Gymnázium, Praha 6, Arabská 14

Obor programování

Ročníkový projekt

Procvičování nepravidelných sloves

Vojtěch Nejedlý

květen 2021

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 Praze dne 3.května 2021 Vojtěch Nejedlý

Název práce: Procvičování nepravidelných sloves

Autor: Vojtěch Nejedlý

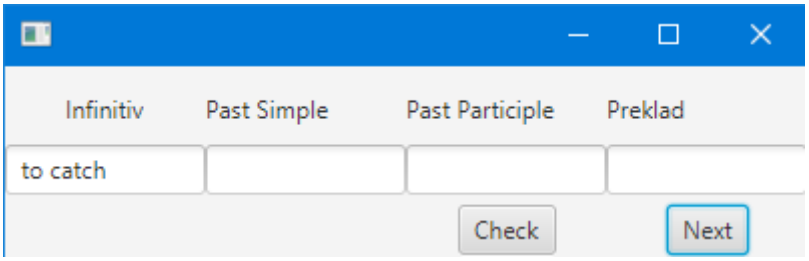
Anotace

Cílem projektu bylo vytvořit program v programovacím jazyce Java, který by pomohl procvičit slovní zásobu anglických nepravidelných sloves a jejich časování. Programu má za úkol vykreslit jednoduchou tabulku, do které uživatel doplní správné tvary sloves a následně bude moct ověřit správnost doplněných sloves. Pokud se splete, bude se moci pokusit opravit tvar, ve kterém si myslí chyboval nebo bude moci přejít na další sloveso.

Obsah

1. Úvod

Program na procvičování sloves se mi zdál jako dobrý nápad, který nebude mít každý. Navíc se mělo jednat o můj první program s grafickým rozhraním, tak jsem nechtěl vymýšlet něco více složitějšího.
Program vždy ukazuje jeden řádek složený ze čtyř textových polí. Do tří z těchto polí uživatel doplní své odpovědi a následně si může zkontrolovat svojí správnost.
Postup řešení funkčnosti mého programu najdete níže.



Obrázek 1

2. Třídy

Program využívá objektově orientované třídy. Třídy jsem rozdělil na následující:

2.1 Slovicko

Třída slovíčko je základní třída, na které stojí ty ostatní. Třída obsahuje konstruktor tvořený Stringy, který je následně použit v dalších třídách.

2.2 Repository

Třída Repository obsahuje konstruktor vytvářející pole z objektů Slovicko, kdy každý objekt Slovicko představuje jeden možný řádek v zobrazované tabulce. Tato třída tedy slouží jako úložiště správných odpovědí.

2.3 FXMLDocumentController

Tato třída obsahuje parametry, které se během běhu programu mohou měnit, a říká, co se má stát, když dojde k interakci s uživatelem. Také je to třída, ve které se dá vše dohromady.

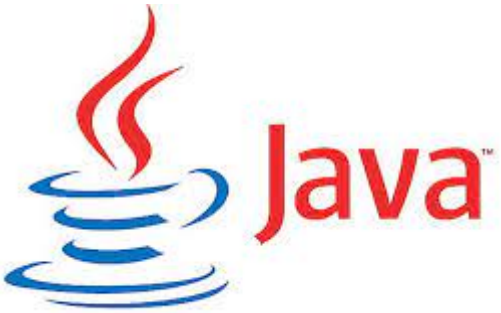
3. Použité knihovny a nástroje

3.1 Použitá grafická knihovna

Jako grafickou knihovnu jsem si zvolil JavaFX. Práci s grafickým rozpoložením objektů mi pomohl program JavaFX Scene builder 2.0 (Obrázek 4).

3.2 Použité nástroje

Během vývoje bylo použito IDE NetBeans 8.2, JDK 8 a JavaFX Scene builder 2.0



Obrázek 2 NetBeans 8.2

Obrázek 3 Java JDK 8

Obrázek 4 JavaFX Scene builder 2.0

4. Tvorba programu

4.1 Popis třídy Slovicko

Třída cílem třídy Slovicko bylo uspořádat Stringy: infinitiv, simple, participle a preklad do objektu, ze kterého se budou moct jednotlivě zavolat getteramy a díky tomu by se dal jeden z nich doplnit do tabulky. Třída Slovicko také obsahuje metodu isCorrect(Obrázek 5), kterou

využívá třída FXMLDocumentController (2.3) pro porovnání uživatelského vstupu s aktuálně vybraným Slovickem.

```
public boolean isCorrect(String infinitiv, String simple, String partici
    return infinitiv.equalsIgnoreCase(this.infinitiv) &&
            simple.equalsIgnoreCase(this.simple) &&
            participle.equalsIgnoreCase(this.participle) &&
            preklad.equalsIgnoreCase(this.preklad);
}
```

Obrázek 5

4.2 Popis třídy Repository

Třída Repository si vytvoří soukromé pole typu Slovicko s názvem slovicka. Třída si ještě vytvoří celočíselnou proměnou lastIndex a proměnou nahoda typu Random. Konstruktor třídy Repository naplní pole Slovicky a následně naplní proměnou nahoda novym Random a nastavy lastIndex na -1.

Repository ještě obsahuje metodu getSlovicko (Obrázek 6), které prvně zkontroluje jestli není slovicka menší než 2. V takovém případě by se program přerušil.
Následně se vytvoří proměná typu int nahodCislo, která se nastaví na náhodné číslo. Náhodné číslo nikdy nebude větší než počet Slovicek v poli. Poté se lastIndex nastaví na nahodCislo. Když nastane při příštím volání metody, že se nahodCislo rovná lastIndex tak se bude generovat nové náhodné číslo dokud nebude jíné než lastIndex a tím se zabrání opakování stejného Slovicka. Pokud bude vše v pořádku, tak metoda vrátí Slovicko z pole slovicka pod indexem nahodCislo.

```
public Slovicko getSlovicko(){
    if(slovicka.length < 2){
        throw  new AssertionError("Pole je moc krátké");
    }
    int nahodCislo = nahoda.nextInt(slovicka.length);
    while(nahodCislo == lastIndex){
        nahodCislo = nahoda.nextInt(slovicka.length);
    }
    lastIndex = nahodCislo;

    return slovicka[nahodCislo];
}
```

Obrázek 6

4.3 Popis třídy FXMLDocumentController

K proměnným, již vytvořeným jsem přidal finální soukromou proměnou typu Repository res a soukromou typu Slovicko current_slovicko. Již vytvořené proměnné typu TextField jsou: inf, pastSim, pastPar, czech a vysledek. Proměné typu Button jsou nextBut a checkBut. Konstruktor nastaví rep na knihovnu sloves Repository.

Tlačítko nextBut při stisknutí zavolá metodu next, která má za úkol vytvořit Slovicko sl za pomoci metody rep.getSlovicko
a current_slovicko se nastavi na sl. Následně si metoda vytvoří String infi, který se nastaví na String infinitiv díky metodě getInfinitiv z třídy Slovicko(4.1). (Obrázek 7)
Následně se začnou nastavovat proměné czech,pastSim a pastPar na prázdný String a proměná inf se nastaví na infi. Zároveň se nastaví, že proměná inf se nebude moct upravovat ručně během běhu programu. Jako poslední krok se nastaví vysledek na prázdný String.
Stisknutí tlačítka checkBut zavolá metodu kontrola, která zkontroluje jestli je současné slovíčko zapsáné správně díky metodě isCorrect(4.1) a podle toho napíše do výsledku správně nebo špatně. (Obrázek 7)
Metoda initialize, která se spustí vždy při zapnutí programu zavolá metodu next.

```
@FXML
void next(ActionEvent event) {
    Slovicko sl = rep.getSlovicko();
    this.current_slovicko = sl;

    String infi = sl.getInfinitiv();

    czech.setText("");
    pastSim.setText("");
    pastPar.setText("");
    inf.setText(infi);
    inf.setEditable(false);
    vysledek.setText("");

}

@FXML
void kontrola(ActionEvent event) {
    if (this.current_slovicko.isCorrect(inf.getText(), pastSim.getText(),
        vysledek.setText("spravně");
    } else {
        vysledek.setText("špatně");
    }

}
```

Obrázek 7

5. Závěr

Myslím, že jsem svůj cíl splnil a projekt má všechny základní části, které jsem si od programu představoval. Řešení jsem očekával složitější, jelikož jsem při zadání nic o grafickém rozhraní nevěděl a vůbec jsem si nedokázal představit, jak to bude vypadat. Vzhled mě však nakonec mile překvapil.
Mezi možné vylepšení do budoucna bych chtěl udělat náhodné vyplnění jednoho ze čtverečku místo neměnného vyplněného jednoho, aby si uživatel procvičil i doplňování infinitivu.

6. Seznam obrázků