Receptkönyv Fejlesztői dokumentáció

Alapanyag.java:

Ez az osztály tárolja a receptek alapanyagainak adatait, melyek az ételek elkészítéséhez szükségesek.

Adattagjai:

- nev - az alapanyag neve
- mertek - az alapanyag mértékegysége(pl. kg, dkg..)
- mennyiseg - az alapanyag mennyisége

Metódusai:

- konstruktor: Létrehozza az alapanyag objektumot és beállítja a kezdeti értékeit.

```
public Alapanyag(String nev, String mertek, int mennyiseg) {
    this.nev = nev;
    this.mertek = mertek;
    this.mennyiseg = mennyiseg;
}
```

- getterek és setterek: Setterek az adattagok értékeit módosítják, getterek pedig visszaadják az adattagok értékeit.

```
public String getNev() {
    return nev;
}

public void setNev(String nev) {
    this.nev = nev;
}
```

Etel.java:

Ez az osztály tárolja a receptek adatait.

ADATTAGJAI:

- nev - az étel neve
- tipus - az alapanyag típusa(Leves, Főétel, Sütemény)
- ido - az étel mennyi perc alatt készül el
- fo - az étel mennyi főre van
- elkeszites - az étel elkészítése

METÓDUSAI:

- konstruktor: Létrehozza az etel objektumot és beállítja a kezdeti értékeit.

```
public Etel(String nev, String tipus, int ido, int fo, String elkeszites) {
    this.nev = nev;
    this.tipus = tipus;
    this.ido = ido;
    this.fo = fo;
    this.elkeszites = elkeszites;
}
```

- getterek és setterek: Setterek az adattagok értékeit módosítják, getterek pedig visszaadják az adattagok értékeit.

```
public String getNev() {
    return nev;
}

public void setNev(String nev) {
    this.nev = nev;
}
```

DBConnection.java:

A lokális adatbázist összekapcsaló osztály

Ez az osztály kapcsolja össze a lokális adatbázist a programmal, és kezeli az sql parancsokat.

ADATTAGJAI:

- con - Connection, ami létrehozza a kapcsolatot
- st - Statement, az sql lekérdezések parancsait tárolja
- rs - ResultSet, sql lekérdezés után az eredmények ebbe a változóba kerülnek eltárolásra
- csat - boolean ami a konstruktorban kap egy true értéket ha a csatlakozott a db-hez, ha nem akkor false, ami a hiba kiírását eredményezi

METÓDUSAI:

- konstruktor: A db csatlakozásának konstruktora, itt csatlakozik a lokális adatbázishoz, aminek a neve: recept. Amennyiben a csatlakozás sikertelen hibát ír ki

```
public DbConnection() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/recept?useUnicode=true&characterEncoding=utf-8",
                "root", "");
        st = con.createStatement();
        csat = true;

    } catch (Exception e) {
        System.out.println("nem csat");
        csat = false;
    }
}
```

- addEtel(): Étel hozzáadása az adatbázisba, az etel táblába az INSERT INTO sql paranccsal. Hiba esetén hibaüzenetet ad és a folyamat nem megy végbe.
  paraméter: az adott ételhez szükséges adatok
  return: egy logikai érték amivel lelehet ellenőrizni, hogy sikeres volt-e az insert

```
public boolean addEtel(String nev, String tipus, int ido, int adag, String elkeszites) {
//INSERT INTO `etel` (`id`, `nev`, `tipus`, `ido`, `fo`, `elkeszites`) VALUES (NULL, 'Palacsinta', 'Főétel', '30',

    try {
        String str = "INSERT INTO `etel` (`id`, `nev`, `tipus`, `ido`, `fo`, `elkeszites`) VALUES "
                + "(NULL, '" + nev + "', '" + tipus + "', '" + ido + "', '" + adag + "', '" + elkeszites + "');";
        st.executeUpdate(str);
        return true;

    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

- addAlapanyag(): Alapanyag hozzáadása az adatbázisba, az alapanyag táblába az INSERT INTO sql paranccsal. Hiba esetén hibaüzenetet ad és a folyamat nem megy végbe.
  paraméter: az adott alapanyag neve
  return: egy logikai érték amivel lelehet ellenőrizni, hogy sikeres volt-e a query

```
public boolean addAlapanyag(String nev) {
//INSERT INTO `alapanyag` (`id`, `nev`) VALUES (NULL, 'liszt');

    try {
        String str = "INSERT INTO `alapanyag` (`id`, `nev`) VALUES (NULL, '"+ nev +"');";
        st.executeUpdate(str);
        return true;

    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

- addAlapanyagToEtel(): Ételhez alapanyag hozzáadása az adatbázisban a hozzadad kapcsolótáblába sql INSERT INTO paranccsal. Hiba esetén hibaüzenetet ad és a folyamat nem megy végbe.
  paraméter: az adott etel es alapanyag idje, plusz a további adatai(mennyiseg, mertekegyseg)
  return: egy logikai érték amivel lelehet ellenőrizni, hogy sikeres volt-e a query

```
public boolean addAlapanyagToEtel(int etelId, int alapanyagId, int mennyiseg, String mertekegyseg ) {
//INSERT INTO `hozzadad` (`etel.id`, `alapanyag.id`, `mennyiseg`, `mertekegyseg`) VALUES ('3', '18', '20', 'dkg');

    try {
        String str = "INSERT INTO `hozzadad` (`etel.id`, `alapanyag.id`, `mennyiseg`, `mertekegyseg`) VALUES "
                + "('"+ etelId +"', '"+ alapanyagId +"', '"+ mennyiseg +"', '"+ mertekegyseg +"');";
        st.executeUpdate(str);
        return true;

    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

- isAlapanyag(): Megnézi, hogy az adott alapanyag benne van már az adatbázisban SELECT sql lekérdezéssel. Hiba esetén hibaüzenetet ad és a folyamat nem megy végbe.
  paraméter: az adott alapanyag neve
  return: egy logikai érték amivel lelehet ellenőrizni, hogy sikeres volt-e a query

```
public boolean isAlapanyag(String nev) {
    try {
        String query = "SELECT * FROM `alapanyag` WHERE nev='" + nev + "';";
        //System.out.println(query + "\n");
        rs = st.executeQuery(query);
        while (rs.next()) {
            return true;
        } return false;

    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

- isEtel(): Megnézi, hogy van-e ilyen étel az adatbázisban SELECT sql lekérdezéssel. Hiba esetén hibaüzenetet ad és a folyamat nem megy végbe.
  paraméter: az adott alapanyag neve
  return: egy logikai érték amivel lelehet ellenőrizni, hogy sikeres volt-e a query

```
public boolean isEtel(String nev) {
    try {
        String query = "SELECT * FROM `etel` WHERE nev='" + nev + "';";
        //System.out.println(query + "\n");
        rs = st.executeQuery(query);
        while (rs.next()) {
            return true;
        } return false;

    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

- etellista(): Lekérdezi az összes ételt az adatbázisból és egy Arraylistben adja vissza SELECT sql lekérdezéssel. Hiba esetén hibaüzenetet ad és a folyamat nem megy végbe.
  return: ArrayList ami Etel tipusu, mely etel objektumokból áll.

```
public ArrayList<Etel> etellista() {

    try {
        ArrayList<Etel> etelek = new ArrayList<Etel>();
        String query = "SELECT * FROM `etel`;";
        rs = st.executeQuery(query);
        while (rs.next()) {
            String nev = rs.getString("nev");
            String tipus = rs.getString("tipus");
            int ido = rs.getInt("ido");
            int fo = rs.getInt("fo");
            String elkeszites = rs.getString("elkeszites");
            Etel etel = new Etel(nev, tipus, ido, fo, elkeszites);
            etelek.add(etel);
        }
        return etelek;

    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

Először lekéri az adatbázis elemeit és változókban eltárolja az adatokat külön-külön, majd ezeket az értékeket értékül adja az étel objektumoknak, melyeket hozzáad egy listához.

- getData(): Bármilyen étel vagy alapanyag adat lekérdezése SELECT sql lekérdezéssel. Hiba esetén hibaüzenetet ad és a folyamat nem megy végbe.
  paraméter: a tábla, étel/alapanyag neve és hogy mit akarunk lekérdezni a táblából
  return: a lekérdezett adat, ha hiba volt a lekérdezéssel akkor -1

```
public int getData(String tabla, String nev, String mit) {

    try {
        String query = "select " + mit + " from "+ tabla +" where nev='" + nev + "'";
        query = query.toLowerCase();
        rs = st.executeQuery(query);
        while (rs.next()) {
            int value = rs.getInt(mit);
            return value;
        }
        return -1;

    } catch (Exception e) {
        e.printStackTrace();
        return -1;
    }
}
```

- getEtelAlapanyag(): Egy adott étel alapanyagainak adatainak lekérdezése az alapanyag és a kapcsólótáblából, és ezt egy ArrayListben adja vissza SELECT sql lekérdezéssel. Hiba esetén hibaüzenetet ad és a folyamat nem megy végbe.
  paraméter: az adott étel neve
  return: egy Alapanyag tipusu ArrayList

```
public ArrayList<Alapanyag> getEtelAlapanyag(String nev) {
    //SELECT alapanyag.nev, hozzadad.mennyiseg, hozzadad.mertekegyseg FROM `alapanyag` INNER JOIN `hozzadad
    try {
        ArrayList<Alapanyag> alapanyagok = new ArrayList<Alapanyag>();
        String query = "SELECT alapanyag.nev, hozzadad.mennyiseg, hozzadad.mertekegyseg FROM `alapanyag` "
                + "INNER JOIN `hozzadad` ON alapanyag.id = `alapanyag.id` INNER JOIN `etel` "
                + "ON `etel.id` = etel.id WHERE etel.nev='"+ nev +"'";
        query = query.toLowerCase();
        rs = st.executeQuery(query);
        while (rs.next()) {
            String alapanyag = rs.getString("nev");
            int mennyiseg = rs.getInt("mennyiseg");
            String mertek = rs.getString("mertekegyseg");
            Alapanyag a = new Alapanyag(alapanyag,mertek,mennyiseg);
            alapanyagok.add(a);
        }
        return alapanyagok;

    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
```

Először lekéri az adatbázis elemeit és változókban eltárolja az adatokat külön-külön, majd ezeket az értékeket értékül adja az étel objektumoknak, melyeket hozzáad egy listához.

- torol(): Étel törlése az adatbázisból DELETE FROM sql lekérdezéssel. Hiba esetén hibaüzenetet ad és a folyamat nem megy végbe.
  paraméter: az adott étel neve
  return: egy logikai érték amivel lelehet ellenőrizni, hogy sikeres volt-e a query

```
public boolean torol(String nev) {
    //DELETE FROM `etel` WHERE `etel`.`nev` = nev
    try {
        PreparedStatement pst = con.prepareStatement("DELETE FROM `etel` WHERE `etel`.`nev` = '"+ nev +"'");
        pst.executeUpdate();
        return true;

    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

- updateAlapanyag(): Nem használt metódus

ReceptJFrame.java:

ADATTAGJAI:

- db - DbConnection, ami létrehozza a kapcsolatot
- alapanyagok - Alapanyag tipusu ArrayList, az Alapanyagok mentéséhez
- alapanyagokLista - String tipusu ArrayList, csak az Alapanyag nevét tartalmazza, egyszeri feltételhez szükséges(van-e már ilyen alapanyagnév)
- etelek - Etel típusú ArrayList, az Etelek mentéséhez

METÓDUSAI:

- Konstruktor: A Frame konstruktora, itt hozzuk létre a adatbázissal való csatlakozást
  ha nem sikerül a csatlakozás: a lista panelt láthatóvá tesszük, mindent lezárunk, és kiírjuk neki a hibát
  ha sikerül: példányosítunk, és láthatóva tesszük a lista panelt, és itt már kilistázzuk az ételeket
  etelek = db.etellista(): a DbConnection metódusa: lekérdezi az összes ételt és listában visszaadja

```
public ReceptFrame() {
    initComponents();
    db = new DbConnection();

    if(db.csat == false) {
        errorMessage("Nem csatlakozott az adatbázishoz!", 1);
        receptLista.setVisible(true);
        receptHozzaad.setVisible(false);
        recept.setVisible(false);
        receptHozzaadBtn.setEnabled(false);
        receptListaBtn.setEnabled(false);
        receptTipusBox.setEnabled(false);
        listazBtn.setEnabled(false);
        receptTabla.setEnabled(false);
    } else {
        alapanyagok = new ArrayList<Alapanyag>();
        alapanyagokLista = new ArrayList<String>();
        etelek = db.etellista();
        receptLista.setVisible(true);
        receptHozzaad.setVisible(false);
        recept.setVisible(false);
        kilistaz("Összes");
        alapanyagSzerkeszt.setEnabled(false);
    }
}
```

- frissit(): A recept hozzáadása panelnél, frissítjük az alapanyagok listáját.

```
private void frissit() {
    alapanyagArea.setText("");
    for(int i = 0; i < alapanyagok.size(); i++) {
        Alapanyag a = alapanyagok.get(i);
      //pl: ablakmosó: 2 l
        alapanyagArea.append(a.getNev() + ": " + a.getMennyiseg() + " " + a.getMertek() + "\n");
    }
}
```

- errorMessage(): Egy felugró ablak a user tájékoztatásához JOptionPane segítségével
  paraméter: az adott hiba amit kiír és mennyi idő alatt

```
private void errorMessage(String error, int second) {
    Timer timer = new Timer();
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
              JOptionPane.showMessageDialog(null,error);
            }
          }, second*1000);
}
```

- isNumeric(): Megnézi hogy egy kapott String átváltás után numerikus-e
  paraméter: egy String
  return: egy logikai érték amivel lelehet ellenőrizni, hogy numerikus-e a String

```
private boolean isNumeric(String str) {
    try {
        Integer.parseInt(str);
        return true;
    } catch(NumberFormatException e){
      return false;
    }
}
```

- kilistaz(): A táblába belerakja az ételek tulajdonságait
  paraméter: milyen típus alapján listázzon

```
etelek = db.etellista();
int sorokSzama = etelek.size();
Object data[][] = new Object[sorokSzama][3];
```

Létrehozza a data 2 dimenziós objektumot melynek sorai az etel tábla sorainak száma és az oszlopai pedig 3

```
if(mit == "Összes") {
    for(int i = 0; i < etelek.size(); i++) {
        Etel etel = etelek.get(i);
        data[i][0] = etel.getNev();
        data[i][1] = etel.getIdo();
        data[i][2] = etel.getFo();
    }
    receptTabla.setModel(new javax.swing.table.DefaultTableModel(data,new String[] { "Név", "Idő(perc)", "fő(adag)" }));
```

ha Összes: mindent kilistáz

```
} else {
    int sor = 0;
    for(int i = 0; i < etelek.size(); i++) {
        Etel etel = etelek.get(i);
        if(etel.getTipus().equals(mit)) {
            data[sor][0] = etel.getNev();
            data[sor][1] = etel.getIdo();
            data[sor][2] = etel.getFo();
            sor++;
        }
    }
    receptTabla.setModel(new javax.swing.table.DefaultTableModel(data,new String[] { "Név", "Idő(perc)", "fő(adag)" }));
}
```

Különben csak a kiválasztott adatot

- receptHozzaadBtnActionPerformed(): A receptHozzaad panelhez visz

```
private void receptHozzaadBtnActionPerformed(java.awt.event.ActionEvent evt) {
    receptLista.setVisible(false);
    receptHozzaad.setVisible(true);
    recept.setVisible(false);
}
```

- receptListaBtnActionPerformed(): A recptek listájához visz, és alapbol kilistázza az összes receptet

```
private void receptListaBtnActionPerformed(java.awt.event.ActionEvent evt) {
    receptLista.setVisible(true);
    receptHozzaad.setVisible(false);
    recept.setVisible(false);
    kilistaz("Összes");
}
```

- alapanyagHozzaadActionPerformed(): Alapanyag hozzáadása a TextArea-ba
  hibák kezelése:
  Ha nincs kitöltve az összes mező

```
if(alapanyagNevField.getText().length() > 0 && mennyisegField.getText().length() > 0) {
```

Ha hibás adato(ka)t adnak meg:

```
if(isNumeric(mennyisegField.getText()) && !isNumeric(alapanyagNevField.getText())) {
```

Ha az alapanyag már benne van a listában:

```
if(!alapanyagokLista.contains(alapanyagNev)) {
```

Ha az alapanyag már benne van az adatbázisban:

```
if(db.isAlapanyag(alapanyagNev)) {
```

metódus menete:

1. az alapanyag nevét kisbetűvel, és a mennyiseget, merteket is elmentjük
2. megnézi, hogy nincs-e benne a TextAreaba az alapanyag neve, ha nincs akkor belerakja (nem lehet egyszerre több ugyanolyan alapnyagnév)!
3. csinál egy alapanyagot, majd hozzadja a listához, és frissíti a TextAreát
4. a lenyíló listába is belementi az alapanyagot, hogy tudjuk majd szerkeszteni
5. Az adatbázisba elmenti az alapanyagot a neve alapján
6. Az inputokat kinullázuk

- alapanyagSzerkesztActionPerformed(): Alapanyag szerkesztése az alapanyagok indexe alapján
  metódus menete:
  1. az alapanyag lenyíló indexét elmentjük
  2. ha nincs név megadva, akkor törli az alapanyagot, a legelső index az 'Alapanyagok', ezért kikell vonni 1-et
  3. Különben: az Alapanyag adatai alapján feltöltjük az input mezőket, csak ekkor szerkeszthetünk a gombbal

- alapanyagBoxActionPerformed(): Az alapanyag nevének kiválasztása a szerkesztéshez
  metódus menete:
  1. az első elem kijelölésekor a mezők alapállapotba kerülnek
  2. különben: csinál egy Alapanyag objektumot az index alapján, és hozzárendel egy mértékegységet is.
  3. kitölti a mezőket az alapanyag adataival

- receptAddActionPerformed(): Recept hozzáadása
  metódus menete:
  1. Hibakezelés: ellenőrzi, hogy minden ki lett-e töltve, az adatok helyesek-e, van-e ilyen étel az adatbázisban
  2. Az Etel adatainak mentése változókba
  3. megnézi hogy van-e már ilyen nevű étel az adatbázisban, ha nem akkor hozzáadja
  4. feltölti a kapcsolótáblát
  5. mezőket kinullázzuk
  6. az adott étel alapanyagainak elemeit töröljük a listákból

- listazBtnActionPerformed(): Listázás a lenyíló lista alapján

```
private void listazBtnActionPerformed(java.awt.event.ActionEvent evt) {
    String tipus = (String)receptTipusBox.getSelectedItem();
    kilistaz(tipus);
}
```

- receptTablaMouseClicked(): A kiválasztott recept adatait jeleníti meg a táblából egy új panelban
  metódus menete:
  1. index változóba menti a kiválasztott sor számát
  2. lekéri a modelt, és a nevet(első oslop) menti el
  3. Hibakezelés: ki lett-e választva recept
  4. végigmegy az ételek listán, a egyezik a kiválasztott név, kiírja az adatokat
  5. az étel alapanyagainak adatait is megszerzi és ezt is kiirja
  6. megjeleníti a recept panelt a kivalasztott receptet az adatokkal

- visszaBtnActionPerformed(): A recept panelból vissza “lép” a receptek listájához, a láthatóságok beállításával

```
private void visszaBtnActionPerformed(java.awt.event.ActionEvent evt) {
    receptLista.setVisible(true);
    receptHozzaad.setVisible(false);
    recept.setVisible(false);
    alapanyagokArea.setText("");
    ennyiFore.setText("");
}
```

- torolBtnActionPerformed(): A recept törlése
  ha 0000 amit beírtunk akkor tudja törölni az adatbázisból a receptet.

```
private void torolBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // ha 0000 amit beirtunk akkor tudja torolni az adatbázisbol
    if(pwField.getText().equals("0000")) {
        if(db.torol(receptNev.getText())) {
            errorMessage("Sikeresen törölve: " + receptNev.getText(),0);
            receptLista.setVisible(true);
            receptHozzaad.setVisible(false);
            recept.setVisible(false);
            kilistaz("Összes");
            ennyiFore.setText("");
        }
    }
    pwField.setText("");
}
```

- jButton1ActionPerformed(): Kiszámolja mennyi főre mennyi alapanyag kell
  Metódus menete:
  1. hibakezelés: megfelelően vannak-e a mező̈l kitöltve, 1-100 közötti létszám van-e megadva
  2. eltárolja egy változóban a létszámot
  3. kerekít
  4. alapanyagok listat az adott etelhez lekeri
  5. az uj mennyiséget kiszámolja, és utánna kiirja a TextArea-ba
  6. kiirja a további adatait
  7. felugró ablakban kiírja az eredményt