
  + Get - meetodid.
* Mängija – klass, kus luuakse mängu mängija.
  + Konstruktor.
  + Get - meetodid

projekti tegemise protsessi kirjeldus (erinevad etapid ja rühmaliikmete osalemine neis);

1. Idee – idee peale tuli Priit.
2. Küsimused – küsimuste failil kindla formaadi ning küsimused mõtles välja Kärt, Priit vormistas mõned asjad paremaks.
3. Baasprogramm – algse programmi, kus olid 3 erinevat klassi nii, et programm oskas failist küsimusi, vastuseid, panuseid ja vihjeid eristada, tegi Priit. Klasside Andmed ja Mängija olemasolu muutsid koodi palju ilusamaks ning lihtsamini jälgitavaks.
4. Programmi, mis kasutajaga „suhtles“, kus programm küsis user input’i ning väljastas küsimusi tegi Kärt, Priit parandas paari viga ning kirjutas sissejuhatava teksti.
5. Punktiskoori uuendus peale iga küsimuse vastuse esitamist, tegi Priit.
6. Meetod randomVastused, mis kuvas valikvariandid ekraanile suvalises järjekorras tegi Kärt.
7. Taimer, mille abil oli võimalik vastamisele panna ajapiirang. See oli üks programmi keerulisem osa ning sellega tekkis palju probleeme, kuid kõik probleemid said lahendatud. Selle tegi Priit.
8. Uued küsimused ning kasutajale set’i valikuvõimaluse lisamine. Vastuse sisestamise järel ilmub ekraanile lisatekst eraldi failist, tegi Kärt.

iga rühmaliikme panus (sh tehtud klassid/meetodid) ja ajakulu (orienteeruvalt);

Priit – tegi 3 klassi (KesTahabSaadaMiljonäriks, Andmed, Mängija), tegi meetodid: küsimusteAndmed, setInterval ning arendas meetodit väljastamine. Ajakulu: 13h.

Kärt – tegi küsimuste set’id ning meetodid: randomVastused, lisaTekst ning arendas meetodit väljastamine.

tegemise mured (nt millistest teadmistest/oskustest tundsite projekti tegemisel puudust);

Kärt – Erinevate klasside kasutus tekitas segadust, kuid muutis koodi selgemaks. Siiamaani ei saa ma aru Get meetodi ja konstruktori vajalikkusest. Samuti tekitas palju probleeme alguses java failide kompileerimine (kood ei läinud käima), kuid need probleemid said lahedatud. Algselt kasutasin GitHub Desktopi, kuid ükshetk (settingute muutmisest) näitas GH Desktop nii palju tehtud muutmisi (27000+), et laadisin edaspidi manuaalselt programme Git’i üles.

Priit – Kõige suurem mure oli taimeri tegemine, sest sellega polnud kunagi varem kokku puutunud. Selle implementeerimine programmi oli järgmine suurem mure, sest taimer ei tahtnud õigel ajal seisma jääda.

hinnang oma töö lõpptulemusele (millega saite hästi hakkama ja mis vajab arendamist);

Vahva programm, mida me mõlemad oleme juba oma pereliikmete peal katsetanud. Taimeri üle oleme uhked, aga samas saaks seda veel paremaks teha.

selgitus ja/või näited, kuidas programmi osi eraldi ja programmi tervikuna testisite ehk kuidas veendusite, et programm töötab korrektselt.

Testisime iseseisvalt enne GitHubi laadimist, kuigi vahepeal jäi sisse apsakaid, mida kohe ei märganud. Enne juhendajale esitamist kontrollisime mõlemad eraldi programmi ja katsetasime kõikvõimalikke „eriolukordi“. Taimeri testimiseks tegin täiesti teise projekti Intellijs ja hakkasin sealt ükshaaval asju üle tõstma põhiprogrammi.