Telefonkönyv Dokumentáció

Készítette: Frank Benedek (QOXO70)

Rövid ismertetés:

A program egy szöveges menürendszer alapú telefonkönyv alkalmazás. Lehetőséget biztosít a felhasználónak a kontaktok mentésére, szerkesztésére, törlésére, keresésére, valamint vCard alapú exportálására is. Ez egy egyszerű program, ezért elkészítésénél az esztétikát illetve az egységes felépítést tartottam szem előtt.

Osztálydiagram (ItnelliJ Ultimate által készített):

Felépítés:

A program 3 fő osztályt tartalmaz, ezek mellett 3 JUnit 4 tesztosztályt.

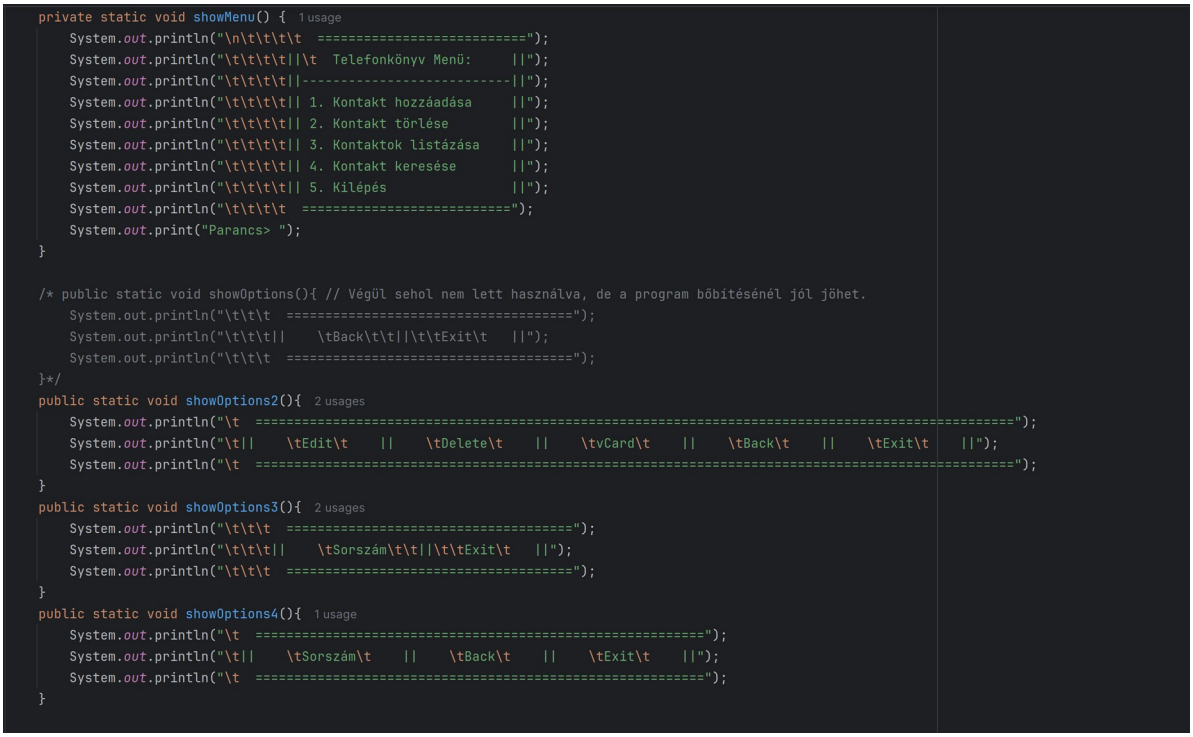
A fő osztályok:

1. **Main.java**
    - **main()**
        - A szöveges felhasználói felület megjelenítéséért, illetve a ContactManager osztállyal való kommunikációért felel.
        - While ciklusban kiíratja a menüt, aztán kér egy bemenetet. A bemenet szerint hívja meg a funkciók methódusait. A ciklus addig tart, amíg a „Kilépés”-t nem választja a felhasználó az „5” inputtal.
        - Érvénytelen bemenet esetén hibaüzenetet printel.

```java
public class Main {
    private static ContactManager manager = new ContactManager();  14 usages
    private static Scanner scanner = new Scanner(System.in);  18 usages

    public static void main(String[] args) {
        manager.loadContacts();
        boolean exit = false;
        while (!exit) {
            showMenu();
            String choice = scanner.nextLine();
            switch (choice) {
                case "1":
                    addContact();
                    break;
                case "2":
                    deleteContact();
                    break;
                case "3":
                    listContacts();
                    break;
                case "4":
                    searchContacts();
                    break;
                case "5":
                    System.out.println("Kilépés...");
                    exit = true;
                    break;
                default:
                    System.out.println("Érvénytelen választás.");
                    break;
            }
        }
        scanner.close();
    }
```

- **showMenu()** illetve **showOptions()**



```
private static void showMenu() {  1 usage
    System.out.println("\n\t\t\t\t  ==========================");
    System.out.println("\t\t\t\t||\t  Telefonkönyv Menü:     ||");
    System.out.println("\t\t\t\t||--------------------------||");
    System.out.println("\t\t\t\t|| 1. Kontakt hozzáadása     ||");
    System.out.println("\t\t\t\t|| 2. Kontakt törlése        ||");
    System.out.println("\t\t\t\t|| 3. Kontaktok listázása    ||");
    System.out.println("\t\t\t\t|| 4. Kontakt keresése       ||");
    System.out.println("\t\t\t\t|| 5. Kilépés                ||");
    System.out.println("\t\t\t\t  ==========================");
    System.out.print("Parancs> ");
}

/* public static void showOptions(){ // Végül sehol nem lett használva, de a program bővítésénél jól jöhet.
    System.out.println("\t\t\t  ====================================");
    System.out.println("\t\t\t||    \tBack\t\t||\t\tExit\t   ||");
    System.out.println("\t\t\t  ====================================");
}*/
public static void showOptions2(){  2 usages
    System.out.println("\t  ============================================================================================");
    System.out.println("\t||    \tEdit\t    ||    \tDelete\t    ||    \tvCard\t    ||    \tBack\t    ||    \tExit\t    ||");
    System.out.println("\t  ============================================================================================");
}
public static void showOptions3(){  2 usages
    System.out.println("\t\t\t  ====================================");
    System.out.println("\t\t\t||    \tSorszám\t\t||\t\tExit\t   ||");
    System.out.println("\t\t\t  ====================================");
}
public static void showOptions4(){  1 usage
    System.out.println("\t  ==========================================================");
    System.out.println("\t||    \tSorszám\t    ||    \tBack\t    ||    \tExit\t    ||");
    System.out.println("\t  ==========================================================");
}
```

  - Printek használatával a menü illetve az egyes menüpontok alatti választási lehetőségek vizuális megjelenítéséért felelek.
  - Tabulátorok valamint „space” karakterek használatával lett testreszabva, a felhasználói szemszögű átláthatóságot és egységességet biztosítva.
  - A showMenu() a főmenü megjelenítésekor kerül meghívásra, a különböző showOptions() változatokat pedig a különböző menüpontok hívják meg.
- **addContact()**
  - Bekéri a kontakt adatait(név, becenév, cím, munkahelyi telefonszám, privát telefonszám) és létrehoz ezekből egy új Contact-ot, majd erre meghívja a ContactManager addContact() methódusát ami elmenti a kontaktot.
- **deleteContact()**
  - Kilistázza az összes kontaktot a ContactManager segítségével, meghívja a 3-as számú showOptions()-t, amivel két lehetőség tárul a felhasználó elé:
    1. Sorszám választása -> a beírt sorszám kontaktja törlésre kerül
    2. Exit (főmenü)

- **listContacts()**
  - A deleteContact()-hoz hasonlóan listázza a kontaktokat, valamint megjelenik a showOptions3() viszont itt egy while (true) ciklusban történik ez, és a sorszám kiválasztását követően egy újabb, belső while(true) ciklus indul. A belső ciklusban megjelenik a showOptions2(), és lehetőség nyílik a kiválasztott kontakt szerkesztésére, törlésére, vCard exportálására.
  - A két ciklus azért szükséges, hogy folyamatos legyen a program, bármikor vissza lehet lépni a „back” paranccsal az előző ciklusba, vagy az „exit” paranccsal a főmenübe.
- **editContact()**
  - a listContacts() vagy a searchContacts() methódusokon keresztül kerül meghívásra, a kiválasztott kontakt indexe alapján bekéri az új adatokat, az üresen hagyott mezők nem változnak. A csere megvalósítása a ContactManager osztályban történik.
- **searchContacts()**
  - A listContacts()-ban található ciklusokhoz hasonló felépítés. Kér egy keresési kritériumot, amit felhasználva a ContactManager elvégzi a keresést, és listázza a keresőszónak megfelelő találatokat.
  - A találatok listázása után a működés megegyezik a listContacts() működésével, sorszámválasztás után a felhasználó elé tárul a kontakt szerkesztése, törlése, vCard exportálása.
  - „back” és „exit” parancsok itt is használhatóak.

2. **ContactManager.java**
    - **ContactManager()**
        - Létrehoz egy listát „contacts” néven
        - Fájl létrehozása, későbbiekben „FILENAME”
        - Google Gson használata a fájlkezeléshez

```
public class ContactManager {  4 usages
    private List<Contact> contacts;  17 usages
    private static String FILENAME = "contacts.json";  3 usages
    private Gson gson;  3 usages
    // Alapértelmezett konstruktor
    public ContactManager() { this( filename: "contacts.json"); }

    // Tesztkonstruktor, amely lehetővé teszi a fájlnév módosítását
    public ContactManager(String filename) {  2 usages
        FILENAME = filename;
        this.contacts = new ArrayList<>();
        this.gson = new Gson();
        loadContacts();
    }
```

        - Alap- és tesztkonstruktor, utóbbi a tesztelések miatt volt szükséges, hogy létrehozhassunk más néven fájlokat
    - **addContacts()** és **deleteContacts()**
        - A mainből kapott kontakton hajtják végre a műveleteket.
        - Az addContact() hozzáadja a kapott kontaktot a „contacts” listához, a deleteContact() pedig a kapott indexű kontaktot törli a listából

```
    // Kontakt hozzáadása
    public void addContact(Contact contact) {  5 usages
        contacts.add(contact);
        saveContacts();
    }
    // Kontakt törlése
    public void deleteContact(int index) {  7 usages
        if (index >= 0 && index < contacts.size()) {
            contacts.remove(index);
            System.out.println("Kontakt törölve.");
            saveContacts();
        } else {
            System.out.println("Érvénytelen index.");
        }
    }
```

- **listContacts()**
  - Ellenőrzi, hogy üres-e a „contacts” lista, ha igen printeli, hogy „Nincsenek kontaktok”
  - Ha nem üres, for ciklussal végigiterál a tartalmán és indexelve (formázva) kiíratja
- **editContact()**
  - A mainből való meghívásával kapott argumentumokra cseréli a kiválasztott indexű kontaktot
- **searchContacts()**
  - A kapott argumentuma a keresőszó, végigiterál egy for-each ciklussal a „contacts” lista elemein, és összehasonlítja a keresőszót a lista elemeinek adataival. Ha találat van azt kilistázza formázott indexeléssel.
- **saveContacts()**
  - FileWriter segítségével a létrehozza a fájlt, gson ezt Json formátummá alakítja.

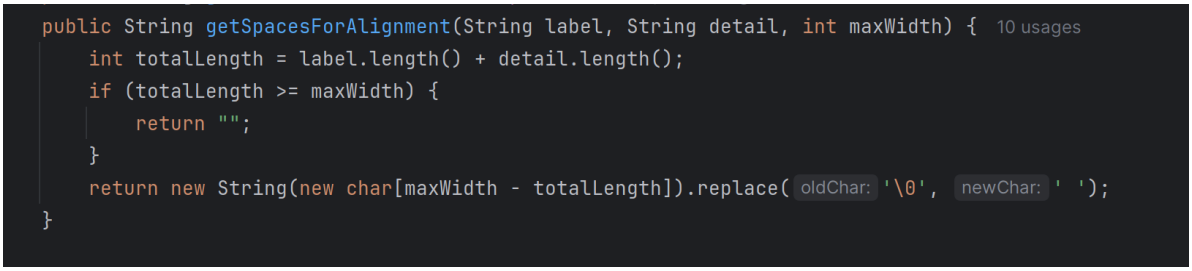
```
// Kontaktok mentése
private void saveContacts() {  3 usages
    try (FileWriter writer = new FileWriter(FILENAME)) {
        gson.toJson(contacts, writer);
    } catch (IOException e) {
        System.out.println("Hiba történt az adatok mentése közben: " + e.getMessage());
    }
}
```

- **loadContacts()**
  - FileReader segítségével betölti a Json fájlt ami gson segítségével bekerül a „contacts” listába

```
// Kontaktok betöltése
public void loadContacts() {  2 usages
    try (FileReader reader = new FileReader(FILENAME)) {
        Type type = new TypeToken<List<Contact>>() {}.getType();
        contacts = gson.fromJson(reader, type);
        if (contacts == null) {
            contacts = new ArrayList<>();
        }
    } catch (IOException e) {
        System.out.println("Hiba történt az adatok betöltése közben: " + e.getMessage());
    }
}
```

- **exportVCard()**
  - Az argumentumként kapott index alapján a választott fájlt a vCard formátumnak megfelelően exportálja egy .vcf fájlba, ami névként a kontakt nevét kapja.
  - A fájlnévben a „space” karakter „-” karakterre van cserélve

3. Contact.java
    - **Konstruktor**
    - **Getterek** és **Setterek**
    - **getSpacesForAlignment()** és **calculateMaxWidth()**
        - A toString() miatt bevezettem, hogy megszámolja mennyi „space” karakter szükséges, hogy jól nézzen ki a kiíratás.



```
public String getSpacesForAlignment(String label, String detail, int maxWidth) {  10 usages
    int totalLength = label.length() + detail.length();
    if (totalLength >= maxWidth) {
        return "";
    }
    return new String(new char[maxWidth - totalLength]).replace( oldChar: '\0',  newChar: ' ');
}
```

```
public int calculateMaxWidth() {  7 usages
    int labelExtra = "Munkahelyi szám:  ".length();
    int extraChars = "||".length() + 1;
    int maxLen = 33;
    maxLen = Math.max(maxLen, labelExtra + name.length());
    maxLen = Math.max(maxLen, labelExtra + nickname.length());
    maxLen = Math.max(maxLen, labelExtra + address.length());
    maxLen = Math.max(maxLen, labelExtra + workPhone.length());
    maxLen = Math.max(maxLen, labelExtra + privatePhone.length());
    return maxLen + extraChars;
}
```

```
// toString
@Override
public String toString() {
    int maxWidth = calculateMaxWidth();
    return "\n\t\t\t|| Név: " + name + getSpacesForAlignment( label: "Név: ", name, maxWidth) + "||" +
            "\n\t\t\t|| Becenév: " + nickname + getSpacesForAlignment( label: "Becenév: ", nickname, maxWidth) + "||" +
            "\n\t\t\t|| Cím: " + address + getSpacesForAlignment( label: "Cím: ", address, maxWidth) + "||" +
            "\n\t\t\t|| Munkahelyi szám: " + workPhone + getSpacesForAlignment( label: "Munkahelyi szám: ", workPhone, maxWidth) + "||" +
            "\n\t\t\t|| Privát szám: " + privatePhone + getSpacesForAlignment( label: "Privát szám: ", privatePhone, maxWidth) + "||" + "\n";
}
```

        - A calculateMaxWidth() megkeresi a leghosszabb adatot az adott kontaktban a leghosszabb „Munkahelyi szám” és „||” hozzáadásával, és visszaadja az értékét
        - A toString() methódus megkapja a maxWidth-et, ami a getSpacesForAlignment() egyik argumentuma lesz a sztringgé alakításnál. Minden adatra meghívjuk, és annyi „space” karaktert rak a sztringbe, hogy a teljes sztring „maxWidth” hosszúságú legyen.