The Java Cookbook

Dokumentáció

Bevezetés

Célkitűzés

A “The Java Cookbook” egy receptkezelő alkalmazás, amely lehetővé teszi a felhasználók számára, hogy recepteket tároljanak, szerkesszenek, töröljenek, és egyszerűen keressenek a már meglévő receptek között. Az alkalmazás célja, hogy könnyen használható legyen, minél áttekinthetőbb felületet biztosítson a felhasználók számára a receptek kezelésekor.

Célcsoport

A program létrejöttét legfőképpen édesanyám inspirálta, aki minden fontosabb ünnep közeledtekor 8 féle receptkönyvet búj a kedvenc receptjeiért, mindig szóvá téve hogy írnia kéne egy sajátot, amibe csakis az Ő kedvencei kerülnének. A ‘The Java Cookbook’ pontosan ezt a gondot kívánja kiküszöbölni. A felhasználó olyan recepteket tárolhat gyűjteményébe amiket csak szeretne, ezeket külön gyűjteményekbe is rakhatja, megoszthatja másokkal. A célcsoport minden olyan főzni-sütni vágyó személy, aki szeretné organizálni receptjeit.

Felhasználói dokumentáció

Funkciók

Recept megtekintése

Az alkalmazást elindítva egy lista jelenik meg, először minden receptet megmutatva. Ennek sorrendje a recept felvételének időpontja szerint van rendezve. Duplán kattintva egy-egy receptre megjelennek annak adatai. Az also ‘bezár’ gombra nyomva visszatérünk a főmenübe.

Recept hozzáadása

A program felső részén található egy navigációs bar. Itt a ‘Recept hozzáadása’ opciót választva felvehetünk egy új receptet a gyűjteményünkbe. Hogyan is történik ez?

1. A gombra nyomás után egy új ablak nyílik meg, ahol az alábbi adatokat kell kitöltened:
    - **Recept neve:** Ide írjuk be az étel nevét
    - **Elkészítési idő:** Megadjuk hány perc alatt készíthető el az étel
    - **Hozzávalók:** Ide írjuk a hozzávalókat, vesszővel, vagy új sorral elválasztva
    - **Elkészítés:** Leírjuk lépésről lépésre, hogyan készül az étel
2. Nyomjuk meg a **"Hozzáad"** gombot
3. Sikeres hozzáadás esetén a recept bekerül az alkalmazásba

Recept keresése

A ‘Recept keresése’ opciót választva 3 feltétel alapján kereshetünk receptjeink között.

1. A keresés opció választása után a megjelenő ablakban:
    - Írjuk be a keresett kifejezést a **"Keresés"** mezőbe
    - Válasszuk ki a keresés típusát:
        - **Recept neve** – *Minden olyan recept megjelenik amelynek neve tartalmazza a keresett szót/szavakat*
        - **Hozzávalók** – *Minden olyan recept megjelenik megjelenik amelynek hozzávalói között előfordula keresett szó/szavak*
        - **Elkészítési idő** – *Minden olyan recept megjelenik, amelynek elkészítési ideje* ***kevesebb, mint*** *a keresett szám*
2. Kattintsunk a **"Keresés"** gombra
3. Az eredmények listájában kattintsunk kétszer egy receptre, hogy részletesen megtekinthessük

Recept szerkesztése

A ‘Recept szerkesztése’ opcióval lehetőségünk van egy meglévő recept adatait megváltoztatni. Szimplán kattintsunk egyszer a módosítani kívánt receptre, nyomjuk meg a ‘Recept szerkesztése’ opciót, és a megjelenő adat-leíró ablakban írjuk át a módosítani kívánt adatokat.

FONTOS!

A recept cask akkor kerül valóban módosításra, ha azt a ‘Módosítás’ gombra nyomva el is mentjük. Ha az ablakot bezárjuk, vagy a ‘Mégse’ gombra nyomunk, semmi sem változik.

Recept törlése

Képesek vagyunk egy nem kívánt receptet törölni a listánkból. A szerkesztéshez hasonlóan válasszunk ki egyszeri kattintással egy receptet a listából, nyomjunk a ‘Recept törlése’ opcióra, majd ezt megerősítve törlődik a recept.

A Fájl menü

A program egyik legnagyobb előnye hogy modulárisan képes kezelni több receptgyűjteményt. Mivel JSON fájlkezelést használunk , a program lehetővé teszi, hogy könnyen menthessünk és betölthessünk különböző receptgyűjteményeket anélkül, hogy azok keverednének. Az alábbi lehetőségeket találod a Fájl menüben:

Json betöltése...

Lehetővé teszi, hogy betöltsünk egy korábban elmentett receptgyűjteményt JSON formátumban.
Használat:

1. Kattintsunk a **Fájl** menüben a **"JSON betöltése..."** opcióra
2. Válasszunk ki egy érvényes JSON fájlt a számítógépről
3. Az alkalmazás automatikusan betölti és megjeleníti a gyűjtemény receptjeit

Mentés másként...

Ezzel az opcióval a jelenlegi receptgyűjteményt menthetjük el egy új fájlba. E tökéletes arra az esetre ha úgy vinnénk fel új receptet hogy az ne keveredjen a meglévőkkel.
Használat:

1. Kattintsunk a **"Mentés másként..."** lehetőségre
2. Adjunk meg egy fájlnevet és mentési helyet
3. Az alkalmazás létrehozza a JSON fájlt

Üres...

Létrehozhatunk egy teljesen új, üres gyűjteményt, amelyben még nincsenek receptek.
Használat:

1. Kattintsunk az **"Üres..."** opcióra
2. Adjunk meg egy fájlnevet, oda ahová az új gyűjteményünket menteni szeretnénk
3. Az alkalmazás automatikusan betölti az üres gyűjteményt

Kilépés

Az alkalmazás megfelelően bezárja magát.

Fejlesztői dokumentáció

Fejlesztési eszközök

- **Programozási nyelv:** Java 22
- **Build-eszköz:** Maven
- **Grafikus felület:** Java Swing
- **Adatkezelés:** JSON fájlformátum (Gson könyvtár segítségével)
- **Tesztelési keretrendszer:** JUnit 5
- **Fejlesztői környezet:** IntelliJ IDEA

Architektúra

Többrétegű felépítés:

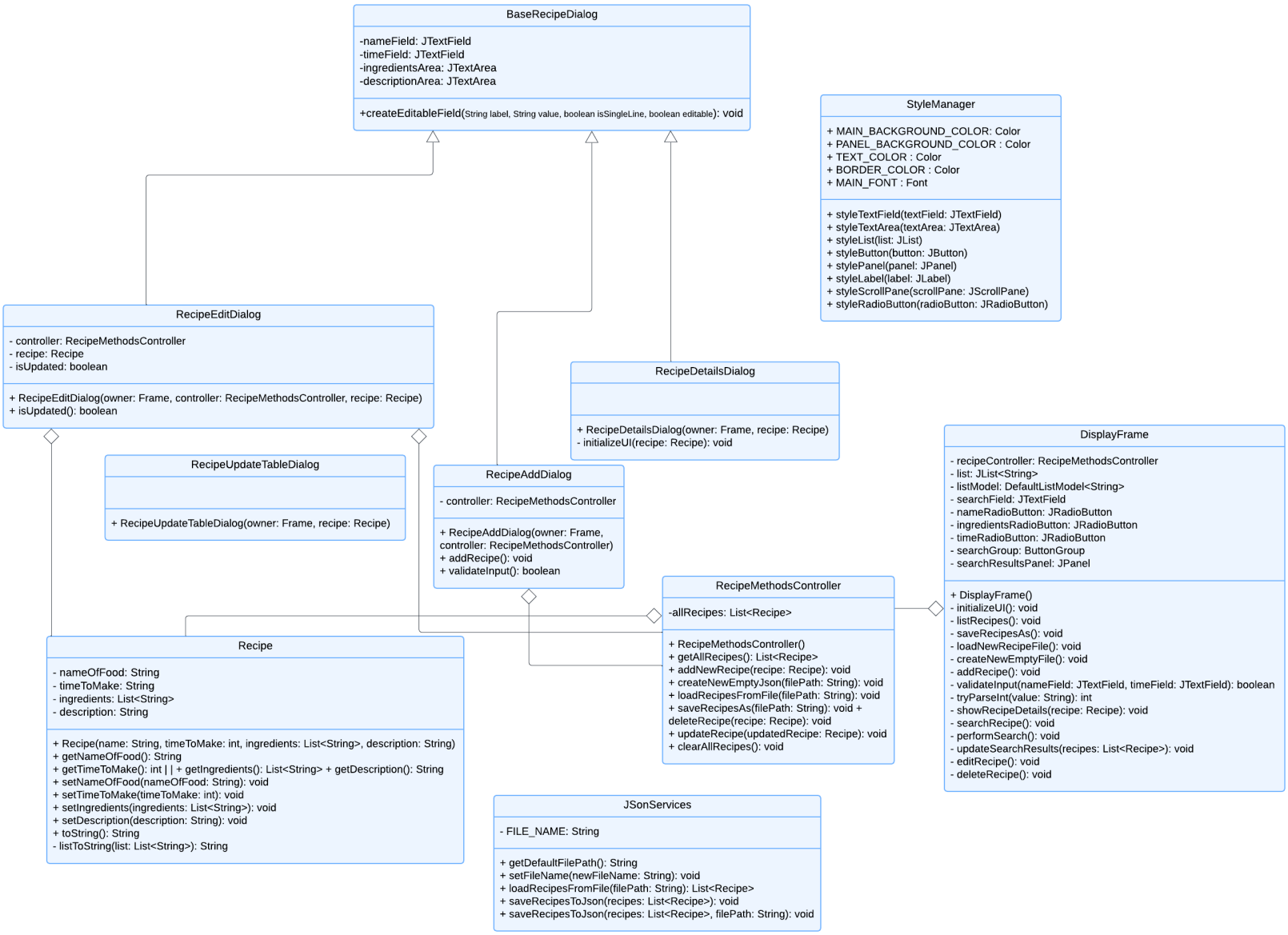
Az alkalmazás jól szervezett package-ekből áll, amelyek különböző feladatköröket látnak el:

1. **Controller package**: Felelős a legtöbb logika végrehajtásáért (pl. RecipeMethodsController)
2. **View package**: Felhasználói felület kezelése, grafikai függvények (pl. DisplayFrame, RecipeAddDialog)
3. **Model package**: Alaposztáylok, a főadatok meghatározása (pl. Recipe)
4. **Service package**: Fájlkezelés (pl. JSonServices)
5. **Main**: a main() metódust tartalmazza

Az osztályok

- Main
- Recipe
- BaseRecipeDialog
- RecipeMethodsController
- JSonServices
- DisplayFrame
- RecipeAddDialog
- RecipeDetailsDialog
- RecipeEditDialog
- RecipeUpdateDialog
- StyleManager

UML Diagram


BaseRecipeDialog
-nameField: JTextField
-timeField: JTextField
-ingredientsArea: JTextArea
-descriptionArea: JTextArea
+createEditableField(String label, String value, boolean isSingleLine, boolean editable): void
StyleManager
+ MAIN_BACKGROUND_COLOR: Color
+ PANEL_BACKGROUND_COLOR : Color
+ TEXT_COLOR : Color
+ BORDER_COLOR : Color
+ MAIN_FONT : Font
+ styleTextField(textField: JTextField)
+ styleTextArea(textArea: JTextArea)
+ styleList(list: JList)
+ styleButton(button: JButton)
+ stylePanel(panel: JPanel)
+ styleLabel(label: JLabel)
+ styleScrollPane(scrollPane: JScrollPane)
+ styleRadioButton(radioButton: JRadioButton)
RecipeEditDialog
- controller: RecipeMethodsController
- recipe: Recipe
- isUpdated: boolean
+ RecipeEditDialog(owner: Frame, controller: RecipeMethodsController, recipe: Recipe)
+ isUpdated(): boolean
RecipeDetailsDialog
+ RecipeDetailsDialog(owner: Frame, recipe: Recipe)
- initializeUI(recipe: Recipe): void
RecipeUpdateTableDialog
+ RecipeUpdateTableDialog(owner: Frame, recipe: Recipe)
RecipeAddDialog
- controller: RecipeMethodsController
+ RecipeAddDialog(owner: Frame, controller: RecipeMethodsController)
+ addRecipe(): void
+ validateInput(): boolean
DisplayFrame
- recipeController: RecipeMethodsController
- list: JList<String>
- listModel: DefaultListModel<String>
- searchField: JTextField
- nameRadioButton: JRadioButton
- ingredientsRadioButton: JRadioButton
- timeRadioButton: JRadioButton
- searchGroup: ButtonGroup
- searchResultsPanel: JPanel
+ DisplayFrame()
- initializeUI(): void
- listRecipes(): void
- saveRecipesAs(): void
- loadNewRecipeFile(): void
- createNewEmptyFile(): void
- addRecipe(): void
- validateInput(nameField: JTextField, timeField: JTextField): boolean
- tryParseInt(value: String): int
- showRecipeDetails(recipe: Recipe): void
- searchRecipe(): void
- performSearch(): void
- updateSearchResults(recipes: List<Recipe>): void
- editRecipe(): void
- deleteRecipe(): void
RecipeMethodsController
-allRecipes: List<Recipe>
+ RecipeMethodsController()
+ getAllRecipes(): List<Recipe>
+ addNewRecipe(recipe: Recipe): void
+ createNewEmptyJson(filePath: String): void
+ loadRecipesFromFile(filePath: String): void
+ saveRecipesAs(filePath: String): void +
deleteRecipe(recipe: Recipe): void
+ updateRecipe(updatedRecipe: Recipe): void
+ clearAllRecipes(): void
Recipe
- nameOfFood: String
- timeToMake: String
- ingredients: List<String>
- description: String
+ Recipe(name: String, timeToMake: int, ingredients: List<String>, description: String)
+ getNameOfFood(): String
+ getTimeToMake(): int | | + getIngredients(): List<String> + getDescription(): String
+ setNameOfFood(nameOfFood: String): void
+ setTimeToMake(timeToMake: int): void
+ setIngredients(ingredients: List<String>): void
+ setDescription(description: String): void
+ toString(): String
- listToString(list: List<String>): String
JSonServices
- FILE_NAME: String
+ getDefaultFilePath(): String
+ setFileName(newFileName: String): void
+ loadRecipesFromFile(filePath: String): List<Recipe>
+ saveRecipesToJson(recipes: List<Recipe>): void
+ saveRecipesToJson(recipes: List<Recipe>, filePath: String): void

Osztályok ismertetése

Recipe

A Recipe osztály az alkalmazás alapvető adatmodellje, amely egy receptet reprezentál. Ez tartalmazza a recept legfontosabb adatait.

- Attribútumai:
  - nameOfFood: A recept neve
  - timeToMake: Az elkészítési idő percben
  - ingredients: A hozzávalók listája
  - description: A recept elkészítésének szöveges leírása
- Metódusai:
  - Getterek és setterek az attribútumok kezeléséhez
  - **toString** metódus: Egy recept olvasásra alkalmas formátumban való megjelenítéséhez
  - Segédmetódus a hozzávalók listájának szöveggé alakításához.

RecipeMethodsController

A RecipeMethodsController az alkalmazás logikai rétegének központi osztálya, amely a receptekkel kapcsolatos műveleteket kezeli. Ez az osztály felelős a receptadatok kezeléséért, betöltéséért, mentéséért és módosításáért.

- Attribútumai:
  - allRecipes: A programban aktuálisan kezelt receptek listája

Funkciók és felelősségek:

- Receptkezelés:
  - Új recept hozzáadása (addNewRecipe).
  - Recept törlése (deleteRecipe).
  - Recept módosítása (updateRecipe).
  - Összes recept listázása (getAllRecipes).
  - Az összes recept törlése memóriából (clearAllRecipes).
- Adatmentés és betöltés:
  - Receptek betöltése JSON fájlból (loadRecipesFromFile).
  - Receptek mentése JSON fájlba (saveRecipesAs, addNewRecipe).
  - Új üres JSON fájl létrehozása (createNewEmptyJson).
- Alapértelmezett fájlkezelés:
  - Alapértelmezett fájl betöltése inicializáláskor.
  - Az aktuális fájlkezeléshez a JSonServices osztályt használja.

Ez az osztály a rendszer központi vezérlője, amely az adatok konzisztens kezeléséért és tárolásáért felelős. Az összes receptkezelési funkció innen indul ki, és itt kerül szinkronizálásra a fájlokkal.

JSonServices

A JSonServices osztály egy segédosztály, amely a receptek JSON formátumú fájlba történő mentéséért és onnan való betöltéséért felel. Az osztály közvetlen kapcsolatot teremt a fájlok és az alkalmazás adatai között.

- Attribútumai:
    - FILE_NAME: Az aktuális JSON fájl elérési útját tárolja Alapértelmezettként a 'src/main/resources/recipes.json' fájl

Metósudok:

Ez az osztály személyem szerint a legfontosabb a program működéséhez, így szeretném metódusait az előzőekhez képest jobban ismertetni.

loadRecipesFromFile(String filePath): feladata egy JSON fájl betöltése. Az adatokat deszerializálja egy List<Recipe> objektumlistába a Google Gson segítségével. Ha hibás a fájl vagy a JSON formátum nem megfelelő a Recipe osztálynak, üres listát ad vissza.

saveRecipesToJson(List<Recipe> recipes): A receptek mentése a jelenlegi fájlba (FILE_NAME). Általában receptkezelős műveletek után hívódik automatikusan.

saveRecipesToJson(List<Recipe> recipes, String filePath): A receptek mentése egy megadott/kiválasztott fájlba. A mentés másként opciókor hívódik meg. Google Gson használ az objektumlista JSON formátummá alakítására és fájlba írására.

getDefaultFilePath(): Visszaadja az alapértelmezett fájl elérési útját.

setFileName(String newFileName): Beállítja az aktuális fájl nevét.

StyleManager

A **StyleManager** osztály egy segédosztály, amely az alkalmazás grafikai elemeinek egységes megjelenítéséért felel. Statikus metódusokat és attribútumokat biztosít a komponensek vizuális stílusának egyszerű beállításához. A főinspiráció itt a HTML-CSS kapcsolata volt, létrehozva előre megírt, stílust meghatározó metódusokat cask meghívjuk a szükséges grafikai elemekre, és mind garantáltan ugyanolyan stílust fog képviselni.

- Attribútumok:
  - MAIN_BACKGROUND_COLOR: Az alkalmazás alapértelmezett háttérszíne
  - PANEL_BACKGROUND_COLOR: A panelek világosabb háttérszíne
  - TEXT_COLOR: A szövegek alapértelmezett színe
  - BORDER_COLOR: A keretek alapértelmezett színe
  - MAIN_FONT: Az alapértelmezett betűtípus az alkalmazásban

- Metódusok:
  - styleTextField(JTextField textField):
    - Egy szövegmező (JTextField) kinézetét állítja be:
      - Betűtípus, háttérszín, szöveg színe, balra igazítás és keret.
  - styleTextArea(JTextArea textArea):
    - Egy szövegterület (JTextArea) stílusának beállítása:
      - Betűtípus, háttérszín, szöveg színe, keret és balra igazítás.
  - styleList(JList list):
    - Egy lista (JList) megjelenését szabályozza:
      - Betűtípus, háttérszín, szöveg színe, kiválasztott elem színe és keret.
  - styleButton(JButton button):
    - Egy gomb (JButton) stílusát adja meg:
      - Betűtípus, háttérszín, szöveg színe, keret és fókusz elrejtése.
  - stylePanel(JPanel panel):
    - Egy panel (JPanel) megjelenésének beállítása:
      - Háttérszín és keret.
  - styleLabel(JLabel label):
    - Egy címke (JLabel) stílusát állítja be:
      - Betűtípus és szöveg színe.
  - styleScrollPane(JScrollPane scrollPane):
    - Egy görgetősáv (JScrollPane) megjelenésének testreszabása:
      - Keret és görgetősáv sebességének beállítása.
  - styleRadioButton(JRadioButton radioButton):
    - Egy rádiógomb (JRadioButton) stílusának meghatározása:
      - Betűtípus, háttérszín, szöveg színe, keret és fókusz elrejtése.

RecipeUpdateTableDialog

A RecipeUpdateTableDialog egy egyszerű, nem szerkeszthető JTabel táblázat, amely vizuálisan megjeleníti egy módosított recept adatait. Alapvetően információs célokat szolgál, és lehetővé teszi az ablak bezárását egy "OK" gomb segítségével.
Attribútuma nincsen.

DisplayFrame

A DisplayFrame osztály az alkalmazás fő felhasználói felületét (UI) biztosítja. Ez az osztály kezeli az alkalmazás ablakát, a menüsávot, és a különböző funkciók megvalósításához szükséges felületet.

- Attribútumai:
  - RecipeMethodsController: A receptkezelést végzi
  - JList és DefaultListModel: A receptek megjelenítésében segít
  - JTextField és JRadioButton: Recept kereséséhez
  - JPanel: Az ablak különböző részeinek tárolására

Metódusok:

- initializeUI: Beállítja az ablak címét, méretét, menüsávját, és a fő tartalmat
- Menüpont metódusok:
  - loadNewRecipeFile: JSON fájl betöltése
  - saveRecipesAs: Receptek mentése új fájlba
  - createNewEmptyFile: Új, üres receptgyűjtemény létrehozása
  - listRecipes: A jelenlegi receptek listájának frissítése
- Receptek kezelése:
  - addRecipe: Új recept hozzáadása
  - searchRecipe: Receptek keresése név, összetevők vagy idő alapján
  - editRecipe: Kiválasztott recept szerkesztése
  - deleteRecipe: Kiválasztott recept törlése
- Keresés és Szűrés:
  - performSearch: A keresési feltételek alapján szűri a recepteket
  - updateSearchResults: Frissíti a keresési eredmények megjelenítését

Kapcsolatok:

- RecipeMethodsController:
  - Függ a kontrollertől, amely az adatkezelést végzi (pl. betöltés, mentés, manipuláció).
- Recipe:
  - Használja a receptmodell objektumot az adatok megjelenítéséhez és kezeléséhez.
- RecipeAddDialog, RecipeEditDialog, RecipeDetailsDialog:
  - Ezeket a párbeszédablakokat hívja meg új receptek hozzáadására, meglévők szerkesztésére vagy részletek megtekintésére. Ezekről bővebben később.
- StyleManager:
  - Az UI komponensek egységes stílusát biztosítja.

JMenu Implementációja

Az DisplayFrame osztály fontos része a menüsáv, amely (amellett hogy előírt követelmény volt a programhoz) a felhasználók számára biztosítja az alkalmazás fő funkcióinak elérését. A menüsávot a JMenu és annak alkomponensei segítségével hoztam létre. Ez lehetővé teszi az egyszerű és intuitív navigációt az alkalmazás különböző funkciói között.

JMenu Felépítése

1. Fájl Menü:
    - JSON betöltése...: JSON fájl betöltése külső forrásból
    - Mentés másként...: Receptek mentése új fájlba
    - Üres...: Új, üres receptgyűjtemény létrehozása
    - Kilépés: Az alkalmazás bezárása
2. Egyéb Funkciók:
    - Receptek listázása: Az aktuális receptgyűjtemény megtekintése
    - Recept hozzáadása: Új recept hozzáadása a gyűjteményhez
    - Recept keresése: Receptek szűrése megadott feltételek alapján
    - Recept szerkesztése: Kiválasztott recept szerkesztése
    - Recept törlése: Receptek eltávolítása a gyűjteményből

```
// Menüsáv létrehozása
JMenuBar menuBar = new JMenuBar();
// Fájl menü létrehozása
JMenu fileMenu = new JMenu( s: "Fájl");
JMenuItem loadMenuItem = new JMenuItem( text: "JSON betöltése...");
JMenuItem saveAsMenuItem = new JMenuItem( text: "Mentés másként...");
JMenuItem newEmptyMenuItem = new JMenuItem( text: "Üres...");
JMenuItem exitMenuItem = new JMenuItem( text: "Kilépés");
fileMenu.add(loadMenuItem);
fileMenu.add(saveAsMenuItem);
fileMenu.add(newEmptyMenuItem);
fileMenu.add(exitMenuItem);
menuBar.add(fileMenu);
// Egyéb menük létrehozása
JMenuItem listMenuItem = new JMenuItem( text: "Receptek listázása");
JMenuItem addMenuItem = new JMenuItem( text: "Recept hozzáadása");
JMenuItem searchMenuItem = new JMenuItem( text: "Recept keresése");
JMenuItem editMenuItem = new JMenuItem( text: "Recept szerkesztése");
JMenuItem deleteMenuItem = new JMenuItem( text: "Recept törlése");
menuBar.add(listMenuItem);
menuBar.add(addMenuItem);
menuBar.add(searchMenuItem);
menuBar.add(editMenuItem);
menuBar.add(deleteMenuItem);
setJMenuBar(menuBar);
```

Interakció a JMenu-vel

A menüelemekhez ActionListener van rendelve, amely meghatározza, hogy egy-egy menüpont aktiválásakor milyen művelet hajtódik végre. Példa:

```
// Menü eseménykezelők
loadMenuItem.addActionListener(e -> loadNewRecipeFile());
saveAsMenuItem.addActionListener(e -> saveRecipesAs());
newEmptyMenuItem.addActionListener(e -> createNewEmptyFile());
exitMenuItem.addActionListener(e -> System.exit( status: 0));

listMenuItem.addActionListener(e -> listRecipes());
addMenuItem.addActionListener(e -> addRecipe());
searchMenuItem.addActionListener(e -> searchRecipe());
editMenuItem.addActionListener(e -> editRecipe());
deleteMenuItem.addActionListener(e -> deleteRecipe());
```

Miért fontos a JMenu használata?

A JMenu segítségével:

- Központosított, jól strukturált módon érhetők el az alkalmazás funkciói
- A felhasználói élmény intuitívvá válik
- A további bővítések és új funkciók hozzáadása egyszerűvé válik

Összegzés

Ahhoz az a hosszas osztályleírásból is látszik, a DisplayFrame osztály az alkalmazás fő belépési pontja a felhasználó számára, amely egy átlátható és interaktív kezelőfelületet nyújt. Ez az osztály koordinálja a felhasználói műveleteket, és biztosítja a különböző funkciók elérését a menü és az interaktív komponensek segítségével.

BaseRecipeDialog

A BaseRecipeDialog osztály egy alaposztály, amely a recepthez kapcsolódó párbeszédablakok (pl. recept hozzáadása, szerkesztése, részletek megtekintése) közös funkcionalitását és stílusát biztosítja. A Swing alapú JDialog osztályt örökli.

- Attríbútumai:
  - nameField: A recept nevét tartalmazó szöveges mező
  - timeField: A recept elkészítési idejét tartalmazó szöveges mező
  - ingredientsArea: A recept hozzávalóinak listáját tartalmazó szöveges terület
  - descriptionArea: Az elkészítési útmutatót tartalmazó szöveges terület

- Funkciók és felelősségek:
  - Egységesített dialógusstruktúra biztosítása:
    - Az osztály sablont ad a dialógusablakok elrendezésére és stílusára.
  - Komponenslétrehozás és stílusozás:
    - Dinamikusan hoz létre szerkeszthető vagy csak olvasható mezőket a receptek attribútumai számára.
  - Továbbfejleszthető alapként szolgál:
    - Más dialógusok, például RecipeAddDialog, RecipeEditDialog, RecipeDetailsDialog, ezt az osztályt öröklik és bővítik.

createEditableField:

Feladata egy mező dinamikus létrehozása a következő paraméterek alapján:

- **label**: A mező címkéje (pl. "Recept neve").
- **value**: Az alapértelmezett érték (pl. a recept neve).
- **isSingleLine**: Meghatározza, hogy egysoros szövegmezőt vagy többsoros szövegdobozt hoz létre.
- **editable**: Meghatározza, hogy a mező szerkeszthető-e.

Kimenete: Egy **JPanel**, amely tartalmazza a címkét és a szöveges mezőt, görgetési lehetőséggel.

Amiért szerettem volna ezt a metódust kiemelni, az az ‘editable’ paraméter és annak fontossága. Ebből az osztályból három osztály is leszármazik amelyeknek ugyan azonos a kinézeteik, nem mindegyikük módosítható, ezt vagyunk képesek manipulálni ezzel az egy változó segítségével. Ebből az osztályból származik le a RecipeAddDialog, RecipeEditDialog és a RecipeDetailDialog osztály hármasa.

RecipeAddDialog

A RecipeAddDialog osztály a receptek hozzáadására szolgáló felhasználói párbeszédablak. Ez az osztály a BaseRecipeDialog osztályból örököl, így annak általános funkcióit használja, és kiegészíti az új recept hozzáadásához szükséges logikával.

Fontos megemlítések:

- super(owner, "Új Recept Hozzáadása"):
  - Meghívja a szülő osztály (BaseRecipeDialog) konstruktorát, amely beállítja az ablak tulajdonosát és címét
  - Ez biztosítja, hogy az alap dialógusablak tulajdonságai (méret, pozíció, modalitás) öröklődjenek
- createEditableField:
  - A BaseRecipeDialog metódusát használva mezőket hoz létre a recept neve, elkészítési ideje, hozzávalói és leírása számára
  - Az editable = true biztosítja, hogy a mezők szerkeszthetők legyenek a felhasználók számára
- validateInput:
  - Ellenőrzi, hogy minden mező megfelelően van kitöltve, és az elkészítési idő számként adható meg
  - Ha a validáció hibás, értesítést küld a felhasználónak
- addRecipe:
  - Új Recipe objektumot hoz létre a felhasználó által megadott adatok alapján
  - Hozzáadja a receptet a RecipeMethodsController segítségével, amely kezeli az adatok tárolását
  - Ha sikeres, értesíti a felhasználót

RecipeEditDialog

A RecipeEditDialog osztály a meglévő receptek szerkesztésére szolgál. Az osztály a BaseRecipeDialog-ból örököl, és biztosítja a mezők szerkeszthetőségét.

Fontos megemlítések:

- updateRecipe:
  - A felhasználó által módosított mezők értékeit beállítja a meglévő Recipe objektumra
  - Frissíti a receptet a RecipeMethodsController segítségével, amely elmenti a változtatásokat
  - Értesítést küld a felhasználónak, ha a frissítés sikeres
- validateInput:
  - Ellenőrzi, hogy az összes mező helyesen van kitöltve, és az elkészítési idő számként értelmezhető

- isUpdated:
  - Egy boolean értéket ad vissza, amely jelzi, hogy történt-e módosítás a dialógusablakban. Ez a módosítás után megjelenő JTable miatt szükséges

RecipeDetailsDialog

A RecipeDetailsDialog osztály a receptek megtekintésére szolgáló dialógusablak. Ez az osztály szintén a BaseRecipeDialog-ból örököl, és nem teszi lehetővé a mezők szerkesztését.

Fontos megemlítések:

- createEditableField:
  - A mezők **editable = false** beállítással jönnek létre, ami biztosítja, hogy a felhasználó csak olvasni tudja az adatokat. Ez a funkció fontos, mivel ebben az osztályban nincs lehetőség a recept szerkesztésére.

Tesztelés

A programhoz átfogó tesztelési rendszert készítettem, amely különböző aspektusokat ellenőriz, beleértve az adatkezelést, fájlműveleteket, és a felhasználói felület bizonyos funkcióit is. A tesztek célja, hogy biztosítsák a program helyes működését normál és hibás körülmények között. A tesztelési környezetet JUnit 5 segítségével alakítottuk ki, és különös figyelmet fordítottunk az éles helyzetek szimulációjára.

Tesztelési környezet

A tesztek természetesen Java nyelven, JUnit 5 keretrendszerrel készültek. A tesztelés során a programhoz tartozó egyedi JSON fájlokat használunk, amelyeket minden teszt előtt inicializálunk, így biztosítva, hogy a tesztelés mindig ugyanazon kezdeti állapotból induljon. A tesztek automatikusan futtathatók az IDE integrált JUnit támogatásával.

Tesztelési kategóriák

Adatkezelési tesztek

RecipeMethodsController tesztelése:

- Tesztelésre kerülnek a receptek hozzáadásáért, törléséért, frissítéséért és betöltéséért felelős metódusok
- Teszteljük, hogy egy új JSON fájl helyesen jön létre, illetve hogy egy üres lista megfelelően betöltődik
- Kiemelt figyelmet kap az adatkezelési funkciók konzisztenciája, például az, hogy a módosított receptek valóban mentésre kerülnek

JSonServices tesztelése:

- Teszteljük a receptek JSON fájlba való mentését és onnan történő betöltését.
- Tesztelésre kerülnek a hibás JSON fájlok, amelyekkel a program várhatóan megfelelő hibakezelési eljárást hajt végre (felugró hibaüzenet jelenik meg a GUI-n keresztül)
- Például hibás JSON szerkezetű fájl betöltésekor a program üres receptlistát ad vissza, és erről értesíti a felhasználót

GUI tesztelés

DisplayFrame osztály tesztelése:

- Mivel a DisplayFrame osztály a program fő grafikus felületét biztosítja, a tesztelés során több funkcióját is ellenőrizzük:
  - Keresés név, hozzávalók és elkészítési idő alapján.
  - A keresés során a tesztek biztosítják, hogy csak a megfelelő eredmények jelenjenek meg.

Model osztályok tesztelése

- A Recipe osztály tesztjei a getterek, setterek és a toString metódus helyes működését ellenőrzik.
- Kiemelt teszt a hozzávalók listájának szöveggé alakítása, amely a receptek megjelenítése során fontos funkció.

Felmerülő felhasználói interakciók a tesztelés során

A GUI tesztelésének sajátosságai miatt bizonyos tesztek manuális interakciókat igényelnek:

- **Felugró ablakok**: Hibás JSON szerkezet vagy mentési hibák esetén a program egy GUI alapú figyelmeztetést jelenít meg. Ezeket a figyelmeztetéseket a tesztelési folyamat során le kell okézni.
- **Jóváhagyás szükségessége**: A felugró ablakok a program helyes működésének részét képezik, így a tesztelés során ezek figyelmen kívül hagyása nem lehetséges.

Ezt nagyon fontos figyelembe venni a tesztek futtatása során. Valóban egy elsőre megrémisztő hibaüzenetet kapunk, de ezt leokézva igenis egy sikeresen futott teszt fog fogadni minket, hisz ez a hibaüzenet az elvárt hibakezelésünk.

Teszt példák

1. **Keresési funkció tesztelése**
   - Teszteljük, hogy a performSearch metódus helyesen szűri ki azokat a recepteket, amelyek megfelelnek a megadott feltételeknek (pl. név, hozzávaló, idő)
   - A Reflection API lehetővé teszi a privát metódus elérését és tesztelését
2. **Hibás JSON fájlok kezelése**
   - A tesztek garantálják, hogy hibás JSON fájl betöltése nem okoz programösszeomlást. Ehelyett a program figyelmezteti a felhasználót, és üres listát ad vissza
3. **Adatkezelési műveletek tesztelése**
   - Egy új recept hozzáadása után a tesztek ellenőrzik, hogy a recept helyesen megjelenik az adatok között, és elmentésre is kerül
   - Egy recept frissítésekor a módosított adatok pontosan mentésre kerülnek
4. **Fájlrendszer-interakció tesztelése**
   - A JSON fájlok mentése és betöltésekor a tesztek biztosítják, hogy a fájlok ténylegesen létrejönnek, és helyesen töltődnek vissza

Hibakezelés

A programban kiemelt figyelmet fordítottam a hibák kezelésére, legyen szó akár felhasználói hibáról, akár rendszerhibáról. A hibakezelés célja, hogy megelőzze az alkalmazás összeomlását, és segítse a felhasználót a problémák felismerésében és megoldásában. Az alábbiakban részletezzük a program hibakezelési stratégiáit és módszereit.

Hibakezelés típusai és példák

Felhasználói hibák

- **Hibás vagy hiányos adatbevitel**:
  - Probléma: A felhasználó nem tölti ki az összes kötelező mezőt, vagy érvénytelen formátumú adatot ad meg (pl. szöveget ír be egy számot igénylő mezőbe)
  - Kezelés: A program hibaüzenetet jelenít meg egy felugró ablakban, amely pontosan jelzi, hogy mely mező hibás.
    - Példa: "Kérlek, minden mezőt helyesen tölts ki!" – ha a recept hozzáadása során valamelyik mező üres vagy érvénytelen
- **Nem választott recept**:
  - Probléma: A felhasználó szerkeszteni vagy törölni szeretne egy receptet anélkül, hogy kiválasztotta volna azt a listából
  - Kezelés: A program figyelmeztető ablakot jelenít meg, például: "Válassz egy receptet a szerkesztéshez!" vagy "Válassz egy receptet a törléshez!"

Fájlkezelési hibák

- **Hiányzó vagy nem elérhető JSON fájl**:
  - Probléma: A felhasználó egy nem létező vagy nem elérhető fájlt próbál betölteni.
  - Kezelés: Felugró hibaüzenet jelenik meg, például: "A fájl nem található: [fájlnév]" A program ilyenkor nem omlik össze, hanem üres listával folytatja a működést
- **Hibás JSON formátum**:
  - Probléma: A fájl tartalma nem felel meg a JSON formátumnak, vagy a JSON struktúrája nem kompatibilis a programmal
  - Kezelés:
    - A program hibaüzenetet jelenít meg: "Hibás JSON formátum: [hiba részletei]"
    - A hibás JSON betöltése helyett az alkalmazás üres listát hoz létre, hogy elkerülje az adatkezelési hibákat
    - A felhasználónak lehetősége van a fájl helyreállítására
- **Mentési hibák**:
  - Probléma: Az adatok mentése során probléma lép fel (pl. az adott hely nem írható)
  - Kezelés: A program hibaüzenetet jelenít meg, például: "Hiba történt a receptek mentésekor: [hiba részletei]"

Keresési hibák

- **Érvénytelen keresési feltétel**:
  - Probléma: A felhasználó olyan keresési értéket ad meg, amely nem felel meg a keresés típusának (pl. szöveg helyett számot keres név alapján)
  - Kezelés: A program a keresést üres eredménnyel zárja, és üzenetet jelenít meg: "Nincs találat"

JSON Fájlformátum

Az alkalmazás a receptek adatainak tárolására és betöltésére JSON fájlokat használ. Ahhoz, hogy a program helyesen tudja értelmezni a fájlokat, a JSON szerkezetének meg kell felelnie a következő formátumnak:

```
[
  {
    "nameOfFood": "Paprikás krumpli",
    "timeToMake": 40,
    "ingredients": ["Burgonya", "Paprika", "Hagyma"],
    "description": "Süssük meg a hagymát, adjuk hozzá a paprikát, majd a burgonyát. Főzzük puhára."
  },
  {
    "nameOfFood": "Gulyásleves",
    "timeToMake": 90,
    "ingredients": ["Hús", "Burgonya", "Sárgarépa", "Zeller", "Fűszerpaprika"],
    "description": "Főzzük meg a húst, adjuk hozzá a zöldségeket és a fűszereket. Főzzük puhára."
  }
]
```

Mezők magyarázata

1. **nameOfFood** *(String)*:
   - A recept neve
   - Példa: "Paprikás krumpli"
2. **timeToMake** *(Integer)*:
   - Az elkészítés ideje percben kifejezve.
   - Példa: 40
3. **ingredients** *(String tömb)*:
   - A hozzávalók listája, ahol minden hozzávaló egy külön szövegként van megadva
   - Példa: ["Burgonya", "Paprika", "Hagyma"]
4. **description** *(String)*:
   - Az étel elkészítési leírása, szöveges formában
   - Példa: "Süssük meg a hagymát, adjuk hozzá a paprikát, majd a burgonyát. Főzzük puhára."

Hibás szerkezetű vagy nem JSON formátumú fájl esetén a program figyelmeztető üzenetet jelenít meg, és a hibás fájl betöltése helyett az alkalmazás üres listával indul. Minden olyan JSON fájl ahol nem helyesek a mezőnevek, nem jók az adattípusok, a rendszer hibásként fog kezelni.

Továbbfejlesztési lehetőségek

- Képek tárolása recepteknél
- Felhőalapú mentés és szinkronizáció
- Felhasználók közötti receptmegosztás
- Recepértékelés, megjegyzés funkció
- Receptgyűjtemények exportálása más fájlformátumokba (pl. PDF)

Végszó

A 'The Java Cookbook' egy olyan alkalmazás, amely nemcsak a háztartások számára nyújt praktikus megoldást a receptek rendszerezésére, hanem alapot biztosít további funkciók és bővítések hozzáadására. Az intuitív felület és a robusztus adatkezelési rendszer garantálja, hogy a felhasználók könnyedén és megbízhatóan kezelhessék kedvenc receptjeiket.

A fejlesztés során számos kihívással szembesültem, kezdve a grafikus felhasználói felület megtervezésétől a JSON fájlok kezelésén át egészen a hibák hatékony kezeléséig. Bár a program nem tökéletes, és van még hova fejlődni, úgy érzem, hogy sikerült egy olyan alapot létrehoznom, amely jól tükrözi a célkitűzéseimet és a felhasználói igényeket.

Ez az alkalmazás nemcsak egy technikai projekt volt számomra, hanem egy személyes kihívás is, amely során sokat tanultam az adatkezelés, a programozás és a tervezés területén. A programot inspiráló gondolat – hogy édesanyám receptjeit könnyen rendszerezhesse – végig motivációt adott, hogy minél jobb és hasznosabb eszközt hozzak létre.

Bízom benne, hogy a 'The Java Cookbook' nemcsak nekem, hanem másoknak is hasznos eszköz lehet, és ha a saját anyukám a karácsonyi receptjei legalább felét itt tárolja idén, akkor már megérte elkészítenem.