Fakulta informatiky a informačných technológií,

Slovenská technická univerzita, Ilkovičova 2, Bratislava, 842 16

Zadanie 3 –

„Predajňa stromov“

Objektovo orientované programovanie

Cvičenia: Streda 14:00-15:50

Cvičiaci: Ing. Anna Považanová

autor:
Riccardo Kiss
ročník štúdia: **druhý**

Bratislava
2020

Zadanie 3 – Záverečné odovzdanie

- Zámer

Zámer projektu je simulovať chod predajne stromov. V predajni budú zamestnaní predavači, účtovníci a vedúci. Všetci zamestnanci budú mať základné atribúty ako napr. poradové číslo, vek a pohlavie. Predavači majú na starosť predaj stromov zákazníkom, preberajú peniaze od zákazníkov, ktoré potom prepošlú ďalej účtovníkom. Účtovníci zaobchádzajú s firemným rozpočtom, t.j. kúpa drevín od dodávateľov a pripočítavanie zárobkov od zákazníkov. Takže prístup k firemnému rozpočtu majú len účtovníci. Vedúci je len jeden človek a jeho úlohou je najímať ostatných zamestnancov. Jediná podmienka, ktorú zohľadňuje je, aby mali nad 18 rokov. Zákazníci disponujú peniazmi, ktorými nakupujú.

Sklad obchodu môže obsahovať rôzne druhy stromov (listnaté – dub, breza, javor, lipa, jabloň, …; ihličnaté – smrek, borovica, jedľa, …). Každý strom bude mať určité vlastnosti (napr. výška, vek a cena). Rôzne druhy stromov majú rozličné ceny. Výška a vek stromov môžu byť náhodne generované hodnoty. Celková cena stromu je potom závislá od všetkých týchto atribút (napr. nízky strom bude lacnejší ako vysoký).

Príklad klasickej interakcie medzi zákazníkom a predajňou:
- Zákazník nakúpi strom od predavača za svoje peniaze.
- Predavač ďalej prepošle peniaze, ktoré zarobil, účtovníkovi.
- Účtovník pričíta zárobky predavačov do firemného rozpočtu.

- Balíky

1. **treeShop.main** – hlavná trieda, ktorá je zodpovedná za chod programu
   1.1. Main.java
   1.2. GUI.java
   1.3. GenericList.java
   1.4. Stopky.java
   1.5. LambdaInfo.java
2. **treeShop.predajna** – balík tried zamestnancov
   2.1. FiremnyRozpocet.java
   2.2. Zamestnanec.java
   2.3. Veduci.java
   2.4. Uctovnik.java
   2.5. Predavac.java
   2.6. Observer.java
3. **treeShop.stromy** – balík tried stromov
   3.1. Strom.java
   3.2. ListnatyStrom.java
   3.3. IhlicnatyStrom.java
4. **treeShop.zakaznici** – balík tried pre zákazníkov
   4.1. Zakaznik.java
5. *treeShop_UMLdiagram*.uxf – diagram tried pomocou UMLet-u

- Použité princípy

1. Trieda *(class)*

   V projekte mám aktuálne vytvorených 15 tried. Napr.

   *public class **Zakaznik()** {...}*

2. Objekt *(object)*

   Každá trieda ma svoje vlastné inštancie. Napr.

   *FiremnyRozpocet **fr** = new FiremnyRozpocet (...)*

3. Zapuzdrenie *(encapsulation)*

   V rodičovských triedach (Zamestnanec.java, Strom.java) používam modifikátory atribútov *„protected“*, inak v ostatných triedach *„private“* a pristupujem k nim pomocou **get** a **set** metód, ktoré sú *„public“*. Napr.

   *private double **peniaze**;*
   *public double **getPeniaze()** {return **peniaze**;}*

4. Preťažovanie *(overloading)*

   Mám preťažené konštruktory v triedach (napr. Zamestnanec, Veduci, Uctovnik, ...). Napr.:

   *public **Uctovnik(**int vek**)** {...}*
   *public **Uctovnik(**boolean pohlavie, int vek**)** {...}*
   *public **Uctovnik(**boolean pohlavie, int vek, double peniaze**)** {...}*

5. Prekonávanie *(overriding)*

   V triedach Veduci, Uctovnik, Predavac mám metódu **info()**, ktorá prekonáva metódu **info()** z triedy nadtypu Zamestnanec: .

   public String info() { return this.isPohlavie() ? ... ;}

6. Dedičnosť *(inheritance)*

   V projekte som použil dedenie 5-krát: Triedy Veduci (*Veduci.java*), Uctovnik (*Uctovnik.java*) a Predavac (*Predavac.java*) dedia atribúty a metódy z triedy Zamestnanec (*Zamestnanec.java*), a triedy IhlicnatyStrom (*IhlicnatyStrom.java*) a ListnatyStrom (*ListnatyStrom.java*) dedia z triedy Strom (*Strom.java*). Napr.

   *public class Veduci **extends** Zamestnanec {...}*
   *public class IhlicnatyStrom **extends** Strom {...}*

7. Agregácia *(aggregation)*

   Referencia na objekt triedy FiremnyRozpocet (*FiremnyRozpocet.java*) v triede Uctovnik (*Uctovnik.java*). Napr.

   *public class **Uctovnik** extends **Zamestnanec** {*
   *private static **FiremnyRozpocet** fr = new **FiremnyRozpocet**(0);*

8. Abstraktná trieda *(abstract class)*

   Trieda Zamestnanec (*Zamestnanec.java*) je abstraktnou triedou, lebo nikde v projekte sa nevytvára jej nová inštancia **new Zamestnanec(...)**.

   *public **abstract** class **Zamestnanec** {*
   ...
   }

9. Rozhranie (*interface*)

Rozhranie Observer (*Observer.*java) mám implementované v triede Účtovník (*Uctovnik.java*) v hlavičke triedy pomocou kľúčového slova **implements**. Každá z tried, ktoré implementujú zadefinované metódy z rozhrania ich tak môžu prekonávať (*overriding*) podľa vlastnej potreby.

```
public abstract interface Observer {
        public abstract void update();
}

public class Uctovnik extends Zamestnanec implements Observer {
        ...
}
```

10. Výnimky (*exception handling*)

Výnimky mám použité v grafickom rozhraní pri zadávaní čísla do textového poľa. Od užívateľa sa očakáva, že zadá číselnú hodnotu. V prípade zadania písmen vypíše, že zadal neplatný vstup.

```
try {
        vybratyVek = Integer.parseInt(vyberVek.getText());
        ...
}
catch(NumberFormatException e) {
        vypisInfo.setText("Neplatný vstup! Zadajte iné hodnoty!");
        e.printStackTrace();
}
```

11. Návrhový vzor **Observer**

V triede FiremnyRozpocet je vytvorené pole observerov, ktorí su inštanciami triedy Uctovnik. Uctovnik reaguje na každú zmenu stavu firemného rozpočtu a vypíše aktuálnu sumu firemného rozpočtu.

12. Grafické rozhranie (*GUI*)

Mám vytvorenú vlastnú triedu GUI, kde sú zadeklarované metódy pre jednotlivé scény. Grafické rozhranie sa spúšťa z triedy Main:

```
public static void main(String[] args) {
        Application.launch(GUI.class, args);
}
```

13. Spracovatelia udalostí (*event handlers*)

V triede GUI mám vytvorené tlačidlá, ktoré majú všetky event handler na akciu stlačenia tlačidla. Napr.:

```
vytvorRandom.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent actionEvent) {
                ...
        }
});
```

14. Viacniťovosť (*multithreading*)

Jedna niť sa automaticky spustí po zavolaní ***Application***.*launch(****GUI****.class, args)*, konkrétne JavaFX Application Thread. Ďalšiu niť mám explicitne vytvorenú v triede GUI. Táto niť slúži na meranie času počas celého behu programu na pozadí.

```
Stopky stopky = new Stopky();
Thread threadStopky = new Thread(stopky);
```

15. Generickosť

Mám vytvorenú triedu GenericList, ktorou môžem vytvárať polia rôznych objektov.

```
public class GenericList<T> {
        private List<T> list = new ArrayList<T>();
        ...
}
```

16. Lambda výrazy (*lambda expressions*)

V triedach zamestnancov (napr. Veduci) mám

```
public String info() {
        LambdaInfo infoVeduci = () -> this.isPohlavie() ? "[Vedúci@..." : "[Vedúca@... ";
        return infoVeduci.vypisInfo();
}
```

17. Implicitná implementácia metód v rozhraní (*default method implementation*)

V rozhraní Observer mám implicitne implementovanú metódu update() pomocou kľúčového slova **default** na vypísanie aktuálnej sumy vo firemnom rozpočte.

```
public abstract interface Observer {
        default void update(Uctovnik u) {
                if(u.isPohlavie())
                        System.out.println("[Účtovník@" + ... );
                else
                        System.out.println("[Účtovníčka@" + ... );
}
```

18. Serializácia (*serialization*)

V triede GUI si ukladám a čítam prvotne vytvorené objekty tried Veduci a Uctovnik z textového súboru **output.txt** takže mám uložené ich informácie po každom spustení aplikácie.

```
ObjectInputStream ois = new ObjectInputStream(new FileInputStream(f));
Veduci povodnyVeduci = (Veduci) ois.readObject();
Uctovnik povodnyUctovnik = (Uctovnik) ois.readObject();
ois.close();
```

19. Javadoc komentáre

V celom projekte mám použité **javadoc** komentáre na popísanie tried a aj niektorých metód. Napr.:

```
/**
 * Trieda <code>Main</code> na spustenie aplikacie.
 *
 * @author Riccardo Kiss
 * @version 2.0.0
```

```
 * @since May 24, 2020
 */
```

- Funkčnosť

Spustenie aplikácie z Main.java otvorí grafické rozhranie. Vytvorenie človeka je možné manuálne (zadávaním vlastných hodnôt) alebo aj automaticky náhodne vygenerovanými hodnotami. V prvom rade treba skontrolovať súbor **output.txt** v koreňovom priečinku projektu.

1. *Output.txt* **je** prázdny:

   Prvá scéna slúži na vytvorenie vedúceho pre predajňu. Po vytvorení vedúceho nasleduje scéna na vytvorenie účtovníka. Po vytvorení účtovníka (ktorého najíma nami vytvorený vedúci z predošlej scény) sa ešte pred prepnutím do hlavnej scény uložia objekty **Veduci** a **Uctovnik** do textového súboru **output.txt**.

2. *Output.txt* **nie je** prázdny:

   Keďže už máme vytvoreného vedúceho aj účtovníka z prvého spustenia aplikácie, nemusíme ich vytvárať znovu, ale rovno sa načítajú objekty z textového súboru **output.txt.** Hneď po spustení prvej scény na vytvorenie vedúceho môžeme rovno kliknúť na „Ďalej“ a aplikácia preskočí scénu na vytvorenie účtovníka a prepne sa rovno do hlavnej scény.

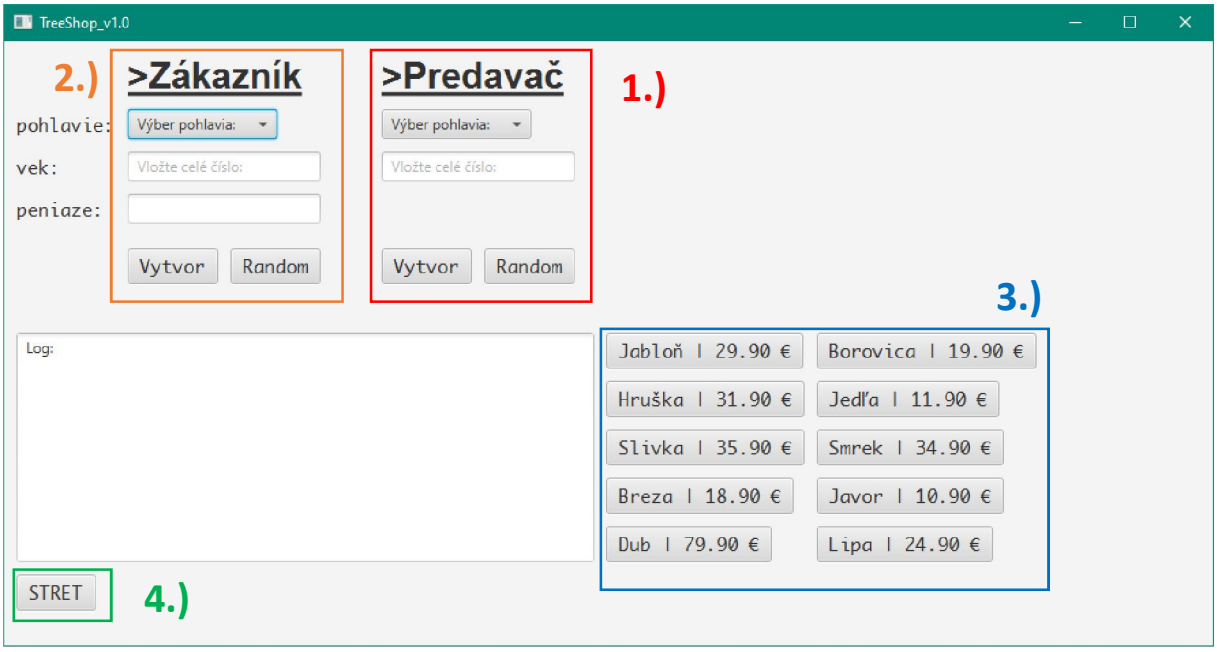
Hlavná scéna (viď. obrázok nižšie):

V hlavnej scéne je najskôr potrebné vytvoriť predavačov **(1.)** a až následne potom zákazníkov **(2.)**. Nikdy nemôže byť väčší počet zákazníkov ako predavačov. Môžeme si to predstaviť tak, že aj v skutočnosti môže naraz každý predavač obslúžiť len jedného zákazníka. Ak chceme vytvoriť zákazníka, po ktorom by ich počet bol väčší ako počet predavačov (vypíše aj varovnú správu), tak musíme najskôr vytvoriť ďalšieho predavača.

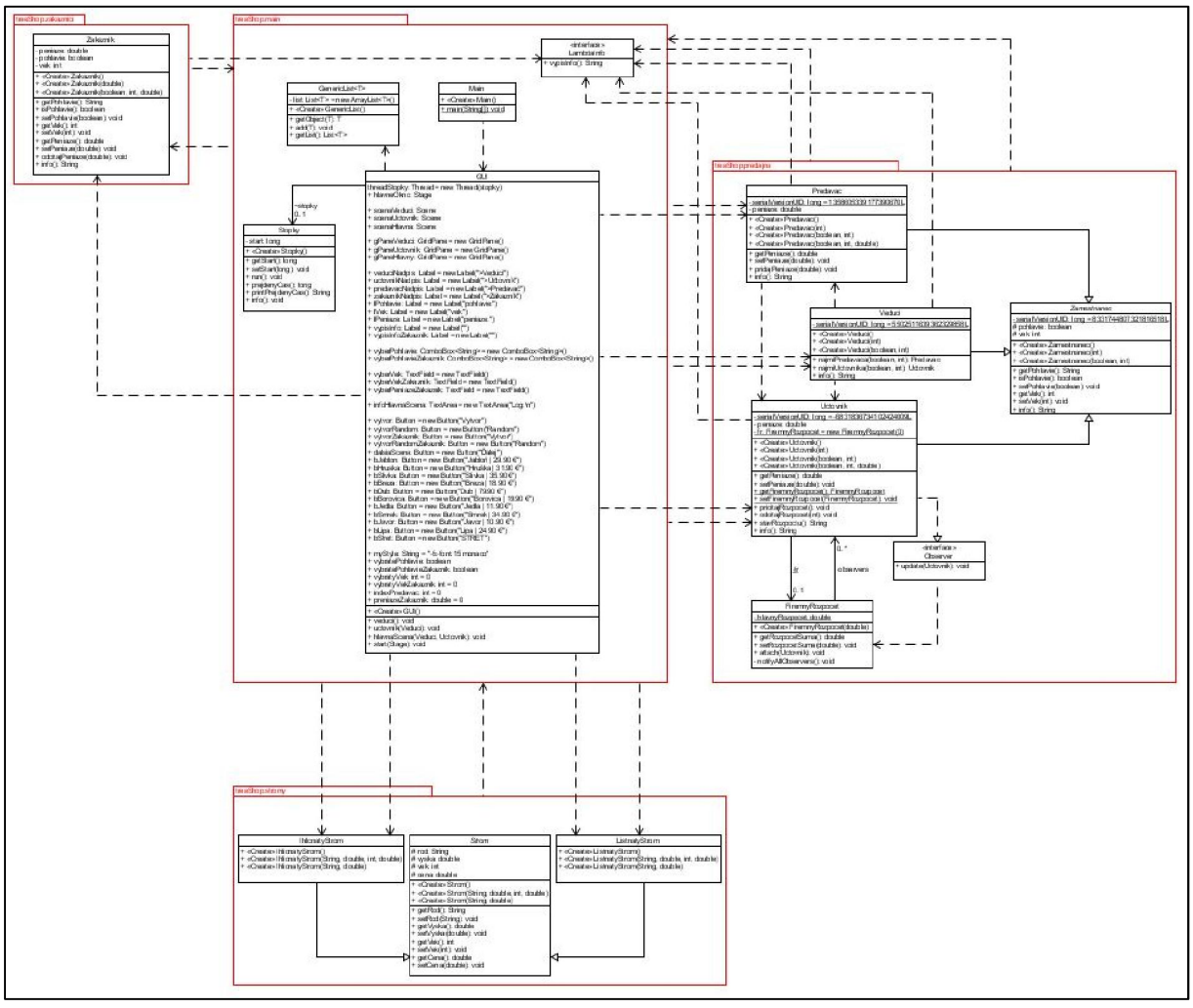
Po každom vytvorení zákazníka si môžeme vybrať z ponuky stromov na nákup, ktoré si bude daný zákazník kupovať **(3.)**. Po vytvorení ďalšieho zákazníka už vyberáme nákup pre tohto aktuálne vytvoreného a k predchádzajúcim zákazníkom sa už nevieme vrátiť a tým pádom nevieme zmeniť už ich nákup.

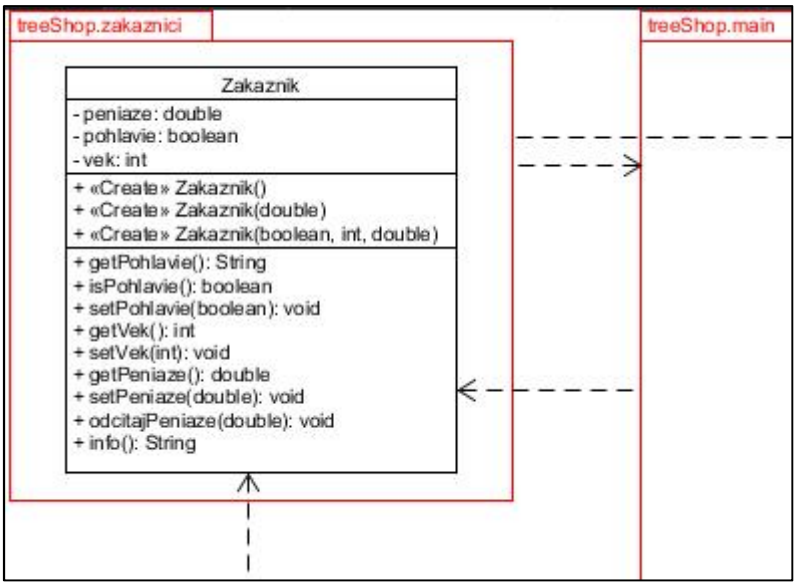
Ak už nechceme ďalej nakupovať, tak po stlačení tlačidla „**STRET“ (4.)** sa všetky peniaze, ktoré každý predavač zarobil od každého zákazníka prepošle každý predavač účtovníkovi, ktorý ich následne pričíta do firemného rozpočtu.

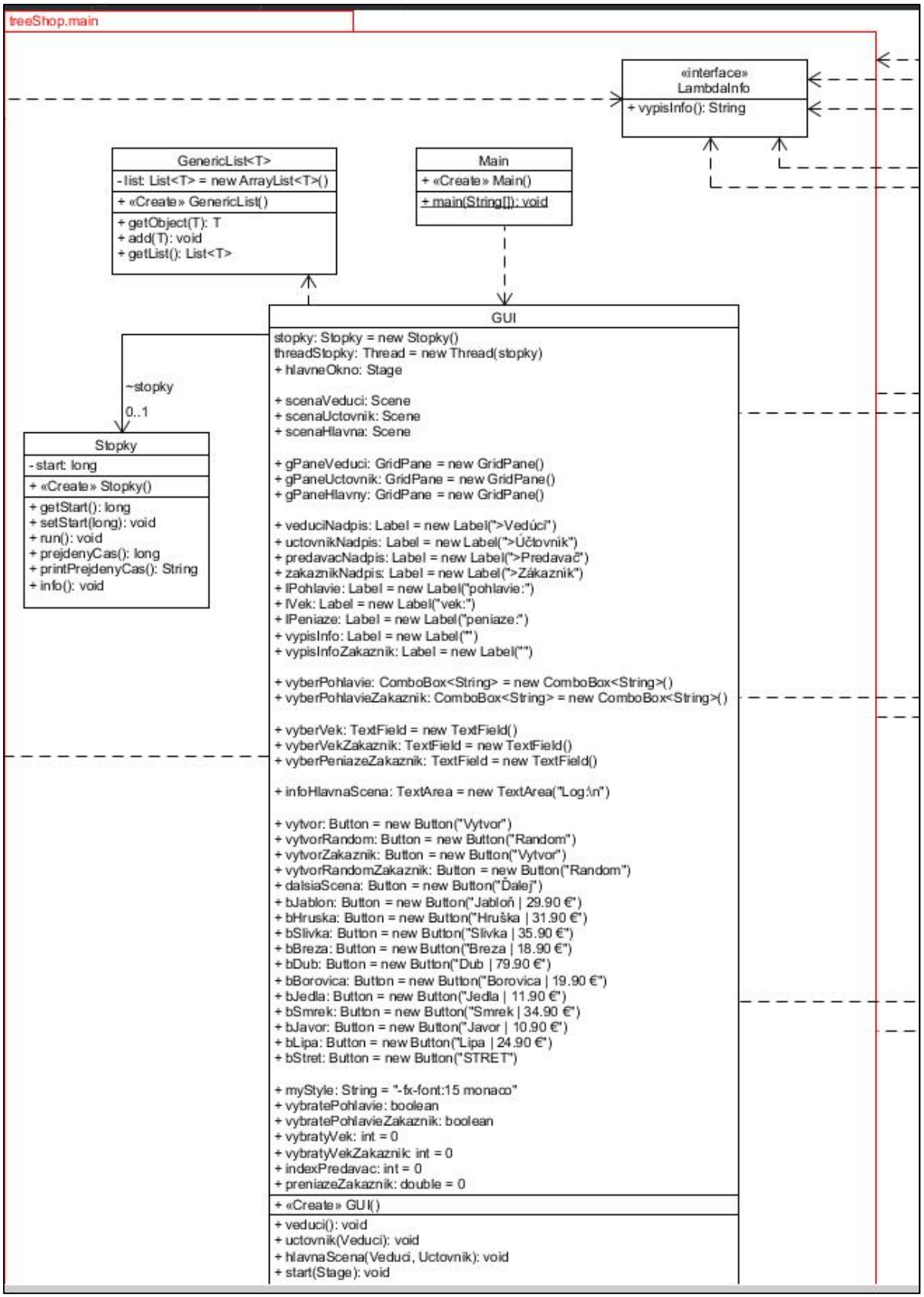
Zoznam zákazníkov sa po strete vymaže, ale zoznam predavačov zostáva a predaj stromov môže ďalej pokračovať vytváraním nových zákazníkov a kupovaním stromov.

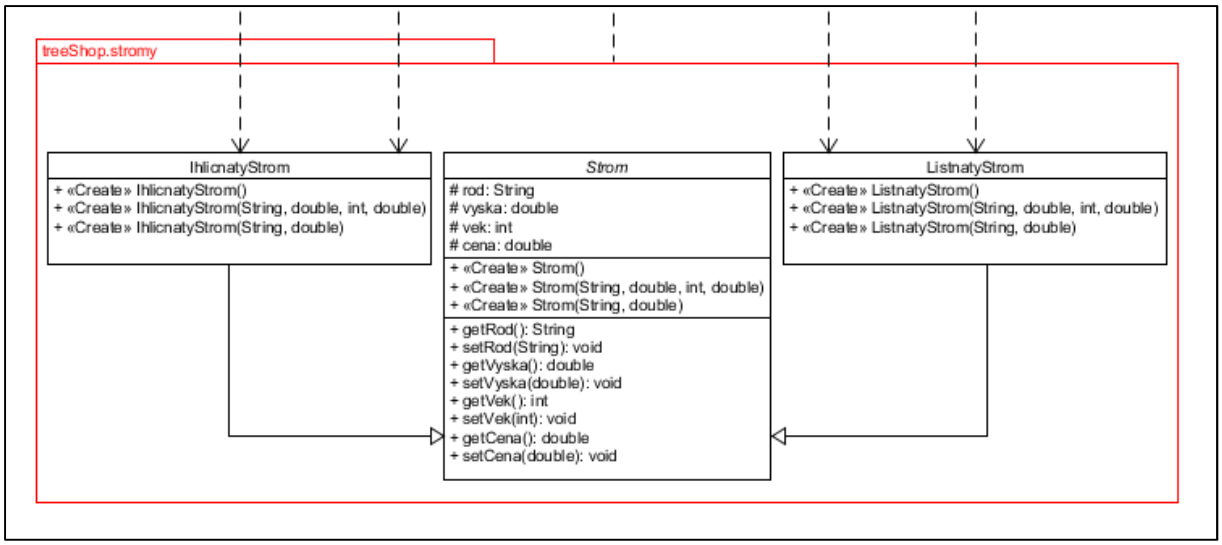

TreeShop_v1.0
2.)
>Zákazník
pohlavie:
Výber pohlavia:
vek:
Vložte celé číslo:
peniaze:
Vytvor
Random
>Predavač
Výber pohlavia:
Vložte celé číslo:
Vytvor
Random
1.)
3.)
Log:
Jabloň | 29.90 €
Borovica | 19.90 €
Hruška | 31.90 €
Jedľa | 11.90 €
Slivka | 35.90 €
Smrek | 34.90 €
Breza | 18.90 €
Javor | 10.90 €
Dub | 79.90 €
Lipa | 24.90 €
STRET
4.)

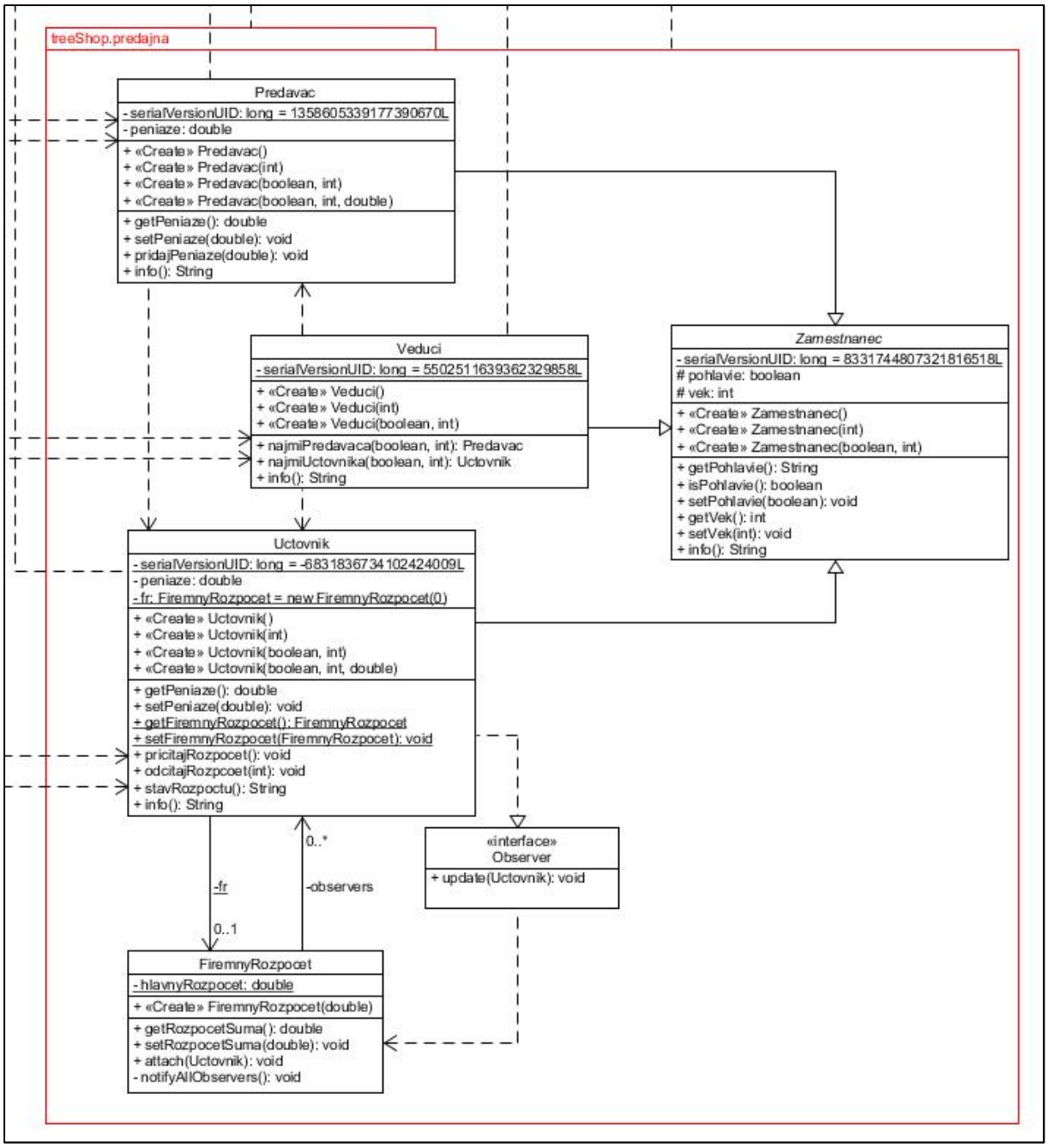
- UML diagram




treeShop.zakaznici
treeShop.main
Zakaznik
- peniaze: double
- pohlavie: boolean
- vek: int
+ «Create» Zakaznik()
+ «Create» Zakaznik(double)
+ «Create» Zakaznik(boolean, int, double)
+ getPohlavie(): String
+ isPohlavie(): boolean
+ setPohlavie(boolean): void
+ getVek(): int
+ setVek(int): void
+ getPeniaze(): double
+ setPeniaze(double): void
+ odcitajPeniaze(double): void
+ info(): String


treeShop.main
«interface»
LambdaInfo
+ vypisInfo(): String
GenericList<T>
- list: List<T> = new ArrayList<T>()
+ «Create» GenericList()
+ getObject(T): T
+ add(T): void
+ getList(): List<T>
Main
+ «Create» Main()
+ main(String[]): void
GUI
stopky: Stopky = new Stopky()
threadStopky: Thread = new Thread(stopky)
+ hlavneOkno: Stage
+ scenaVeduci: Scene
+ scenaUctovnik: Scene
+ scenaHlavna: Scene
+ gPaneVeduci: GridPane = new GridPane()
+ gPaneUctovnik: GridPane = new GridPane()
+ gPaneHlavny: GridPane = new GridPane()
+ veduciNadpis: Label = new Label(">Vedúci")
+ uctovnikNadpis: Label = new Label(">Účtovník")
+ predavacNadpis: Label = new Label(">Predavač")
+ zakaznikNadpis: Label = new Label(">Zákazník")
+ lPohlavie: Label = new Label("pohlavie:")
+ lVek: Label = new Label("vek:")
+ lPeniaze: Label = new Label("peniaze:")
+ vypisInfo: Label = new Label("")
+ vypisInfoZakaznik: Label = new Label("")
+ vyberPohlavie: ComboBox<String> = new ComboBox<String>()
+ vyberPohlavieZakaznik: ComboBox<String> = new ComboBox<String>()
+ vyberVek: TextField = new TextField()
+ vyberVekZakaznik: TextField = new TextField()
+ vyberPeniazeZakaznik: TextField = new TextField()
+ infoHlavnaScena: TextArea = new TextArea("Log:\n")
+ vytvor: Button = new Button("Vytvor")
+ vytvorRandom: Button = new Button("Random")
+ vytvorZakaznik: Button = new Button("Vytvor")
+ vytvorRandomZakaznik: Button = new Button("Random")
+ dalsiaScena: Button = new Button("Ďalej")
+ bJablon: Button = new Button("Jabloň | 29.90 €")
+ bHruska: Button = new Button("Hruška | 31.90 €")
+ bSlivka: Button = new Button("Slivka | 35.90 €")
+ bBreza: Button = new Button("Breza | 18.90 €")
+ bDub: Button = new Button("Dub | 79.90 €")
+ bBorovica: Button = new Button("Borovica | 19.90 €")
+ bJedla: Button = new Button("Jedla | 11.90 €")
+ bSmrek: Button = new Button("Smrek | 34.90 €")
+ bJavor: Button = new Button("Javor | 10.90 €")
+ bLipa: Button = new Button("Lipa | 24.90 €")
+ bStret: Button = new Button("STRET")
+ myStyle: String = "-fx-font:15 monaco"
+ vybratePohlavie: boolean
+ vybratePohlavieZakaznik: boolean
+ vybratyVek: int = 0
+ vybratyVekZakaznik: int = 0
+ indexPredavac: int = 0
+ preniazeZakaznik: double = 0
+ «Create» GUI()
+ veduci(): void
+ uctovnik(Veduci): void
+ hlavnaScena(Veduci, Uctovnik): void
+ start(Stage): void
~stopky
0..1
Stopky
- start: long
+ «Create» Stopky()
+ getStart(): long
+ setStart(long): void
+ run(): void
+ prejdenyCas(): long
+ printPrejdenyCas(): String
+ info(): void


treeShop.stromy
IhlicnatyStrom
+ «Create» IhlicnatyStrom()
+ «Create» IhlicnatyStrom(String, double, int, double)
+ «Create» IhlicnatyStrom(String, double)
Strom
rod: String
vyska: double
vek: int
cena: double
+ «Create» Strom()
+ «Create» Strom(String, double, int, double)
+ «Create» Strom(String, double)
+ getRod(): String
+ setRod(String): void
+ getVyska(): double
+ setVyska(double): void
+ getVek(): int
+ setVek(int): void
+ getCena(): double
+ setCena(double): void
ListnatyStrom
+ «Create» ListnatyStrom()
+ «Create» ListnatyStrom(String, double, int, double)
+ «Create» ListnatyStrom(String, double)


treeShop.predajna
Predavac
- serialVersionUID: long = 1358605339177390670L
- peniaze: double
+ «Create» Predavac()
+ «Create» Predavac(int)
+ «Create» Predavac(boolean, int)
+ «Create» Predavac(boolean, int, double)
+ getPeniaze(): double
+ setPeniaze(double): void
+ pridajPeniaze(double): void
+ info(): String
Veduci
- serialVersionUID: long = 5502511639362329858L
+ «Create» Veduci()
+ «Create» Veduci(int)
+ «Create» Veduci(boolean, int)
+ najmiPredavaca(boolean, int): Predavac
+ najmiUctovnika(boolean, int): Uctovnik
+ info(): String
Zamestnanec
- serialVersionUID: long = 8331744807321816518L
pohlavie: boolean
vek: int
+ «Create» Zamestnanec()
+ «Create» Zamestnanec(int)
+ «Create» Zamestnanec(boolean, int)
+ getPohlavie(): String
+ isPohlavie(): boolean
+ setPohlavie(boolean): void
+ getVek(): int
+ setVek(int): void
+ info(): String
Uctovnik
- serialVersionUID: long = -6831836734102424009L
- peniaze: double
- fr: FiremnyRozpocet = new FiremnyRozpocet(0)
+ «Create» Uctovnik()
+ «Create» Uctovnik(int)
+ «Create» Uctovnik(boolean, int)
+ «Create» Uctovnik(boolean, int, double)
+ getPeniaze(): double
+ setPeniaze(double): void
+ getFiremnyRozpocet(): FiremnyRozpocet
+ setFiremnyRozpocet(FiremnyRozpocet): void
+ pricitajRozpocet(): void
+ odcitajRozpcoet(int): void
+ stavRozpoctu(): String
+ info(): String
«interface»
Observer
+ update(Uctovnik): void
-fr
0..1
0..*
-observers
FiremnyRozpocet
- hlavnyRozpocet: double
+ «Create» FiremnyRozpocet(double)
+ getRozpocetSuma(): double
+ setRozpocetSuma(double): void
+ attach(Uctovnik): void
- notifyAllObservers(): void