

## Programozás alapjai 3 Házi feladat dokumentáció

### Fitness Tracker

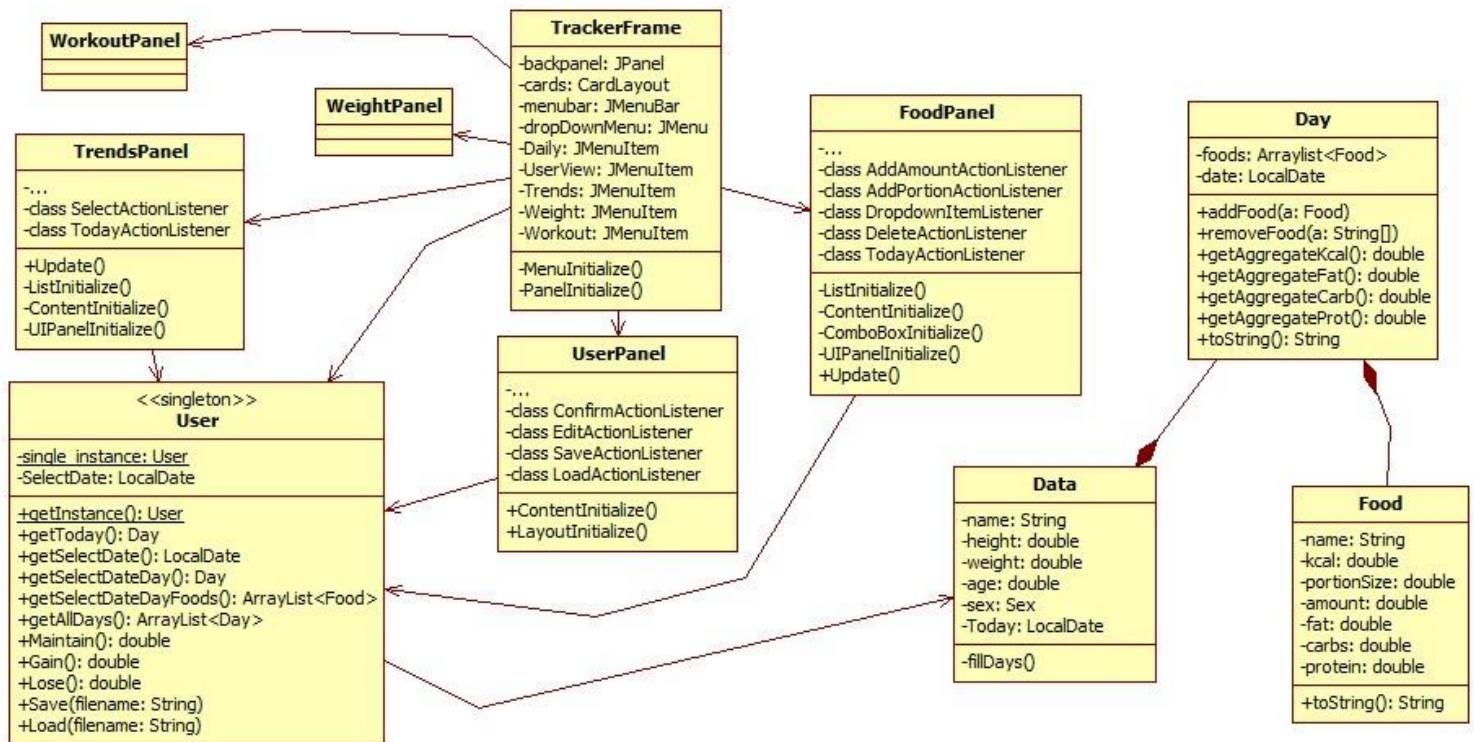
2020/2021 I. félév

Készítette: Tóth Gergely György, GGI1K7

Rövid ismertetés: Az elkészült program egy Fitness Tracker alkalmazás, melyen megtekinthető a felhasználó által felvitt adatokból a napi kalóriaigénye és az elfogyasztott ételek naplózhatóak.

- i. Osztálydiagram
- ii. Adatszerkezetek, osztályok mire valók és miért
- iii. Osztályok részletes leírása minden attribútumra és metódusra
- iv. User Manual
- v. Use Case-k leírása
- vi. Be- és kimenetek (egér, textboxok, profilmentés, listaolvasás)

## Osztálydiagram



Magyarázat: nincs feltüntetve a következő:

-getter, setter függvények

-TrendsPanel, UserPanel, FoodPanel grafikus megjelenítő attribútumai nagy számuk miatt

-WorkoutPanel, WeightPanel tartalma. Ezek a specifikációban említettek, de egyeztetés után az implementációjuktól eltértem. Jelenleg egy placeholder szerepben benne vannak a kész programban, ám csak saját maguk megjelenítését végzik. Bővebb részletezésüket így kihagyom a dokumentációból

## Adatszerkezetek

### User

A felhasználó osztálya, ez az, amit a konzisztencia miatt minden más osztály modifikál.

Mivel sok helyen van rá szükség, és minden helyen ugyan az az instance kell, hogy módosítva legyen, ezért singleton. Ez azt váltotta ki, hogy minden egyes panelba, minden panel minden függvényébe mint argumentum meg kelljen adni.

Maga a User class attribútumot csak három dolgot tárol: a kiválasztott dátumot, az egyetlen instance-ét a singleton-ság miatt, és egy **Data** belső classt, ami a valódi adattárolója. Erre azért van szükség, mert a felhasználó állapotát menteni és betölteni kell a programban, ám az előbb említett singleton tulajdonság a beolvasáskori új instance-ek létrehozásával nincs összhangban. Így minden panel szabadon módosíthatja, töltheti és mentheti a User-t, az úgymint saját magán, hanem ezen a belső osztályon dolgozik.

Fő feladatai az előbb említett panelek közti konzisztencia biztosítása egyetlen instance-ével, a bevitt adatok felvitele a Data-ba, kommunikáció biztosítása a Panelek és Data között, Data mentése és betöltése

### Data

A Data class egy serializable osztály, melynek fő feladata a User (itt mint felhasználó és mint User class értelemben is) által adott információk tárolása. Ezt a személyes információk mellett egy Day objektumokat tartalmazó ArrayList-el teszi. A személyes információk mind double típusúak, mert a napi kalóriamennyiségek kiszámításának úgymint double értékek a szorzótényezők.

### Day

Egy nap reprezentációja, tárolja a saját dátumát, és egy listát az aznap elfogyasztott ételekről. Főbb feladatai közé tartozik, hogy a napi összesítést elkészítse az ételek értékeiről. Így az adatszerkezet egy dinamikusan növekvő napok listája, melyek önmagukban dinamikusan növekvő ételek listáját tárolják, mint ahogy azt egy kvázi-adatbázistól elvárható.

### Food

Tárol egy ételnevet, egy súlyt(amount) és az ehhez a súlyhoz kapcsolódó makrókat az ételről, double formátumban. Logikus úgy gondolni, hogy nem csak egész gramm-nyi ételt lehet enni. Bonyolult dolgokat nem kell csinálnia.

### TrackerFrame

A legfőbb megjelenítő osztály, közrefogja a többi panelt és a User-t, létrehozza a felső sor menüjét, és inicializálja az ablak nagyobb részét kitevő paneleket. Felelős a panelváltásokért, de az egyéb műveleteket meghagyja a paneleknek.

CardLayoutot használ, mely több egymásba ágyazott panel megjelenítéséért és köztük való váltásért felel.

### UserPanel

A felhasználó által elsőnek meglátott ablakelem, mely kéri a nevét, súlyát, stb. Egyfajta regisztrációs oldal, mely menteni és betölteni tudja a felhasználó profilokat. Az elrendezését GroupLayout végzi,

mely kicsit hosszadalmas kódhoz vezet, de a legátláthatóbb UI felületet lehet vele létrehozni nagy számú grafikai elem esetén. Ezért ezt alkalmazom a további paneleken is. Minden Panelen továbbá bármilyen felhasználó okozta változás után újra kirajzoltatom az adatokat.

### **FoodPanel**

A legbonyolultabb panel, ez felel a kézzel megadható ételek beviteléért. Két részből áll a panel, melyek további belső panelek. A felső rész egy Lista, ami az adott napi ételek írja ki, az alsó rész a vezérlőpanel, ahol megjelennek a felhasználó adatai alapján a kalóriamennyiségek, és legördülő listán kiválasztható a megadni kívánt étel. Ehhez egy listába be kell olvasni egy előre megadott ételekkel teli txt file-t, emberi szemmel olvasható formátumban, bővíthetőség miatt.

### **TrendsPanel**

A legutolsó panel, melynek listája a napokat jeleníti meg, az előző ételeivel ellentétben, így mindkét dinamikus adattárolási szintre van megjelenítő. Itt fő funkcióként a több napi összesítéseket nézhetjük meg, és kiválaszthatunk Select Date-et, mely megjelenik a FoodPanel-en. Mind itt, mind a FoodPanelen bármikor át lehet ugrani az adott (mai) napra.

.

## Osztályok részletes leírása

### User

**private class Data implements Serializable**

A belső adat osztály, mely megvalósítja a menthető tartalmát a User classnak

**public Data data**

Public, mert a serializációhoz ez kell. Az adat osztály objektuma

**private static User single\_instance**

A belső változó ami az egyetlen létező User-t tárolja

**private LocalDate SelectDate=LocalDate.now()**

A SelectDate változó határozza meg, melyik nap van aktívan kiírva FoodPanelen és Trends Panelen mint Date

**private User()**

Privát konstruktor, hogy singleton legyen

**public static User getInstance()**

Konstruktor helyett ezt hívják a panelek

**public String getName()**

A data osztályból kiveszi és hívónak visszaadja a nevéből adódó attribútumot

**public void setName(String a)**

A data osztály megfelelő értékét beállítja

**public double getHeight()**

A data osztályból kiveszi és hívónak visszaadja a nevéből adódó attribútumot

**public void setHeight(double a)**

A data osztály megfelelő értékét beállítja

**public Sex getSex()**

A data osztályból kiveszi és hívónak visszaadja a nevéből adódó attribútumot

**public void setSex(Sex a)**

A data osztály megfelelő értékét beállítja

**public double getAge()**

A data osztályból kiveszi és hívónak visszaadja a nevéből adódó attribútumot

**public void setAge(double a)**

A data osztály megfelelő értékét beállítja

**public double getWeight()**

A data osztályból kiveszi és hívónak visszaadja a nevéből adódó attribútumot

**public void setWeight(double a)**

A data osztály megfelelő értékét beállítja

**public Day getToday()**

A Data-ból kiveszi a legutolsó napot a napok listájából, ugyanis ez a mai nap

**public LocalDate getSelectDate()**

Visszatér a kiválasztott dátummal

**public void setSelectDate(String a)**

Beállítja a kiválasztott dátumot a stringből beolvasva.

**public ArrayList<Food> getSelectDateFoods()**

Visszatér a Data-ból a Select Date-nek megfelelő napjának foods listájával

**public Day getSelectDateDay()**

Visszatér a Data-nak a megfelelő dátumú napjával

**public ArrayList<Day> getAllDays()**

A data osztályból kiveszi és hívónak visszaadja a nevéből adódó attribútumot

**public double Maintain()**

Egy Mifflin-St. Jeor egyenlet alapján kiszámolja a felhasználó napi kalóriaigényét

**public double Lose()**

A Maintain-ből származtatva kiszámolja a fogyáshoz szükséges napi kalóriát

**public double Gain()**

Maintain-ből származtatva kiszámolja a hízáshoz szükséges napi kalóriát

**public void Save(String filename)**

A megadott névre elmenti a Data szegmenst. ProfileX.txt formátumban vannak megadva ahol hívódik

**public void Load(String filename)**

Az előzőnek a megfelelője, a ProfileX.txt-ből visszatölti a Data-t, ahol X 1,2,3 lehet

## Data

```
private static final long serialVersionUID = 1L
```

Serializáció miatt

```
private String name
```

```
private double height
```

```
private double age
```

```
private double weight
```

A felhasználó testi értékei double típussal, mert a fél kg-ok, stb létező és valid értékek, továbbá a számítási egyenletben ahol szükség van rájuk double értékekkel vannak szorozva. Age évet jelöl, height centimétert, weight Kilogrammot

```
private Sex sex
```

enum {male,female} típusú, akár boolean is lehetne, a felhasználó biológiai nemét tárolja

```
private ArrayList<Day> days
```

A napok dinamikusan bővülő listája

```
private LocalDate Today
```

A mai nap dátuma

```
private void fillDays()
```

A bekapcsoláskor, betöltéskor feltölti üres napokkal a napok listáját, hogy a legutóbbi feljegyzés óta ne legyen feljegyezhetetlen nap(mivel nap hozzáadása funkció nincs, ám jellemző ha valaki elfelejti vezetni a kalória naplóját)

```
private Data()
```

Konstruktor, melyet csak a User hív ezért privát

## Day

```
private static final long serialVersionUID = 363266L;
    Serializáció miatt
private ArrayList<Food> foods
    Dinamikusan növekvő avagy csökkenő lista a felhasználó bemenetei
    alapján
private LocalDate date
    Az adott naphoz tartozó dátum, elég csak napra pontosan, sőt
    kifejezetten problémás lenne attól pontosabb időt tárolni

public Day()
    Konstruktor
public Day(String a)
    A Data osztály konstruktor ezt hívja meg egy előre megadott
    kezdődátummal
public Day(LocalDate prevDay)
    A Data osztály fillDays() metódusa ezt használja újabb napok
    készítésére amíg fel nem tölti a mai napig a listáját
public LocalDate getDate()
    A dátum lekérése
public void addFood(Food a)
    Étel hozzáadása a listához
public void removeFood(String[] a)
    Kételemű string array alapján megkeresi a név+amount kombináció
    alapján az ételek listájában a kitörölendőt. Azért kell így, mert a
    Daily view listája szövegesen adja vissza a listaelemeit.
public ArrayList<Food> getFoods()
    Az ételek listáját adja vissza
public double getAggregateKcal()
public double getAggregateFat()
public double getAggregateCarb()
public double getAggregateProt()
    A nevükben lévő étel attribútumot összegzik a foods listában
    szereplőkből
public String toString()
    A Trends panel ez alapján írja ki a listájára
```



## Food

```
private static final long serialVersionUID = 393812L
```

Szerializáció miatt

```
private String name
```

```
private double kcal
```

```
private double portionSize
```

```
private double amount
```

```
private double fat
```

```
private double carbs
```

```
private double protein
```

Szorzásban való részvétel, és való élet alapján double értékek

```
public String getName()
```

```
public double getKcal()
```

```
public double getPortionSize()
```

```
public double getAmount()
```

```
public double getFat()
```

```
public double getCarbs()
```

```
public double getProtein()
```

A nevében szereplő attribútumot adja vissza

```
public String toString()
```

A Daily view/FoodPanel ez alapján írja a ki az adott napra felvitt ételeket

```
public Food(String name,double kcal,double portionSize,double amount,double fat,double carbs,double protein)
```

Egy elsőre komplikált konstruktor, mely főleg a FoodPanel legördülő listája miatt van. Így egyszerűen megadható egyetlen függvényben az összes értéke, nem kell mindhez settert hívni

## TrackerFrame

```
private User UserInfo=User.getInstance()
```

Konstruktor helyett így szerzik be a panelek a User példányát

```
private JPanel backpanel
```

Az alap panel melyre az 5 másik fel van véve

```
private CardLayout cards
```

A CardLayout teszi lehetővé a panelek közti váltásokat

```
private JPanel userpanel
```

```
private JPanel foodpanel
```

```
private JPanel workoutpanel
```

```
private JPanel weightpanel
```

```
private JPanel trendspanel
```

A nevüknek megfelelő panelek

```
private JMenuBar menubar;
```

A Views menüt tároló felső „csík”

```
private JMenu dropDownMenu
```

A Views menü maga

```
private JMenuItem UserView
```

```
private JMenuItem Daily
```

```
private JMenuItem Workout
```

```
private JMenuItem Weight
```

```
private JMenuItem Trends
```

A Views menü elemei

```
private void MenuInitialize()
```

A felső Views menüt létrehozza

```
private class MenuButtonActionListener implements ActionListener
```

A Views menüben levő kattintásért felelős, hogy a megfelelő panelra váltson az ablak

```
private void PanelInitialize()
```

Az 5 panelt létrehozza és a cardlayout-ba integrálja

```
public TrackerFrame()
```

Konstruktor, ami meghívja a felette lévő két függvényt, melyek tovább hívják saját tartalmaikra konstruktoraikat. Gyakorlatilag a Main egyetlen sora ennek a konstruktornak a hívása.

## UserPanel

```
private User UserInfo
```

A User-t tároló mutató

```
private JTextField NameInput  
AgeInput  
WeightInput  
HeightInput
```

A nevüknek megfelelő értékek írandók beléjük, belőlük lesz beolvasva a user data-ba

```
private JComboBox SexInput
```

Egy legördülő menü a nem kiválasztására

```
private JLabel NameLabel  
AgeLabel  
WeightLabel  
HeightLabel  
SexLabel
```

A nevüknek megfelelő címkék az inputokat jelezve

```
private JLabel ProfileLabels[]=new JLabel[3]
```

A Profile 1,2,3 címkék a mentés, betöltés gomboknál

```
private JButton Confirm  
Edit
```

A véglegesítés és megváltoztatás gombok a felhasználói felületen

```
private JButton Saves[]=new JButton[3]
```

```
private JButton Loads[]=new JButton[3]
```

A tömbökbe rendezett mentés és betöltés gombok mind ugyan azt a feladatot játják el, tömbbe vannak rendezve, mert működésük egyetlen karakterrel tér el (hogya milyen számú profilt mentsen/töltsön)

```
private void ContentInitialize()
```

A fentebbi rengeteg grafikus elemet inicializálja

```
private void LayoutInitialize()
```

A grafikus elemek elhelyezésért felel

```
private class ConfirmActionListener implements ActionListener
```

A Confirm gomb megnyomásakor a felhasználói adatokat beállítja

```
private class EditActionListener implements ActionListener
```

A már létező felhasználói beállításokat ismét írhatóvá teszi

```
private class SaveActionListener implements ActionListener
```

Mentés gomb alapján megfelelő fileba menti a Data class

```
private class LoadActionListener implements ActionListener
```

A megfelelő mentett fileból visszatölti a Data class értékeit

```
public UserPanel()
```

Konstruktor, amely létrehozza a panelt az inicializáló függvényeivel

## FoodPanel

**private User `UserInfo`**

A User-t tároló mutató

**private ArrayList<Food> `FoodList`**

A fileből olvasott lista, ami a legördülő menübe kerül

**private JPanel `ListPanel`**

A felső fele a panelnek, mely a napi ételek listáját mutatja

**private JPanel `UIPanel`**

Az alsó fele a panelnek, mely a felhasználói felületet, gombokat mutatja

**private JScrollPane `scrollpane`**

A napi feljegyzett ételeket egy tekerhető keretbe zárja

**private JList `list`**

A lista ami a tekerhető keretbe kerül

**private JButton `Today`**

A mai napra változtatja a SelectDate-et

**private JLabel `DateLabel`**

„Date” szó kiírása

**private JTextField `DateOutput`**

Select Date dátumának helye

**private JButton `AddAmount`**

**`AddPortion`**

**`Delete`**

A három fő működtető gomb ezen a panelen. A nevüknek megfelelő működésű ActionListener-t tartalmaznak

**private JTextField `kcalOutput`**

**`fatOutput`**

**`carbsOutput`**

**`protOutput`**

**`maintain`**

**`gain`**

**`lose`**

A panel bal oldalának kimeneti textfieldjei, az első 4 az aznapi összes étel összesített értékeit írja ki, a második három a felhasználói testi adatok alapján számolt súlycsökkenés, tartás és fogyás kalórikus értékei

**private JTextField `kcalInput`**

**`fatInput`**

**`carbsInput`**

**`protInput`**

**`amountInput`**

**`portionInput`**

**`portionSize`**

**`NameInput`**

A panel jobb oldalának textfieldjei, melyekre a legördülő menü ír, majd a megfelelő Add... gomb megnyomásakor a gomb ActionListener-e beolvas és belőle új ételt ad a felhasználó napjához

```
private JLabel    DailyLabel
                  kcalLabel
                  fatLabel
                  carbsLabel
                  protLabel
                  maintainLabel
                  gainLabel
                  loseLabel
                  CustomNameLabel
```

A panelen megtalálható címkék, mind a nevükkel megegyező, nevüknek megfelelő szöveggel rendelkeznek

```
private JComboBox FoodDropDown
    A legördülő menü
```

```
private DefaultListModel listModel
    A felhasználó által felvitt ételek listáját vezérli
```

```
private void ListInitialize()
    A panel felső részének inicializálása
```

```
private void ContentInitialize()
    Minden combobox, textfield, label, stb inicializálása
```

```
private void ComboBoxInitialize()
    DropdownMenut feltölti
```

```
private void UIPanelInitialize()
    Inicializálta a panel alsó felét
```

```
public void Update()
    Újra kirajzolja a panelt az új értékekkel
```

```
private class TodayActionListener implements ActionListener
    A Selected Date-et a mai napra állítja
```

```
Private class AddAmountActionListener implements ActionListener
    Az Add Amount gomb működéséért felel, az input mezőkből beolvassa új ételt ad a listához
```

```
private class AddPortionActionListener implements ActionListener
    A beírd szám*porció métere mennyiségű ételt ad hozzá a listához
```

```
private class DropdownItemListener implements ItemListener
    A legördülő menüből való választást viszi fel az input mezőkre
```

```
private class DeleteActionListener implements ActionListener
    Kitörli a jelenleg kiválasztott sort a listából
```

```
public FoodPanel()
    Konstruktor
```

## TrendsPanel

**private User UserInfo**

A user-t tároló mutató

**private JScrollPane scrollpane**

**private JList list**

Mint a FoodPanelnél, a gördülő lista elemei

**private JButton Today**

Működése egyezik a FoodPanelen találhatóval

**private JButton Select**

A listából kiválasztott dátumot állítja be Select Date-nek

**private JLabel DateLabel**

A „Date” címke

**private JTextField DateOutput**

A Select Date kimenete

**private JPanel ListPanel**

**private JPanel UIPanel**

FoodPanellel megegyzik e két panel működése

**private DefaultListModel listModel**

A listát vezérli

**private void ListInitialize()**

A lista inicializálása

**private void ContentInitialize()**

A grafikus elemek inicializálása

**private void UIPanelInitialize()**

A UIPanel elemeinek inicializálása

**public void Update()**

A felület újra kirajzolása felhasználói esemény után

**private class SelectActionListener** implements ActionListener

A Select date gomb megnyomásakor a Select Date értékét beállítja a kiválasztott listaelemre, ha van ilyen

**private class TodayActionListener** implements ActionListener

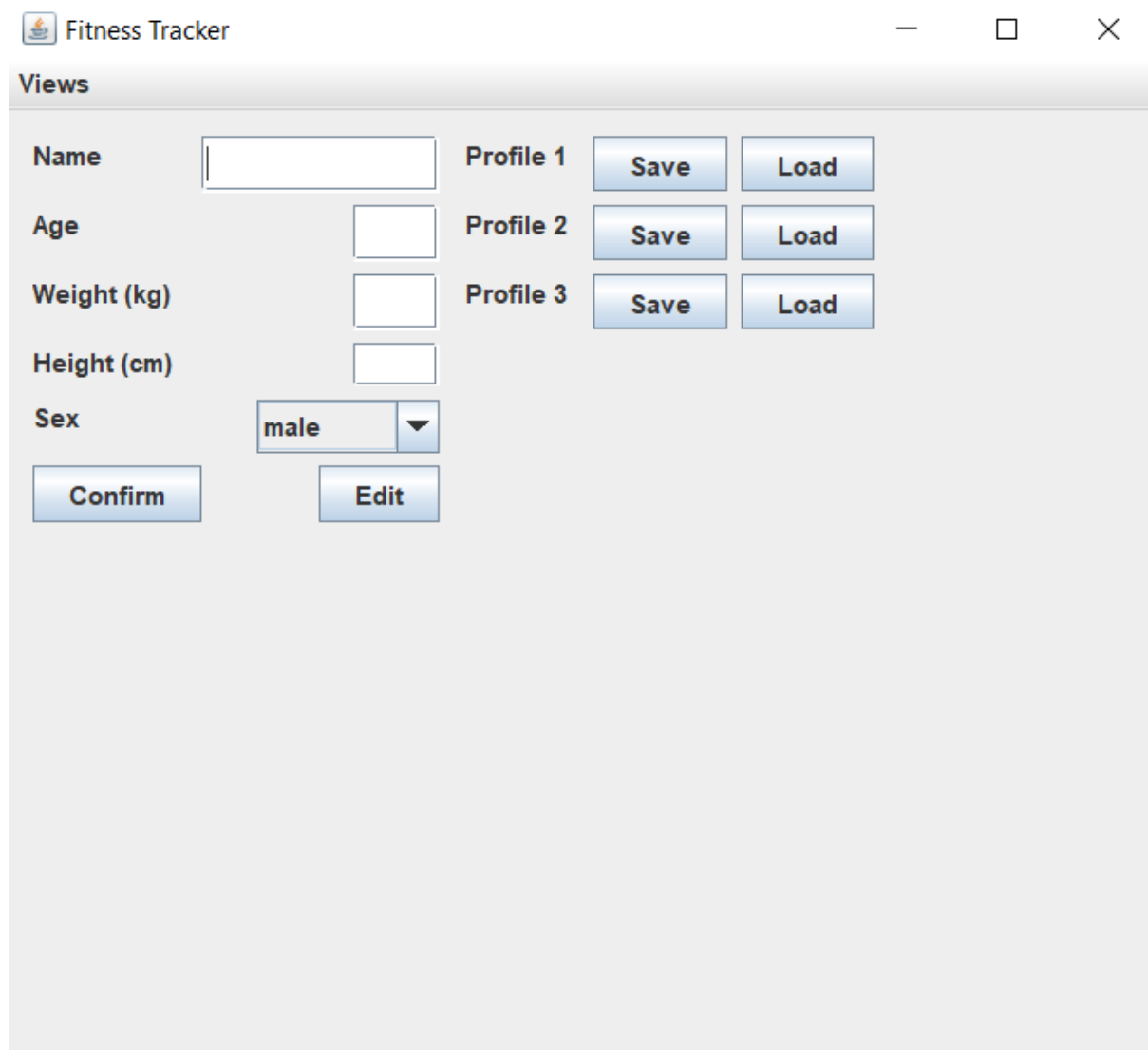
Today gomb megnyomásakor a Select Date-t a mai napra állítja át

**public TrendsPanel()**

Konstruktor, mely létrehozza a panelt. Egyéb függvényét nem kell meghívni a működéshez

## User Manual

## 1. Felhasználó személyes adatai



The screenshot shows a window titled "Fitness Tracker" with standard window controls (minimize, maximize, close). Below the title bar is a "Views" tab. The main content area contains a form for user profile data. On the left, there are labels for "Name", "Age", "Weight (kg)", "Height (cm)", and "Sex". Each label is followed by an input field: a text box for Name, and numeric boxes for Age, Weight, and Height. The Sex field is a dropdown menu currently showing "male". To the right of these fields are three profile sections, each with a label ("Profile 1", "Profile 2", "Profile 3") and two buttons ("Save" and "Load"). At the bottom left of the form are two buttons: "Confirm" and "Edit".

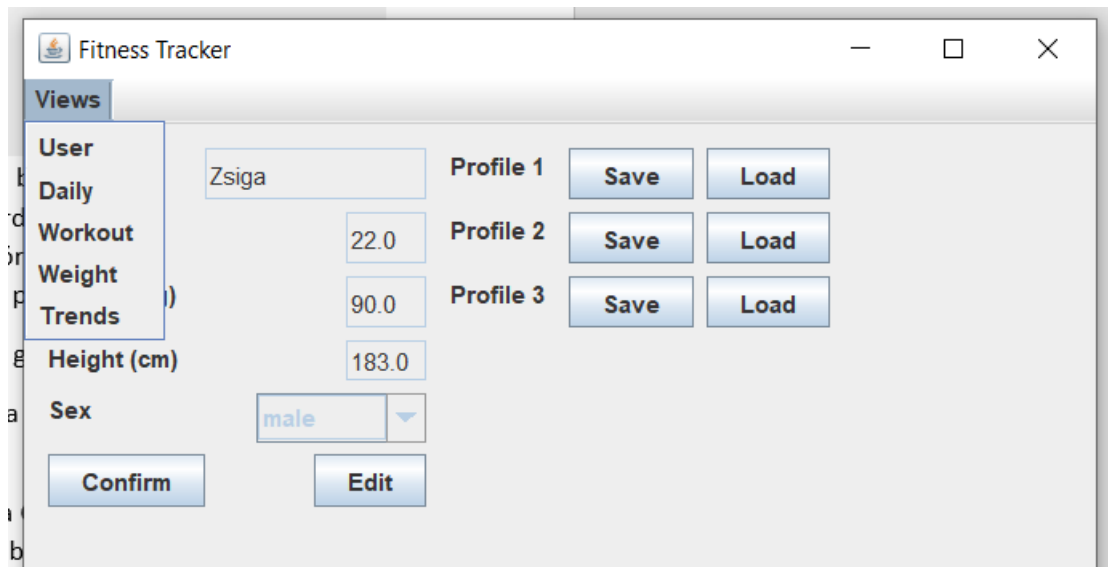
Az első látható oldal ami fogadja a felhasználót. A szabad textboxokba belekattintva lehet beléjük írni, névnek bármit, alatta pedig csak számokat lehet megadni. A legördülő menü a felhasználó biológiai nemét állítja be. Ezek az adatok mind szükségesek a napi kalóriaszükséglet kiszámításához, ezért amíg nincsenek megadva, majd confirm gombbal elfogadva más panel nem jeleníthető meg.

Korábban elfogadott profil is betölthető a megfelelő sorban levő Load gombbal.

Elfogadott profil továbbá bármikor menthető a Save gombbal, felülírva az arra a gombra korábban mentett profilt.

Elfogadott profil bármikor editálható, az Edit gombra kattintva, majd a Confirm gombbal véglegesítve a változtatást. Edig gomb egymás utáni kétszeri megnyomása a legutóbbi Confirm vagy Load művelet utáni állapotra reseteli a kijelzett értékeket.

## 2. Új panel választása



A profil létrehozása után a Views menüt lenyitva választhatunk 5 különböző panel közül (A Workout és Weight csupán esztétikai értékkel bír, korábban jelezve ezeknek nincs mélyebb implementációjuk)



## 3. Daily Panel

**Fitness Tracker**

**Views**

Egg, 200.0g, kcal:136.0, fat:20.0, carbs:1.2, protein:21.6  
 Egg, 72.0g, kcal:81.6, fat:12.0, carbs:0.72, protein:12.96  
 Egg, 100.0g, kcal:68.0, fat:10.0, carbs:0.6, protein:10.8  
 Hamburger, 120.0g, kcal:352.8, fat:56.4, carbs:28.8, protein:20.4

**Date** 2020-12-04 **Today**

**Daily** White bread, 100.0g, kcal:261.0.. ▾

**kcal** 638.4

**fat** 98.4

**carbs** 31.32

**protein** 65.76

**kcal to maintain** 1938.75 **Add amount**

**kcal to gain** 2288.75 **Add Portion**

**kcal to lose** 1438.75 **Custom Name**

**Delete**

A felső listán látható a már feljegyzett ételek sora, értékeikkel együtt.

A bal oldali részen látható a dátum, napi összegzés az ételek tartalmáról, majd a User Panelen megadott testi értékek alapján kiszámolt súlytartáshoz, hízáshoz és fogyáshoz tartozó értékek.

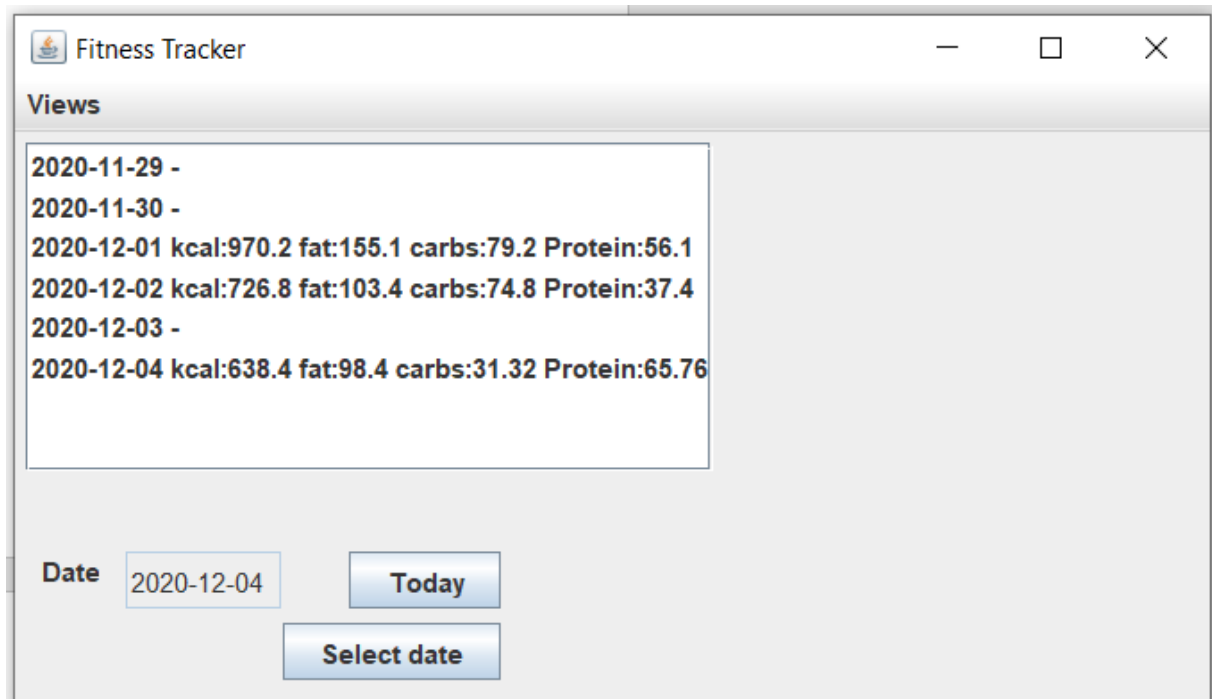
A Today gomb mindig átváltja a listát az aznapira.

A legördülő menüben ételek sora vár, hogy kiválasszák, ekkor a lezárt textfieldeken megjelennek az értékeik. Az Add amount gombhoz tartozó mezőbe beírva a beírt érték grammal hozzáadódik a kiválasztott ételből a listához. Az Add Portion gombbal a beírt érték szorozva a kiválasztott étel egy átlagos adagjával grammal megadva kerül a listába.

A legelső kiválasztható elem a lisztban a Custom food, ekkor minden mező írhatóvá válik, és miután minted kitöltötte a felhasználó az Add amount-al hozzáadhatja a listában esetleg nem szereplő ételt.

A Delete gomb kitörli a felső listából a kiválasztott sort.

#### 4. Trends Panel



The screenshot shows a window titled "Fitness Tracker" with a "Views" section. A list of dates and their corresponding nutritional data is displayed. The data is as follows:

Date	kcal	fat	carbs	Protein
2020-11-29 -				
2020-11-30 -				
2020-12-01	970.2	155.1	79.2	56.1
2020-12-02	726.8	103.4	74.8	37.4
2020-12-03 -				
2020-12-04	638.4	98.4	31.32	65.76

Below the list, there is a "Date" label, a text input field containing "2020-12-04", a "Today" button, and a "Select date" button.

A Trends Panel megmutatja az elmúlt napok feljegyzéseinek összegét napokra bontva. Itt a listában egy elemet kiválasztva, majd a Select Date gombot megnyomva az adott nap jelenik meg a Date mezőben, továbbá a FoodPanelen ennek a napnak a feljegyzései kerülnek ki.

## Use Case-k

### Új profil létrehozása

- Program betöltése
- Load gomb megnyomása

alternatíván:

- Program betöltése
- Írható textfieldek kitöltése
- Confirm gomb megnyomása

### Profil mentése

- Profil betöltése vagy létrehozása
- Bármilyen programon belüli tevékenység
- Save gomb megnyomása

### Étel hozzáadása

- Elfogyasztott étel adatainak összegyűjtése (pl csomagolásról)
- Daily view megnyomása
- Custom food kiválasztása legördülő listán
- Adatok felvétele
- Add Amount gomb megnyomása

alternatíván:

- Daily view megnyomása
- Elfogyasztott étel kikeresése legördülő listában
- Megfelelő mennyiség vagy adag beírása
- Megfelelő Add... gomb megnyomása

### Nap kiválasztása

- Trends view megnyomása
- listából megfelelő dátumra kattintás
- Select Date megnyomása

.

## Be- és kimenetek

### **Profile1.txt, Profile2.txt, Profile3.txt**

A felhasználó mentési profiljai, serializáció során keletkeznek, így emberi szemmel nem olvashatóak.

Tartalmazzák a mentéskori Data osztályt, mely tartalmazza a Day listáját, melyek tartalmaznak Food listákat.

### **FoodList.txt**

A FoodPanel által beolvasott file, emberi szemmel olvashatóan plaintext formátumban. A benne lévő sorok tartalmaznak egy nevet, majd a Food constructor sorrendjében, annak megfelelő számértékeket. Ezeket az értékeket majd betölti a FoodPanel a legördülő menüjébe, innen pedig majd az Add gombokkal a felhasználó viszi fel a napjaira.

### **Egér**

Elsődlegesen a kattintásokkal irányítható az ablak, Jbutton és Jmenu-t kattintásával.

### **Szöveges bemenetek**

A paneleken textboxok találhatók, melyeket a felhasználónak kell kitöltenie, majd főszabályként egy elfogadó gombra kattintva veszi be a program ezeket a szövegeket(és számokat)