Gymnázium, Praha 6, Arabská 14

Programování

Ročníková práce

Hana Machalíková

ToDo List

2.E

duben 2022

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Kladně dne 2. května 2022 ……..……..

Hana Machalíková

Název práce: ToDo List

Autor: Hana Machalíková

Anotace: Program je složen ze tří částí – část s hlavními tlačítky, část nazvaná „soubor“ a část nazvaná „seznam“. V těchto oblastech se zobrazují jednotlivé položky. Program by měl být schopný po načtení položek ze souboru do souborové části okna vytvářet a upravovat seznamy. Mělo by být možné měnit pořadí a dále přidávat a upravovat názvy na jednotlivých položkách. Upravené verze lze kdykoliv uložit.

Vytvořený soubor lze použít například jako nákupní seznam: aby měl uživatel lepší přehled o tom, co má koupit, seřadí si položky tak, jak postupně bude procházet obchodem, aby na nic nezapomněl. Program lze použít také k vytvoření seznamu písní, které chce kapela hrát na koncertě. Na jedné straně bude mít výpis všech písní, které hraje, a na druhé straně písně, které chce hrát právě teď, včetně pořadí.

Zadání: Program načte ze souboru množinu položek, které následně vypíše do nabídky položek. Z nabídky je možné položky vybírat a přesouvat je na jednotlivé pozice v seznamu. Bude možné měnit pořadí položek. Bude to užitečné např. pro kapely, které potřebují před koncertem sepsat seznam písní, co a v jakém pořadí budou hrát, nebo jako nákupní seznam s daným pořadím. Výsledkem bude seřazená podmnožina původní množiny položek.

Těchto seznamů bude možné vytvořit více a uložit si je do aplikace. Bude možné vytvořit nový na základě dříve uloženého a upravit ho.

1. Obsah

2. Úvod

Tento dokument se zabývá popisem mé ročníkové práce s úkolem naprogramovat ToDo List dle předem schváleného stručného zadání. Program spravuje položky, které si uživatel načte a případně ještě i vytvoří nebo jinak upraví v samotném programu.

Program je složen ze tří částí – tlačítek seznamu a souboru. Má zajistit jednoduché organizování položek, které uživatel programu poskytne.

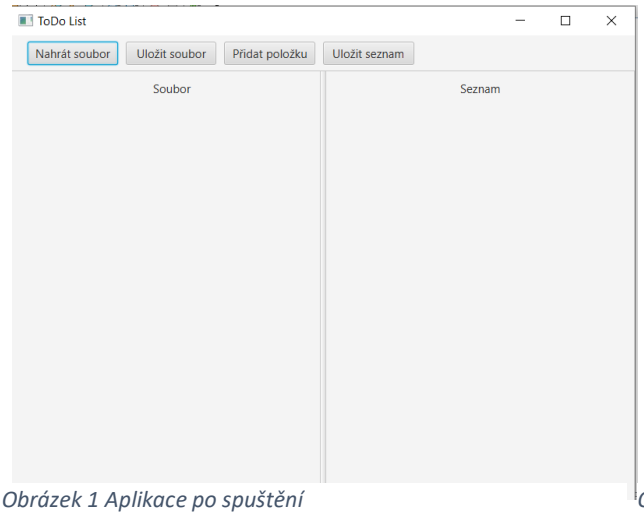
3. Jak program funguje

3.1. Implementace grafického uživatelského rozhraní

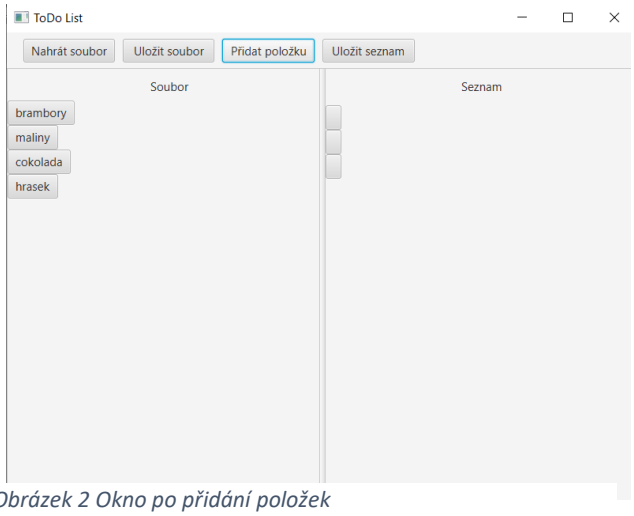
Aplikace je naprogramována v platformě JavaFX – JavaFXML. Jazykem programu je java. Jako vývojová prostředí jsem střídala NetBeans IDE a IntelliJ IDEA. Při výuce jsme ještě donedávna používali výhradně NetBeans, proto jsem začala zde. Postupem času jsem však došla k závěru, že některé věci je lepší dělat v IntelliJ, které poskytují více nápověd a celkové podpory uživateli, a tak jsem přepínala mezi oběma prostředími. Naštěstí jsou i částečně propojená skrze samotný program, tudíž to nebylo příliš složité. Pro grafickou úpravu jsem použila SceneBuilder, který uživatelům umožňuje jednoduché vytváření a upravování XML souborů.

Jelikož minulou ročníkovou práci jsem psala v JavaFX, bylo zajímavé vyzkoušet si opět něco nového. Protože tentokrát je kód složitější než v loňském roce, bez pomoci SceneBuilderu a jeho automaticky vytvářených XML souborů by se mi s ním pracovalo mnohem obtížněji.

Jako komponenty pro jednodušší obsluhu aplikace využívám primárně 2 VBoxy – kontejnery, ve kterých se zobrazují jednotlivé položky jako tlačítka kvůli následné možné manipulaci. Jeden VBox reprezentuje souborovou část a druhý seznamovou. Je prakticky jedno, kterou z těchto dvou oblastí si uživatel zvolí jako primární a kterou jako pomocnou, jež slouží jako úložiště pro aktuálně nepotřebné položky. Ve třetí ze stěžejních částí celého okna se nacházejí 4 tlačítka pro jednoduché



Obrázek 1 Aplikace po spuštění



Obrázek 2 Okno po přidání položek

užívání aplikace. Tlačítka slouží k nahrání souboru, uložení souborové nebo seznamové části okna a k přidání nové položky bez názvu do pravé (seznamové) části okna.

Ke každému z VBoxů je dále vytvořený ArrayList na tlačítka. Obsah je po většinu doby totožný, mění se pouze ve chvílích překreslování. Teoreticky jsem mohla používat pouze VBoxy, takto je ale práce mnohem jednodušší.

3.2. Důležité metody

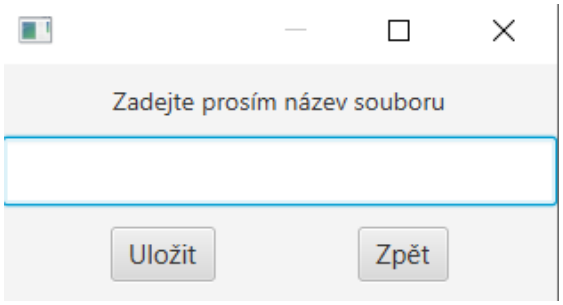
3.2.1. Dialogové okno

Za jednu z velmi důležitých metod, přestože není úplně stěžejní, považuji vytváření dialogového okna *(Obrázek 3)*. Ve svém kódu ji využívám mnohokrát, jelikož mi umožňuje, abych dala uživateli prostor si zvolit například název souboru, do kterého bude chtít seznam uložit, nebo aby si mohl přejmenovat jednotlivé položky.

Jelikož jsem dialogová okna *(Obrázek 4)* nikdy dříve nevytvářela, nechala jsem se inspirovat tutoriálem na YouTube, díky kterému jsem alespoň z větší části pochopila princip dialogového okna a přepínání mezi jednotlivými scénami a stagemi (Martens,

```
void Dialog(ActionEvent event, Button bt1, Button bt2, String fxml) throws IOException {
	Stage stage;
	Parent root;
	if (event.getSource() == bt1) {
		stage = new Stage();
		root = FXMLLoader.load(Objects.requireNonNull(getClass().getResource(fxml)));
		stage.setScene(new Scene(root));
		stage.initModality(Modality.APPLICATION_MODAL);
		stage.initOwner(bt1.getScene().getWindow());
		stage.showAndWait();
	} else {
		stage = (Stage) bt2.getScene().getWindow();
		stage.close();
	}
}
```

Obrázek 3 Metoda pro vytvoření dialogového okna



Obrázek 4 Ukázka dialogového okna

2015).

Atributy, které jsou třeba pro vytvoření, je především ActionEvent, který umožní rozpoznat, kdy se aktivovat. Dále to jsou dvě tlačítka, každé pro jednu scénu. Aby program věděl, kterou scénu má uživatel na mysli, je nutné mu dát k dispozici jeden element, který se v této scéně vyskytuje. Poslední atribut je řetězec fxml, což je odkaz na FXML soubor, aby aplikace věděla, jak má scéna, na kterou má přepnout, vypadat.

3.2.2. Odstranění položky a její vytvoření jinde

Pokud chceme přesouvat položky i v rámci stejného seznamu, ve kterém záleží na pořadí, je nutné při přemisťování objektu ji vymazat na jejím původním působišti a naopak vytvořit novou na jejím novém místě.

Při přepsání jsou tudíž použity dvě metody. Bylo by možné použít jen jednu, ale kód by byl méně přehledný.

```
void OdstranitPolozku(ArrayList<Button> arr, VBox vbox, Button bt) throws IOException {
	int pozice = Integer.parseInt(bt.getId());
	arr.remove(pozice);
	PrepisVBoxu(arr, vbox);
}
```

Obrázek 6 Metoda pro odstranění položky ze souboru

```
void VlozitPolozku(ArrayList<Button> arr, VBox vbox, Button co, Button kam) throws
IOException {
	int pozice = Integer.parseInt(kam.getId());
	Button k = new Button("");
	k.setId(arr.size() + "");
	arr.add(k);
	Button pred;
	if ((arrSouboru.contains(co) && arrSouboru.contains(kam) &&
Integer.parseInt(co.getId()) < Integer.parseInt(kam.getId()))
	|| (arrSeznamu.contains(co) && arrSeznamu.contains(kam) &&
Integer.parseInt(co.getId()) < Integer.parseInt(kam.getId()))) {
		for (int i = arr.size() - 1; i >= pozice; i--) {
			pred = arr.get(i - 1);
			arr.set(i, pred);
		}
	} else {
		for (int i = arr.size() - 1; i >= pozice + 1; i--) {
			pred = arr.get(i - 1);
			arr.set(i, pred);
		}
		arr.set(pozice, co);
		PrepisVBoxu(arr, vbox);
	}
}
```

Obrázek 5 Metoda pro přidání položky do souboru

Metoda pro odebrání položky *(Obrázek 5)* má za parametry ArrayList seznamu nebo souboru (podle toho, odkud má být položka odebrána), k němu příslušný VBox a jako poslední je tlačítko, které chceme smazat.

Když naopak položku přidáváme *(Obrázek 6)*, je třeba jako parametr použít ještě jedno tlačítko navíc. Jedno opět reprezentuje přesouvané tlačítko, druhé ukazuje, před kterým tlačítkem se má položka objevit.

3.2.3. SetOnAction

Poslední metoda, která není přímo spjata s některým z mých FXML souborů, je SetOnAction *(Obrázek 7)*, která nastavuje fx metodu seOnAction. Využívá se u tlačítek, která nejsou předdefinována v žádném z FXML souborů. Jelikož ale chci takto tlačítka využívat minimálně na to, aby mohla být někam přesunuta, je nutné jim tuto metodu zprovoznit. A jelikož tato tlačítka vytvářím na více místech, je jednodušší mít na to zvláštní metodu, aby kód nemusela opakovat.

Moje metoda SetOnAction je propojená s odebíráním a přidáváním položek v seznamech.

Parametrem je pouze Button, u kterého má být metoda setOnAction

```
void SetOnAction(Button bt) throws IOException {
	bt.setOnAction((ActionEvent e) -> {
		Button n = (Button) e.getSource();
		if (!aktivni) {
			tlacitko = n;
			aktivni = true;
		} else {
			if (tlacitko != n) {
				try {
					Presun1(tlacitko);
				} catch (IOException ex) {
Logger.getLogger(FXMLDocumentController.class.getName()).log(Level.SEVERE, null, ex);
				}
				try {
					Presun2(e, tlacitko, n);
				} catch (IOException ex) {
Logger.getLogger(FXMLDocumentController.class.getName()).log(Level.SEVERE, null, ex);
				}
				aktivni = false;
			} else {
				zmacknute = n;
				aktivni = false;
				try {
					Okno(e, n, zpet4, "upravaTlacitka.fxml");
				} catch (IOException ex) {
					throw new RuntimeException(ex);
				}
			}
		}
	});
}
```

Obrázek 7 Metoda SetOnAction

nastavena.

3.2.4. Přepis VBoxu

Když se v ArrayListu změní pořadí položek nebo se dokonce položky přesunou mezi oběma seznamy, uživatel to nemá jak zjistit. Je tedy nutné, abychom VBox, což je viditelné rozhraní pro seznamu položek, překreslili nebo aktualizovali.

Zásadní částí této metody *(Obrázek 8)* je vymazání samotného VBoxu, aby se položky při přepisování neřadily za sebe, ale přepisovaly se.

```
void PrepisVBoxu(ArrayList<Button> arr, VBox vbox) {
	vbox.getChildren().clear();
	pomoc = (ArrayList<Button>) arr.clone();
	arr.clear();
	for (int a = 0; a < pomoc.size(); a++) {
		Button b = new Button(pomoc.get(a).getText());
		b.setId(a + "");
		SetOnAction(b);
		arr.add(b);
		vbox.getChildren().add(arr.get(a));
	}
}
```

Obrázek 8 Metoda pro přepsání VBoxu

Kdykoliv je tedy změněn obsah nebo pořadí v ArrayListu, je nutné tuto metodu zavolat, aby se změna projevila i v grafickém prostředí programu.

Při procházení jednotlivých položek se aktualizují jejich ID v seznamu. Pro jednodušší zjišťování aktivního tlačítka je lepší používat jeho ID zároveň jako index v seznamu. Jediným problémem je, že neexistuje žádná metoda, která by byla schopná přepsat ID elementu. Jeho identita je tedy pevně daná po prvním nastavení. Jediné řešení, které jsem byla schopná vymyslet, bylo vytvořit duplikát ArrayListu jako pomocnou proměnou typu ArrayList a ten původní zcela vymazat. Při dalším kroku jsou pak veškeré položky nahrány zpět do původního seznamu, ale s jiným ID.

3.3. Soubory

Součástí celého mého programu je dohromady 7 souborů.

Soubor, který obsahuje metodu start a main, má název RP.java.

V souboru FXMLDocumentController.java jsou rozepsány jednotlivé metody a hlavně všechny potřebné informace pro správný chod programu. Tedy RP.java program spustí a předá řízení Controlleru.

Zbylých 5 souboru jsou FXML soubory, které definují jednotlivé scény. Soubory napis.fxml, napis2.fxml, napis3.fxml obsahují různé prvky, které se využívají trochu odlišně, a proto mají vlastní soubor. Soubor UpravaTlacitka.fxml je důležitý pro přejmenování a případné odstranění položky. V souboru FXMLDocument.fxml je uložen vzhled celé scény, která je zobrazena po celou dobu, kdy je aplikace spuštěná. Někdy je zčásti překrytá dialogovými okny.

K dispozici je také doplňkový soubor seznam.txt obsahující položky, které chci do seznamu nahrát, a případně i zkušební soubor pro ukládání seznamů.

3.4. Zadání názvu souboru

Jedním z problémů, se kterými jsem se při vývoji aplikace setkala, je zadání názvu souboru, který chce uživatel načíst. Alespoň do jisté míry se mi podařilo zprovoznit dvě varianty.

V prvním případě bylo možné zadat název dokumentu, ale ve VBox se jeho obsah nezobrazoval. Nad tímto problémem jsem strávila spoustu času, ale na žádné řešení jsem nepřišla. Paradoxem bylo, že ve VBox reprezentujícím seznam se nové prázdné položky zobrazovaly bez jakýchkoliv obtíží.

Jako náhradní řešení jsem proto použila pevné nastavení souboru seznam.txt. Za této situace se jeho obsah ve VBoxu zobrazuje, ale proč tomu předtím tak nebylo, nevím.

První možnost je zakomentovaná na konci Controlleru.

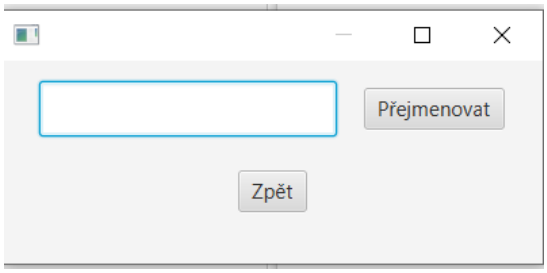
3.5. Přemístění položky a její změna názvu

K přemisťování položky slouží již zmíněné dvě metody. Jedna položku smaže z aktuálního místa, ale předtím si ji uloží do proměnné, aby se obsah položky neztratil. Následně se vytvoří nová položka v označeném místě, do které se uloží původní údaj z proměnné.

V praxi to vypadá tak, že uživatel klikne na položku, kterou chce přesunout nebo přejmenovat.

Druh akce záleží na tom, kde se provede druhé kliknutí. Pokud uživatel stiskne stejné tlačítko i podruhé, zobrazí se dialogové okno *(Obrázek 8)*, které mu umožní zadat nový název položky. Po stisknutí tlačítka Přejmenovat se název položky aktualizuje.

Pokud uživatel klikne na jakékoliv jiné tlačítko v interaktivní části okna, položka se zařadí před toto tlačítko.



Obrázek 9 Dialogové okno pro změnu názvu položky

4. Závěr

Se svou prací jsem relativně spokojená. Mrzí mě, že se mi nepovedlo zprovoznit načítání souboru z dialogového okna. Jsem ráda, že veškeré stěžejní prvky fungují alespoň relativně tak, jak mají. Určitě je zde prostor pro zlepšení, kromě zmíněného zadávání by například bylo možné udělat aplikaci uživatelsky přívětivější (například použít jinou než šedou barvu nebo vytvořit motivy, ze kterých by si uživatel mohl vybrat, aby se mu lépe pracovalo).

5. Seznamy

5.1. Citace

5.2. Zdroje

Add Button to VBox : VBox « JavaFX « Java. *java2s.* [Online] http://www.java2s.com/Code/Java/JavaFX/AddButtontoVBox.htm.

Čápka, David. 2013. Lekce 3 - Práce s textovými soubory v Javě. *ITnetwork.* [Online] 6. 5 2013. [Citace: 24. 4 2022.] https://www.itnetwork.cz/java/soubory/java-tutorial-prace-se-soubory-txt.

2015. Is there a way to remove all the contents in vBox in JavaFx? *stackoverflow.* [Online] 8. 3 2015. [Citace: 27. 4 2022.] https://stackoverflow.com/questions/28925381/is-there-a-way-to-remove-all-the-contents-in-vbox-in-javafx.

Jenkov, Jakob. 2021. JavaFX VBox. *Jenkov.* [Online] 9. 3 2021. [Citace: 27. 4 2022.] https://jenkov.com/tutorials/javafx/vbox.html.

Martens, John. 2015. JavaFX Beginner Tutorial: Switching Scenes and Making a Popup Window. *YouTube.* [Online] 22. 1 2015. [Citace: 26. 4 2022.] https://www.youtube.com/watch?v=w7r698c3yaw.

5.3. Seznam obrázků