1. **Henkilötiedot**

   Rahan hallinta

   Tiina Mikkola, 1024674

   Kauppatieteiden kandidaatti

   12.5.2023

2. **Yleiskuvaus**

   Ohjelma, joka lukee tilitapahtumat tiedostosta (.csv), erottelee tulot ja menot toisistaan sekä taulukoi menot ja tulot. Toteutettavan ohjelma laskee yhteen samaan kauppaan tehdyt ostokerrat. Käyttäjän pystyy myös ryhmittelemään (ja poistamaan ryhmittely) kauppoja ja tuloja haluamansa nimikkeen alle, esimerkiksi ”ruokakaupat”, johon käyttäjä lisää haluamansa kaupat yksitellen.

   Käyttäjän saa sekä ryhmitellyn että ryhmittelemättömän kuvaajan rahan käytöstä. Kulujen lisäksi myös tulojen jakautumista kuvataan taulukolla.

3. **Käyttöohje**

   Ohjelma käynnistetään ajamalla main.py -tiedosto.

   Tekstipohjainen käyttöliittymä kommunikoi käyttäjän kanssa, kertoen mitä ohjelmalla voi tehdä ja mitä komentoja käyttäjältä milloinkin vaaditaan.

   Käyttäjältä pyydetään ensin tilitapahtumatiedoston nimi (projektissa on tiedosto ’tilitapahtuma.csv’ jota käyttää). Kun tiedosto on luettu onnistuneesti, käyttäjä voi valita, haluaako hän ryhmitellä tuloja ja menoja, tulostaa taulukon tulojen ja menojen jakautumisesta vai lopettaa ohjelman käytön.

   Yhden suoritetun pyynnön (esim. uuden ryhmän muodostamisen jälkeen) käyttäjä pääsee uudelleen valitsemaan, mitä haluaa tehdä seuraavaksi. Ohjelma jatkaa seuraavan pyynnön kysymistä kunnes käyttäjä valitsee ”exit” -vaihtoehdon, jolloin ohjelma pysähtyy.

4. **Ulkoiset kirjastot**

   Unittest-kirjastoa käytetty yksikkötestaukseen.

5. **Ohjelman rakenne**

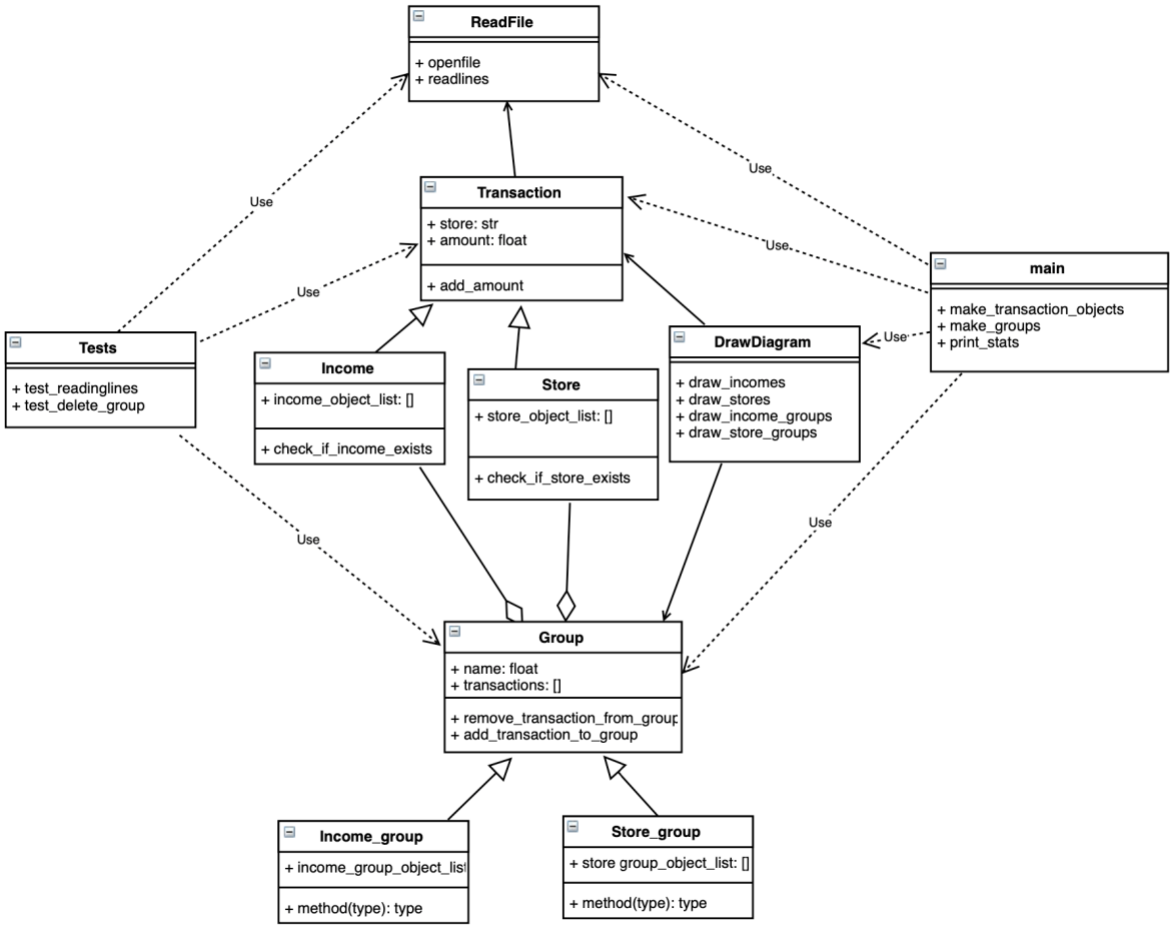
   ReadFile-luokka sisältää tiedoston avaavan openfile-funktion sekä tiedoston rivit lukevan readlines-funktion. Readlines-funktio palauttaa listan, jossa yksittäinen transaktio (kaupan nimi;summa) on listan alkio.

   Ohjelman keskeiset luokat ovat Transaction ja Group, joista ensimmäinen kuvaa yksittäistä kauppaa ja siihen tehtyjä ostoksia, ja jälkimmäinen yhtä

käyttäjän luomaa ryhmää ja siihen kuuluvia Transaction-luokan olioita. Molemmilla luokilla on alaluokat, jotka käsittelevät tuloja ja menoja erikseen.

Transaction ja Group-luokkien alaluokat sisältävät listat luokan objekteista. Alaluokissa on myös funktiot, jotka tarkistavat, onko jokin luokan ilmentymä jo olemassa (check_if_x_exists), ja tarvittaessa löytävät oikean luokan ilmentymän sen nimen perusteella (get_group). Toimintoja tarvitaan, kun esimerkiksi halutaan poistaa jokin ryhmä. Nämä funktiot korvaavat alkuperäisen projektisuunnitelman GroupHandler-luokan.

DrawDiagram-luokka taas sisältää algoritmit jotka laskevat menojen ja tulojen suhteellisen jakautumisen sekä kaupoittain että ryhmittäin. Kommunikointi käyttäjän kanssa tapahtuu main.py -tiedostosta käsin.



6. **Algoritmit**

   DrawDiagram-luokka:

DrawDiagram-luokassa on omat funktiot tuloille ja menoille yksittäin, sekä tuloille ja menoille ryhmittäin. Kaikissa funktioissa lasketaan ensin tulojen/menojen kokonaissumma. Tämän jälkeen lasketaan suhteelliset osuudet jakamalla yksittäisen transaktion summa kokonaissummalla (yksittäiset transaktiot) tai luokan transaktioiden summa kokonaissummalla (ryhmät). Kyse on siis yksinkertsaisesta prosenttilaskusta.

get_best_match -funktio (main.py):

Käyttäjältä pyydetään ohjelmassa usein luokan tai transaktion nimeä, jonka käyttäjä antaa inputina käyttöliittymässä. Funktio get_best_match pyrkii löytämään parhaan vastineen inputille.

Funktio pilkkoo ensin käyttäjän antaman sanan sen kaikiksi mahdollisiksi substringeiksi. Tämän jälkeen funktio käy listan, joka sisältää mahdolliset vaihtoehdot (esim. ryhmästä löytyvät transaktiot), pilkkoo jokaisen näistä kaikiksi mahdollisiksi substringeiksi, ja pyrkii löytämään pisimmän substringin, joka on yhteinen sekä käyttäjän inputille että mahdollisille vaihtoehdoille.

Mikäli funktio löytää yli 2 merkkiä pitkän yhteisen substringin, se palauttaa sanan, joka on paras vastine annetulle inputille. Mikäli funktio ei löydä vastinetta, se palauttaa käyttäjän alun perin ehdottaman sanan.

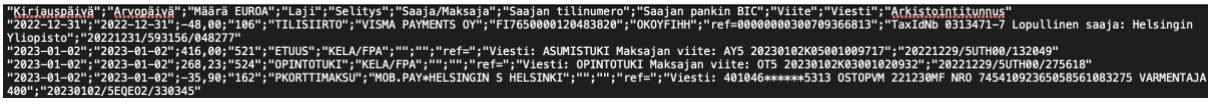
7. **Tietorakenteet**

   Transaction ja Groups-luokkien ilmentymiä tallennettiin listoihin, jotta ryhmittelyä tehtäessä pystyttiin käymään läpi kaikki luokkien ilmentymät, löytämään oikea luokan ilmentymä ja tarkistamaan, onko ilmentymä olemassa.

   Myös ReadFile-luokassa tiedot (kauppa;summa) tallennettiin listan alkoiksi. Tällöin listan läpi iterointi ja Transactions-olioiden alustaminen oli sujuvaa. DrawDiagram-luokan funktiot palauttivat samanlaisen listan (alkiot tyyppiä kauppa;summa), jolloin listan data voitiin pilkkoa ja tulostaa riveittäin.

8. **Tiedostot**

   Tiedostoformaatti on .csv.. Tiedosto sisältää riveittäin tilitapahtumat. Yksittäinen tilitapahtuma (rivi) sisältää tiedon kirjauspäivästä, määrästä, selityksestä, viitteestä sekä saajan tiedoista. Tiedostot ovat esimerkiksi seuraavanlaisia:



   Tiedostossa rivien osat on pilkottu ; -merkeillä. Rivillä kolmantena on summa, ja kuudentena saajan/maksajan nimi. Ensimmäinen rivi on otsikkorivi.

9. **Testaus**

Tests-luokka:

test_readinglines: Yksikkötesti varmistaa, että tiedostoa lukiessa readlines-funktio löytää oikean datan (kaupan nimi;summa) ja palauttaa sen oikein muotoiltuna listan alkiona.

test_delete_group: Yksikkötesti varmistaa, että kun jokin ryhmä poistetaan, ryhmän Transactions-oliot ei poistu.

Molemmat yksikkötestit läpäistään. Suunnitelmassa oli lisäksi tarkoitus testata piirakkadiagrammin piirtämistä, mutta koska projektissa ei lopulta ole graafista visualisointia, tämä testi jätettiin tekemättä.

Muu testaus:

Main-funktiossa on try-except -rakenne, joka antaa virheilmoituksen, mikäli tiedostonimi on virheellinen (OSError).

ReadFile-luokan readlines-funktio heittää poikkeuksen, mikäli annetussa tiedostossa ei ole yhtään virheetöntä tilitapahtumariviä. Lisäksi readlines hyppää rivien yli, jos rivillä on selvästi liian vähän dataa. Funktio readlines on kuitenkin edelleen melko altis tiedostovirheille, eikä se esimerkiksi tunnista, mikäli yksittäisellä rivillä on tietoa väärässä järjestyksessä.

Ohjelmaa tehtäessä sitä on testattu ensisijaisesti ajamalla ohjelmaa yhden tilitapahtuman sisältävää testitiedostoa (testfile) käyttäen.

10. **Ohjelman tunnetut puutteet ja viat**

    Main-funktio:

    Mikäli käyttäjä antaa muun kuin kokonaisluvun kysyttäessä, mitä käyttäjä haluaisi tehdä (vaihtoehdot 1-6), heittää ohjelma ValueErrorin.

    ReadFile:

    Lukee vain OP:n csv-muotoiden tilitapahtumatiedoston, ja huomaa siinä ainoastaan hyvin karkean tason virheitä.

    Group, Transactions ja DrawDiagram:

    Group ja Transactions-luokkien alaluokissa on keskenään hyvin samanlaisia funktioita (get_group, check_if_x_exists jne), jotka olisi voinut todennäköisesti siirtää yläluokkaan. Myös DrawDiagram-luokassa on paljon toistoa, sillä tuloille ja menoille on omat, samaa asiaa suorittavat funktiot. Toisto ja koodin suora kopiointi ei ole tehokasta tai hyvän koodaustavan mukaista.

11. **3 parasta ja 3 heikointa kohtaa**

    Parhaat kohdat:

Käyttäjän kanssa kommunikointi. Tekstipohjainen käyttöliittymä on mielestäni selkeä ja kuljettaa käyttäjän ohjelman läpi. Käyttäjälle ilmoitetaan, mikäli jokin yritetty toimi on virheellinen ja kerrotaan miten käyttäjän on toimittava, jotta ohjelma toimisi.

Funktio get_best_match. Funktio helpottaa oikean transaktion ja ryhmän löytämistä, sillä ei vaadita täydellistä osumaa.

Heikkoudet:

Toiston määrä. Koska tuloja ja menoja käsitellään erikseen, mutta niille tehtävät toimet ovat samanlaisia, koodissa on paljon toistoa. Erityisesti Group ja Transactions-luokkien alaluokkia voisi yhdistää yläluokkaan, tai rakentaa erillinen luokka (GroupHandler) joka hoitaa alaluokkien tehtäviä.

Tiedoston lukeminen. Ohjelma on suunniteltu lukemaan vain OP:n csv-muotoisen tilitapahtumatiedoston, ja ohjelma huomaa siinä ainoastaan hyvin karkean tason virheitä. Ohjelma ei toimi halutunlaisesti muilla tiedostoilla, eikä käyttäjä saa helposti tietoonsa, mikäli tiedoston pohjalta alustetuissa Transaction-olioissa on virheitä. Ohjelman voisi muokata niin, että tietyn, määrätyn indeksin sijaan se löytäisi indeksin, jolla on otsikkorivillä ”määrä”, sekä indeksin, jolla on otsikkorivillä ”saaja/maksaja”. Tällöin readlines osaisi lukea myös tiedostoja, joissa haluttu tieto on rivillä eri kohdassa.

Visualisointi. Tuloja ja menoja tulostetaan taulukkona, josta löytyy sekä yhteen kohteeseen käytetty summa että sen osuus kokonaissummasta. Jonkinlainen graafinen kuvaaja toisi kuitenkin lisäarvoa ja auttaisi hahmottamaan menojen ja tulojen jakautumista. Lisäksi heikkoutena on, että esimerkiksi omien tilien väliset siirrot näkyvät taulukossa ja vääristävät suhteellisia osuuksia (esimerkiksi tilisiirto käyttötililtä säästötilille näkyisi käyttötilillä menoeränä). Ohjelmaan olisi järkevää lisätä mahdollisuus poistaa joitakin transaktioita tulostuksesta.

12. **Poikkeamat suunnitelmasta**

    Jätin GroupHandler-luokan tekemättä ajanpuutteen vuoksi. Group-luokka ja sen alaluokat huolehtivat GroupHandler-luokalle alun perin tarkoitetut tehtävät (oikean ryhmän löytäminen, ryhmän olemassaolon tarkistus, ryhmän poisto).

    Ajanpuutteen vuoksi päädyin toteuttamaan työn helppojen vaatimusten mukaan, vaikka alkuperäisenä suunnitelmana oli täyttää keskivaikean työn vaatimukset. Toteutusjärjestys noudatti kuitenkin melko hyvin suunniteltua: projektin teko alkoi ReadFile-luokasta, jonka jälkeen tein Transaction-luokan ja lopulta Group-luokan sekä viimeisenä DrawDiagram-luokan.

13. **Toteutunut työjärjestys ja aikataulu**

Ajankäyttöarvio ei toteutunut, ja projektin tekeminen alkoi kunnolla vasta huhti-toukokuun vaihteessa. Tämä johtui liian kiireisestä aikataulusta ja muiden kurssien odotettua suuremmasta työmäärästä.

Vko17:

ReadFile-luokan aloitus.

Vko18:

ReadFile, Transaction ja Group-luokan tekoa.

Vko19:

Transaction ja Group-luokat loppuun. DrawDiagram-luokan teko. Koodin siistiminen. Dokumentaation teko.

14. **Arvio lopputuloksesta**

    Ohjelma toteuttaa helpon työn vaatimukset. Ohjelma on käyttäjäystävällinen ja kommunikoi käyttäjän kanssa selkeästi. Ohjelmassa hyödynnetään runsaasti olioita ja perintää, ja ohjelma havaitsee ainakin osan mahdollisista virhetilanteista.

    Ohjelman haasteena on toiston määrä, jota voisi vähentää esimerkiksi uusia luokkia lisäämällä. Ohjelma toimii vain tietynlaisilla tiedostoilla. Ohjelma visualisoi tuloja ja menoja melko karkealla tasolla.

15. **Viitteet**

    Stack Overflow, stackoverflow.com

    Python Official Documentation: https://docs.python.org

16. **Liitteet**

Esimerkkiajo:

Please enter the name of file: tilitapahtuma.csv

File read. Now you can organize transactions into groups (example: Groceries).

What would you like to do? (1 = add new groups, 2 = add transactions to existing groups, 3 = remove transactions from groups 4 = delete groups, 5 = print statistics, 6 = exit): 1

Do you want to add a group for incomes or costs? (income/cost): cost

Please enter name of new group: Groceries

Cost group Groceries added.

What would you like to do next? (1 = add new groups, 2 = add transactions to existing groups, 3 = remove transactions from groups, 4 = delete groups, 5 = print statistics 6 = exit): 2

Do you want to add transactions to income or cost groups? (income/cost): cost

Current cost groups:['Groceries']

Choose a group in which you want to add transactions: groc

Did you mean 'Groceries'? (yes/no) yes

Current cost transactions: ['MIKKOLA TIINA MARIA', 'PRISMA ISO OMENA ESPOO', 'LIDL ESPOO-ISO-OMEN ESPOO', 'R_ESPOO_Otaniementi ESPOO', 'Compass Group Finla Espoo']

Which cost transaction do you want to add to group Groceries?: prisma

Did you mean 'PRISMA ISO OMENA ESPOO'? (yes/no) yes

Transaction added to group.

What would you like to do next? (1 = add new groups, 2 = add transactions to existing groups, 3 = remove transactions from groups, 4 = delete groups, 5 = print statistics 6 = exit): 2

Do you want to add transactions to income or cost groups? (income/cost): cost

Current cost groups:['Groceries']

Choose a group in which you want to add transactions: lidl

Cost group lidl does not exist. Choose another group: groceries

Did you mean 'Groceries'? (yes/no) yes

Current cost transactions: ['MIKKOLA TIINA MARIA', 'PRISMA ISO OMENA ESPOO', 'LIDL ESPOO-ISO-OMEN ESPOO', 'R_ESPOO_Otaniementi ESPOO', 'Compass Group Finla Espoo']

Which cost transaction do you want to add to group Groceries?: lidl

Did you mean 'LIDL ESPOO-ISO-OMEN ESPOO'? (yes/no) yes

Transaction added to group.

What would you like to do next? (1 = add new groups, 2 = add transactions to existing groups, 3 = remove transactions from groups, 4 = delete groups, 5 = print statistics 6 = exit): 5

What data do you want to print? (1 = incomes individually, 2 = costs individually, 3 = income groups, 4 = cost groups) 4

****

Group sum % of total spending

Groceries 27.06 3.61

****

What would you like to do next? (1 = add new groups, 2 = add transactions to existing groups, 3 = remove transactions from groups, 4 = delete groups, 5 = print statistics 6 = exit): 6

Process finished with exit code 0