Projektisuunnitelma – Rahan hallinta

CS-A1121 - Ohjelmoinnin peruskurssi Y2 14.01.2021-31.05.2021

1. Henkilötiedot

Edward Daka – 705952, Kauppatieteiden kandidaattitutkinto, vsk 2019. 25.2.2021

2. Yleinen kuvaus ja vaikeustaso

Suunnitelmanani on toteuttaa rahanhallinnan ohjelma Pythonissa. Ohjelma lukee tilitapahtumat tiedostosta, erottelee tulot ja menot toisistaan sekä tekee selkeän kuvaajan menojen jakautumisesta piirakkadiagrammilla.

Toteutettavan ohjelman osaa automaattisesti laskea yhteen samaan kauppaan tehdyt ostokerrat. Käyttäjän pitää halutessaan pystyä myös ryhmittelemään (ja poistamaan ryhmittely) kauppoja haluamansa nimikkeen alle, esimerkiksi ”ruokakaupat”, johon käyttäjä lisää haluamansa kaupat.

Tarkoitukseni on alustavasti tehdä Keskivaikea-versio sovelluksesta. Tämä sisältää seuraavat ominaisuudet :

- helpon vaatimukset
- käyttöliittymä ja rahankäytön visualisointi graafisena
- yksikkötestit edes jollekin osalle ohjelmaa

Yksilöivänä suunnitelmana projektilleni olen miettinyt ”Dark Mode”-tilaa, eli käyttöliittymän värin saisi vaihdettua tummaksi painikkeesta.

3. Käyttötapauskuvaus ja käyttöliittymän luonnos

Ohjelmalle luodaan käyttöliittymä. Tämä käyttöliittymä on pääasiallinen kommunikointi tapa käyttäjän ja ohjelman välillä. Käyttäjälle mahdollistetaan tekstin syöttö kolmella eri painikkeella :

1. Lue tiliotetiedot
2. Ryhmittele

Ensimmäisessä syötteessä käyttäjä antaa tiliotteen sisältävän tiedoston nimen. Toisessa tapauksessa käyttäjä syöttää ensin luotavan ryhmän nimen ja tämän jälkeen antaa yksi kerrallaan ryhmään lisättävän tiliotetapahtuman.

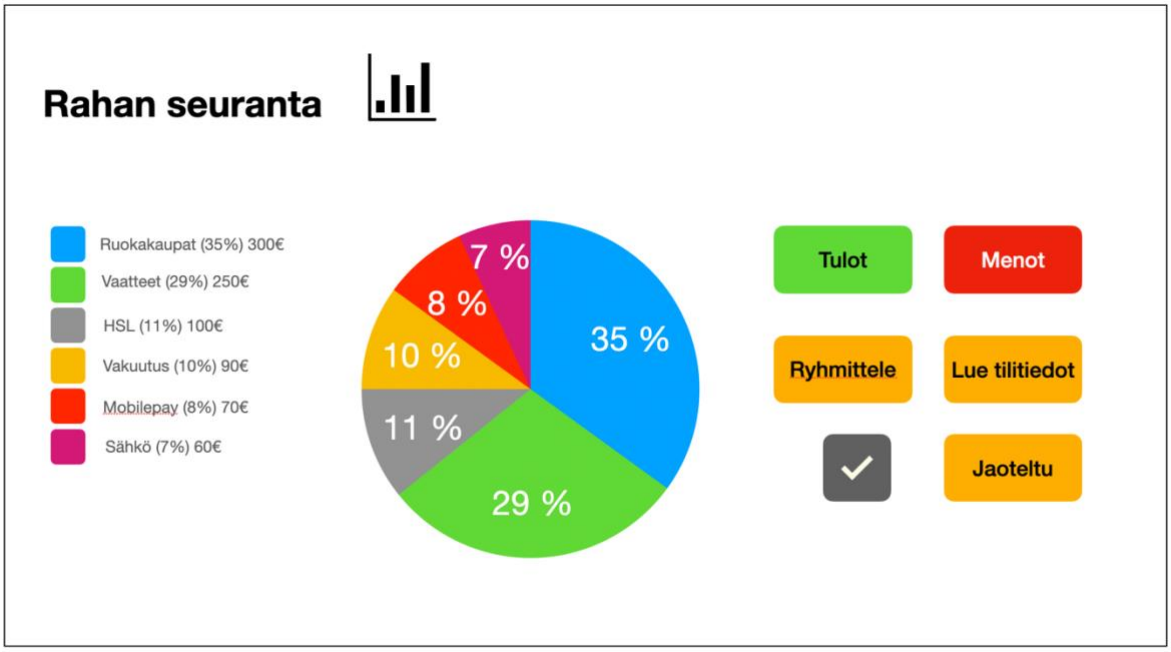
Klikkauksella toimivat seuraavat painikkeet:

1. Tulot

2. Menot
3. Jaoteltu

Tulot ja Menot painikkeet päivittävät grafiikan kuvaaman, joko tiliotteen menoja tai tuloja nimiensä mukaisesti. Jaoteltu painike toimii valintaoptiona. Valintaoptiossa ollessa ruksi, tarkoittaa että grafiikkaa näyttää kuvaajan ja listauksen aiemmin tehdyn ryhmittelyn perusteella. Mikäli valintaoptiossa ei ole ruksi, ei kuvaaja eikä listaus huomioi ryhmittelyä.

Syötteiden syötön jälkeen käyttöliittymä aina päivittää näkymän grafiikat eli piirakkadiagrammin ja listauksen tiliotetapahtumista.



Käyttäjätarina : *”Haluan nähdä tilitietojeni menot”.*

Aloittaakseen käyttäjä käynnistää ohjelman. Tällöin näkyvillä on vain käyttöliittymässä näkyvät painikkeet.

Tämän jälkeen käyttäjä painaa **Lue tilitiedot**-painiketta, joka avaa syöttölaatikon. Tähän asiakas syöttää tilitiedot sisältävän tiedoston nimen. Tämän jälkeen ohjelma hakee tiedoston, ja lukee siinä olevat tilitiedot. Ohjelma luo yksittäiselle tilitapahtumalle oman olion, jolla on nimi ja summa.

Mikäli tilitapahtuman summa on positiivinen, lisätään se *Incomes*-luokassa muodostettavaan ryhmään, mikäli se on negatiivinen, lisätään se *Expenses*-luokassa olevaan ryhmään.

Kun yllä olevat toimenpiteet on suoritettu, välitetään tiedot graafisen käyttöliittymästä vastaavaan *Graph*-luokkaan. Täällä luodaan uusi graafi. Oletuksena graafi näyttää Incomes-luokassa olevat tilitapahtumat.

Käyttäjä tässä tilanteessa painaa **Menot**-painiketta, jotta hän saa menonsa näkyviin.

4. Ohjelman rakennesuunnitelma

Olen suunnitellut ohjelman toteutukseen seuraavanlaisen luokkajaon:

Käyttöliittymän käsittävät luokat :

Interface-luokka = Käsittää käyttöliittymän muodostamisen ja painikkeiden luomisen.
- Sisältää metodit *create_interface, add_buttons, push_buttons*.

Graph-luokka = Käsittää piirakkakuvaajan luomisen ja päivittämisen tietojen/tilojen muuttuessa. On assosioitu Interface-luokkaan.
- Sisältää metodit *create_chart, update_chart, create_listing, update_listing*
  - *Create_* metodit luovat alustavan kuvaajan ja listauksen
  - *Update_* metodit päivittävät nämä, kun vaihdetaan menot/tulot/ryhmitelty/ryhmittelemätön

Tiliotetietojen käsittävät luokat :

ReadStatement-luokka = Lukisi syötteenä saadun tiliotetiedoston ja lähettäisi sen tiedot *StatementManagement*-luokkaan.
- Sisältää metodin *Readfile*, joka hoitaa lukemisen ja palauttaa listan *StatementManagementiin.* Tämä metodi myös varmistaa, ettei samannimistä tilitapahtumaa lueta erillisiksi tilitapahtumiksi, vaan niiden summat summataan.

StatementManagement-luokka = Hallinnoi tiliotetapahtumista muodostuvien olioiden tekemisen ja niiden jaottelun tuloihin ja menoihin.
- Luo oliot tilitapahtumille, nimen ja summa.
- Määrittelee onko tilitapahtuma tulo vai meno *which_type*-metodissa.

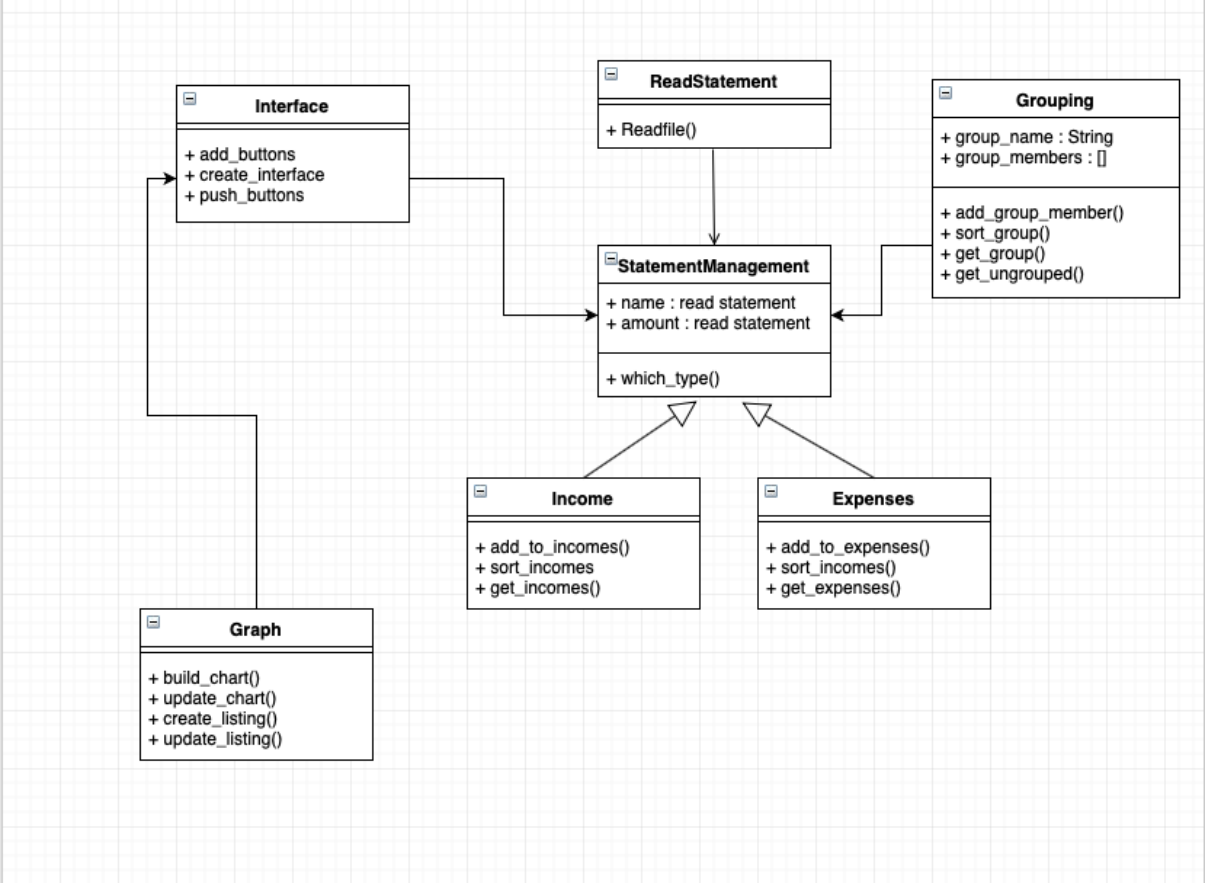
Incomes-luokka = Lisää positiiviset tilitapahtumat omalle listalleen ja osaa jaotella ne suuruus järjestykseen.
- Sisältää metodit *add_incomes, sort_incomes ja get_incomes*

Expenses-luokka = Lisää negatiiviset tilitapahtumat omalle listalleen ja osaa jaotella ne.
- Sisältää metodit *add_expenses, sort_expenses ja get_expenses*.
- Eli *Incomes* ja *Expenses*in metodit luovat listan, järjestävät sen suuruus järjestykseen ja palauttavat listan.
- Molemmat luokat ovat *StatementManagement*-luokan alaluokkia.

Ryhmittelyn käsittelevä luokka :

Grouping-luokka = Tämä pyytäisi luokan nimen, lisäisi siihen käyttäjän haluamat tapahtumat. Luokka myös säilyttäisi alkuperäisen ryhmittelemättömän listauksen.
- Sisältää metodit *add_group, sort_group, get_group ja get_ungrouped*

- Nämä lisäisivät uuden ryhmän, järjestäisivät suurus järjestykseen, antaisi listan ryhmästä ja ryhmittymättömästä.



5. Tietorakenteet

Olen alustavasti suunnitellut, että ohjelmani hyödyntää pääasiassa listoja ja olioihin tallennettuja attribuutteja. Listat ovat mielestäni ohjelman luonteen huomioiden sopiva tapa tallentaa ja järjestellä tietoa. Listat ovat riittävä tapa käsitellä tietoa, sillä tiedon asetetaan pääsääntöisesti suuruus järjestykseen summan perusteella. Myös *Grouping*-luokassa olisi tärkeä saada tiliotetiedot jaettua omiin listoihinsa, joita voi tämän jälkeen järjestellä.

6. Tiedostot ja tiedostoformaatit

Ohjelma hyödyntämät tiliotetiedot ovat CSV-tiedostossa. Käyttämäni pankki on Osuuspankki, joten tilitietojen lukeminen tapahtuu omilla tiliotetapahtumillani, jotka latasin omasta verkkopankistani.

Osuuspankin tilitapahtumat CSV-tiedostossa, relevantteja kohtia ovat ”Summa” ja ”Saajan / Maksajan nimi”. Nämä tiedot tallennetaan luotaville oliolle. Ohjelman on kyettävä erottelemaan vain nämä tiedot tiedostosta, muut tiedot ovat ohjelman toiminnan kannalta irrelevantteja.

7. Algoritmit

Keskeisiä algoritmeja, joita ohjelmisto tarvitsee ovat yksinkertaisia summa, vähennys, jako ja kertolaskuja. Tässä seuraavat laskutoimitukset :

1. Menojen ja tulojen jakautuminen : Edellä mainittujen yhteenlasku ja tämän jälkeen jakaminen, jotta saadaan kunkin tulon/menon suhteellinen osuus. (Yksittäisen tilitapahtuman summa / Tilitapahtumien yhteissumma)

   Tätä suhteellista osuutta hyödynnetään graafin ja listauksen luonnin yhteydessä.

2. Listan järjestäminen : Lista voisi järjestäisi summien perusteella (esim. sort()-metodia hyödyntäen)

8. Testaussuunnitelma

Ohjelman tulisi kyetä seuraavaan:

- Ottaa tiliotetiedosto, lukea se onnistuneesti ja antaa virhe, jos ei lukeminen onnistu.
  - Tätä voi testata erilaisilla rikkinäisille ja toimivilla tiedostoilla. Antaa virhe reaktio, jos kyseessä ei ole *csv*-tiedosto.
- Erottamaan summien perusteella tulot ja menot.
  - Tätä voidaan testata *Income* ja *Expenses*, luokkien kyvyn jaotella summat oikein. Tulisi huomioida myös tilanne, jossa summa on 0.
- Ryhmittelyn tulisi onnistuneesti laskea nimetä uusi ryhmä, laskea käyttäjän syöttämien tapahtumien summat yhteen.
  - Tätä voi testata antamalla *Grouping*-luokalle syötteenä erilaisia testitapahtumia. Osaako luokka niputtaa ryhmän oikein?

Tärkeä että käyttöliittymä palauttaa käyttäjälle virheviestin, mikäli syöte on väärässä muodossa ja pyytää tämän jälkeen uutta syötettä. Tärkeää myös kiinnittää huomiota *ReadStatement*-luokan toimintaan. Mikäli tiedosto luetaan väärin, niin ohjelman antamat rahan seuranta tiedot ovat myös virheellisiä. Oikea oppinen lukeminen ja tiedon esittäminen on kriittistä ohjelman toiminnan kannalta.

9. Kirjastot ja muut työkalut

Ohjelma hyödyntää PyQt-kirjastoa. Kirjastoa hyödynnetään graafisen käyttöliittymän rakentamisessa ja käyttöliittymässä näkyvien kuvaajien muodostamisessa.

Tarkoitus on hyödyntää juuri uusinta versiota PyQt:stä, eli PyQt5.

10.Aikataulu

Ensimmäinen vaihe sisältää käyttöliittymän toteutuksen. Tällä hetkellä omaan rajallisen osaamisen käyttöliittymän luomisesta. Tärkeä oppia PyQt:n käyttö, ja saada ainakin karkea käyttöliittymä valmiiksi, ennen ensimmäistä deadlinea.

Luokkien toteutus etenee seuraavassa järjestyksessä:

1. ReadStatement = Tiliotetiedoston lukeminen on ohjelman toiminnon aloittava tekijä.
2. Statement = Luo oliot yksittäisille tilitapahtumille.
    a. Income = Erittelee tulot
    b. Expenses = Erittelee menot
3. Graph = Luo kuvaajat
4. Grouping = Voidaan toteutaa, kun ollaan ensin saatu perustoimivuus. Toimii ns. bonuksena.

11. Kirjallisuusviitteet ja linkit

The Python Standard Library — Python 3.9.2 documentation. (2021). Retrieved 26 February 2021, from https://docs.python.org/3/library/

PyQt5 Reference Guide — PyQt v5.15 Reference Guide. (2021). Retrieved 26 February 2021, from https://www.riverbankcomputing.com/static/Docs/PyQt5/

Matthes, E. (2020). *Python Crash Course, 2nd Edition* (2nd ed.). San Francisco: No Starch Press Inc.

PyQt5 tutorial 2021 — Create GUI applications with Python and Qt. (2021). Retrieved 26 February 2021, from https://www.learnpyqt.com

Kurssimateriaali Y2. (2021). Retrieved 26 February 2021, from https://plus.cs.aalto.fi/y2/2021/toc/

1. Tietokoneista ja ohjelmista — Ohjelmoinnin peruskurssi Y1 0.0.1 documentation. (2021). Retrieved 26 February 2021, from https://grader.cs.aalto.fi/static/y1/moniste/tietokoneista_ja_ohjelmista.html