Gymnázium Brno, Vídeňská, příspěvková organizace

Pololetní projekt
pro Programování

Kvízová aplikace

11. 06. 2024

Roman Grmela, 5.G

Abstrakt

Tato práce popisuje postup myšlení při tvorbě kvízové aplikace s grafickým uživatelským rozhraním. Umožňuje uživatelům testovat a rozšiřovat jejich znalosti německého jazyka. Aplikace nabízí více typů kvízových otázek zaměřených na gramatiku, slovní zásobu a kulturní znalosti. Ideální pro studenty, učitele i samouky.

Klíčová slova

metoda, soubor, data, GUI, třída, konstruktor

Citace

Roman Grmela: *Kvízová aplikace*, Pololetní projekt, Brno, Gymnázium Brno, Vídeňská, příspěvková organizace, 2024.

Kvízová aplikace

Prohlášení

Prohlašuji, že jsem tuto práci vytvořil samostatně a poctivě jsem uvedl všechny zdroje, ze kterých jsem vycházel.

.....

Roman Grmela

17. června 2024

Obsah

Kapitola 1 Zadání

Vytvoř program s GUI v Javě, který bude realizovat kvíz/dotazník. Otázky bude načítat ze souboru s předem daným formátem. Program bude umět takový soubor načíst a otázky pak bude uživateli klást v náhodném pořadí s náhodně seřazenými variantami odpovědí. Bude umět počítat body za správné odpovědi a na konci uživateli zobrazí, jak dopadl. Program bude mít grafické uživatelské rozhraní (GUI), které bude postupně zobrazovat načtené otázky, počítat odpovědi a zobrazovat výsledky.

Kapitola 2 Analýza a návrh

2.1 Specifikace zadání

Vytvoř program s GUI v Javě, který bude realizovat kvíz/dotazník. Otázky bude načítat ze souboru s následujícím formátem:

Název:Test VV 01
Otázka 1zN:Která barva je nejlepší?
TODO: Udělat si seznam barev od
nejlepší po nejhorší.
Hodnota:5
Správně:Modrá
Špatně:Růžová
Špatně:Purpurová
Špatně:Fialová
Otázka 1zN:Který tvar je hranatý?
Hodnota:10
Špatně: Kruh
Správně: Čtverec
Špatně: Elipsa

Text na začátku řádku před dvojtečkou je klíčové slovo označující, co se na řádku nachází. Za klíčovým slovem Název: se nachází název dotazníku. Za klíčovým slovem Otázka 1zN: se nachází text otázky, která má právě jednu správnou odpověď. Počet bodů za otázku je označen klíčovým slovem Hodnota:. Správné varianty odpovědí začínají klíčovým slovem Správně:, nesprávné varianty začínají klíčovým slovem Špatně:. V souboru může být libovolné množství otázek. Řádek začínající mřížkou (#) se může vyskytovat kdekoli a považuje se za komentář. Prázdné řádky se ignorují. Program bude umět takový soubor načíst a otázky pak bude uživateli klást v náhodném pořadí s náhodně seřazenými variantami odpovědí. Bude umět počítat body za správné odpovědi a na konci uživateli zobrazí, jak dopadl. Program bude mít grafické uživatelské rozhraní (GUI), které bude postupně zobrazovat načtené otázky, počítat odpovědi a zobrazovat výsledky.

2.2 Téma projektu a budoucí představa

Rozhodl jsem se vytvořit sebevzdělávací aplikaci, která umožní uživatelům učit se základy německého jazyka. Na začátku je důležité se zamyslet, co bych od takové aplikace jakožto běžný uživatel očekával. Jakým způsobem bude aplikace fungovat? Pro efektivní učení cizího jazyka je vhodné kombinovat úkoly a otázky z různých okruhů, jako je slovní zásoba, gramatika, čtení a poslech. Aplikace bude kombinovat různé typy úkolů a otázek, aby se uživatelé mohli efektivně učit.

Dále bych jako uživatel ocenil různorodost otázek. Např. doplňování slov do věty aby dávala smysl, překlad slovíček, výběr z odpovědí, kdy může být správná pouze 1, nebo dokonce více. Zkrátka, aby bylo v aplikaci co nejvíce relevantních typů úloh.

V mém nápadu se nabízí práce s obrázky. Pro příklad uvedu úkol, kdy je potřeba přeložit slovíčko z češtiny do němčiny. Pro každého uživatele by bylo určitě velkým přínosem vidět obrázek slovíčka, které má přeložit (např. obrázek psa).

Po každé správné odpovědi se uživateli budou přičítat body. Každá otázka bude jinak bodovaná podle náročnosti. Hodnota otázky je uložena v souboru jako všechny další relevantní informace o otázce. Pokud uživatel zadá odpověď nesprávně, záporné body se udělovat nebudou. Po zadání špatné odpovědi se bude zobrazovat správné řešení.

Tvorba tohoto projektu pro mě bude určitě velmi cenná zkušenost, jelikož si v tento moment ani nedokážu představit, jak aplikaci budu implementovat. Doposud nemám žádné zkušenosti s prací s obrázky ani s implementací a obsluhou GUI v Javě, tak jsem velmi zvědavý, jak zvládnu aplikaci realizovat. Takto bych si představoval finální verzi mého projektu.

2.3 Analýza problémů

V první etapě projektu bylo za úkol napsat program, který zpracuje soubor v zadaném formátu (zatím nemusí obsahovat GUI). Ve finální verzi bude realizovat kvíz nebo dotazník.
V budoucnu bude program obsahovat GUI a bude počítat body uživateli v závislosti na hodnotě otázky. Program má umět načíst otázky do seznamu, náhodně je zamíchat a nakonec vytisknout na výstup. Program musí také reagovat na případné chyby ve vstupním souboru pomocí výjimek.

S tím se pojí několik problémů, které je třeba při implementaci zvážit a vyřešit. Např. jak bude uživatel zadávat soubor se vstupy? Jak přesně má vypadat formát vstupního souboru, tedy co je správný vstup a co ne? Co se stane, když uživatel zadá vstupní soubor chybně, nebo pokud bude soubor v chybném formátu?

Další problémy k řešení souvisí s grafickým uživatelským rozhraním. Je potřeba např. zvážit, v jakém momentu se budou přičítat body. Vhodnější by bylo nejspíše sečtení bodů na konci kvízu, aby měl uživatel ještě možnost změny své odpovědi a návratu k předchozí otázce.

Jelikož programuji v jazyce Java, bylo by vhodné vymyslet a implementovat vhodné třídy a metody, které pomohou snáze zpracovávat otázky v souboru.

Po realizaci 1. etapy projektu bude potřeba projekt rozšířit. Konkrétně vytvořit grafické uživatelské rozhraní (GUI) a ošetřit počítání bodů. Proto bude klíčové navrhovat třídy tak, aby se do nich při rozšiřujících úpravách téměř nemuselo zasahovat a bylo snadno rozšiřitelné.

2.4 Návrh řešení

Na začátek je vhodné vymyslet třídy, které mi pomohou realizovat zadání. Mohl bych vytvořit např. třídy Otázka, jejíž objekty by si v sobě uchovávaly relevantní informace jako hodnotu, text otázky

a také texty odpovědí. Teď je tu ale problém, že každá otázka může mít jiný počet odpovědí. Je tedy zbytečné vytvářet určitý počet atributů pro uchování odpovědí. Je lepší si uchovávat seznam odpovědí na otázku. Proto by byla vhodná také třída Odpověď, jejíž atributy by měly informace o tom, zda je odpověď správná či špatná a také text odpovědi. Podle konceptu atomizace je nejlepší, aby třídy obsahovaly pouze to, co s nimi souvisí, a to pouze to nejnutnější, nic navíc. Bude lepší vymyslet další třídy, které by řešily načítání dat ze souboru, výpis pro implementační účely atd. Dále bych mohl vytvořit třídu uživatel, která by uchovávala informace o uživateli – třeba jméno a počet bodů, které v kvízu obdržel.

Další problém se týká zpracování souboru.
Budu zpracovávat soubor ve vlastní třídě. Je mnoho způsobů, jak soubor zpracovat. Velmi časté je obalování čtenářů souborů něčím chytřejším. Nikdy se k souboru nepřistupuje přímo. Je možné použít např. „FileReader“, „Bufferedreader“, „FileInputStream“ atd.

Chyby můžu testovat buď synchronně (tj. hned v místě vzniku), nebo asynchronně. Výhodnější je asynchronní zpracovávání chyb. Mimo jiné to zvyšuje přehlednost kódu. Formát souboru je složitě strukturovaný. Bude dobré vymyslet co nejlepší způsob, jak data ze souboru získat do programu. Je možné číst soubor po znacích, po slovech, ovšem za mě bude nejlepší číst soubor po řádcích a poté testovat klíčová slova na začátku každého řádku. Po přečtení 1 otázky v souboru je nejlogičtější vytvořit instanci třídy Otázka a vložit do seznamu. Seznam může mít různé podoby. Zvažoval jsem možnosti „List/LinkedList/ArrayList“.

Co se týče výběru souboru, máme více možností. První je manuální zadání cesty k souboru. Uživatel bude mít možnost ručně zadat cestu k souboru pomocí textového pole v GUI. Toto řešení je jednoduché a přímočaré, ale vyžaduje, aby uživatel znal přesnou cestu k souboru. Druhou možností je využití dialogového okna. Implementace dialogového okna v GUI umožní uživateli vybrat soubor prostřednictvím standardního průzkumníka souborů ve svém operačním systému. Tento přístup je uživatelsky přívětivější a eliminuje potřebu znalosti přesné cesty k souboru.

Řešení chybných vstupů je možné řešit vyhazováním výjimek, nebo pomocí zobrazení dialogových oken. Výjimky (objekty nesoucí informaci o chybě) související s chybným formátem souboru jsou následující: „FileNotFoundException“ - pokud zadaná cesta k souboru neexistuje. Dále „IOException“ - pokud dojde k chybě při čtení souboru. A nakonec „FormatException“. Pokud soubor není ve správném formátu.

Aplikace musí obsahovat GUI (grafické uživatelské rozhraní). GUI aplikace může být navrženo tak, aby poskytovalo uživatelsky přívětivé prostředí pro interakci s kvízem nebo dotazníkem v německém jazyce. Bude podporovat načítání otázek a odpovědí ze souboru, zobrazení otázek uživateli, sběr odpovědí a zobrazení výsledků. Prostředí NetBeans umožňuje jednoduché grafické navržení tohoto rozhraní pomocí přetažení jednotlivých komponentů – „drag and drop“.

Navrhuji vytvořit hlavní formulář „Jframe“ s metodou „main“. Ten by mohl obsahovat název kvízu v záhlaví, v zápatí tlačítka „Další“ a „Předchozí“, pomocí nichž by uživatel překlikával mezi jednotlivými otázkami. Největší část formuláře může zabírat „mainPanel“, kam bych poté dynamicky generoval jednotlivé panely v závislosti na typu otázky.

2.5 Změny

Kvůli časovým důvodům jsem se rozhodl své předchozí plány redukovat. Tento půlrok byl pro mě velmi nabitý. Zpočátku jsem byl až příliš ambiciózní, i přesto, že jsem neměl ani tušení, jak budu své plány realizovat. Chtěl jsem se naučit spoustu nových věcí, ovšem to jsem si neuvědomil, jaké budu mít časové vytížení. Dlouhodobé výpadky internetu, absolventský klavírní koncert a konzistentní příprava na MČR juniorů a posléze i MČR dospělých mi nezbyl téměř žádný čas pro tvorbu projektu.

Rozhodl jsem se proto realizovat pouze základní možnosti rozšíření a vytvořit jednoduchý projekt, i přesto, že jsem od sebe očekával mnohem více. Vynechám práci s obrázky a také pokročilejší typy otázek jako např. doplňování slov do věty. Dále mi nezbyl čas ani na kvízové úlohy s důležitou oblastí učení se němčiny, a to čtení.

2.6 Možné vylepšení

V současné verzi implementace není dostupná funkcionalita, která měla být zahrnuta. Konkrétně se jedná o počítání bodů a zobrazování konkrétních panelů s otázkami. Tento nedostatek plánuji v budoucnu implementovat, abych tak docílil 100% funkčnost vzdělávací aplikace. Dále by bylo vhodné vytvořit úvodní okno, které by se uživatele ptalo na jméno, příjmení či možnost změny vstupního souboru.

Kapitola 3 Popis řešení - implementace

3.1 Prostředí

Celý projekt je vypracován v programovacím jazyce Java. Práci jsem tvořil v aplikaci NetBeans na mém domácím PC. Toto prostředí je mi známé i z praktických vyučovacích hodin programování. NetBeans jsem si oblíbil pro jeho uživatelsky přívětivé rozhraní, široké spektrum funkcí a silnou podporu pro vývoj v jazyce Java. Díky tomu jsem byl schopen efektivně organizovat kód, spravovat projekty a „debugovat“ aplikaci. V celém projektu můžeme narazit na dokumentační komentáře upřesňující funkčnost programu.

3.2 Implementace jednotlivých tříd

3.2.1 Třída otázka a potomci

Vytvořil jsem třídu Otázka, kde se nacházejí atributy související s otázkou v souboru (text otázky, Hodnota a seznam odpovědí). Jak již bývá tradicí, nacházejí se zde také přetěžované (overloaded) konstruktory, „gettery", „settery“ a pro případ textové reprezentace objektu překrytá metoda „toString“.

Později jsem si uvědomil, že by bylo vhodnější vytvořit co nejvíce obecnou abstraktní třídu a z ní odvozovat potomky – jednotlivé typy otázek. Dodržovalo by to tak jeden ze základních konceptů OOP v Javě – dědičnost a vytvoření vztahu předek-potomek. V abstraktní třídě se nachází informace, které o sobě nese každá konkrétní otázka – hodnota a její text. Ostatní konkrétnější atributy uchovávají potomci. Tím pádem by každý specifický typ otázky rozšiřoval tuto obecnou třídu.

V této třídě jsem implementoval následující metody. Konstruktory, které vždy volají potomci při inicializaci svých atributů. Klíčové metody budou ovšem implementovat potomci této třídy, a to konkrétně metodu „checkAnswer“, která bude ověřovat, zda se uživatelova odpověď shoduje se správnou odpovědí – vrací tudíž „true“, nebo „false“. Dále getter pro odpovědi, aby bylo možné je z daného objektu vytáhnout a následně zpracovat.

Potomci abstraktní třídy „Otázka“ jsou „Otazka1zN“ což je specifický typ otázky, který má pouze 1 odpověď správnou z N možností. Nachází se zde konstruktory, které ovšem prvně volají konstruktor předka. Dále gettery, settery, překrytá metoda „toString“ a zásadní implementace metody „checkAnswer“, která otestuje a rozhodne, zda je uživatelova odpověď správná. Tato metoda ve zkratce projde všechny možnosti odpovědí a ptá se, zda je aktuální odpověď správná a zda se zároveň shoduje s odpovědí uživatele.

Další potomek abstraktní třídy „Otázka“ je „OtazkaText“, což je druhý typ otázky v mém projektu. Tento typ je obdobný jako předchozí typ, ovšem reprezentující otázku s textovou odpovědí, nikoliv výběrem z možností. Jelikož jsem načítání začal vcelku nešikovně, bylo to velmi pracné. Bohužel je mé řešení nerozšiřitelné. Neměl jsem čas začít dělat načítání od znovu.

3.2.2 Třída odpověď

Přišlo mi na místě vytvořit také třídu Odpověd, a to hlavně z důvodu, abych rozeznal správnou odpověď od špatné. Jsou zde pouze 2 atributy: text odpovědi a boolenová hodnota, která uchovává informaci o tom, zda je odpověď správná či špatná. Opět je zde metoda „toString“ pro textovou reprezentaci instance třídy Odpověď, gettery, settery a konstruktory. Důležitá metoda „isSpravne“ vrací „true“, pokud je odpověď správná a uplatní se pří testování správnosti odpovědi.

3.2.3 Hlavní třída

Hlavní třída „main“ se nachází ve formuláři „Jframe“. Viz část dokumentu „návrh GUI“.

3.2.4 Třída Kvíz

Stěžejní třída je třída Kviz, která se stará o načtení dat ze souboru a následný výpis na výstup.
V této třídě jsem vytvořil seznam typu Otázka, kam si budu postupně ukládat objekty nesoucí informace o otázce.

Co se týká zpracovávání souboru, rozhodl jsem se použit blok „try-catch se zdrojem“. Potom není nutné používat větev „finally“, jelikož se všechny věci zavřou automaticky. Ve větvi try zpracovávám soubor po řádcích. Po načtení každého řádku testuji klíčové slovo na začátku řádku a podle toho ukládám hodnoty atributů. Pokaždé, když narazím na klíčové slovo otázky, tvořím objekt otázka a vkládám to do seznamu. Vybral jsem „ArrayList“, jelikož otázky v seznamu nebudu muset odstraňovat či přidávat uprostřed. Otázky budu vždy přidávat na konec seznamu, a pro tento účel je „ArrayList“ optimální. Soubor jsem se rozhodl zpracovat pomocí třídy Scanner, která bude obalovat „FileInputStream“, protože s tím mám dobré zkušenosti z minulých měsíců v programovacích hodinách.

Před tvorbou objektu zamíchám odpovědi v seznamu pomocí metody „shuffle“ třídy „Collections“. Na závěr se zamíchá také seznam otázek. Díky tomu se později budou zobrazovat náhodně zamíchané otázky se zamíchanými odpověďmi. Větev „catch“ odchytává výjimky (objekty nesoucí informace o chybě) a starají se o bezpečnost programu.

Po nasbírání zkušeností jsem si uvědomil nešikovnost tohoto načítání. Nevyužil jsem tak naplno možnosti OOP a jazyka Javy. Bylo by lepší řešit načítání v samostatných třídách, které by načítaly své konkrétní typy otázek. Např. pomocí „Readerů“. V mém případě bylo pak každé další rozšíření velmi pracné, a tomu bych se chtěl při tvorbě příštích projektů určitě vyhnout.

Při implementaci této třídy jsem narazil na menší problém. Ještě musím vyřešit problém týkající se ukládání odpovědí do seznamu, protože nyní je můj projekt ve stavu, kdy se do jediného objektu třídy Otázka ukládají úplně všechny odpovědi, jež se nachází v souboru. Možná budu muset časem své řešení upravit. Zkoušel jsem vždy před načtením další otázky vymazat obsah seznamu, kam si ukládám odpovědi pro 1 otázku, tj. použít metodu seznamu „odpovedi.clear()“, ovšem ani to nepomohlo. Po nějaké době jsem problém vyřešil - a to tak, že jsem pro každou otázku vytvořil svou vlastní kolekci odpovědí a zamezil tak chybě.

Pro implementační účely je zde i výpis otázek na konzoli. Výpis otázek řeším pomocí for cyklu, konkrétně průchodu typu „for-each loop“. Před tímto cyklem volám svoji metodu, která zamíchá seznam s otázkami, aby se uživateli zobrazovaly v náhodném pořadí. Implementace různých typů otázek je mé rozšíření projektu.

3.2.5 Třída BadQuestionException

Třída „BadQuestionException“ je moje vlastní výjimka, která rozšiřuje třídu „Exception“. Své uplatnění má při inicializaci otázky. Když je prázdný text. Inspiroval jsem se ve videu Davida Martínka v soukromém kurzu. Implementoval jsem zde konstruktory, které předávají svému předkovi zprávu o chybě. Textová informace o chybě může být zadána při vyhazování této výjimky někde výše, nebo se pošle obecná zpráva o chybně zadané otázce – např. chybí-li její text. Konstruktory jsou přetěžovány.

3.2.6 Třída Uživatel

Třída Uživatel reprezentuje relevantní informace o uživateli kvízu. Následná instance této třídy by si pamatovala jméno, příjmení a počet dosažených bodů z testu. V této třídě jsem implementoval konstruktory – přetěžované. V případě, že by člověk nezadal své jméno se automaticky nastaví na „noName“, body se vždy nastaví na 0, pokud nezadá uživatel jinak při inicializaci a volání daného konstruktoru. Dále zde můžeme najít gettery, settery a metodu pro přidání bodů uživateli. Tato třída bude mít uplatnění při počítání bodů. Aktuální verze pololetního projektu tuto třídu zatím nevyužívá.

3.3 Ošetření chyb

V programu testuji velké množství chyb, které může uživatel způsobit. Většina z nich se týká špatného formátu souboru, či objektu typu Otázka jako takového. Pomocí výjimek testuji, zda se podařilo soubor otevřít pro čtení, či pokud je v souboru špatný formát čísla. Tyto výjimky jsou odchytávány v té samé metodě, což není optimální. Obvykle chceme, aby byly výjimky propagovány výše. Takto propaguji pouze svoji vlastní výjimku.

Další chyby souvisejí s neúplným či špatným formátem objektu typu „Otázka“. Nejvýhodnější mi přišlo vytvořit metodu, která by zjistila, zda je otázka v pořádku a zda je možné ji bez problému přidat do seznamu. Tato metoda je implementována v třídě „Kviz“, kde probíhá načítání. Možná by bylo výhodnější tuto metodu přesunout do třídy „Otázka“, aby sama otázka dokázala rozhodnout, zda je kompletní, či nikoliv. Tímto by byl zachován koncept zapouzdření. Metodě „testOtazky“ předávám již vytvořenou otázku. Metoda testuje, zda otázce nechybí text, hodnota, zda chybí odpovědi či jenom správná odpověď.

V případě chyby formátu souboru zobrazím dialogové okno s chybovým hlášením a chybná otázka se do seznamu otázek nepřidá. Zamezím tak výskytu neúplných otázek. Bylo by vhodnější řešit chyby jednotným způsobem, a dialog vyvolávat pouze, když je to nutn

3.4 Návrh GUI

Grafické uživatelské rozhraní je důležitá součást mého projektu. Navrhoval jsem ho následovně. Vytvořil jsem hlavní formulář typu „Jframe“. Chtěl jsem, aby byl design jednoduchý a přehledný, jelikož jsem neměl žádné předchozí zkušenosti s tvorbou GUI. Uživatelé tak snadno pochopí způsob použití aplikace. V hlavním formuláři je vložen „mainPanel“, ve kterém se budou dynamicky měnit panely reprezentující jednotlivé typy otázek. Nachází se zde také záhlaví, kam generuji název kvízu (klíčové slovo Název: v souboru). V zápatí jsem přetáhl 3 tlačítka pojmenované: „Předchozí“, „Konec“ a „Další“. Pomocí nich by se posouvala aktuální otázka, nebo ukončil celý kvíz.

Vytvořil jsem design jednotlivých panelů: „JPanel1of4“ reprezentující otázku se 4 odpověďmi. Na místo otázek a „radioButtonů“ se bude generovat text otázky a možnosti odpovědí. Další připravené panely jsou „JpanelVysledky“, který na konci zobrazí počet dosažených bodů konkrétního uživatele. Dále „JpanelText“, který reprezentuje otázku s textovou odpovědí a čeká na textovou odpověď uživatele. Ve formuláři „mainFrame“ řeším výběr souboru uživatelem. Viz „Výběr souboru pomocí dialogu“

Design všech panelů je „responzivní“. Všechny komponenty se se zvětšením okna roztahují na vhodnou velikost.

V budoucnu plánuji dořešit obsluhu jednotlivých komponentů v GUI a vytvořit třídu „PanelFactory“, která by tvořila panely v závislosti na typu přečtené otázky. Z časových důvodů jsem to v termínu nestihl.

3.5 Výběr souboru pomocí dialogu

Výběr souboru v aplikaci řeším ve třídě „mainFrame“, a to v konstruktoru. Kromě již vygenerovaných povinných metod a příkazů jsem zde přidal příkaz pro vycentrování okna. Je to tak pro uživatele příjemnější. Pokaždé po spuštění aplikace se zobrazí dialogové okno s možností změnit soubor. Nastavuji zde relativní cestu, aby se vždy předem zvolily právě ty soubory, které se nachází ve složce projektu. Pokud uživatel dialog zruší, aplikace se ukončí. Pokud se soubor úspěšně vybere a otevře, přejde se k načtení potřebných dat. Pomocí příkazu „try-catch“ zkouším soubor otevřít a načíst data do seznamu otázek. Ten si pomocí getteru propaguji právě sem do hlavní třídy.

3.6 Zdroje

Jelikož bylo při tvorbě projektu důležité a nutné rozsáhlé samostudium novým konceptů programování (např. tvorba GUI), čerpal jsem z mnoha zdrojů. Hlavní podporou mi byly prezentace a naučná videa Davida Martínka. Pro hlubší pochopení tématu jsem čerpal také na internetu z různých článků, YT videí či z učebnice jazyka Java. Při programování jsem rovněž čerpal z ChatGPT.

Všechny použité zdroje cituji níže.

3.7 Citace použité literatury

HEROUT, Pavel. *Učebnice jazyka Java*. České Budějovice: Kopp, 2018.

Java Project Tutorial - How To Create a Quiz Program In Java NetBeans. Online. In: . 2021. Dostupné z:https://www.youtube.com/watch?v=D0cVLO1AT58&t=16s. [cit. 2024-06-17].

How to Create a Java Quiz using NetBeans. Online. In: . 2020. Dostupné z: https://www.youtube.com/watch?v=epdXrFEpoqQ. [cit. 2024-06-17].

Introduction to GUI Building. Online. Dostupné z: https://netbeans.apache.org/tutorial/main/kb/docs/java/gui-functionality/. [cit. 2024-06-17].

MARTÍNEK, David. Video: Cvičení - Multipanelová aplikace s GUI. Online. In: . [cit. 2024-06-17].

MARTÍNEK, David. Video - Cviko - Java, Tvorba aplikací s GUI. Online. In: . [cit. 2024-06-17].

Kapitola 4 Závěr

Na závěr bych rád shrnul přínos a nedostatky tohoto řešení. Toto řešení bylo navrženo s cílem poskytnout efektivní a užitečný nástroj pro daný problém, což se dle mého názoru podařilo. Přesto jsem si vědom, že existuje několik oblastí, které by mohly být dále vylepšeny. Například by bylo možné optimalizovat výkon aplikace, zlepšit uživatelské rozhraní nebo implementovat pokročilejší funkce.

Tento projekt mi poskytl cenné zkušenosti a prohloubil mé znalosti v dané oblasti. Uvědomuji si, že kdybych měl více času, bych mohl dosáhnout ještě lepších výsledků a důkladněji se věnovat detailům. Bohužel, časová omezení mi neumožnila provést všechna vylepšení, která jsem plánoval.

Toto byla moje první zkušenost s tvorbou grafického uživatelského rozhraní a tvorba rozsáhlejšího projektu v Javě pomocí OOP. Postupem času jsem zjišťoval, že jsem určité věci mohl implementovat jinak a lépe.

I přes tyto výzvy věřím, že toto řešení přineslo užitek a splnilo základní cíle, které jsem si stanovil na začátku projektu. Věřím, že mi tvorba projektu přinesla mnoho nových znalostí a zkušeností. Jsem si jistý, že příští projekt v Javě bude o dost snazší, nejen díky získaným znalostem, ale také díky zkušenostem a dovednostem, které jsem během práce na tomto projektu nabyl.

Je mi líto, že jsem na projekt neměl dostatek času z důvodu mistrovství republiky a absolventského klavírního koncertu. Programování mě moc baví a každý nový projekt beru jako cennou zkušenost v mé kariéře.

Děkuji za příležitost pracovat na tomto projektu a za veškerou podporu, kterou jsem během jeho realizace obdržel. Pevně věřím, že mi v budoucnu tyto zkušenosti pomohou dosáhnout lepších výsledků. Těším se na možnost pokračovat ve zdokonalování tohoto projektu v budoucnosti a na další výzvy, které mě čekají.