Älykkään reseptikirjan dokumentaatio

Release 1.0

Kimi Päivärinta, 351393, AIT 3 vk.

05.05.2016

Sisältö

LUKU 1

Yleiskuvaus

Toteutin Älykkään reseptikirjan keskivaikeana. Ohjelmassa on graafinen käyttöliittymä, jossa voi selata tallennettuja reseptejä, raaka-aineita sekä varastossa olevia raaka-aineita. Näitä kaikkia on myös mahdollista muokata, sekä reseptejä on mahdollista luoda ohjelmassa. Käyttäjä voi myös tallentaa tekemänsä muutokset tiedostoihin tai halutessaan ladata tiedostosta alkuperäiset reseptit, raaka-aineet ja/tai varastotilanteen. Älykkään reseptikirjasta tekee sen hakutoiminnot. Reseptikirjalla voi hakea reseptit, joihin löytyy raaka-aineet varastosta tai jotka sisältävät esim. tiettyä raaka-ainetta.

Projekti täyttää keskivaikean toteutuksen vaatimukset. Lisätoiminnallisuuksien implementoimisen sijaan keskityin koodin hyvään luettavuuteen, kommentointiin sekä koodin automaattiseen dokumentointiin. Oikeastaan 99% koodin metodeista on kommentoitu ja tämän dokumentin lopussa on koodista automaattisesti generoitu dokumentointi. Siitä käy ilmi jokaisen metodin toiminnallisuus. Näiden lisäksi tein myös kattavat yksikkötestit ohjelman tärkeimmille toiminnoille.

Ohjelman vaatimukset:

- Python 3.x versio
- PyQt5

Projektin toiminnallisuudet alla. Suurimmilta osin kuvaus on projektin vaatimuksista, mutta osaa kohtia on muokattu.

- Ohjelman käyttäjällä on jääkaappi ja muut kaapit täynnä ruoan raaka-aineita ja tarkat ajanmukaiset tiedot siitä kuinka paljon mitäkin ainetta on varastossa. Ohjelman avulla voidaan etsiä vain ne ruokalajit, jotka voidaan valmistaa käymättä kaupassa tai ruokalajit, jotka vaativat N puuttuvaa tai osittain puuttuvaa ainetta.
- Lisäksi voidaan hakea erityisesti reseptit jotka sisältävät tiettyjä aineita. Ruoka-aineisiin voidaan liittää merkintä siitä, että se sisältää jotakin allergisoivaa tekijää. (Esim maito, kerma jne sisältävät laktoosia) Ohjelmalla tulee voida rajoittaa haku niin, että tietyssä haussa vältettäväksi halutut allergeenit jäävät pois.
- Jotkut raaka-aineet voidaan myös rakentaa itse reseptistä kuin ruokalajit. Esim jauhelihapihvin resepti sisältää lihamureketaikinaa jolla taas on oma resepti. Vastaavasti joulutortut tehdään voitaikinasta, joka sekin on itse valmistettavissa. Jos jääkaapissa ei ole voitaikinaa, ohjelman tulee yrittää koota taikina raaka-aineista. (Huomaa että tämä tekee raaka-aineista ja tuotetuista ruoista hyvin samankaltaisia...)
- Lienee itsestään selvää että jotkin aineet kuten munat tai sipulit ovat laskettavissa kappaleittain. Moni raaka-aine kuitenkin mitataan erilaisilla mitoilla. Kaapissa oleva jauho myös ostetaan kiloittain, mutta mitataan resepteissä desilitroissa. Tee ohjelmaasi luokka joka hoitaa kaikki muunnokset mittojen välillä. (vinkki: aineella on tiheys)
- Graafinen käyttöliittymä
- (Komentorivi käyttöliittymä, jossa osa toiminnoista, mutta testaaminen hyvin vähäistä.)
- Ohjelman käyttäjällä on jääkaappi ja muut kaapit täynnä ruoan raaka-aineita. Ohjelmalla voidaan pitää näistä kirjaa.

- Ohjelman avulla voidaan etsiä vain ne ruokalajit, jotka voidaan valmistaa käymättä kaupassa tai ruokalajit, jotka vaativat korkeintaan N puuttuvaa tai osittain puuttuvaa (riittää vain pieneen määrään) ainetta.
- Lisäksi voidaan hakea erityisesti reseptit jotka sisältävät tiettyjä aineita. Vaikkapa kala täytyy käyttää pois.
- Raaka-aineisiin tulee voida liittää merkintä siitä, että kyseinen raaka-aine sisältää jotakin allergisoivaa tekijää. (Esim maito, kerma jne sisältävät laktoosia) Ohjelmalla tulee voida rajoittaa reseptien haku niin, että kyseisessä haussa vältettäväksi halutut allergeenit jäävät pois.
- Jotkut raaka-aineet voidaan myös rakentaa itse reseptistä samoin kuin ruokalajit. Esim jauhelihapihvin resepti sisältää lihamureketaikinaa, jolla taas on oma resepti. Vastaavasti joulutortut tehdään voitaikinasta, joka sekin on itse valmistettavissa. Jos jääkaapissa ei ole valmista voitaikinaa, ohjelman tulee katsoa sen sijaan onko taikinan tekoon tarvittavat raaka-aineet tarjolla.
- Lienee itsestään selvää että jotkin aineet kuten munat tai sipulit ovat laskettavissa kappaleittain. Moni raaka-aine kuitenkin mitataan erilaisilla mitoilla. Kaapissa oleva jauho myös ostetaan kiloittain, mutta mitataan resepteissä desilitroissa. Tee ohjelmaasi luokka joka hoitaa kaikki muunnokset mittojen välillä.
- Raaka-aineet allergiamäärityksineen, ruokareseptit ja jääkaapin sisältö tulee tallentaa sopiviin tiedostoihin tekstimuodossa.
- reseptien interaktiivinen rakentaminen ja muokkaaminen käyttöliittymän kautta

Käyttöohje

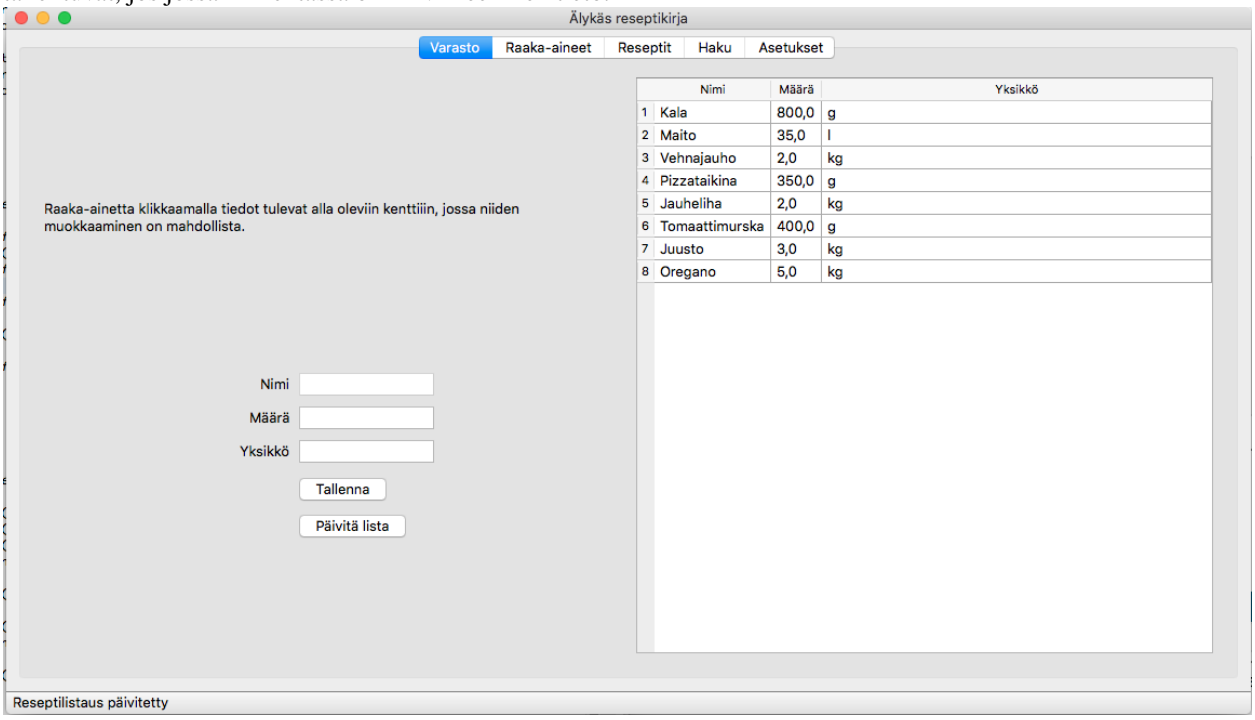
Ohjelman vaatimukset:

- Python 3.x
- PyQt5

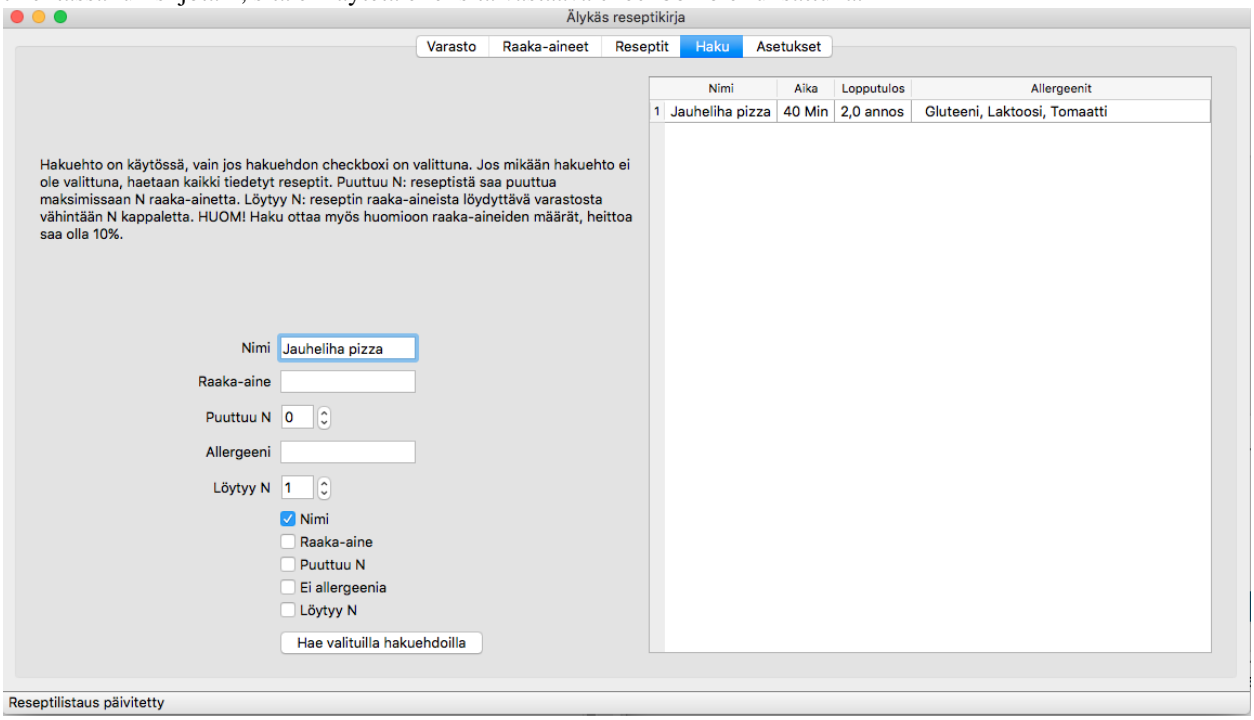
Ohjelma käynnistetään ajamalla mainGUI.py tiedosto python3:lla. Ohjelman käynnistyessä ladataan reseptit, raaka-aineet ja varastolistaus tiedostoista ’reseptit.txt’, ’raaka_aineet.txt’ ja ’varasto.csv. Näiden tiedostojen struktuuri on ihmiselle helposti luettavissa sekä muokattavissa.

Ohjelman käynnistyttyä aukeaa ikkuna, jossa on viisi välilehteä. Varasto, raaka-aineet ja reseptit -välilehdillä on listattuna kaikki ohjelmaan juuri ladatut kyseiset raaka-aineet/resepetit/varastotilanne. Näillä välilehdillä on myöskin mahdollista muokata näiden tietoja. Taulukon riviä klikkaamalla tulevat kyseisen rivin tiedot automaattisesti tekstikenttiin, josta niitä voi muokata. Jokaisella välilehdellä on lyhyt käyttöohje.

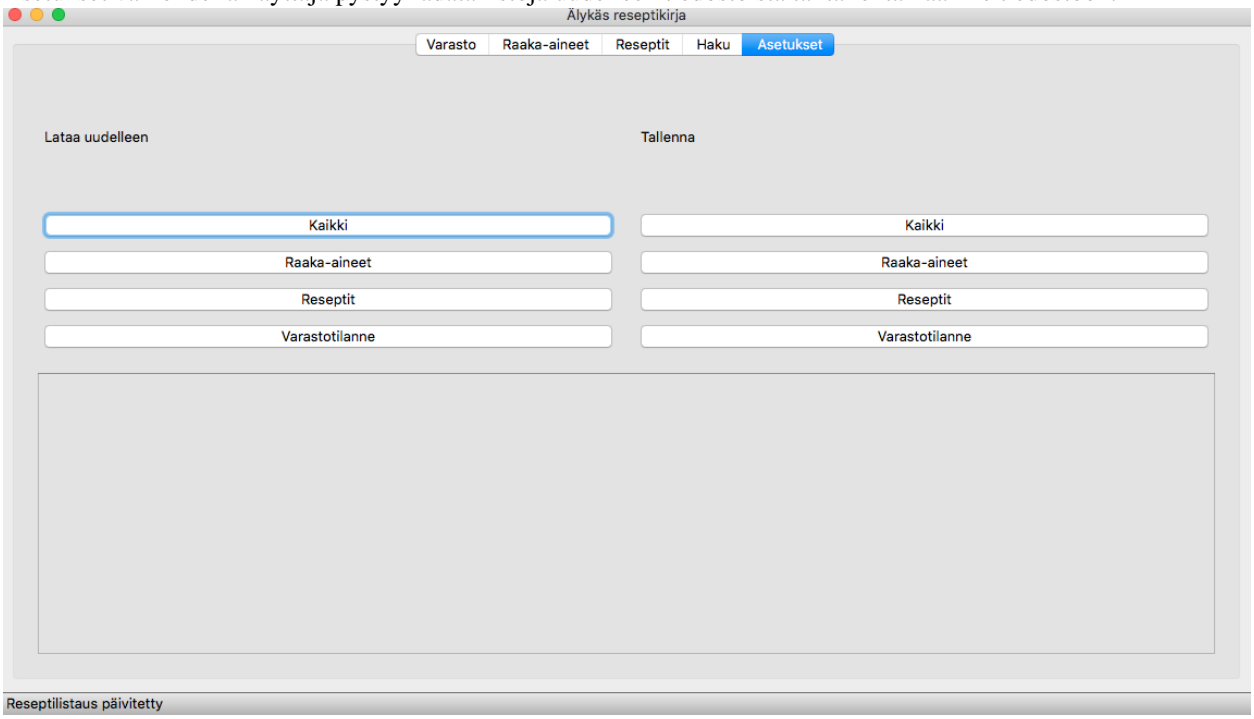
Tekstikenttiin pystyy syöttämään vain tiettyjä arvoja, kuten määrä kenttään vain desimaalilukuja. Lähes kaikissa kentissä on kuitenkin myös minimipituus vaatimus, joka käy ilmi vasta tallennettaessa. Huomioitavaa on, että muut tiedot tallentuvat, jos jossakin kentässä onkin virheellinen tieto.



Haku välilehdellä pystyy hakemaan reseptejä eri vaihtoehdoilla tai täysin ilman vaihtoehtoja. Vaikka hakuehdon tekstikentässä lukisi jotain, sitä ei käytetä ellei sitä vastaava checkbox ole ruksattuna.



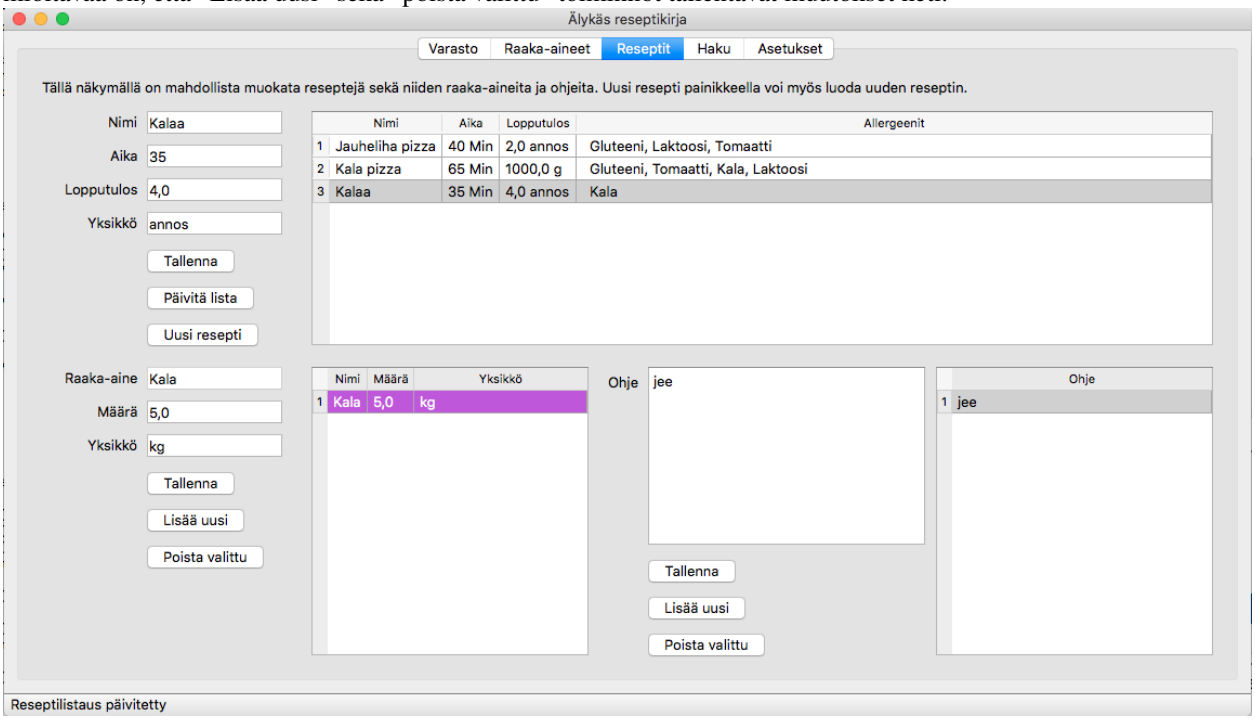
Asetukset välilehdellä käyttäjä pystyy ladata listoja uudelleen tiedostoista tai tallentamaan ne tiedostoon.



Reseptit näkymä on kaikista ”monimutkaisin” ja siellä on eniten toiminnallisuuksia. Näkymällä voidaan muokata sekä reseptin perustietoja, raaka-aineita että ohjeita. Raaka-aineita ja ohjeita pystyy lisäämään reseptille sekä poistamaan reseptiltä. Näiden lisäksi näkymällä on mahdollista luoda täysin uusi resepti painamalla ”uusi resepti” -painiketta. Tästä painikkeesta avautuu uusi ikkuna, johon annetaan reseptin tiedot. Reseptin ensimmäinen raaka-aine ja ohje annetaan dialogissa, loput lisätään reseptit näkymällä. Cancel painike peruuttaa luonnin, OK painike tallentaa reseptin,

sikäli kun tiedot ovat oikeelliset.

Raaka-aineet ja ohjeet valitaan muokattavaksi myös klikkaamalla taulukosta riviä, kuten muillakin näkymillä. Huomioitavaa on, että ”Lisää uusi” sekä ”poista valittu” toiminnot tallentavat muutokset heti.



Ohjelmasta on myös komentorivi versio, jossa on vain osa toiminnallisuuksista sekä sitä on testattu vähemmän. Sen saa toimimaan ajamalla main.py python3:lla.

LUKU 3

Ohjelman rakenne

3.1 Yleiskatsaus ja UML-kaavio

Ohjelman rakenne on nähtävissä alla olevasta UML-kaaviosta. Kaaviossa ei ole luokkien kaikkia metodeja tai attribuutteja, vaan muutama, jotta käy paremmin ilmi mitä niiltä odotetaan.

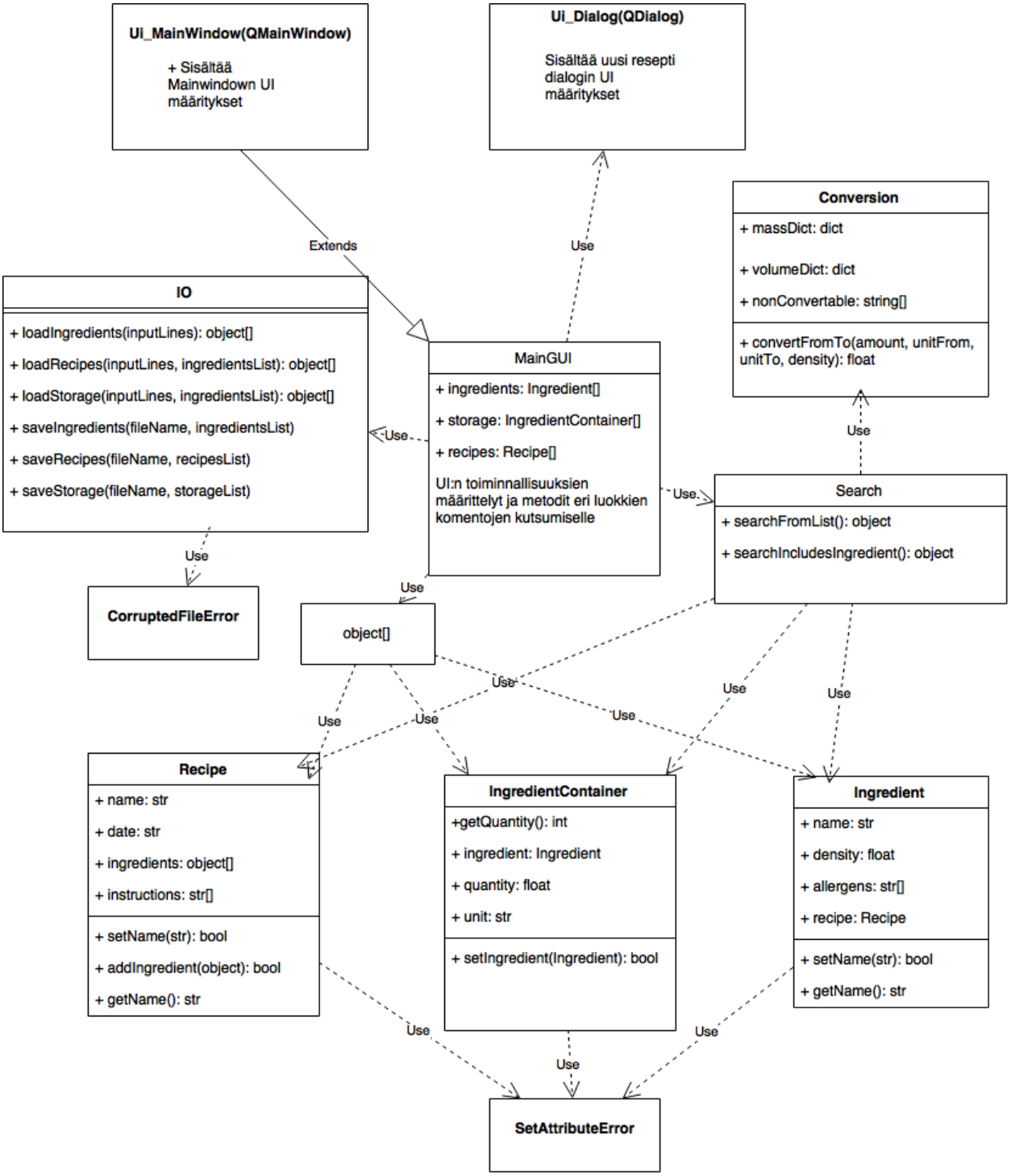
Käyttöliittymä on tehty PyQt kirjastoa ja QT designeria hyödyntäen. Käyttöliittymän ulkoasu on Qt designerin generoimaa koodia, jossa painikkeet ja taulut laitetaan oikeille paikoille. Pääikkunan design on luokka nimelta Ui_MainWindow, jonka ohjelman pääluokka MainGUI perii.

MainGUI luokassa määritellään mitä metodia kutsutaan, kun käyttöliittymässä esim. painetaan painiketta. Myös nämä kutsuttavat metodit ovat MainGUI luokassa ja sisältävät pääosin taulujen ja kenttien populoimista sekä niistä tietojen lukemista.

MainGUI luokassa myös säilytetään listaa kaikista raaka-aine, resepti ja varastotuote (ingredientContainer) olioista. Oikeastaan kaikki käyttöliittymään liittymättömät toiminnallisuudet toteutetaan hyödyntäen eri luokkien kuten Search ym. metodeja.

Toteutettu rakenne on hyvin toimiva, mutta muutamien metodien suhteen herää kysymys, että mihin luokkaan niiden tulisi kuulua. Esimerkiksi joitain aika yleisiä apumetodeja olisi voinut laittaa uuteen omaan luokkaan.

Yleiskuvaus luokista ja niiden tärkeimmistä metodeista on generoitu automaattisesti koodista alla olevan kuvan alapuolelle.



3.2 MainGUI

MainGUI luokasta on vaikea valita mitään erityisen tärkeitä metodeja, mutta kuvauksessa on selitetty mitä minkäkin niminen metodi yleensä tekee sekä jokaisesta on yksi esimerkki. Jokainen metodi on myös kommentoitu, joten halutessaan tämän dokumentin lopusta voi katsoa mitä metodeja luokka sisältää ja mitä ne tekevät. Created on 27.4.2016

@author: Kimi Päivärinta

class mainGUI.**MainGUI**

Tämä luokka perii pyqt:n QMainWindow luokan sekä QT Designerilla luodun Ui_MainWindow luokan, joka on moduulissa GUIDesign. Ui_MainWindow luokka sisältää graafisen käyttöliittymän designin.

Luokka sisältää paljon metodeja, joilla tehdään muutoksia, kun käyttöliittymällä tapahtuu muutoksia. Metodit voidaan jakaa karkeasti osiin

Metodit:

init* Luokan luomisen yhteydessä asetetaan nappuloiden toiminnallisuudet, ladataan tiedostoja ym.

populate* Piirretään data johonkin tauluun

save* Tallennetaan muuttunutta dataa

add* Lisätään uusi ohje/raaka-aine/ym.

get*InDataListForTable Metodit palauttavat niille annetun listan olennaisimmat tiedot "data" tyyppinä, joka voidaan antaa populateTableWith() metodille populoitavaksi

addNewRecipeIngredient()

Tämä metodi tarkastaa ja tallentaa reseptinäkymällä raaka-aine tekstikentissä olevat tekstit uudeksi raaka-aineeksi reseptille. Oikean reseptin löytymiseksi hyödynnetään self.recipeToEdit muuttujaa.

Tallennuksen jälkeen metodi populoi uudelleen reseptin raaka-ainelistaus taulun.

deleteRecipeIngredient()

Tämä metodi poistaa valitun reseptin valitun raaka-aineen.

Metodia kutsutaan painamalla poista painiketta. Oikea resepti ja raaka-aine selviää muuttujista self.recipeToEdit ja self.recipeIngredientToEdit

getRecipesInDataListForTable(*recipeList*)

Muodostaa annetujen reseptien tiedoista listojen listan, jotka on helppo populoida QTabletWidget tauluun.

initButtons()

Tässä metodissa määritellään käyttöliittymän painikkeiden toiminallisuudet

loadFromFileToList()

Tämä metodi päättelee kutsujan (self.sender()) perusteella mikä tiedosto tulee ladata uudelleen ja myöskin lataa sen.

Jos self.sender() on None, niin ladataan kaikki tiedostot. Muulloin kutsuja on esim. self.buttonLoadRecipes, jolloin ladataan reseptit tiedostosta.

populateIngredientsEditFields(*mi*)

Tämä metodi populoi raaka-ainenäkymällä olevat tekstikentät, joilla voi muokata raaka-aineen tietoja. Tätä metodia kutsutaan, kun raaka-ainelistaus taulua klikataan.

Rivin indeksi tallennetaan self.ingredientToEdit muuttujaan, jotta muutoksia tallennettaessa tiedetään mitä raaka-ainetta täytyy muokata.

Args:

mi mi muuttuja/olio, joka sisältää klikatun rivin ja kolumnin indeksin.

populateIngredientsTable()

Tämä metodi populoi raaka-ainelistaus tauluun kaikki tiedetyt raaka-aineet

saveIngredientsEdit()

Tämä metodi tarkistaa raaka-aineen perustietojen tekstikenttien arvot, jonka jälkeen muutokset tallennetaan raaka-aineeseen.

Oikea raaka-aine löydetään self.ingredientToEdit muuttujan avulla.

3.3 Recipe

Created on 19.4.2016

@author: Kimi Päivärinta

class `recipe.Recipe`

Luokka reseptejä varten. Tämä luokka pitää sisällään kaikki reseptille ominaiset attribuutit sekä tarvittavat metodit niiden arvojen muuttamiseksi.

Attributes:

self.date Luontipäivä

self.name Reseptin nimi

self.time Reseptin tekemiseen menevä aika minuuttina (int)

self.instructions Ohjeet (str[])

self.outcomeSize Reseptin lopputuloksen koko, esim. 4 (kg)

self.outcomeUnit Reseptin lopputuloksen yksikkö, esim (4) kg

self.ingredients Raaka-aineet (object[])

Returns: Attribuuttien muuttamiseen käytettävät metodit (set* & add* & delete/remove*) palauttavat True, jos muutos onnistuu

Raises: Kaikki attribuuttien asettamiseen käytettävät metodit (set* & add*) heittävät SetAttributeErrorin, jos attribuutin arvon asetus epäonnistuu.

`addIngredientContainer`(*ingredientContainer*)

Validoi, että lisättävä raaka-aine on IngredientContainer-olio sekä lisää raaka-aineen(Container) self.ingredients[] listaan

`deleteIngredient`(*index*)

Poistaa reseptiltä raaka-aineen. Argumenttina annetaan ohjeen sijainti listassa (index)

`getAllergensDistinctGUI`()

Palauttaa reseptien raaka-aineiden stringinä pilkulla erotettuna. Allergeeni esiintyy listassa vain kerran, vaikka se olisi monessa raaka-aineessa.

`getIngredients`()

Palauttaa reseptin raaka-aine oliot listana

`getName`()

Palauttaa reseptin nimen

`setName`(*name*)

Validoi, että nimi on yli 2 merkkiä pitkä ja asettaa sen: self.name

`setTime`(*time*)

Asettaa reseptin tekemiseen menevän ajan (min) inttinä: self.time

3.4 Ingredient

Created on 19.4.2016

@author: Kimi Päivärinta

class ingredient.**Ingredient**

Raaka-aine luokka. Tämä luokka sisältää perustiedot raaka-aineesta, varastossa ja resepteissä olevat "raaka-aineet" sisältävät tämän olion.

Attributes:

self.date Luontipäivä

self.name Nimi

self.density Tiheys yksikkömuunnoksia varten (float)

self.allergens Allergeenit (str[])

self.recipe Mahdollinen resepti (object)

self.recipeLoaded Kertoo onko raaka-aineen resepti ladattu. None = Ei reseptiä, False = Resepti on, mutta oliota ei ladattu, True = Olio ladattu

Returns: Attribuuttien muuttamiseen käytettävät metodit (set* & add* & delete/remove*) palauttavat True, jos muutos onnistuu

Raises: Attribuuttien asettamiseen käytettävät metodit (set* & add*) heittävät SetAttributeErrori:n, jos validointi epäonnistuu

getAllergens()

Palauttaa allergeenit listana

getDensityGUI()

Palauttaa tiheyden stringinä desimaalipilkulla

loadRecipe(*recipesList*)

Etsii nimen perusteella reseptilistasta raaka-aineelle halutun reseptin ja asettaa olion: self.recipe

Args:

recipesList Lista kaikista resepteistä

Returns:

Onnistuessa True

Ei ladattavaa None

Raises:

SetAttributeError Reseptiä ei löytynyt

removeAllergens()

Poistaa kaikki raaka-aineen allergeenit

removeRecipe()

Asettaa self.recipe = None sekä self.recipeLoaded = None

setName(*name*)

Validoi, että nimi on yli 2 merkkiä pitkä ja asettaa sen self.date

setRecipe(*recipe*)

Raaka-aineet luetaan sisälle ennen reseptejä, joten reseptin oliota ei todennäköisesti ole vielä olemassa. Validoi, että resepti on yli kaksi merkkiä pitkä sekä asettaa halutun reseptin nimen stringinä: self.recipe sekä asettaa self.recipeLoaded = False

3.5 IngredientContainer

Created on 19.4.2016

@author: Kimi Päivärinta

class ingredient.**IngredientContainer**

Tämä luokka sisältää viittauksen raaka-aine olioon ja tämän lisäksi omat attribuutit määrästä sekä yksiköstä. Tätä luokkaa hyödynnetään varastolistauksen sekä reseptien raaka-aineiden tallentamisessa. Samaa raaka-ainetta käytetään hyvin todennäköisesti useassa eri reseptissä, jos Ingredient luokassa olisi määrä, niin silloin tietoa joutuisi kopioimaan sekä säilyttämään moneen kertaan. Tämän luokan avulla esim. Jauhelihan tiedot ovat vain kertaalleen Ingredient oliossa ja resepteissä olevat IngredientContainer oliot sisältävät viittauksen tähän.

Attributes:

self.ingredient Raaka-aine olio

self.quantity Raaka-aineen määrä

self.unit Määrän yksikkö

Returns: :Attribuuttien muuttamiseen käytettävät metodit (set* & add* & delete/remove*) palauttavat True, jos muutos onnistuu

Raises: :Attribuuttien asettamiseen käytettävät metodit (set* & add*) heittävät SetAttributeErrori:n, jos validointi epäonnistuu

getAllergensStr()

Palauttaa allergeenit stringinä pilkulla erotettuna sekä alkussa teksti “Allergeenit: “

getQuantityStr()

Palauttaa määrän stringinä, desimaalipilkulla

hasRecipe()

Palauttaa True, jos raaka-aineella on tallennettu resepti, muuten false

setIngredient(*ingredient*, *ingredientsList*)

Etsii halutun raaka-aineen annetusta raaka-ainelistasta nimen perusteella sekä asettaa sen: self.ingredient

Attributes:

ingredient Etsittävä raaka-aine (string)

ingredientsList Lista kaikista raaka-aineista

Returns:

Onnistuessa True

Raises: :SetAttributeError epäonnistuessa

3.6 IO

Created on 19.4.2016

@author: Kimi Päivärinta

class IO.**IO**

Input output luokka, jolla ladataan raaka-aineet, reseptit ja varastolistaus tiedostoista. Tämän lisäksi nämä voidaan tallentaa tiedostoihin annetusta listasta.

loadIngredients(*inputLines*)
Tämä metodi lukee raaka-aineet tiedostosta ja palauttaa ne listana.

Args:

inputLines tiedoston kahva (fileIO), josta tiedot luetaan.

Returns:

ingredientList Raaka-aine oliot listana

successCount Montako luettiin onnistuneesti sisään

errorCount Montako jäi virheeseen luettaessa

Raises:

CorruptedFileError Jos kohdataan tuntematon tiedostotyyppi

loadRecipes(*inputLines*, *ingredientsList*)
Tämä metodi lukee reseptit tiedostosta ja palauttaa ne listana.

Args:

inputLines tiedoston kahva (fileIO), josta tiedot luetaan.

ingredientList Tunnetut raaka-aineet listana

Returns:

recipesList Resepti oliot listana

successCount Montako luettiin onnistuneesti sisään

errorCount Montako jäi virheeseen luettaessa

Raises:

CorruptedFileError Jos kohdataan tuntematon tiedostotyyppi

loadRecipesForIngredients(*ingredientsList*, *recipesList*)
Tällä metodilla voidaan ladata kaikkien raaka-aineiden reseptit. Käytännössä tämä metodi kutsuu raaka-aineen loadrecipe() metodia

Args:

ingredientList Raaka-aine oliot listana

recipesList Reseptio oliot listana

loadStorage(*inputLines*, *ingredientsList*)
Tämä metodi lukee varastolistauksen tiedostosta ja palauttaa ne listana.

Args:

inputLines tiedoston kahva (fileIO), josta tiedot luetaan.

ingredientList Tunnetut raaka-aineet listana

Returns:

storageList Raaka-aine oliot listana

successCount Montako luettiin onnistuneesti sisään

errorCount Montako jäi virheeseen luettaessa

Raises:

CorruptedFileError Jos kohdataan tuntematon tiedostotyyppi

saveIngredients (*fileName*, *ingredientsList*)
Tämä metodi tallentaa raaka-aineet tiedostoon ohjelman luettavassa muodossa. Tiedot tallennetaan ensin temp.ing tiedostoon, jonka jälkeen kyseinen tiedosto nimetään uudelleen halutun nimiseksi.

Args:

fileName Tiedostonimi, johon raaka-aineet tallennetaan

ingredientList Raaka-aine oliot listana

saveRecipes (*fileName*, *recipesList*)
Tämä metodi tallentaa reseptit tiedostoon ohjelman luettavassa muodossa. Tiedot tallennetaan ensin temp.rec tiedostoon, jonka jälkeen kyseinen tiedosto nimetään uudelleen halutun nimiseksi.

Args:

fileName Tiedostonimi, johon reseptit tallennetaan

recipesList Resepti oliot listana

saveStorage (*fileName*, *storageList*)
Tämä metodi tallentaa varastossa olevat raaka-aineet tiedostoon ohjelman luettavassa muodossa. Tiedot tallennetaan ensin temp.sto tiedostoon, jonka jälkeen kyseinen tiedosto nimetään uudelleen halutun nimiseksi.

Args:

fileName Tiedostonimi, johon varaston raaka-aineet tallennetaan

storageList Varaston raaka-aine oliot listana

3.7 Search

Created on 19.4.2016

@author: Kimi Päivärinta

class search.**Search**
Tämän luokan metodeilla on mahdollista tehdä erilaisia hakuja. Ohjelma hyödyntää tätä luokkaa reseptien etsimisessä tietyillä hakuehdoilla

searcForhRecipesNIngredientsInStorage (*recipesList*, *N*, *storageList*, *NNotInStorage*)
Tällä metodilla voidaan etsiä reseptejä, joihin löytyy vähintään N raaka-ainetta varastosta tai puuttuu maksimissaan N raaka-ainetta.

Args:

recipesList Lista kaikista reseptiolioista

N Monta raaka-ainetta löydyttävä/puututtava varastosta (int)

storageList Lista varastossa olevista raaka-aineolioista

NNotInStorage (Boolean) True arvolla N merkitsee sitä montako saa puuttua. False arvolla N merkitsee montako täytyy löytyä

Returns:

recipesFound Lista reseptio-olioista

searchFromList (*searchFor*, *searchList*)
Tällä metodilla etsitään olio listasta tiettyä oliota sen nimen perusteella. Vertailu tapahtuu stringeinä ja vaatii, että listassa olevilla olioilla on getName() metodi.

Args:

searchFor Etsittävä nimi (string)

searchList Lista olioista, joilla on getName() metodi

returns:

recipesFound Lista löydetyistä olioista. Tyhjä, jos ei löytynyt

searchIncludesIngredient(*ingredientStr*, *recipesList*)

Tällä metodilla etsitään reseptejä, jotka sisältävät tietyn raaka-aineen. Vertailutapahtuu stringeinä

Args:

ingredientStr Raaka-aineen nimi (string)

recipesList Lista reseptiolioista

Returns:

recipesFound Lista reseptiolioista. Tyhjä, jos ei löytynyt

searchNoAllergen(*allergenStr*, *recipesList*)

Tämmä metodilla etsitään reseptejä, jotka eivät sisällä tiettyä allergeeniä. Allergeenien vertailu tapahtuu stringeinä

Args:

allergenStr Allergeeni (string)

recipesList Lista reseptiolioista

Returns:

recipesFound Lista reseptiolioista. Tyhjä, jos ei löytynyt

3.8 Conversion

Created on 19.4.2016

@author: Kimi Päivärinta

class conversion.**Conversion**

Conversion luokka sisältää kaikki ohjelman tuntemat yksiköt sekä niiden "suhdeluvut", jotka mahdollistaa muunnokset. Tämän luokan avulla onnistuu yksikkömuunnokset.

convertFromTo(*amount*, *unitFrom*, *unitTo*, *density*)

Tekee yksikkömuunnoksen ja palauttaa muunnetun arvon. Jos ei muunnettavissa, palauttaa alkuperäisen arvon.

Args:

amount Muunnettava määrä (float)

unitFrom Alkuperäinen yksikkö

unitTo Haluttu yksikkö

density aineen tiheys, tarvitaan vain kun muunnetaan massa<->tilavuus (float)

Returns:

Muunnetun yksikön (float)

convertMassToMass(*amount*, *unitFrom*, *unitTo*)
Muuttaa massa yksikön massa yksiköksi, esim. g->kg

convertMassToVolume(*amount*, *unitFrom*, *unitTo*, *density*)
Muuttaa massa yksikön tilavuus yksiköksi, esim. kg -> l. Hyödyntää kaavaa v = m / r

convertVolumeToMass(*amount*, *unitFrom*, *unitTo*, *density*)
Muuttaa tilavuus yksikön massa yksiköksi, esim. l -> kg. Hyödyntää kaavaa m = v * r

convertVolumeToVolume(*amount*, *unitFrom*, *unitTo*)
Muuttaa tilavuus yksikön tilavuus yksiköksi, esim. l -> m3

isValidUnit(*unit*)
Tällä metodilla voidaan tarkistaa onko sille parametrina annettu yksikkö ohjelman tuntema

Returns:

Tunnettu True

Ei tunnettu False

3.9 CustomErrors

Created on 19.4.2016

@author: Kimi Päivärinta

exception customErrors.**CorruptedFileError**
Korruptoituneen tiedoston exception

exception customErrors.**SetAttributeError**
Käytetään, kun olion attribuutin asettamisessa tapahtuu virhe

LUKU 4

Algoritmit

4.1 Yksikkömuunnokset

Älykäs reseptikirja vaatii algoritmeja yksikkömuunnosten tekemiselle. Yksinkertaisessa tilanteessa on kyse kertoimen muuntamisesta eli massa <-> massa (g<->kg) tai tilavuus <-> tilavuus.

Haastavimmissa tilanteissa pitää myös pystyä tehdä muutoksia massa <-> tilavuus. Näissä muunnoksissa ohjelma hyödyntää kaavaa m = v * r, jossa m = massa, v = tilavuus sekä r = aineen tiheys. Jokaisesta aineesta täytyy siis olla tallennettuna tiheys, jotta muunnokset ovat mahdollisia. Tämän lisäksi ohjelmaan on tallennettu lista muista tunnetuista yksiköista ja niiden suhdeluvut.

Erikoistilanteen luo yksiköt, jotka eivät ole muunnettavissa kuten 'kpl' ja 'tlk'.

Ohjelmassa on omat metodit jokaiselle eri muunnostyypille. Nämä metodit ovat absrahoitu yhden metodin taakse, jolla pystyy tehdä kaikkia muunnoksia. Alla metodeista automaattisesti generoitu dokumentaatio. Created on 19.4.2016

@author: Kimi Päivärinta

class conversion.**Conversion**

Conversion luokka sisältää kaikki ohjelman tuntemat yksiköt sekä niiden "suhdeluvut", jotka mahdollistaa muunnokset. Tämän luokan avulla onnistuu yksikkömuunnokset.

convertFromTo(*amount*, *unitFrom*, *unitTo*, *density*)

Tekee yksikkömuunnoksen ja palauttaa muunnetun arvon. Jos ei muunnettavissa, palauttaa alkuperäisen arvon.

Args:

amount Muunnettava määrä (float)

unitFrom Alkuperäinen yksikkö

unitTo Haluttu yksikkö

density aineen tiheys, tarvitaan vain kun muunnetaan massa<->tilavuus (float)

Returns:

Muunnetun yksikön (float)

convertMassToMass(*amount*, *unitFrom*, *unitTo*)

Muuttaa massa yksikön massa yksiköksi, esim. g->kg

convertMassToVolume(*amount*, *unitFrom*, *unitTo*, *density*)

Muuttaa massa yksikön tilavuus yksiköksi, esim. kg -> l. Hyödyntää kaavaa v = m / r

convertVolumeToMass(*amount*, *unitFrom*, *unitTo*, *density*)
Muuttaa tilavuus yksikön massa yksiköksi, esim. l -> kg. Hyödyntää kaavaa m = v * r

convertVolumeToVolume(*amount*, *unitFrom*, *unitTo*)
Muuttaa tilavuus yksikön tilavuus yksiköksi, esim. l -> m3

4.2 Haku

Reseptejä voi hakea eri hakuehdoilla. Ohjelmaa tehdessä on oletettu, ettei raaka-aineita, reseptejä tai varastotuotteita tule kovinkaan paljon, joten tehokkaita hakualgoritmejä ei ole tarve tehdä. Kaikki edellä mainitut asiat ovat ohjelmassa tallennettuna listoissa tallennusjärjestyksessä, joten esimerkiksi binäärihaun suoraan hyödyntäminen ei onnistu. Ohjelman haut yksinkertaisesti käyvät listaa läpi alusta loppuun mitään oletuksia tekemättä.

Ohjelman muuttaminen käyttämään ns. binäärihakua ei pitäisi olla hirveän vaikeaa. Tällöin lista pitäisi kuitenkin pitää jollain tapaa aina järjestyksessä, joka tuottaa kuormaa aina kun raaka-aine lisätään tai poistetaan.

Oikeastaan kaikki haut perustuvat nimien (string) vertailuun ja lista, josta etsitään käydään kokonaan tai osittain läpi. Osa algoritmeista laskee vastaavuuksien määrän ja toiset palauttavat vastaavuuden. Yksi haku algoritmeista myös kutsuu rekursiivisesti itseään, koska ”hakulista” saattaa sisältää toisen ”hakulistan. Created on 19.4.2016

@author: Kimi Päivärinta

class search.**Search**
Tämän luokan metodeilla on mahdollista tehdä erilaisia hakuja. Ohjelma hyödyntää tätä luokkaa reseptien etsimisessä tietyillä hakuehdoilla

searcForhRecipesNIngredientsInStorage(*recipesList*, *N*, *storageList*, *NNotInStorage*)
Tällä metodilla voidaan etsiä reseptejä, joihin löytyy vähintään N raaka-ainetta varastosta tai puuttuu maksimissaan N raaka-ainetta.

Args:

recipesList Lista kaikista reseptiolioista

N Monta raaka-ainetta löydyttävä/puututtava varastosta (int)

storageList Lista varastossa olevista raaka-aineolioista

NNotInStorage (Boolean) True arvolla N merkitsee sitä montako saa puuttua. False arvolla N merkitsee montako täytyy löytyä

Returns:

recipesFound Lista reseptio-olioista

searchFromList(*searchFor*, *searchList*)
Tällä metodilla etsitään olio listasta tiettyä oliota sen nimen perusteella. Vertailu tapahtuu stringeinä ja vaatii, että listassa olevilla olioilla on getName() metodi.

Args:

searchFor Etsittävä nimi (string)

searchList Lista olioista, joilla on getName() metodi

returns:

recipesFound Lista löydetyistä olioista. Tyhjä, jos ei löytynyt

searchIncludesIngredient(*ingredientStr*, *recipesList*)
Tällä metodilla etsitään reseptejä, jotka sisältävät tietyn raaka-aineen. Vertailutapahtuu stringeinä

Args:

ingredientStr Raaka-aineen nimi (string)

recipesList Lista reseptiolioista

Returns:

recipesFound Lista reseptiolioista. Tyhjä, jos ei löytynyt

searchNoAllergen(*allergenStr*, *recipesList*)

Tämmä metodilla etsitään reseptejä, jotka eivät sisällä tiettyä allergeeniä. Allergeenien vertailu tapahtuu stringeinä

Args:

allergenStr Allergeeni (string)

recipesList Lista reseptiolioista

Returns:

recipesFound Lista reseptiolioista. Tyhjä, jos ei löytynyt

LUKU 5

Tietorakenteet

Raaka-aineista tehdään jokaisesta oma olio, jossa on raaka-aineen perustiedot kuten nimi, tiheys, allergeenit ym. Useissa resepteissä esiintyy samat raaka-aineet, mutta ei ole kovin mielekästä monistaa samaa pohja dataa useisiin olioihin. Tämän vuoksi ohjelmassa on ns. ”ingredientContainer” olio, jossa on viittaus perus raaka-aineolioon sekä erikseen määritelty määrä sekä määrän yksikkö. Näin Jauhelihan tietoja ei tarvitse tallentaa useaan olioon, ainoastaan viittaus Jauhelihan olioon.

Olioita (reseptit, raaka-aineet sekä ingredientContainer) säilytetään yksinkertaisesti listassa. Pythonin lista on muuttuvatilainen rakenteinen. Jos ohjelmaa kehittäisi eteenpäin, niin yksi ominaisuus olisi reseptin tai raaka-aineen poistaminen. Olioita pitää siis olla mahdollista poistaa myös listan keskeltä, joten muuttuvatilainen rakenne on parempi.

Massa- sekä tilavuusyksiköt ovat ohjelmassa tallennettuna sanakirjaan. Sanakirja on tähän sopiva valinta, koska yksiköistä ei riitä ainoastaan nimi, vaan täytyy tietää myös jonkinlainen suhdeluku muihin yksikköihin verrattuna ja tämän suhdeluvun pitää löytyä nimenomaan yksikön nimellä.

Jo ohjelmaa suunnitellessa tiesin, ettei ”tietokannan” luominen listoilla ole kovin järkevää, mutta kurssin alueen puitteissa päädyin tähän ratkaisuun. Fiksumpi vaihtoehto olisi ollut esimerkiksi käyttää Pythonin SQLite3 kirjastoa. Näin data olisi ollut tallennettuna tiedostoihin. Tämä olisi myös tehnyt ohjelmasta paljon helpommin skaalattavan, koska datan hakeminen tietokannasta olisi jo perustunut SQL kyselyihin ja siirtyminen käyttämään oikeaa tietokantapalvelinta olisi ollut verrattain helppoa. Myöhemmin dokumentissa puhutaan myös hakualgoritmeista ja niiden tehokkuudesta. Oikean tietokannan käyttäminen helpottaisi myös paljon hakujen tekemistä ja niiden tehokkuudesta huolehtiminen siirtyisi tietokannan puolelle, joka on suunniteltu tehokkuus ja haut mielessä.

Created on 19.4.2016

@author: Kimi Päivärinta

class ingredient.**Ingredient**

Raaka-aine luokka. Tämä luokka sisältää perustiedot raaka-aineesta, varastossa ja resepteissä olevat “raaka-aineet” sisältävät tämän olion.

Attributes:

self.date Luontipäivä

self.name Nimi

self.density Tiheys yksikkömuunnoksia varten (float)

self.allergens Allergeenit (str[])

self.recipe Mahdollinen resepti (object)

self.recipeLoaded Kertoo onko raaka-aineen resepti ladattu. None = Ei reseptiä, False = Resepti on, mutta oliota ei ladattu, True = Olio ladattu

Returns: Attribuuttien muuttamiseen käytettävät metodit (set* & add* & delete/remove*) palauttavat True, jos muutos onnistuu

Raises: Attribuuttien asettamiseen käytettävät metodit (set* & add*) heittävät SetAttributeErrori:n, jos validointi epäonnistuu

addAllergen(*allergen*)
Validoi, että allergeeni on yli 2 merkkiä pitkä ja lisää sen raaka-aineen allergeeni listaan

getAllergens()
Palauttaa allergeenit listana

getAllergensGUI()
Palauttaa allergeenit stringinä pilkulla ja välilyönnillä erotettuna

getAllergensStr()
Palauttaa allergeenit stringinä pilkulla erotettuna sekä alkussa teksti “Allergeenit: “

getDate()
Palauttaa luontipäivän

getDensity()
Palauttaa tiheyden floattina

getDensityGUI()
Palauttaa tiheyden stringinä desimaalipilkulla

getName()
Palauttaa nimen

getRecipe()
Palauttaa resepti olion, jos se on asetettu ja ladattu. Muulloin False

getRecipeGUI()
Palauttaa reseptin nimen stringinä

getRecipeLoaded()
Palauttaa self.recipeLoaded arvon. None = Ei reseptiä, True = Resepti olio ladattu, False = Reseptiä ei vielä ladattu

getRecipeStr()
Palauttaa “Resepti: ” + reseptin nimi, jos resepti on asetettu ja ladattu

hasRecipe()
Palauttaa True, jos raaka-aineella on tallennettu resepti, muuten false

loadRecipe(*recipesList*)
Etsii nimen perusteella reseptilistasta raaka-aineelle halutun reseptin ja asettaa olion: self.recipe

Args:

recipesList Lista kaikista resepteistä

Returns:

Onnistuessa True

Ei ladattavaa None

Raises:

SetAttributeError Reseptiä ei löytynyt

removeAllergens()
Poistaa kaikki raaka-aineen allergeenit

removeRecipe()
Asettaa self.recipe = None sekä self.recipeLoaded = None

setDate(*date*)
Validoi päivämäärän ja asettaa sen: self.date

setDensity(*density*)
Muuttaa desimaalipilkun pisteeksi, muuntaa tiheyden float luvuksi ja asettaa sen: self.density

setName(*name*)
Validoi, että nimi on yli 2 merkkiä pitkä ja asettaa sen self.date

setRecipe(*recipe*)
Raaka-aineet luetaan sisälle ennen reseptejä, joten reseptin oliota ei todennäköisesti ole vielä olemassa. Validoi, että resepti on yli kaksi merkkiä pitkä sekä asettaa halutun reseptin nimen stringinä: self.recipe sekä asettaa self.recipeLoaded = False

class ingredient.**IngredientContainer**
Tämä luokka sisältää viittauksen raaka-aine olioon ja tämän lisäksi omat attribuutit määrästä sekä yksiköstä. Tätä luokkaa hyödynnetään varastolistauksen sekä reseptien raaka-aineiden tallentamisessa. Samaa raaka-ainetta käytetään hyvin todennäköisesti useassa eri reseptissä, jos Ingredient luokassa olisi määrä, niin silloin tietoa joutuisi kopioimaan sekä säilyttämään moneen kertaan. Tämän luokan avulla esim. Jauhelihan tiedot ovat vain kertaalleen Ingredient oliossa ja resepteissä olevat IngredientContainer oliot sisältävät viittauksen tähän.

Attributes:

self.ingredient Raaka-aine olio

self.quantity Raaka-aineen määrä

self.unit Määrän yksikkö

Returns: :Attribuuttien muuttamiseen käytettävät metodit (set* & add* & delete/remove*) palauttavat True, jos muutos onnistuu

Raises: :Attribuuttien asettamiseen käytettävät metodit (set* & add*) heittävät SetAttributeErrori:n, jos validointi epäonnistuu

getAllergens()
Palauttaa allergeenit listana

getAllergensStr()
Palauttaa allergeenit stringinä pilkulla erotettuna sekä alkussa teksti “Allergeenit: “

getDensity()
Palauttaa raaka-aineen tiheyden floattina

getIngredient()
Palauttaa raaka-aine olion

getName()
Palauttaa raaka-aineen nimen

getQuantity()
Palauttaa määrän floattina

getQuantityStr()
Palauttaa määrän stringinä, desimaalipilkulla

getRecipe()
Palauttaa reseptiolion, jos raaka-aineella on

getRecipeStr()
Palauttaa reseptin nimen stringinä, jos raaka-aineella on

getUnit()
Palauttaa määrän yksikön

hasRecipe()
Palauttaa True, jos raaka-aineella on tallennettu resepti, muuten false

setIngredient(*ingredient*, *ingredientsList*)
Etsii halutun raaka-aineen annetusta raaka-ainelistasta nimen perusteella sekä asettaa sen: self.ingredient

Attributes:

ingredient Etsittävä raaka-aine (string)

ingredientsList Lista kaikista raaka-aineista

Returns:

Onnistuessa True

Raises: :SetAttributeError epäonnistuessa

setQuantity(*quantity*)
Muuttaa desimaalipilkun pisteeksti ja asettaa määrän floattina: self.quantity

setUnit(*unit*)
Validoi, että yksikkö on ohjelman tuntema sekä asettaa sen: self.unit

Created on 19.4.2016

@author: Kimi Päivärinta

class recipe.**Recipe**
Luokka reseptejä varten. Tämä luokka pitää sisällään kaikki reseptille ominaiset attribuutit sekä tarvittavat metodit niiden arvojen muuttamiseksi.

Attributes:

self.date Luontipäivä

self.name Reseptin nimi

self.time Reseptin tekemiseen menevä aika minuuttina (int)

self.instructions Ohjeet (str[])

self.outcomeSize Reseptin lopputuloksen koko, esim. 4 (kg)

self.outcomeUnit Reseptin lopputuloksen yksikkö, esim (4) kg

self.ingredients Raaka-aineet (object[])

Returns: Attribuuttien muuttamiseen käytettävät metodit (set* & add* & delete/remove*) palauttavat True, jos muutos onnistuu

Raises: Kaikki attribuuttien asettamiseen käytettävät metodit (set* & add*) heittävät SetAttributeErrorin, jos attribuutin arvon asetus epäonnistuu.

addIngredientContainer(*ingredientContainer*)
Validoi, että lisättävä raaka-aine on IngredientContainer-olio sekä lisää raaka-aineen(Container) self.ingredients[] listaan

addInstruction(*instruction*)
Validoi, että ohje on yli 2 merkkiä pitkä ja lisää sen self.instruction[] listaan

deleteIngredient(*index*)
Poistaa reseptiltä raaka-aineen. Argumenttina annetaan ohjeen sijainti listassa (index)

deleteInstruction(*index*)
Poistaa reseptiltä ohjeen. Argumenttina annetaan ohjeen sijainti listassa (index)

getAllergensDistinctGUI()
Palauttaa reseptien raaka-aineiden stringinä pilkulla erotettuna. Allergeeni esiintyy listassa vain kerran, vaikka se olisi monessa raaka-aineessa.

getDate()
Palauttaa reseptin luontipäivän

getIngredients()
Palauttaa reseptin raaka-aine oliot listana

getIngredientsGUI()
Palauttaa raaka-aineiden nimet listana

getIngredientsStr()
Palauttaa reseptin raaka-aineet stringinä, hyödyntää raaka-aine luokan __str__() metodia

getInstructions()
Palauttaa ohjeet listana

getInstructionsStr()
Palauttaa ohjeet stringinä, jokainen ohje omalla rivillä ja edessä ohjeen järjestysnumero eli järjestys listassa

getName()
Palauttaa reseptin nimen

getOutcomeSize()
Palauttaa lopputuloksen floattina

getOutcomeSizeGUI()
Palauttaa lopputuloksen stringinä desimaalipilkulla

getOutcomeStr()
Palauttaa reseptin lopputuloksen desimaalipilkulla muodossa “<määrä> <yksikkö>”

getOutcomeUnit()
Palauttaa lopputuloksen yksikön

getTime()
Palauttaa reseptin tekemiseen menevän ajan inttinä

getTimeGUI()
Palauttaa reseptin tekemiseen menevän ajan stringinä

getTimeStr()
Palauttaa reseptin tekemiseen menevän ajan stringinä, jonka lopussa on ” Min”

setDate(*date*)
Validoi päivämäärän ja asettaa sen stringinä: self.date

setName(*name*)
Validoi, että nimi on yli 2 merkkiä pitkä ja asettaa sen: self.name

setOutcomeSize(*outcomeSize*)
Muuttaa desimaalipilkun pisteeksi ja settaa määrän floattina: self.outcomeSize

setOutcomeUnit(*outcomeUnit*)
Validoi, että yksikkö on ohjelman tuntema ja asettaa sen: self.outcomeUnit

setTime(*time*)
Asettaa reseptin tekemiseen menevän ajan (min) inttinä: self.time

LUKU 6

Tiedostot

6.1 Yleisesti

Ohjelma lukee ja tallentaa tekstitiedostoja. Raaka-aineilla, resepteillä ja varastolistauksella on kaikilla omat tiedostoformaattinsa, jotka ovat ihmisen helposti luettavissa sekä muokattavissa. Projektissa on mukana esimerkkitiedostot 'raaka_aineet.txt', 'reseptit.txt' sekä 'varasto.csv'.

6.2 Raaka-aineet

Esimerkki ja tietuekuvaus:

```
INGREDIENTLIST

#Ingredient
Date                    : 18.4.2015
Name                    : Kala
Density                 : 0.5
Allergen                : Kala
Recipe                   : Jauheliha pizza


#Ingredient
Date                    : 18.4.2015
Name                    : Maito
Density                 : 1.0
Allergen                : Laktoosi
Allergen                  : Maito
```

Ekan rivin alussa oltava INGREDIENTLIST, muuten tiedostoa ei lueta.

#Ingredient kertoo uuden raaka-aineen alkamisesta, ei maksimimäärää.

Tiedot kerrotaan yleensä [attribuutti] : [arvo] sekä joillain attribuuteilla on useampia arvoja, kuin 1, nekin ovat erotettu kaksoispisteellä.

Pakolliset attribuutit

Date: Raaka-aineen luontipäivä, oltava muodossa dd.mm.yyyy

Name: Raaka-aineen nimi, oltava yli kaksi merkkiä pitkä

Tiheys: Aineen tiheys (float)

Vapaaehtoiset attribuutit:

Allergen: Allergeeni tekstinä, voi olla useita

Recipe: Resepti

6.3 Reseptit

Esimerkki ja tietuekuvaus:

```
RECIPELIST

#Recipe
Date : 18.4.2016
Name : Jauheliha pizza
Time : 40
Outcome : 2.0 : annos
Ingredient : Jauheliha : 600.0 : g
Ingredient : Pizzataikina : 300.0 : g
Ingredient : Juusto : 0.1 : kg
Ingredient : Oregano : 5.0 : g
Ingredient : Tomaattimurska : 300.0 : g
Instruction : Ruskista jauheliha
Instruction : Kaada kaikki sekoitettuna uuniin
Instruction : Nauti kylman PepsiColan kanssa.
```

Ensimmäisen rivin alussa oltava RECIPELIST, muuten tiedostoa ei tunnisteta.

#Recipe kertoo uuden reseptin alkamisesta. Tiedostossa voi olla useita reseptejä.

Tiedot kerrotaan yleensä [attribuutti] : [arvo] sekä joillain attribuuteilla on useampia arvoja, kuin 1, nekin ovat erotettu kaksoispisteellä.

Pakolliset attribuutit:

Date: reseptin luontipäivä, oltava muodossa dd.mm.yyyy

Name: reseptin nimi, oltava yli 2 merkkiä pitkä

Time: Kauanko reseptin tekemiseen menee minuuteissa, kokonaisluku

Outcome: reseptin lopputuloksen määrä (float) ja yksikkö

Ingredient: raaka-aineen nimi, määrä (float) ja yksikkö. Raaka-aineiden maksimimäärää ei määritelty.

Instruction: Ohje, ohjeet tulee reseptille tässä järjestyksessä.

6.4 Varastotilanne

Csv-tiedosto, jossa soluerottimena puolipiste eikä tekstin ympärillä "" merkkejä. Esimerkki ja tietuekuvaus:

```
STORAGELIST
Kala;800.0;g
Maito;35.0;l
Vehnajauho;2.0;kg
Pizzataikina;350.0;g
Jauheliha;2.0;kg
Tomaattimurska;400.0;g
```

```
Juusto;3.0;kg
Oregano;5.0;kg
```

Ensimmäisellä rivilla oltava STORAGELIST tai tiedostoa ei lueta.

Rivin ensimmäisessä solussa raaka-aineen nimi

Rivin toisessa solussa määrä (float) desimaalipilkulla tai pisteellä

Rivin kolmannessa solussa määrän yksikkö

LUKU 7

Testaus

7.1 Yleisesti

Ohjelma nojaa hyvin paljon siihen, ettei virheellistä dataa ole mahdollista tallentaa olioihin. Dataa ei siis enää validoida, kun se haetaan oliosta tai muualta. Tämän vuoksi oli hyvin tärkeää testata kunnolla kaikki paljon käytetyt metodit, joilla asetetaan arvoja eri olioille. Myös yksikkömuunnoksien toimivuus on tärkeää, mutta myös testaaminen helppoa yksikkötesteillä.

Suunnitelmissa oli testata kaikki ohjelman luokat yksikkötesteillä, mutta se ei tämän kurssin ja ajan puitteissa ole järkevää.

7.2 Yksikkötestit

Tärkeimpien ja eniten käytettyjen metodien toimivuuden varmistamiseksi kirjoitin kattavat yksikötestit. Yksikkömuunnoksista testaan jokaista apumetoda useammalla eri arvoilla. Näiden lisäksi on myös testattu, että apumetodit abstrahoiva metodi toimii halutulla tavalla.

Raaka-aineen attribuuttien asetus metodit ovat myöskin testattu kattavasti. Yksikkötesteillä on varmistettu, ettei vääränlaisia arvoja ole mahdollista asettaa olioille sekä että metodit palauttavat fiksun virheen eivätkä kaada ohjelmaa. Tämän lisäksi on tarkastettu, että metodit hyväksyvät niitä arvoja joita kuuluukin. Samalla on myös varmistettu, että arvojen palautus metodit toimivat.

IngredientContainer sekä resepti oliot ja niiden metodit ovat myöskin testattu hyvin samalla tavalla kuin raaka-aine olion. Koko ohjelma oikeastaan perustuu näiden kolmen luokan eri metodeihin, joten niiden toimivuuden varmistaminen on tärkeää.

Tämän lisäksi yksikkötestein on tarkistettu jokaisen eri tiedostotyypin sisäänluku ja arvojen tallennus olioihin. Testattavia asioita on mm., että tiedoston epäonnistuessa heitetään järkevä virhe eikä ohjelma kaadu.

7.3 Manuaalinen testaaminen

Haun testaaminen yksikkötesteillä olisi järkevää, sillä se on myös yksi ohjelman tärkeistä ominaisuuksista. Sen testaaminen yksikkötesteillä on kuitenkin hyvin työlästä, joten päädyin tekemään sen manuaalisesti ohjelmaa käyttäen. Hakua on testattu eri hakusyötteillä sekä hakuvaihtoehdoilla. Hakutuloksien oikeellisuus on pystytty varmistamaan, kun tiedetään mitä dataa ohjelmassa on sisällä. Testidatan koko on verrattain pieni, joten päätteleminen on mahdollista.

Käyttöliittymää ei myöskään ole yksikkötestattu, koska yksikkötesteillä ns. BDD testien tekeminen ei ole kovinkaan helppoa. Käyttöliittymää on testattu käyttämällä sitä. Muutamia testattuja asioita mainitakseni

- Väärien syötteiden antaminen tekstikenttiin, esim. kopioimalla
- Nappuloiden klikkaaminen useita kertoja
- Taulukoiden klikkaaminen useita kertoja
- Painikkeiden, tekstien, taulukoiden ”drag-n-drop”
- Ikkunan venyttäminen

Ohjelman puutteet

- Raaka-ainetta tai reseptiä tallennettaessa ei tarkasteta onko samalla nimellä jo olemassa olio. Tämä joissain tilanteissa aiheuttaa hieman outoja hakutuloksia. Ratkaisu: Pitäisi tarkastaa onko samannimistä oliota jo olemassa.
- Raaka-aineen reseptin tarkasteleminen ei ole kovinkaan yksinkertaista käyttöliittymällä. Esim. raaka-ainetta kaksoisklikkaamalla voisi aueta raaka-aineen resepti
- Hakunäkymällä resepteistä ei saa kovinkaan paljoa irti, vaan pitää siirtyä reseptit näkymälle. Haku näkymältä pitäisi saada resepti kokonaisuudessaan helposti auki.
- Raaka-aineelta ei voi poistaa reseptiä
- Raaka-aineelta ei voi poistaa kaikkia allergeenejä
- Haku kyllä tarkistaa voiko raaka-aineen tehdä reseptistä, jos sitä ei ole tarpeeksi varastossa. Haku ei kuitenkaan skaalaa määriä eli jos tarvitaan 500g pizzataikinaa, mutta pizzataikinan ohjeesta saadaan 1kg, niin silloin tarkastetaan riittääkö ainekset 1kg tekemiseen eikä 500g. Tämän korjaamiseksi howManyIngredientsFOundInStorage metodissa pitäisi laskea suhdeluku raaka-aineen määrän ja raaka-aineen reseptin lopputuloksen välillä, jota voitaisiin hyödyntää vertailussa. Tätä toimintoa varten ohjelmassa on myös oletus, että annos = 250g, jotta reseptien lopputulosta voi todella verrata tarvittaviin määriin.

Heikoimmat ja parhaat kohdas

Parhaina kohtina pidän koodin virheenkäsittelyä, yksikkötestien kattavuutta sekä koodin helppolukuisuutta. Käyttöliittymää on myöskin testattu paljon, enkä ole millään toiminnolla tai syötteellä saanut ohjelmaa kaatumaan tai näyttämään vääriä tietoja.

Ohjelman heikoin kohta on IO luokka. Sisäänlukujen koodi ei ole kovin lukijaystävällistä sekä se on toisteista. Metodeita olisi helppo jakaa pienempiin osiin, joita voisi hyödyntää kaikissa tiedostonluku metodeissa. Myöskin osa sisäänluvun virheistä tulostetaan konsoliin eikä niitä näytetä ollenkaan käyttöliittymällä. Myöskään virhellisiä syötteitä tiedostoissa ei ole testattu tarpeeksi ja uskon, että nimenomaan siinä on ohjelman heikoin kohta.

LUKU 10

Poikkeamat suunnitelmasta

Poikkesin suunnitelmastani muutamassakin kohtaa. Päädyin lopulta tekemään graafisen käyttöliittymän komentorivi käyttöliittymän sijaan. Tämä oli mielestäni todella hyvä ratkaisu, koska nyt tajuaa, ettei GUI:n tekeminen ole mitenkään erityisen vaikeaa. Toisaalta huomasin myös, että uusien ominaisuuksien toteuttaminen oli jopa helpompaa graafisesti, esim. tietojen syöttäminen. En koskaan ollut tehnyt GUI:ta, joten tämän opettelu vei paljon enemmän aikaa, kun olin käyttöliittymän koodaamiseen suunnitellut käyttäväni.

Toinen pieni muutos liittyy luokkajakoihin. Alkuperäisestä suunnitelmasta poiketen jaoin Ingredient luokan kahteen osaan: Ingredient ja IngredientContainer. Näin vältytään tiedon turhalta kopioimiselta ja säilyttämiseltä useampaan kuin yhteen kertaan. Tämän muutoksen tekeminen ei kauaa vienyt.

Toteutusjärjestys oli myös aivan toisenlainen kuin alunperin suunnittelin sen olevan. Tähän päätökseen päädyin jo projektin demotilaisuudessa, jossa assari suositteli tekemään aluksi toimivan ohjelman vähillä toiminnoilla ja sen jälkeen kasvattaa ohjelmaa. Alkuperäinen suunnitelmani oli tehdä kaikki osiot erikseen valmiiksi ja lopuksi integroida ne. Uskon, että käytetyllä suunnitelmalla säästin aikaa.

Toteutunut työjärjestys ja aikataulu

Toteutunut työjärjestys poikkesi paljon suunnitelmasta. Näin tapahtui myös aikataulun kanssa, koska pääsin aloittamaan projektin vasta paljon suunniteltua myöhemmin. Alla committien perusteella tehty työjärjestys ja aikataulu. Aikataulussa ei näy testaamista, mutta sitä tein jokaisen toiminnallisuuden implementoinnin jälkeen.

Kokonaisuudessaan työjärjestys oli hyvin sekalainen. Työmäärää olisi myös vähentänyt kunnon suunnittelu sekä tavoitteen päättäminen, esim. tein kaksi käyttöliittymää. Myöskin osaa toiminnallisuuksista suunnittelin kyllä hyvin, mutta loppuenlopuksi en ollut vain tyytyväinen koodiin ja keksin selkeämmän ja paremman tavan toteuttaa sama asia.

18.4.2016: 2h

- Gitin kuntoon laittaminen
- Komentorivi käyttöliittymän teko

19.4.2016: 10h

- Yksikkötestien tekoa
- Käyttöliittymän viilaamista
- Koko komentorivi käyttöliittymän refaktorointi, jottei toisteisuutta
- Viime vuoden koodin korjailua ja sisäistämist'
- Tiedostojen lukemisen implementointi loppuun
- Virheenkäsittelyn teko tiedostojen lukemiseen
- Ingredient ja Recipe luokkien get ja set metodien implementointi
- Yksikkötestien teko tiedoston sisäänluvuille

20.4.2016: 2h

- tiedoston avaaminen UTF8:na, joka ei lopulta edes toiminut kunnolla
- tiedostojen lukeminen ohjelman avaamisen yhteydessä ja listojen säilytys

22.4.2016: 5h

- Listan tulokselle metodi (tulosta raaka-aineet tai reseptit ym.)
- __str__ metodit olioille ja tulostusten loppuun tekeminen

24.-25.4.2016: 8h

- Conversion luokan implementointi
- Haku luokan implementointi
- Haun ja Conversion kevyt testaus

- Haun ja conversion liittäminen käyttöliittymään
- Tietojen tallennus tiedostoon
- Tietojen tallennus testaus
- Implementoitu reseptin etsiminen ja tallennus Raaka-aine luokalle

26.4.2016: 2h

- Virheenkäsittelyn lisäys luokkien set metodeihin

27.4.2016: 6h

- GUI:n tekemisen opettelu ja implementointi
- Muutaman GUI:n toiminnallisuuden implementointi
- Luokkille uusia metodeja GUI:n tarpeisiin sopivaksi

28.4.2016: 5h

- Reseptinäkymän toimintojen sekä asetukset näkymän toimintojen implementointi

2.5.2016: 7h

- Haku näkymän toiminnallisuuksien implementointi
- Uusi resepti dialogin suunnittelu ja toiminnallisuuksien implementointi

3.5.2016: 10h

- MainGUI luokan metodien refaktorointi, toiston vähentämiseksi
- Kommenttien lisäys koodiin
- Yksikkötestien kattava kirjoittaminen
- Manuaali testaamista
- Ulkoasun hienosäätöä
- Automaattisen dokumentoinnin opettelu ja aloitus

4.5.2016: 5h

- Dokumentoinnin kirjoitus

5.5.2016: 1h

- Hienosäätöä

Yhteensä: n. 65h

LUKU 12

Arvio lopputuloksesta

Kokonaisuudessaan olen tyytyväinen projektin lopputulokseen. Paljon mieleen tulleita ominaisuuksia jäi toteuttamatta, mutta kuten todettu tämä oli myös tietoinen valinta. Halusin, että palautuksen koittaessa projekti on hyvin tehty sekä dokumentoitu eikä niin, että siinä on paljon toiminnallisuuksia, mutta ne ovat tehty huonosti.

Eniten parannettavaa on mielestäni omissa työskentelytavoissani. Liian useat ominaisuudet tuli tehtyä yhteen tai kahteen kertaan, koska en ollut tyytyväinen toteutukseeni tai keksin paremman tavan. Tulevaisuudessa pyrin myöskin tekemään yhden osion kerralla kuntoon ja sitten jatkaa seuraavaan. Jokaisen osion välissä pitäisi myöskin malttaa suunnitella toiminallisuuden impelementaatio kunnolla ja miettiä mm. toteutuksen skaalautuvuutta.

Ohjelman oleellisen puute lienee se, että raaka-aineiden reseptejä on hieman hankala tarkastella. Haku ottaa huomioon raaka-aineiden reseptit ja näin ollen palauttaa reseptejä, joihin ei suoraan ole raaka-aineita. Näkymä ei kuitenkaan indikoi mitenkään, että mikä raaka-aine puuttuu ja pitää tehdä reseptistä, vaan se jää käyttäjän vastuulle. Muut puutteet on kerrottu tämän dokumentin kohdassa 9. Ohjelman puutteet.

Ohjelman parantamiseksi refaktoroisin IO luokan lähes kokonaan ja parantaisin sen virheenkäsittelyä sekä virheen heittämistä. Muut parannukset koskisivat pääasiassa toiminnallisuuksia, kuten: * raaka-aineen lisäys ja poisto * varastotuotteen lisäys ja poisto * Tallennettavien tai ladattavien tiedostojen polkujen valitseminen käyttöliittymältä * Reseptin katsominen helpommaksi hakunäkymällä

Paremmin projektissa olisi mielestäni voinut tehdä tietorakenteiden valinnan. Kuten kohdassa 5. Tietorakenteet totesin, niin listoilla tietokannan luominen ei ole järin hyvä vaihtoehto. Parempi vaihtoehto olisi ollut esim SQLite3 kirjaston käyttö. Ohjelma on muilta osin mielestäni hyvin skaalautuva ja uusien toiminnallisuuksien lisääminen käyttäen olemassa olevia metodeja pitäisi olla verrattain yksinkertaista - osittain kattavan dokumentaation takia. Oikean tietokannan puuttuminen kuitenkin heikentää ohjelman skaalautuvuutta merkittävästi. Projekti on myös todella kattavasti dokumentoitu, joten myöhemmin projektin jatkaminen pitäisi olla verrattain helppoa.

Viitteet

http://pyqt.sourceforge.net/Docs/PyQt4/modules.html

http://stackoverflow.com

http://thomas-cokelaer.info/blog/2012/10/pyqt4-example-of-tablewidget-usage/

Automaattisesti generoitu dokumentaatio

14.1 MainGUI

Created on 27.4.2016

@author: Kimi Päivärinta

class mainGUI.**MainGUI**

Tämä luokka perii pyqt:n QMainWindow luokan sekä QT Designerilla luodun Ui_MainWindow luokan, joka on moduulissa GUIDesign. Ui_MainWindow luokka sisältää graafisen käyttöliittymän designin.

Luokka sisältää paljon metodeja, joilla tehdään muutoksia, kun käyttöliittymällä tapahtuu muutoksia. Metodit voidaan jakaa karkeasti osiin

Metodit:

init* Luokan luomisen yhteydessä asetetaan nappuloiden toiminnallisuudet, ladataan tiedostoja ym.

populate* Piirretään data johonkin tauluun

save* Tallennetaan muuttunutta dataa

add* Lisätään uusi ohje/raaka-aine/ym.

get*InDataListForTable Metodit palauttavat niille annetun listan olennaisimmat tiedot "data" tyyppinä, joka voidaan antaa populateTableWith() metodille populoitavaksi

addNewRecipeIngredient()

Tämä metodi tarkastaa ja tallentaa reseptinäkymällä raaka-aine tekstikentissä olevat tekstit uudeksi raaka-aineeksi reseptille. Oikean reseptin löytymiseksi hyödynnetään self.recipeToEdit muuttujaa.

Tallennuksen jälkeen metodi populoi uudelleen reseptin raaka-ainelistaus taulun.

addNewRecipeInstruction()

Tämä metodi tallentaa reseptinäkymällä ohjetekstikentässä olevan tekstin uudeksi ohjeeksi reseptille. Oikean reseptin löytymiseksi hyödynnetään self.recipeToEdit muuttujaa.

Tallennuksen jälkeen metodi populoi uudelleen reseptin ohjelistaus taulun.

checkStateChanged(*state*)

Tämä metodi pitää huolen, ettei hakunäkymällä olevat "Puuttuu N" ja "Löytyy N" checkboksit ole valittuna samaan aikaan, koska se ei ole järin järkevä hakuvaihtoehto.

Tätä metodia kutsutaan vaihtamalla jomman kumman checkboksin tilaa.

clearRecipeEditLineEdits()
Tämä metodi tyhjentää reseptin raaka-aine sekä ohjeen tekstikentät. Tätä metodia kutsutaan, kun reseptilistaus taulua klikataan, jotta kyseisissä tekstikentissä ei "vanhoja" tietoja.

deleteRecipeIngredient()
Tämä metodi poistaa valitun reseptin valitun raaka-aineen.

Metodia kutsutaan painamalla poista painiketta. Oikea resepti ja raaka-aine selviää muuttujista self.recipeToEdit ja self.recipeIngredientToEdit

deleteRecipeInstruction()
Tämä metodi poistaa valitun reseptin valitun ohjeen.

Metodia kutsutaan painamalla poista painiketta. Oikea resepti ja ohje selviää muuttujista self.recipeToEdit ja self.recipeInstructionToEdit

getIngredientContainersInDataListForTable(*ingredientContainerList*)
Muodostaa annetujen raaka-aineiden(container) tiedoista listojen listan, jotka on helppo populoida QTabletWidget tauluun.

getIngredientsInDataListForTable(*ingredientList*)
Muodostaa annetujen raaka-aineiden tiedoista listojen listan, jotka on helppo populoida QTabletWidget tauluun.

getRecipesInDataListForTable(*recipeList*)
Muodostaa annetujen reseptien tiedoista listojen listan, jotka on helppo populoida QTabletWidget tauluun.

initButtons()
Tässä metodissa määritellään käyttöliittymän painikkeiden toiminallisuudet

initDialogLineEdits()
Tässä metodissa alustetaan tarvittavat validaattorit sekä asetetaan nämä validaattorit käyttöön käyttöliittymän "Uusi resepti" dialogin tekstikenttiin

initLineEdits()
Tässä metodissa alustetaan tarvittavat validaattorit sekä asetetaan nämä validaattorit käyttöön käyttöliittymän tekstikentille.

Käytettävät validaattorit ovat :min2char: vähintään 2 merkkiä pitkä :min1tomax10char: 1-10 merkkiä aakkosia :float2decimals: desimaaliluku, jossa 2 desimaalia :onlyint: ainoastaan kokonaislukuja

initSettingsAndLists()
Tämä metodi alustaa tarvittavat muuttujat ja oliot sekä lataa reseptit, raaka-aineet ja varastotilanteen tiedostoista listoihin.

initTablesAndLists()
Tässä metodissa määritellään käyttöliittymän taulukkojen toiminnallisuudet, esim. kun taulukon riviä klikataan

initToEditVariables()
Tässä metodissa alustetaan apumuuttujat, joilla pidetään kirjaa mikä rivi mistäkin taulukosta on valittuna. Tätä tietoa tarvitaan taulukon tietojen muuttamisessa, esim. reseptin nimen.

loadFromFileToList()
Tämä metodi päättelee kutsujan (self.sender()) perusteella mikä tiedosto tulee ladata uudelleen ja myöskin lataa sen.

Jos self.sender() on None, niin ladataan kaikki tiedostot. Muulloin kutsuja on esim. self.buttonLoadRecipes, jolloin ladataan reseptit tiedostosta.

openFileUTF8(*file*)
Tämä metodi avaa halutun tiedoston UTF-8 enkoodauksella ja palauttaa sen tiedostokahvan.

Args:

file tiedoston polku

Returns:

Onnistuessa tiedostokahva avattuun tiedostoon

Epäonnistuessa False

populateAllMainTables()
Tämä metodi kutsuu kaikkien päätaulukoiden (reseptit, varastotilanne, raaka-aineet) populointimetodeja

populateIngredientsEditFields(*mi*)
Tämä metodi populoi raaka-ainenäkymällä olevat tekstikentät, joilla voi muokata raaka-aineen tietoja. Tätä metodia kutsutaan, kun raaka-ainelistaus taulua klikataan.

Rivin indeksi tallennetaan self.ingredientToEdit muuttujaan, jotta muutoksia tallennettaessa tiedetään mitä raaka-ainetta täytyy muokata.

Args:

mi mi muuttuja/olio, joka sisältää klikatun rivin ja kolumnin indeksin.

populateIngredientsTable()
Tämä metodi populoi raaka-ainelistaus tauluun kaikki tiedetyt raaka-aineet

populateRecipesEditFields(*mi*)
Tämä metodi populoi reseptinäkymällä olevat reseptin tekstikentät sekä tyhjentää reseptin raaka-aine ja ohje muokkaustekstikentät sekä kutuu metodeja, joilla populoidaan reseptin raaka-aine ja ohjetaulut.

Tätä metodia kutsutaan, kun reseptilistaus taulua klikataan.

Rivin indeksi tallennetaan self.recipeToEdit muuttujaan, jotta muutoksia tallennettaessa tiedetään mitä reseptiä täytyy muokata.

Args:

mi mi muuttuja/olio, joka sisältää klikatun rivin ja kolumnin indeksin.

populateRecipesIngredientEditFields(*mi*)
Tämä metodi populoi reseptinäkymällä olevat raaka-aine tekstikentät, joilla voi muokata reseptin raaka-aineen tietoja. Tätä metodia kutsutaan, kun reseptin raaka-ainelistaus taulua klikataan.

Rivin indeksi tallennetaan self.recipeIngredientToEdit muuttujaan, jotta muutoksia tallennettaessa tiedetään mitä raaka-ainetta täytyy muokata.

Args:

mi mi muuttuja/olio, joka sisältää klikatun rivin ja kolumnin indeksin.

populateRecipesIngredientsTable()
Tämä metodi populoi reseptinäkymällä reseptin raaka-aineet tauluun. Tätä metodia kutsutaan, kun reseptilistaus taulua klikataan.

Oikean reseptin löytämiseksi hyödynnetään self.recipeToEdit muuttujaa.

populateRecipesInstructionsEditFields(*mi*)
Tämä metodi populoi reseptinäkymällä olen ohje tekstikentät, joilla voi muokata reseptin ohjetta. Tätä metodia kutsutaan, kun reseptin ohjelistaus taulua klikataan.

Rivin indeksi tallennetaan self.recipeInstructionToEdit muuttujaan, jotta muutoksia tallennettaessa tiedetään mitä raaka-ainetta täytyy muokata.

Args:

mi mi muuttuja/olio, joka sisältää klikatun rivin ja kolumnin indeksin.

populateRecipesInstructionsTable()
Tämä metodi populoi reseptinäkymällä reseptin ohjeet tauluun. Tätä metodia kutsutaan, kun reseptilistaus taulua klikataan.

Oikean reseptin löytämiseksi hyödynnetään self.recipeToEdit muuttujaa.

populateRecipesTable()
Tämä metodi populoi reseptilistaus tauluun kaikki tiedetyt reseptit

populateSearchTable()
Tämä metodi populoi datan hakunäkymän taulukkoon. Tätä metodia kutsuu hakunäkymän "Hae" painike.

Hakunäkymällä on mahdollista valita nolla tai useampi hakuvaihtoehto, tämä metodi sisältää myös logiikan valittujen vaihtoehtojen tarkastamiselle sekä oikean hakutuloksen saamiseksi.

Haku hyödyntää Search luokan metodeja, jotka palauttavat listan resepteistä, jotka täyttävät hakukriteerin. Hakulogiikka on ns. iteratiivinen ja alussa Search luokan metodeille annetaan listana kaikki tunnetut reseptit, mahdollinen seuraava haku kuitenkin tehdään edellisen haun palauttamasta listasta, jolloin lopuksi listassa on enää kaikki hakutulokset täyttävät reseptit.

populateStorageEditFields(*mi*)
Tämä metodi populoi varastonäkymällä olevat tekstikentät, joilla voi muokata varastossa olevaa raaka-ainetta. Tätä metodia kutsutaan, kun varastolistauksessa klikataan riviä.

Rivin indeksi tallennetaan self.storageToEdit muuttujaan, jotta muutoksia tallennettaessa tiedetään mitä varasto raaka-ainetta täytyy muokata

Args:

mi mi muuttuja/olio, joka sisältää klikatun rivin ja kolumnin indeksin.

populateStorageTable()
Tämä metodi populoi varastolistaus tauluun kaikki varastossa olevat raaka-aineet

populateTableWithData(*table*, *data*)
Tämä metodi populoi annettuun QTableWidget tauluun annetun datan. Datan tulee olla muotoa esim. data = [['Nimi','Määrä'], ['Kala', 'Peruna'], [5,3]]

Taulun sisältö tyhjennetään aluksi, asetetaan asetetaan sarakkeiden ja rivien lukumäärät. Tämän jälkeen taulu populoidaan datalla, jonka jällkeen kolumnien leveydet skaalataan sisällölle sopivaksi.

saveIngredientsEdit()
Tämä metodi tarkistaa raaka-aineen perustietojen tekstikenttien arvot, jonka jälkeen muutokset tallennetaan raaka-aineeseen.

Oikea raaka-aine löydetään self.ingredientToEdit muuttujan avulla.

saveNewRecipe()
Tämä metodi tarkastaa uusi resepti -dialogissa annetut tiedot ja valideja, niin luo uuden reseptin näillä tiedoilla ja lisää sen reseptilistaan. Tämän jälkeen reseptilistaus taulu populoidaan uudelleen.

saveRecipesEdit()
Tämä metodi tarkistaa reseptin perustietojen tekstikenttien arvot, jonka jälkeen muutokset tallennetaan reseptiin.

Oikea resepti löydetään self.recipeToEdit muuttujan avulla.

saveRecipesIngredientEdit()
Tämä metodi tarkistaa reseptin raaka-aineen tekstikenttien arvot, jonka jälkeen muutokset tallennetaan reseptiin.

Oikea resepti sekä raaka-aine löydetään self.recipeToEdit ja self.recipeIngredientToEdit muuttujien avulla.

saveRecipesInstructionsEdit()
Tämä metodi tallentaa annetun ohjetekstin reseptille.

Oikea resepti sekä ohje löydetään self.recipeToEdit ja self.recipeInstructionToEdit muuttujien avulla.

saveStorageEdit()
Tämä metodi tarkistaa varastossa olevan raaka-aineen perustietojen tekstikenttien arvot, jonka jälkeen muutokset tallennetaan raaka-aineeseen.

Oikea raaka-aine löydetään self.storageToEdit muuttujan avulla.

saveToFile()
Tämä metodi päättelee kutsujan (self.sender()) perusteella mikä tiedosto tulee tallentaa ja myöskin tallentaa sen.

Jos self.sender() on self.buttonSaveAll, niin ladataan kaikki tiedostot. Muulloin kutsuja on esim. self.buttonSaveRecipes, jolloin tallennetaan reseptit tiedostoon.

showCreateNewRecipeDialog()
Tämä metodi luo uuden QDialog widgetin, jossa voi täyttää uuden reseptin tietoja. Dialogin graafinen ulkoasu on luotu QT Designerilla ja on tallennettuna moduuliin GUIrecipeDialog.

Jos dialogissa painetaan "OK" painiketta, kutsutaan metodia self.saveNewRecipe(), joka annetut tiedot ja tallentaa reseptin.

14.2 Search

Created on 19.4.2016

@author: Kimi Päivärinta

class search.**Search**
Tämän luokan metodeilla on mahdollista tehdä erilaisia hakuja. Ohjelma hyödyntää tätä luokkaa reseptien etsimisessä tietyillä hakuehdoilla

amountDifferenceMax10Perc(*ingredientRecipe*, *ingredientStorage*)
Apumetodi, jota ei ole tarkoitus kutsua luokan ulkopuolelta.

Tällä metodilla tarkastetaan onko varastossa tarpeeksi tiettyä raaka-ainetta. Jos resepti vaatii 1kg jauhelihaa, niin 900g kin kyllä riittää, joten varastosta saa puuttua maksimissaan 10% vaaditusta määrästä.

Args:

ingredientRecipe Reseptissä oleva ingredientContainer olio

ingredientStorage Varastossa oleva ingredientCOntainer olio

Returns:

True Raaka-aineiden määrän erotus on alle 10% tarvittavasta eli reseptin määrästä

False Erotus on yli 10% tarvittavasta määrästä

howManyIngredientsFoundInStorage(*recipe*)
Apumetodi, jota ei ole tarkoitus kutsua luokan ulkopuolelta.

Metodi käy reseptin raaka-aineet läpi ja montako niistä on tarpeeksi varastossa, metodi siis tarkastaa myös määrän. Jos raaka-ainetta ei ole tarpeeksi varastossa, tarkastetaan voidaanko raaka-aine itse valmistaa reseptistä, jolloin kutsutaan tätä samaa metodia. Tätä metodia voidaan kuitenkin kutsua maksimissaan 2 kertaa, jottei päädytä ikuiseen looppiin. Siis Raaka-aine - resepti ketjua seurataan vain 2 kertaa.

searcForhRecipesNIngredientsInStorage(*recipesList*, *N*, *storageList*, *NNotInStorage*)
Tällä metodilla voidaan etsiä reseptejä, joihin löytyy vähintään N raaka-ainetta varastosta tai puuttuu maksimissaan N raaka-ainetta.

Args:

recipesList Lista kaikista reseptiolioista

N Monta raaka-ainetta löydyttävä/puututtava varastosta (int)

storageList Lista varastossa olevista raaka-aineolioista

NNotInStorage (Boolean) True arvolla N merkitsee sitä montako saa puuttua. False arvolla N merkitsee montako täytyy löytyä

Returns:

recipesFound Lista reseptio-olioista

searchFromList(*searchFor*, *searchList*)
Tällä metodilla etsitään olio listasta tiettyä oliota sen nimen perusteella. Vertailu tapahtuu stringeinä ja vaatii, että listassa olevilla olioilla on getName() metodi.

Args:

searchFor Etsittävä nimi (string)

searchList Lista olioista, joilla on getName() metodi

returns:

recipesFound Lista löydetyistä olioista. Tyhjä, jos ei löytynyt

searchIncludesIngredient(*ingredientStr*, *recipesList*)
Tällä metodilla etsitään reseptejä, jotka sisältävät tietyn raaka-aineen. Vertailutapahtuu stringeinä

Args:

ingredientStr Raaka-aineen nimi (string)

recipesList Lista reseptiolioista

Returns:

recipesFound Lista reseptiolioista. Tyhjä, jos ei löytynyt

searchNoAllergen(*allergenStr*, *recipesList*)
Tämmä metodilla etsitään reseptejä, jotka eivät sisällä tiettyä allergeeniä. Allergeenien vertailu tapahtuu stringeinä

Args:

allergenStr Allergeeni (string)

recipesList Lista reseptiolioista

Returns:

recipesFound Lista reseptiolioista. Tyhjä, jos ei löytynyt

14.3 Conversion

Created on 19.4.2016

@author: Kimi Päivärinta

class conversion.**Conversion**

Conversion luokka sisältää kaikki ohjelman tuntemat yksiköt sekä niiden "suhdeluvut", jotka mahdollistaa muunnokset. Tämän luokan avulla onnistuu yksikkömuunnokset.

convertFromTo(*amount*, *unitFrom*, *unitTo*, *density*)

Tekee yksikkömuunnoksen ja palauttaa muunnetun arvon. Jos ei muunnettavissa, palauttaa alkuperäisen arvon.

Args:

amount Muunnettava määrä (float)

unitFrom Alkuperäinen yksikkö

unitTo Haluttu yksikkö

density aineen tiheys, tarvitaan vain kun muunnetaan massa<->tilavuus (float)

Returns:

Muunnetun yksikön (float)

convertMassToMass(*amount*, *unitFrom*, *unitTo*)

Muuttaa massa yksikön massa yksiköksi, esim. g->kg

convertMassToVolume(*amount*, *unitFrom*, *unitTo*, *density*)

Muuttaa massa yksikön tilavuus yksiköksi, esim. kg -> l. Hyödyntää kaavaa v = m / r

convertVolumeToMass(*amount*, *unitFrom*, *unitTo*, *density*)

Muuttaa tilavuus yksikön massa yksiköksi, esim. l -> kg. Hyödyntää kaavaa m = v * r

convertVolumeToVolume(*amount*, *unitFrom*, *unitTo*)

Muuttaa tilavuus yksikön tilavuus yksiköksi, esim. l -> m3

isValidUnit(*unit*)

Tällä metodilla voidaan tarkistaa onko sille parametrina annettu yksikkö ohjelman tuntema

Returns:

Tunnettu True

Ei tunnettu False

14.4 Recipe

Created on 19.4.2016

@author: Kimi Päivärinta

class recipe.**Recipe**

Luokka reseptejä varten. Tämä luokka pitää sisällään kaikki reseptille ominaiset attribuutit sekä tarvittavat metodit niiden arvojen muuttamiseksi.

Attributes:

self.date Luontipäivä

self.name Reseptin nimi

self.time Reseptin tekemiseen menevä aika minuuttina (int)

self.instructions Ohjeet (str[])

self.outcomeSize Reseptin lopputuloksen koko, esim. 4 (kg)

self.outcomeUnit Reseptin lopputuloksen yksikkö, esim (4) kg

self.ingredients Raaka-aineet (object[])

Returns: Attribuuttien muuttamiseen käytettävät metodit (set* & add* & delete/remove*) palauttavat True, jos muutos onnistuu

Raises: Kaikki attribuuttien asettamiseen käytettävät metodit (set* & add*) heittävät SetAttributeErrorin, jos attribuutin arvon asetus epäonnistuu.

addIngredientContainer(*ingredientContainer*)
Validoi, että lisättävä raaka-aine on IngredientContainer-olio sekä lisää raaka-aineen(Container) self.ingredients[] listaan

addInstruction(*instruction*)
Validoi, että ohje on yli 2 merkkiä pitkä ja lisää sen self.instruction[] listaan

deleteIngredient(*index*)
Poistaa reseptiltä raaka-aineen. Argumenttina annetaan ohjeen sijainti listassa (index)

deleteInstruction(*index*)
Poistaa reseptiltä ohjeen. Argumenttina annetaan ohjeen sijainti listassa (index)

getAllergensDistinctGUI()
Palauttaa reseptien raaka-aineiden stringinä pilkulla erotettuna. Allergeeni esiintyy listassa vain kerran, vaikka se olisi monessa raaka-aineessa.

getDate()
Palauttaa reseptin luontipäivän

getIngredients()
Palauttaa reseptin raaka-aine oliot listana

getIngredientsGUI()
Palauttaa raaka-aineiden nimet listana

getIngredientsStr()
Palauttaa reseptin raaka-aineet stringinä, hyödyntää raaka-aine luokan __str__() metodia

getInstructions()
Palauttaa ohjeet listana

getInstructionsStr()
Palauttaa ohjeet stringinä, jokainen ohje omalla rivillä ja edessä ohjeen järjestysnumero eli järjestys listassa

getName()
Palauttaa reseptin nimen

getOutcomeSize()
Palauttaa lopputuloksen floattina

getOutcomeSizeGUI()
Palauttaa lopputuloksen stringinä desimaalipilkulla

getOutcomeStr()
Palauttaa reseptin lopputuloksen desimaalipilkulla muodossa "<määrä> <yksikkö>"

getOutcomeUnit()
Palauttaa lopputuloksen yksikön

getTime()
Palauttaa reseptin tekemiseen menevän ajan inttinä

getTimeGUI()
Palauttaa reseptin tekemiseen menevän ajan stringinä

getTimeStr()
Palauttaa reseptin tekemiseen menevän ajan stringinä, jonka lopussa on ” Min”

setDate(*date*)
Validoi päivämäärän ja asettaa sen stringinä: self.date

setName(*name*)
Validoi, että nimi on yli 2 merkkiä pitkä ja asettaa sen: self.name

setOutcomeSize(*outcomeSize*)
Muuttaa desimaalipilkun pisteeksi ja settaa määrän floattina: self.outcomeSize

setOutcomeUnit(*outcomeUnit*)
Validoi, että yksikkö on ohjelman tuntema ja asettaa sen: self.outcomeUnit

setTime(*time*)
Asettaa reseptin tekemiseen menevän ajan (min) inttinä: self.time

14.5 Ingredient

Created on 19.4.2016

@author: Kimi Päivärinta

class ingredient.**Ingredient**
Raaka-aine luokka. Tämä luokka sisältää perustiedot raaka-aineesta, varastossa ja resepteissä olevat “raaka-aineet” sisältävät tämän olion.

Attributes:

self.date Luontipäivä

self.name Nimi

self.density Tiheys yksikkömuunnoksia varten (float)

self.allergens Allergeenit (str[])

self.recipe Mahdollinen resepti (object)

self.recipeLoaded Kertoo onko raaka-aineen resepti ladattu. None = Ei reseptiä, False = Resepti on, mutta oliota ei ladattu, True = Olio ladattu

Returns: Attribuuttien muuttamiseen käytettävät metodit (set* & add* & delete/remove*) palauttavat True, jos muutos onnistuu

Raises: Attribuuttien asettamiseen käytettävät metodit (set* & add*) heittävät SetAttributeErrori:n, jos validointi epäonnistuu

addAllergen(*allergen*)
Validoi, että allergeeni on yli 2 merkkiä pitkä ja lisää sen raaka-aineen allergeeni listaan

getAllergens()
Palauttaa allergeenit listana

getAllergensGUI()
Palauttaa allergeenit stringinä pilkulla ja välilyönnillä erotettuna

getAllergensStr()
Palauttaa allergeenit stringinä pilkulla erotettuna sekä alkussa teksti “Allergeenit: “

getDate()
Palauttaa luontipäivän

getDensity()
Palauttaa tiheyden floattina

getDensityGUI()
Palauttaa tiheyden stringinä desimaalipilkulla

getName()
Palauttaa nimen

getRecipe()
Palauttaa resepti olion, jos se on asetettu ja ladattu. Muulloin False

getRecipeGUI()
Palauttaa reseptin nimen stringinä

getRecipeLoaded()
Palauttaa self.recipeLoaded arvon. None = Ei reseptiä, True = Resepti olio ladattu, False = Reseptiä ei vielä ladattu

getRecipeStr()
Palauttaa "Resepti: " + reseptin nimi, jos resepti on asetettu ja ladattu

hasRecipe()
Palauttaa True, jos raaka-aineella on tallennettu resepti, muuten false

loadRecipe(*recipesList*)
Etsii nimen perusteella reseptilistasta raaka-aineelle halutun reseptin ja asettaa olion: self.recipe

Args:

recipesList Lista kaikista resepteistä

Returns:

Onnistuessa True

Ei ladattavaa None

Raises:

SetAttributeError Reseptiä ei löytynyt

removeAllergens()
Poistaa kaikki raaka-aineen allergeenit

removeRecipe()
Asettaa self.recipe = None sekä self.recipeLoaded = None

setDate(*date*)
Validoi päivämäärän ja asettaa sen: self.date

setDensity(*density*)
Muuttaa desimaalipilkun pisteeksi, muuntaa tiheyden float luvuksi ja asettaa sen: self.density

setName(*name*)
Validoi, että nimi on yli 2 merkkiä pitkä ja asettaa sen self.date

setRecipe(*recipe*)
Raaka-aineet luetaan sisälle ennen reseptejä, joten reseptin oliota ei todennäköisesti ole vielä olemassa. Validoi, että resepti on yli kaksi merkkiä pitkä sekä asettaa halutun reseptin nimen stringinä: self.recipe sekä asettaa self.recipeLoaded = False

class ingredient.**IngredientContainer**

Tämä luokka sisältää viittauksen raaka-aine olioon ja tämän lisäksi omat attribuutit määrästä sekä yksiköstä. Tätä luokkaa hyödynnetään varastolistauksen sekä reseptien raaka-aineiden tallentamisessa. Samaa raaka-ainetta käytetään hyvin todennäköisesti useassa eri reseptissä, jos Ingredient luokassa olisi määrä, niin silloin tietoa joutuisi kopioimaan sekä säilyttämään moneen kertaan. Tämän luokan avulla esim. Jauhelihan tiedot ovat vain kertaalleen Ingredient oliossa ja resepteissä olevat IngredientContainer oliot sisältävät viittauksen tähän.

Attributes:

self.ingredient Raaka-aine olio

self.quantity Raaka-aineen määrä

self.unit Määrän yksikkö

Returns: :Attribuuttien muuttamiseen käytettävät metodit (set* & add* & delete/remove*) palauttavat True, jos muutos onnistuu

Raises: :Attribuuttien asettamiseen käytettävät metodit (set* & add*) heittävät SetAttributeErrori:n, jos validointi epäonnistuu

getAllergens()
Palauttaa allergeenit listana

getAllergensStr()
Palauttaa allergeenit stringinä pilkulla erotettuna sekä alkussa teksti “Allergeenit: “

getDensity()
Palauttaa raaka-aineen tiheyden floattina

getIngredient()
Palauttaa raaka-aine olion

getName()
Palauttaa raaka-aineen nimen

getQuantity()
Palauttaa määrän floattina

getQuantityStr()
Palauttaa määrän stringinä, desimaalipilkulla

getRecipe()
Palauttaa reseptiolion, jos raaka-aineella on

getRecipeStr()
Palauttaa reseptin nimen stringinä, jos raaka-aineella on

getUnit()
Palauttaa määrän yksikön

hasRecipe()
Palauttaa True, jos raaka-aineella on tallennettu resepti, muuten false

setIngredient(*ingredient*, *ingredientsList*)
Etsii halutun raaka-aineen annetusta raaka-ainelistasta nimen perusteella sekä asettaa sen: self.ingredient

Attributes:

ingredient Etsittävä raaka-aine (string)

ingredientsList Lista kaikista raaka-aineista

Returns:

Onnistuessa True

Raises: :SetAttributeError epäonnistuessa

setQuantity(*quantity*)
Muuttaa desimaalipilkun pisteeksti ja asettaa määrän floattina: self.quantity

setUnit(*unit*)
Validoi, että yksikkö on ohjelman tuntema sekä asettaa sen: self.unit

14.6 IO

Created on 19.4.2016

@author: Kimi Päivärinta

class IO.**IO**
Input output luokka, jolla ladataan raaka-aineet, reseptit ja varastolistaus tiedostoista. Tämän lisäksi nämä voidaan tallentaa tiedostoihin annetusta listasta.

loadIngredients(*inputLines*)
Tämä metodi lukee raaka-aineet tiedostosta ja palauttaa ne listana.

Args:

inputLines tiedoston kahva (fileIO), josta tiedot luetaan.

Returns:

ingredientList Raaka-aine oliot listana

successCount Montako luettiin onnistuneesti sisään

errorCount Montako jäi virheeseen luettaessa

Raises:

CorruptedFileError Jos kohdataan tuntematon tiedostotyyppi

loadRecipes(*inputLines*, *ingredientsList*)
Tämä metodi lukee reseptit tiedostosta ja palauttaa ne listana.

Args:

inputLines tiedoston kahva (fileIO), josta tiedot luetaan.

ingredientList Tunnetut raaka-aineet listana

Returns:

recipesList Resepti oliot listana

successCount Montako luettiin onnistuneesti sisään

errorCount Montako jäi virheeseen luettaessa

Raises:

CorruptedFileError Jos kohdataan tuntematon tiedostotyyppi

loadRecipesForIngredients(*ingredientsList*, *recipesList*)
Tällä metodilla voidaan ladata kaikkien raaka-aineiden reseptit. Käytännössä tämä metodi kutsuu raaka-aineen loadrecipe() metodia

Args:

ingredientList Raaka-aine oliot listana

recipesList Reseptio oliot listana

loadStorage(*inputLines*, *ingredientsList*)
Tämä metodi lukee varastolistauksen tiedostosta ja palauttaa ne listana.

Args:

inputLines tiedoston kahva (fileIO), josta tiedot luetaan.

ingredientList Tunnetut raaka-aineet listana

Returns:

storageList Raaka-aine oliot listana

successCount Montako luettiin onnistuneesti sisään

errorCount Montako jäi virheeseen luettaessa

Raises:

CorruptedFileError Jos kohdataan tuntematon tiedostotyyppi

saveIngredients(*fileName*, *ingredientsList*)
Tämä metodi tallentaa raaka-aineet tiedostoon ohjelman luettavassa muodossa. Tiedot tallennetaan ensin temp.ing tiedostoon, jonka jälkeen kyseinen tiedosto nimetään uudelleen halutun nimiseksi.

Args:

fileName Tiedostonimi, johon raaka-aineet tallennetaan

ingredientList Raaka-aine oliot listana

saveRecipes(*fileName*, *recipesList*)
Tämä metodi tallentaa reseptit tiedostoon ohjelman luettavassa muodossa. Tiedot tallennetaan ensin temp.rec tiedostoon, jonka jälkeen kyseinen tiedosto nimetään uudelleen halutun nimiseksi.

Args:

fileName Tiedostonimi, johon reseptit tallennetaan

recipesList Resepti oliot listana

saveStorage(*fileName*, *storageList*)
Tämä metodi tallentaa varastossa olevat raaka-aineet tiedostoon ohjelman luettavassa muodossa. Tiedot tallennetaan ensin temp.sto tiedostoon, jonka jälkeen kyseinen tiedosto nimetään uudelleen halutun nimiseksi.

Args:

fileName Tiedostonimi, johon varaston raaka-aineet tallennetaan

storageList Varaston raaka-aine oliot listana

14.7 CustomErrors

Created on 19.4.2016

@author: Kimi Päivärinta

exception customErrors.**CorruptedFileError**
Korruptoituneen tiedoston exception

exception customErrors.**SetAttributeError**
Käytetään, kun olion attribuutin asettamisessa tapahtuu virhe

Indices and tables

- genindex
- modindex
- search

c

i

m

r

s

A

C

D

G