

Meno:	Adrián Hládek	
Cvičenie:	Streda 16:00	
Ak. rok	2020/2021	

Objektovo orientované programovanie

Obsah

Zámer projektu z dňa 12.3.2021.....	3
Porovnanie Zámeru oproti finálnej verzii	4
Hierarchia I.....	5
Hierarchia II	6
Spustenie Programu	7
Zapuzdrenie.....	11
Polymorfizmus	12
Generickosť	13
Výnimka.....	14
GUI + OverLoading	15
GitHub anotácie	16

Zámer projektu z dňa 12.3.2021

Z LESA DO NAŠICH DOMOV

Cieľom mojej práce je navrhnuť a zostrojiť výrobný proces jeho plánovanie a riadenie. Hlavným aspektom v tomto prípade je pre nás imaginárna drevárska firma. Jej úlohou je zo surového dreva vytvoriť požadovaný produkt na základe požiadavky zákazníka. Celý proces prebieha 3 fázovo .

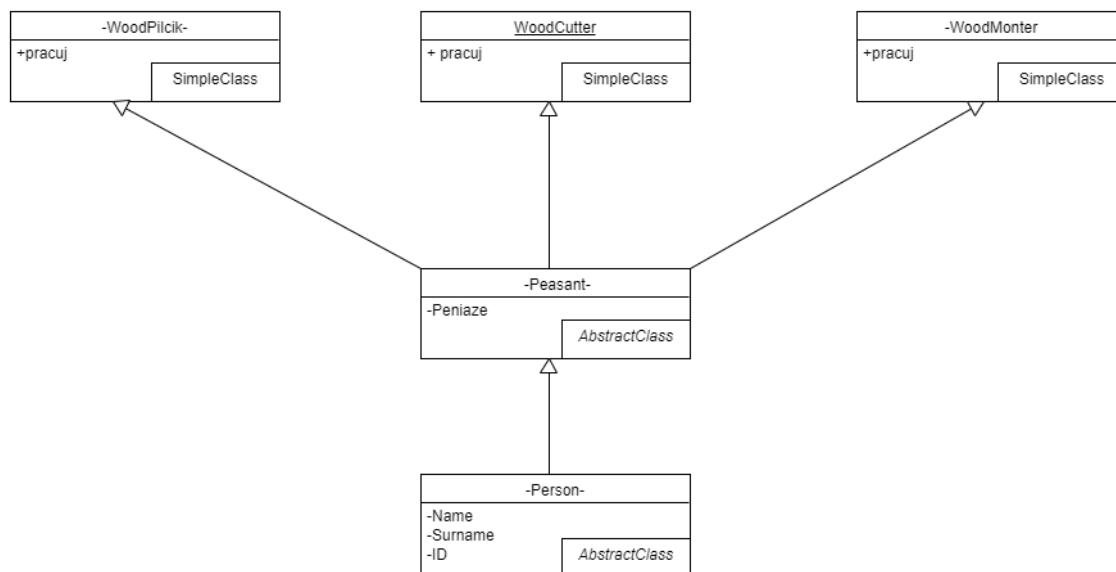
- Prvá fáza zahŕňa Získavanie materiálu
- Do druhej Fázy patrí Transport a distribúcia.
- Do tretej fáze spadá Výroba.

Následne pre každú z uvedených fáz pripadá Nadriadení , ktorý je podriadený Vlastníkovi firmy , v tomto prípade nám. Každá fáza je rozložená na niekoľko pod fáz spolu s určením pracovníkom ktorý k pod fáze prislúcha. K jednej pod fáze môže patriť viac ako 1 pracovník avšak určitý pracovník môže byť priradený/zamestnaný práve iba v jednej z fáz alebo pod fáz . K Pod fázam pre fázu 'Získavanie materiálu' spadá Ťažba, ochrana/zabezpečenie pracoviska a reprodukcia. Transport zahŕňa prepravu materiálu , prepravu produktu a jeho distribúciu zákazníkovi, Taktiež prepravu príslušenstva alebo nástrojov. Fáza výroby ma na starosť spracovať čo najkvalitnejšie a najrýchlejšie Získaný polotovár získaní z prvej fázy a vytvoriť produkt. Výsledný kvalita produktu bude ovplyvnená skúsenosťami a kvalitami pracovníkov , taktiež nástrojmi a príslušenstvom ktoré použijú alebo polotovarom ktorého kvalita závisí od rovnakých aspektov ako kvalita produktu .

Porovnanie Zámeru oproti finálnej verzii

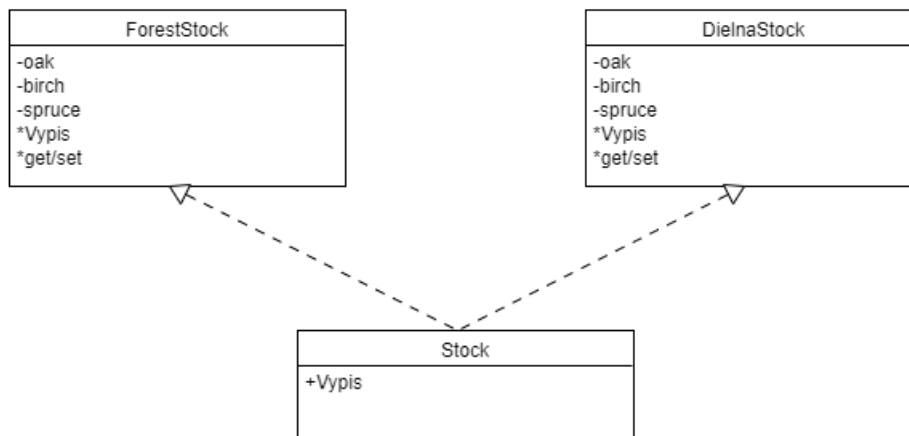
Do finálnej verzie som ponechal 3 fázový proces aj keď s pozmenenými fázami. Pre tieto Fázy nemajú nadriadeného, jediným vedúcim je používateľ. Pracovníkov firmy tvoria predom vygenerovaní ľudia (objekty triedy Peasant) ktorí tvoria väčšiu časť interakcie. Finálnu prácu by sme mohli zaradiť k z časti ku klikacím hrám. Všetky kroky a procesy sme schopný konať iba skrz tlačidlá GUI.

Hierarchia I



Hlavné gro programu tvorí hierarchia odvíjajúca sa od Abstraktnej triedy **Person**. Táto Trieda rozvíja triedu **Peasant** tá následne rozvíja triedy **Woodpilcik**, **WoodCutter**, **WoodMonter** a iné ďalšie menej podstatné. Tieto triedy prekonávajú metódu **pracuj** a dedia getteri a setteri pre svoje atributy.

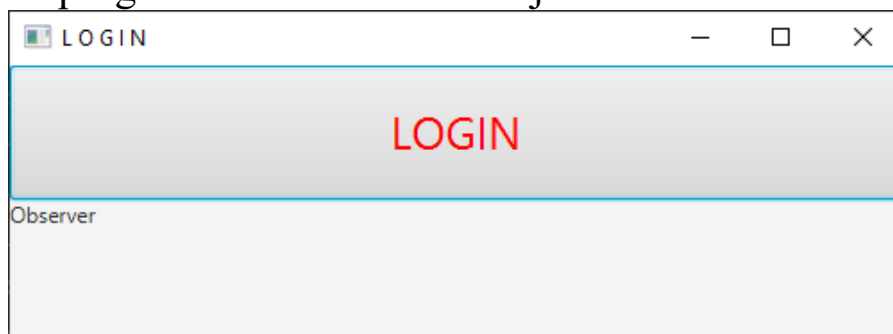
Hierarchia II



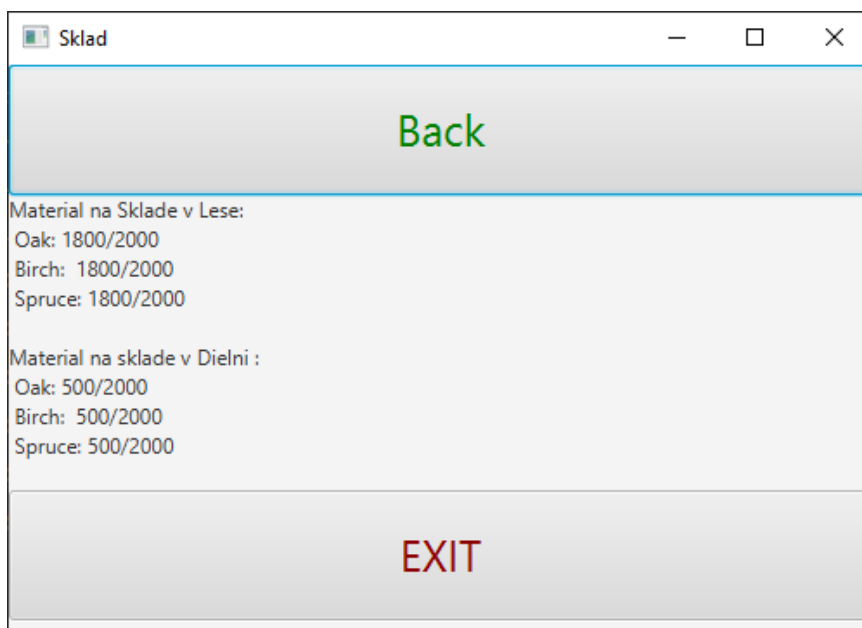
Hierarchia skladov je menej obsiahla prekonáva jednu metódu od rozhrania. Táto hierarchia je taktiež zakomponovaná do hierarchie GUI snímkov.

Spustenie Programu

Po spustení programu sa otvorí nasledujúce okno.

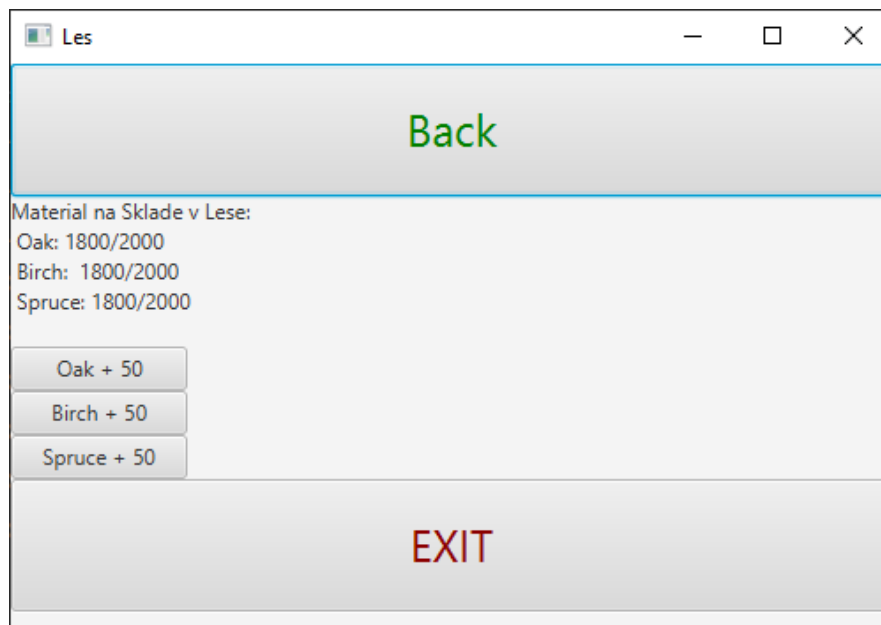


Jediná možnosť interakcie je tlačidlo LOGIN ktorá nás presunie do hlavného menu. V tom sa nachádza 6 tlačidiel z ktorých tlačidlo EXIT nás vráti naspäť do úvodnej obrazovky(screenu), toto tlačidlo sa nachádza vo všetkých scénach GUI okrem úvodnej scény. Ďalej sa tu nachádzajú tlačidlá Shop, Sklad, Les, Dielna, Monterna. Sklad slúži na prehľad aktuálneho materiálu v Skladoch.

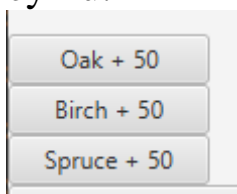


Možeme vidieť že sa tu nachádza už spomínane tlačidlo EXIT, ďalej sa tu nachádza tlačidlo Back ktoré nás vráti do hlavného menu.

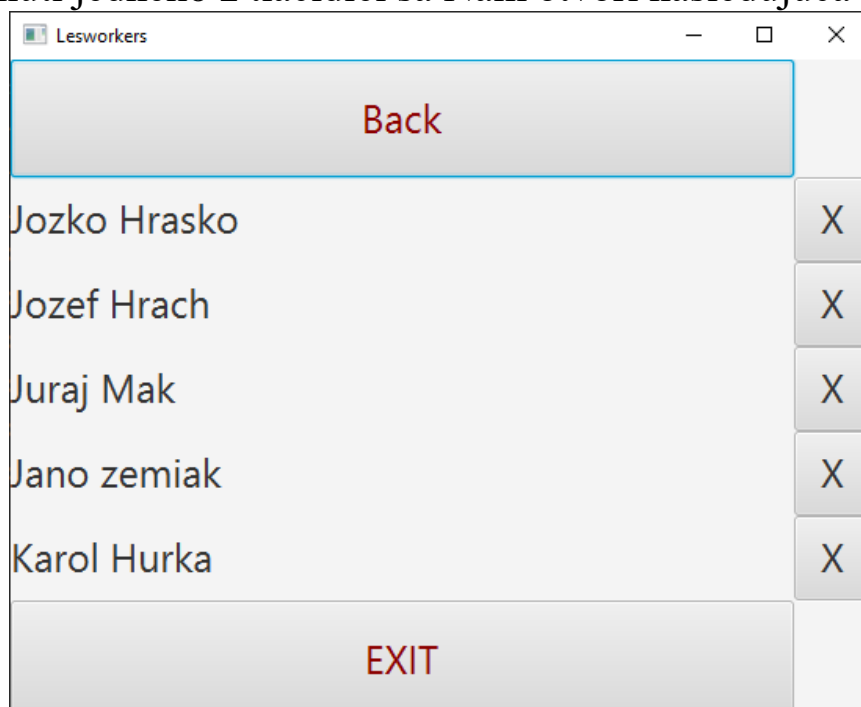
Les je inštanciou kde pracuje object WoodCutter. Ako prvé určujeme ktorý typ dreva ma vytážiť.



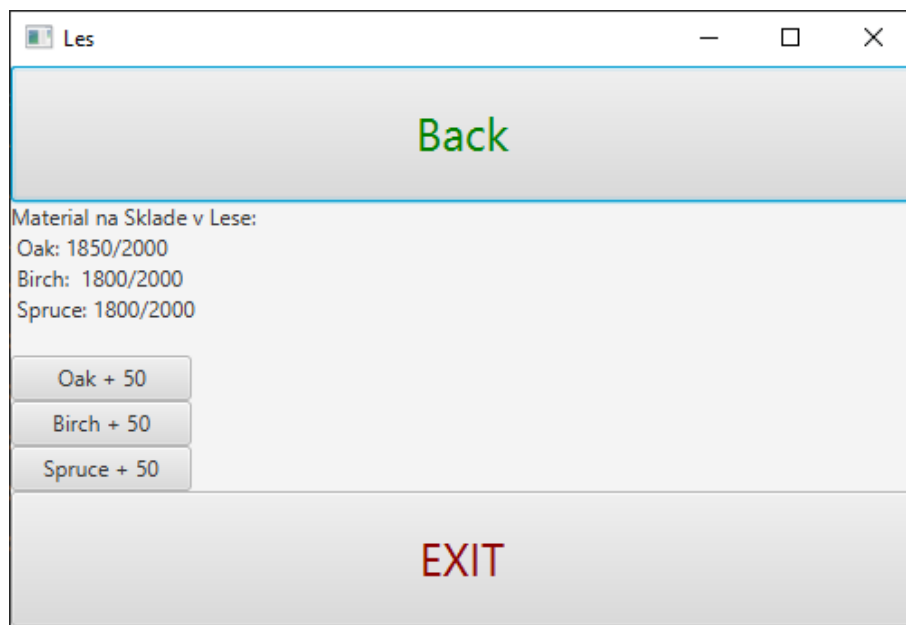
V popise tlačidla vidíme koľko je schopný prispieť počas jedného cyklu.



Po zakliknutí jedného z tlačidiel sa Nám otvorí nasledujúca scéna .



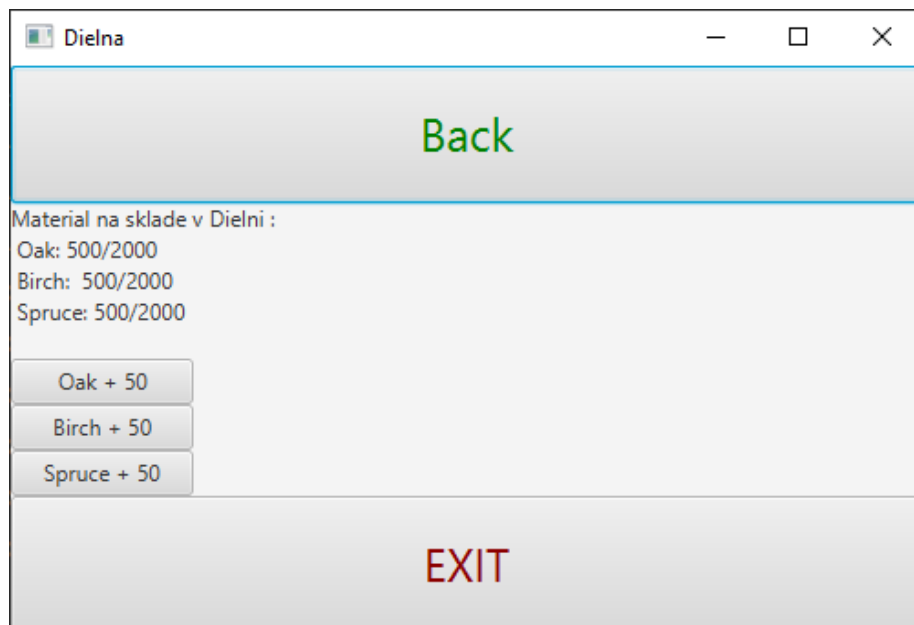
Tu môžeme vidieť Vypísaných všetkých zástupcov triedy Woodcutter . Ktorých tu máme na výber ako na tácke. Po zakliknutí Tlačidla s krížikom v riadku s menom človeka sa vykoná predom vybraná úloha a vykoná ju presne daný človek .Ten za ňu dostane aj zaplatené. Automaticky sa vrátíme do Scény Les.



Môžem si všimnúť že sa výpis Materiálov aktualizoval o danú čiastku.

*Poznámka (V prípade plného skladu 2000/2000 sa nič nevykoná + je tu aj uplatnená výnimka)

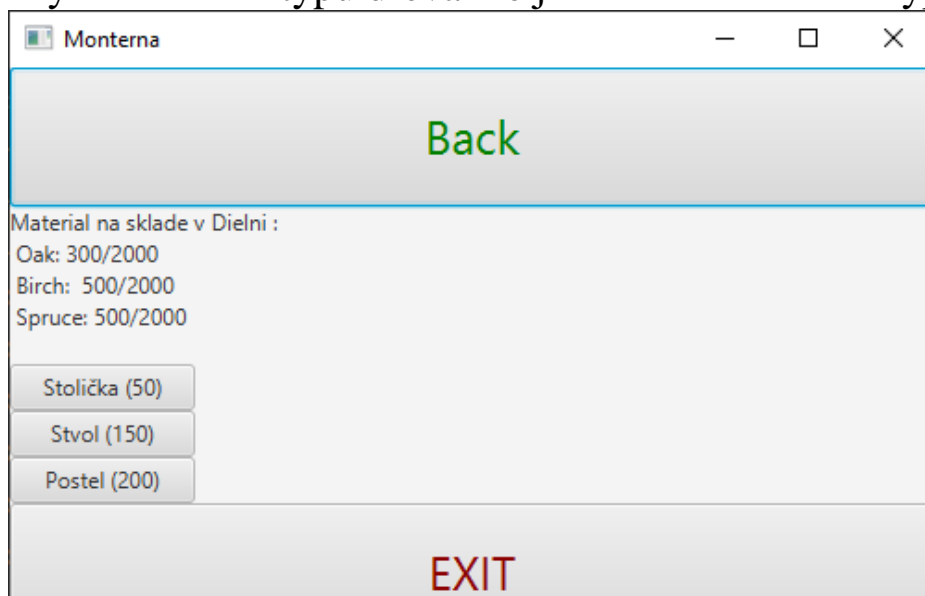
Vraciame sa do menu pomocou tlačidla <Back >následne zaklikávame tlačidlo <Dielna>



Môžeme si všimnúť že sa nachádzame v podobnej scéne ako scéna Les. Rozdiel je ale v tom že podmienky a vyústenia sú tu implikované inak. Pri zakliknutí jedného z tlačidiel dostaneme opäť na výber tento krát ale môžeme byť zamietnutý ako z dôvodu plného skladu, tak aj z dôvodu nedostatku surového dreva ktoré je čerpané zo skladu inštancie LES. Pri tomto kroku teda dochádza iba k prepracovaniu a transportu materiálu. Po zaklikaní tlačidiel sa Aktualizujú obidva sklady.

Tlačidlo <Back>

Ako posledné sa tu nachádza scéna Monterna kde sa vytvárajú dané produkty. Každý produkt ma iné nároky na výroby je ho možné ale urobiť z každého typu dreva nie je ale možné miešať typy.



Zapuzdrenie

```
package sample.Location;

public class DielnaStock implements Stock{
    private static int oak;
    private static int birch;
    private static int spruce;
    public static int getOak() { return DielnaS
    public static void setOak(int count) { Diet
```

```
ava X MonternaStock.java X ForestStock.java X DielnaS
package sample.Worker;

public abstract class Person {

    //private static Person dalsi;
    private String Name;
    private String Surname;
    private int ID;
    private String Povolanie;

    public String getPovolanie() { return
    public void setPovolanie(String povol
    public void setName(String name) { th
    public String getName(){return Name;}

    public void setSurname(String surname)
    public String getSurname(){return Surr
    public void setID(int ID) { this.ID =
    public int getID(){return ID;}
```

Polymorfizmus

```
@Override
```

```
public void render() {
```

```
    Button btn = new Button(text: "Back");
```

```
    btn.setOnAction(e->{
```

```
        System.out.println("->Login");
```

```
    });
```

Generickost'

```
public class DataClass {  
  
    ArrayList<WoodCutter> drevorubaci = new ArrayList<>();  
    ArrayList<BigBoss> bosik = new ArrayList<>();  
    ArrayList<WoodPilcici> pilcici = new ArrayList<>();  
    ArrayList<WoodMonter> monter = new ArrayList<>();  
    ArrayList<Produkt> stvol = new ArrayList<>();  
    ArrayList<Produkt> stolicka = new ArrayList<>();  
    ArrayList<Produkt> postel = new ArrayList<>();  
  
    public DataClass(ArrayList<WoodCutter> drevorubaci,  
        this.drevorubaci = drevorubaci;  
        this.pilcici = pilcici;  
        this.bosik = bosik;  
        this.monter = monter;  
        this.stvol = stvol;  
        this.stolicka = stolicka;  
        this.postel = postel;  
    }  
}
```

Výnimka

```
package sample;
```

```
public class PlnySklad extends Exception{  
}
```

```
Button Oak = new Button (text: "Oak + 50");  
Oak.setOnAction(e->{
```

```
    int pocet = ForestStock.getOak();
```

```
    if (pocet < 2000)  
    {
```

```
        try {
```

```
            Router.goTo(ROUTES.LESWORKEROAK, this.data);
```

```
        } catch (IOException ioException) {
```

```
            ioException.printStackTrace();
```

```
        } catch (ClassNotFoundException classNotFoundException) {
```

```
            classNotFoundException.printStackTrace();
```

```
        }
```

```
    }
```

```
    else
```

```
        try {
```

```
            throw new PlnySklad();
```

```
        } catch (PlnySklad plnySklad) {
```

```
            plnySklad.printStackTrace();
```

```
        }
```

GUI + OverLoading

```
package sample.router;
```

```
public enum ROUTES {
```

```
    SHOP, LOGIN, WELCOME, SKLAD, LES, DIELNA, MONTERNA, TOVAR, LESWORKER,  
    LESWORKEROAK, LESWORKERBIRCH, LESWORKERSPRUCE, DIELNAWORKEROAK,  
    DIELNAWORKERBIRCH, DIELNAWORKERSPRUCE,  
    MONTERNAWORKEROAK, MONTERNAWORKERBIRCH, MONTERNAWORKERSPRUCE,  
    VYBERWOOD, MONTERNAWORKERPOSTEL, MONTERNAWORKERSTOLICKA, MONTERNAWORKERSTVOL,  
    STOLICKAMONTER, STVOLMONTER, POSTELMONTER
```

```
}
```

```
public class Router {
```

```
    public static Stage window;
```

```
    /**
```

```
     *
```

```
     * @param route
```

```
     * @param data
```

```
     * @throws IOException
```

```
     * @throws ClassNotFoundException
```















```
    */
```

```
    public static void goTo(ROUTES route, DataClass data) throws IOException, ClassNotFoundException {...}
```

```
    public static void goTo(ROUTES route, DataClass data, String wood) throws IOException, ClassNotFoundException  
    {...}
```

```
    public static void goTo(ROUTES route, String nabytok, DataClass data) throws IOException, ClassNotFoundException  
    {...}
```

GitHub anotácie

	joj  Adrian-Hladek committed on 13 Apr
	init  Adrian-Hladek committed on 13 Apr
	Add files via upload  Adrian-Hladek committed on 13 Apr
	Update README.md  Adrian-Hladek committed on 13 Apr
	Create README.md  Adrian-Hladek committed on 13 Apr
	Add files via upload  Adrian-Hladek committed on 13 Apr
	Delete Main.java  Adrian-Hladek committed on 13 Apr
	Delete README.md  Adrian-Hladek committed on 13 Apr
	Delete predbezna_verzia.iml  Adrian-Hladek committed on 13 Apr
	Delete MarketStock.class  Adrian-Hladek committed on 13 Apr
	Delete Market.class  Adrian-Hladek committed on 13 Apr
	Delete ForestStock.class  Adrian-Hladek committed on 13 Apr
	Delete Forest.class  Adrian-Hladek committed on 13 Apr
	Update README.md  Adrian-Hladek committed on 13 Apr