UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

# Účetní a fakturovací software

Vratislav Marek

Bakalářská práce
2013

**Univerzita Pardubice**
**Fakulta elektrotechniky a informatiky**
Akademický rok: **2012/2013**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Vratislav Marek** |
| Osobní číslo: | **I09188** |
| Studijní program: | **B2646 Informační technologie** |
| Studijní obor: | **Informační technologie** |
| Název tématu: | **Účetní a fakturovací software** |
| Zadávající katedra: | **Katedra informačních technologií** |

Zásady pro vypracování:

Cílem baklářské práce je realizace fakturačního a účetního programu určeného pro drobné živnostníky.
Teoretická část:
V teoretické části bude popsáno využití jazyka Java s důrazem na serializaci, vlákna a využití kolekcí. Popis technologie XML a možností tisku pomocí Java Print.
Implementační část:
V této části bude navrženo a vytvořeno jednoduché účetnictví včetně tvorby faktur. Aplikace umožní vytvářet smlouvy o dílo, deník živnostníka, knihu jízd a celkové vyučtování.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

***Kolektiv. Jednoduché účetnictví. Computer Press, ISBN 80-7226-967-4, 2007.**
***Eckel, B. Myslíme v jazyku Java. Knihovna zkušeného programátora. Grada Publishing, 2001.**
***McLaughlin, B. Java and XML, Second Edition. OReilly Media, 2001.**

Vedoucí bakalářské práce: **Ing. Zdeněk Šilar**
Katedra informačních technologií

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

**Prohlášení autora**

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 7. 5. 2013 Vratislav Marek

**Poděkování**

Na tomto místě bych chtěl poděkovat panu Zdeňku Šilarovi, za podporu a správné směrování mé osoby při vypracovávání této práce, svým rodičům za podporu a zázemí v průběhu celého mého studia a lékařům, kteří mi zajišťovali zdravotní péči včetně péče pana Milana Novotného.

**Anotace**

Tato bakalářská práce se věnuje realizaci fakturačního a účetního programu určeného pro drobné živnostníky. V práci je popsáno využití jazyka Java s důrazem na serializaci, vlákna a využití kolekcí, technologii XML a možnost tisku pomocí knihovny Java Print. V praktické části je realizováno jednoduché účetnictví včetně tvorby faktur. Aplikace navíc umožňuje tisk smlouvy o dílo, vedení deníku živnostníka a knihy jízd.

**Klíčová slova**

Java, ekonomický software, živnostník, Serializable, XML, DOM, PrinterJob, Runnable

**Title**

Accounting and billing software

**Annotation**

The subject of this thesis is an accounting program aimed at self-employed people or small companies. The theoretical part contains the description of the Java programming language, with emphasis on serialization, threading, the use of collections and XML technology and the ability to print output on a printer using the Java "Print" library. The practical part of this thesis contains a simple accounting program including invoice creation. The application also allows the printing of work contracts, managing a work diary and trip records.

**Keywords**

Java, economic software, tradesman, Serializable, XML, DOM, PrinterJob, Runnable

# Obsah

## Seznam zkratek

| | |
|---|---|
| DPH | Daň z přidané hodnoty |
| JAR | Java Archiv |
| JDK | Java Development Kit |
| JVM | Java Virtual Machine |
| IDE | Integrated Development Enviroment |
| GNU | GNU’s Not Unix |
| GPL | General Public License |
| CDDL | Common Development and Distribution License |
| XML | eXtensible Markup Language |
| UML | Unified Modeling Language |

## Seznam obrázků

# Úvod

Důvodem vzniku této aplikace byla potřeba malého živnostníka vytvořit na míru sestavenou aplikaci podle představ. Žádný dostupný software nevyhovoval buď svými parametry, nebo vysokou cenou. Omezení daného výběru bylo hledáním softwaru pro platformu Linux, páce je vypracovaná v jazyku Java, tím umožňuje uživateli pracovat na platformách Linux i Windows. Cílem práce je zjednodušit shromáždění účetních informací pro jeho další zpracování účetním poradcem. Aplikace shromažduje informace o výkazech vyplývající z činnosti živnostníka, především vytváření a tisk faktur za celkově odvedenou práci, nebo za jednotlivé účetní položky zboží či činností, celkové vyúčtování určené k přehledu zákazníka o provedených činnostech a spotřebovanému materiálu, smlouvy o dílo, jež je vázanou dohodou o provedení a realizaci objednaných činností zákazníkem, dále evidenci knihy jízd živnostníka a deník živnostníka pro časový a úkonový přehled o jeho činnosti.

V teoretické části bude stručně popsána problematika účetnictví, dále se budu věnovat použitým technologiím využitých pro realizaci aplikace.

V praktické části se budu věnovat především popisu postupů vedoucích k vypracování samotné aplikace a způsobu práce s aplikací.

# 1 Ekonomické základy účetnictví

## 1.1 Faktura

Faktura je dokladem vystaveným podnikajícími fyzickými či právnickými osobami pro zaplacení za zboží či služby provedené pro objednavatele, kteří mohou být jiné fyzické či právnické osoby. Je vázána časovým prostorem, kdy může být vystavena přímo proti platbě, nebo libovolně dlouhou dobou splatnosti. Standardní dobou splatnosti je 14 dní.

Nebyl stanoven konkrétní tvar ani podoba jak má faktura vypadat, nicméně jsou určeny informace, které musí každá faktura obsahovat. Tyto informace se liší v závislosti na tom, zda li je vystavovatel plátcem DPH, nebo ne [1], [2], [8], [11], [12].

Faktura neplátce DPH není povahově daňovým dokladem, slouží jako prostý účet za odvedenou práci nebo zboží. Splatit daňový nárok DPH je neplátce DPH povinen až ve chvíli, kdy dostane od zákazníka zaplaceno. Pokud zákazník fakturu neuhradí, není vystavitel faktury vázán platbou daně za nesplacenou fakturu. Každá faktura od neplátce DPH musí obsahovat informace:

- označení
- jméno a adresu toho, kdo fakturu vydal, včetně IČO
- jméno či název firmy a adresu toho, kdo fakturu přijal, včetně IČO
- zmínku o zápisu podnikatele v živnostenském rejstříků
- datum vydání a datum splatnosti
- položky faktury
- informaci o fakturované peněžní částce

Faktura plátce DPH má právě povahu daňového dokladu, vystavením faktury je závazek splatit DPH nezávisle na tom, zda li zákazník svůj závazek splatí či nikoli. Den, kdy je v platnosti závazek splacení daně, je uveden na faktuře, jako den zdanitelného plnění. Každá faktura od plátce DPH musí obsahovat informace:

- označení
- evidenční číslo dokladu
- jméno a adresu toho, kdo fakturu vydal, včetně IČO
- jméno či název firmy a adresu toho, kdo fakturu přijal, včetně IČO
- zmínku o zápisu podnikatele v živnostenském rejstříků
- datum vydání a datum splatnosti
- datum uskutečnění zdanitelného plnění
- položky faktury
- informaci o fakturované peněžní částce
- základ daně
- informaci o % výši DPH
- výši DPH

## 1.2 Celkové vyúčtování

Celkové vyúčtování je detailní přehled účetních položek zákazníkovi a shrnutí provedení veškerých činností a spotřebě materiálu, které budou následně fakturou požadovány k proplacení zákazníkem.

## 1.3 Kniha jízd

Podnikatelé jsou povinni ze zákona č. 586/1992 Sb., o daních z příjmů shromažďovat informace o výdajích na dopravu silničních motorových vozidel, pro doložení vynaložených výdajů správci daně, pokud správce daně nestanoví jinak. Zákonem č. 304/2009 Sb. byl zákon č. 586/1992 Sb. novelizován, tím nově umožňuje aplikaci paušálního výdaje za dopravu silničních motorových vozidel.

Kniha jízd má obsahovat, kdy byla jízda započata a skončena, účel jízdy, ujeté kilometry, množství spotřebovaného paliva a rozlišení, zda se jedná o soukromou či služební jízdu[9].

## 1.4 Deník živnostníka

Tento deník nespadá přímo do jednoduchého účetnictví, ale umožňuje drobnému živnostníkovi vést každý den poznámky o provedených úkonech ve své živnosti.

# 2 Využité technologie

## 2.1 Komponenty knihovny javax.swing

*JFrame*, potomek třídy *Window*, je jeden ze způsobů grafického výstupu aplikace. Na rozdíl od třídy *Window* je obohacen o *hendlery* reagující na akce kurzoru, je vybaven rámem a běžně se vyskytujících funkcí manipulujících s okny, příkladem menu nabídky okna, nebo tlačítka ovládání okna jako jsou minimalizování, maximalizování, obnovení a zavření. Je určen pro samostatné zobrazování aplikace a nepotřebuje výší prvky a proto je samostatně spustitelný.

*JPanel* jakým si kontejnerem zastřešujících další prvky knihovny *javax.swing* a plátnem pro vykreslování grafických obrazců komponenty *Graphics* a *Graphics2D*. Samostatně nemůže být zobrazen, právě proto je nutné, aby mu byl nadřazen prvek typu *Window* nebo již zmiňovány prvek *JFrame*.

*JList* je grafická komponenta, potřebující pro svou funkci nadřazených grafických kontejnerů jako je *JPanel*, *JFrame*, nebo *Window*. Tato komponenta vypisuje uživateli jednotlivé položky v zobrazení listu s možností zpětné vazby vybrání položek, podle nastavení jedna, množinu položek nebo množinu označení. V základních modelech zpracovává vložené komponenty pouze textovou podobou a u ostatních objektů volá získání textové podoby skrz metodu *toString*. Jako modelovou položku využívá *ListModel*, jež zadržuje samotný seznam položek.

*JComboBox* je grafická komponenta, potřebující pro svou funkci nadřazených grafických kontejnerů jako je *JPanel*, *JFrame*, nebo *Window*. Tato komponenta vypisuje uživateli jednotlivé položky v zobrazení jedné lince se zabudovaným seznamem vyvolaným akcí kurzoru a možností připsat novou položku, s možností zpětné vazby vybrání právě jedné položky. V základních modelech zpracovává vložené komponenty pouze textovou podobou a u ostatních objektů volá získání textové podoby skrz metodu *toString*. Jako modelovou položku využívá *ComboBoxModel*, jež zadržuje samotný seznam položek a vybraný index. Při použití jednoho *ComboBoxModelu* do více *JComboBoxů* budou všichni sdílet vybranou položku a zároveň měnit svůj stav.

*JTable* je grafická komponenta, potřebující pro svou funkci nadřazených grafických kontejnerů jako je *JPanel*, *JFrame*, nebo *Window*. Účelem této komponenty je vytvořit tabulku buněk obsahujících objekty. Podle nastavení může *JTable* během svého provozu vytvářet nové řádky, nebo mít pevně stanovený počet řádků. Jednotlivé buňky slouží k informaci, výběru, editaci informací, nebo k vložení nových. Jako modelovou položku využívá *TableModel*, pro znepřístupnění zápisu do jednotlivých buněk je zapotřebí tento model zdědit a přepsat metodu *isCellEditable*. *TableMode*l je možno naplnit konstruktorem a to jednorozměrným polem třídy *String*, jež nese informaci o názvech jednotlivých sloupců a dvourozměrným polem objektů, které nesou informaci o jednotlivých buňkách, ve výchozím stavu pracující s metodou *toString*. *JTable* je možné

nastavit vybírání jednotlivých buněk, množiny skupin buněk, jeden řádek, více řádků a další [3][4].

## 2.2 Datové struktury Java

*Iterator* je genericky otypovatelným interfejsem kolekce, kde jsou implementovány metody „*hasNext*“ a „*Next*“. *Iterator* je určen především k procházení datových struktur od prvního prvku k poslednímu s předem určeným pořadím k jakékoli datové struktuře. Metoda „*hasNext*“ vždy vrací logický výraz pravda, nepravda v závislosti na tom, zda li existuje další prvek v řadě a může se k němu přistoupit. Metoda „*Next*“ pokud existuje další prvek v řadě, pak jej vrátí.

*HashMap* je generickým, dynamický rostoucím kontejnerem knihovny *java.util* s využitím indexování jednotlivých položek pomocí genericky určeného klíče. Vložením jiného prvku s již existujícím klíčem původní záznam přepíše. *HashMap* nemá pevně dané pořadí prvků, proto pokud chceme procházet kontejner, musíme stanovit pořadí právě daným klíčem. *HashMap* nedisponuje metodou navracející *Iterator*, avšak je možné z prvků bez klíče vrátit generickou *Collection* a z té nechat Iterator vygenerovat.

*LinkedList* je generický, dynamicky rostoucí kontejner knihovny *java.util*. Identifikátorem prvku v kontejneru je buď instance prvku samotného, nebo jeho pořadí. Pořadí se vytváří vkládáním, kde každý nový prvek bude dodán na konec pořadníku. *LinkedList* disponuje metodou pro generování Iteratoru v pořadí vnitřního listu [3], [4].

## 2.3 XML

XML je formát zápisu dat v dokumentu, kde k identifikaci jednotlivých datových položek nám slouží tzv. „tagy“. Jednotlivé tagy určují tzv. „elementy“. Element je pojmenován názvem, který není předem určený, a proto si může autor elementu vybrat vlastní název. Element je vždy párový tag, jedině prázdný je vygenerován jenom jeden uzavřený tag. Elementy mohou přímo nést na sobě atributy, jež je mohou identifikovat nebo nést další informace. Uvnitř elementu mohou být potřebné údaje, nebo opět další elementy[10].

## 2.4 Vývojová aplikace NetBeans IDE

NetBeans IDE je velice pokročilý vývojový nástroj programátora od společnosti ORACLE, postavený na Javě a JDK. NetBeans IDE je nezávislý na dané platformě, právě proto že je postavení na Javě. Je licencován duální licencí CDDL a GNU GPL a volně ke stažení na webu společnosti ORACLE. NetBeans IDE nástroj se nespecializuje na vývoj konkrétního programovacího jazyka, ale na skupinu jazyků a formátů dat, které jsou pro tento nástroj přístupné z dodávaných a volně stažitelných pluginů. Mezi podporované jazyky a formáty patří Java ve všech verzích, C/C++, PHP, HTML, CSS a XML. Prostředí programátorový napovídá při tvorbě svého algoritmu příkazy i formáty, kontroluje syntaxi, částečně sémantiku a na pozadí i závislosti jednotlivých tříd. Umožňuje některé část kódu sám generovat. Pro Javu vlastní kompilátor i krokovací nástroj.

## 2.5 Volně šiřitelné kódy třetích stran

Z důvodu absence grafické komponenty kalendáře s výběrem dní v základních knihovnách swing byly v práci použity kromě běžně dostupných knihoven Javy také knihovny třetích stran, avšak volně šiřitelné pod licencí GNU a volně ke stažení.

Název této komponenty je *JCalendar*. Použitá komponenta byla vytvořena Kai Toedterem. Komponenta však pro své použití musela být autorem práce upravena, aby spolupracovala se zbytkem vytvořené aplikace [7].

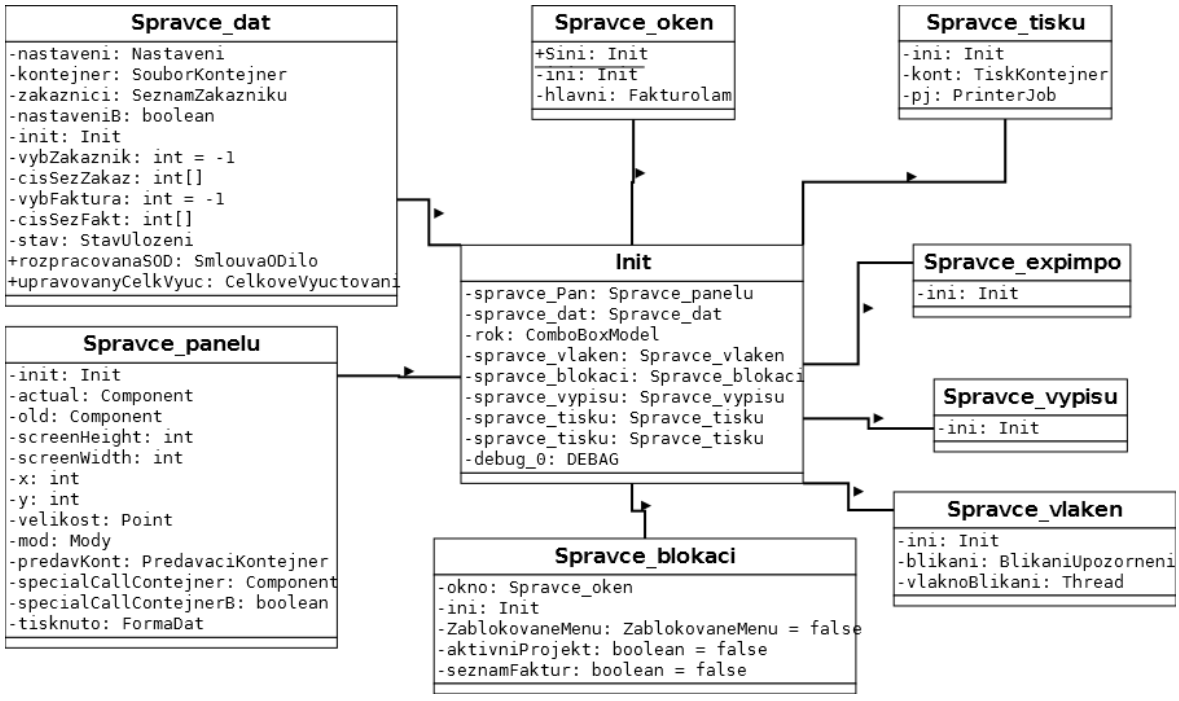
# 3 Účetní a fakturovací software

## 3.1 Stromová struktura

Aplikace je hierarchicky rozložena od jádra až po nejnižší prvky kontejnerů. Jádrem aplikace je tu myšleno konkrétně rozdělení několika tříd, které jej společně tvoří svoji spoluprací a návaznosti. Pokud by nebyly rozděleny, ale sloučeny do jediné, pak by tato jediná třída byla příliš rozsáhlá a nepřehledná.

Jádro aplikace se stará o komunikaci všech prvků, zpracování a ukládáni informací a reakce na jednotlivé stavy, jako například neuložený kontejner projektu. Z panelů jež jsou grafickým prostředím pro uživatele, byly odebrány veškeré algoritmy, které zpracovávají chod aplikace a následně přesunuty do jádra. Ale byly jim ponechány algoritmy, které zpracovávají konkrétní sběr nebo zobrazení informací náležící právě jedinému panelu.

Jádro aplikace se skládá z třídy tvořící vrchol aplikace a tříd tzv. správců služeb.




**Obrázek 1 – UML diagram jádra aplikace**

### 3.1.1 Vrchol aplikace

Vrcholem ve struktuře aplikace byla stanovena třída *Init*, není však samostatně spustitelnou třídou. Spouštění aplikace zajišťuje třída *Main* s jedinou pro chod důležitou statickou metodou *main*, zbylé metody jsou přítomny čistě z odlaďovacích důvodů. Při spuštění je inicializována třída *Init* v metodě *main* třídy *Main* v novém anonymním vlákně.

*Init* v sobě nese asociaci jednotlivých správců služeb a zajišťuje jejich počáteční inicializaci. Svou vlastní referencí zajišťuje přenos, komunikaci a šíření události i dat mezi správci. Někteří správci služeb jsou na jiných závislí, pro svou funkci a to znamená, že je nutné předem určit jejich pořadí inicializace závislých správců:

1. *Spravce_oken*
2. *Spravce_dat*
3. *Spravce_blokaci*
4. *Spravce_panelu*

### 3.1.2 Správci služeb

Jednotlivé funkce jádra byly pro přehlednost rozděleny do několika tříd, přitom každá z nich se stará čistě jenom o své určené odvětví. Nezasahuje do části jiného správce, ale předává informace o událostech, které nastali a potřebná data.

Prvním správcem je třída *Spravce_oken*, tento správce se stará výhradně o správu všech tříd typu *JFrame*. Nabízí reference na tyto okna ostatním správcům a zajišťuje jejich inicializaci a zobrazení. V současné verzi se stará o okna Autor a *Fakturolam*. Třída Autor má za úkol zobrazit informace aplikace uložených v konstantách, konkrétně o verzi programu, autorovi a licenci. Druhá zmiňovaná třída *Fakturolam* má za úkol právě zobrazování a ohraničení celého grafického výstupu aplikace. Sama nese obecné informace a nabídky pro uživatele, přístupné pro celou aplikaci.

Konkrétně třída *Fakturolam* informuje uživatele ve stavovém řádku o otevřeném projektu aplikace a o stavu uložení projektu a společných zákazníků. Dále nabízí nabídky menu aplikace, ty jsou rozděleny do tří skupin Soubor, Zákazníci, Možnosti. V kategorii Soubor dává možnosti uživateli pracovat se souborem projektu, otevřít ho, zavřít, uložit či vrátit do uloženého stavu, poskytuje i možnost vytvoření zálohy a uzavření programu. Skupina Zákazníci nabízí uživateli správu nad uložením, zálohováním či vrácení do uloženého stavu společných informací o zákaznících. Poslední skupina Možnosti, dává uživateli přístup k obecným funkcím aplikace, jako je nastavení, informace o aplikaci a export či import souborů XML.

V řadě dalším správcem je třída *Spravce_dat*, hlavním úkolem tohoto správce je manipulace a přistup k jednotlivým datovým strukturám, současně také zadržuje informace o stavu rozdílu jednotlivých kontejneru proti uloženým souborům. Pokud se liší stavy kontejnerů uložených v paměti a v souboru na disku, kontaktuje správce *Spravce_vypisu*. Třída *Spravce_dat* v sobě konkrétně zadržuje kontejnery *Nastaveni*, *SouborKontejner* a *SeznamZakazniku*. Dále zadržuje informace o vybraných záznamech pro zajištění předání konkrétního záznamu z jednoho panelu do jiného. Konkrétně předává identifikační číslo vybraného zákazníka, pole identifikačních čísel zákazníků seřazených podle abecedy, pro pohyb vybraného zákazníka v rámci stanoveného pořadí, záznam smlouvy o dílo a záznam celkového vyúčtování.

Při své inicializaci třída *Spravce_dat* má za úkol zajistit načtení společných dat pro aplikaci, těmi jsou právě kontejnery *Nastaveni* a *SeznamZakazniku*, o to se starají dvě následující metody:

```
    private void nactiCoJe() {
        UlozitNacist<Nastaveni> nas;
        nas = new UlozitNacist<Nastaveni>(Kons.NASTAVENI +
Kons.KONCOVKA);
        this.nastaveniB = nas.existuje();
        if (this.nastaveniB) {
            this.nastaveni = nas.nacti();
        } else {
            this.nastaveni = new Nastaveni();
        }
        this.nactiZakazniky();
    }

    public void nactiZakazniky() {
        UlozitNacist<SeznamZakazniku> zak;
        zak = new UlozitNacist<SeznamZakazniku>(Kons.ZAKAZNICI +
Kons.KONCOVKA);
        if (zak.existuje()) {
            this.zakaznici = zak.nacti();
        } else {
            this.zakaznici = new SeznamZakazniku();
        }
        this.stav.ulozenoZakaznici();
    }
```

Uživateli se seznam zákazníku ukazuje seřazen podle abecedy, to však znamená, že nebude přímo souhlasit pořadí podle identifikačního čísla. Konkrétní seřazení zajišťuje třída z pracovních kontejnerů *SeznamPanelu*, s jejíž pomocí se vytvoří pole identifikačních čísel zákazníků v požadovaném pořadí. Toto pořadí je předáno do pole, které drží právě třída *Spravce_dat*. Při prohlídce zákazníků se jednotliví následníci a předchůdci posouvají v rámci seřazení pomocí dvou metod správce:

```
    public Zakaznik getVybZakaznikDalsi() {
        if (this.cisSezZakaz != null) {
            for (int i = 0; i < this.cisSezZakaz.length; i++) {
                if (this.vybZakaznik == this.cisSezZakaz[i]) {
                    int u = i + 1;
                    if ((u) >= this.cisSezZakaz.length) {
                        u = 0;
                    }
                    this.vybZakaznik = this.cisSezZakaz[u];
                    return this.getVybZakaznik();
                }
            }
        }
        return null;
    }


    public Zakaznik getVybZakaznikPredchozi() {
        if (this.cisSezZakaz != null) {
            for (int i = 0; i < this.cisSezZakaz.length; i++) {
                if (this.vybZakaznik == this.cisSezZakaz[i]) {
```

```
                    int u = i - 1;
                    if ((u) < 0) {
                        u = this.cisSezZakaz.length - 1;
                    }
                    this.vybZakaznik = this.cisSezZakaz[u];
                    return this.getVybZakaznik();
                }
            }
        }
        return null;
    }
```

Následujícím správce je třída *Spravce_blokaci*, ta se specializuje na blokování uživatelských možností v závislosti na stavech souborů, získaných hlavně z událostí třídy *Spravce_dat*. Události především nastaví informace o blokování a následně vyvolá podmět k obnovení blokovaných stavů komponent, v závislosti na aktuálním zobrazeném panelu, který je spravován třídou *Spravce_panelu*.

Správcem starajícím se o přepínání, inicializaci a přednastavení jednotlivých panelu, o pozici hlavního okna a centrování je třída *Spravce_panelu*. Koncept grafické časti aplikace a velká část uživatelského prostředí stojí právě na tomto správci, při jeho absenci by aplikace nemohla vůbec fungovat.

Při své inicializaci má za úkol třída *Spravce_panelu* první naplnění okna správným panelem. V závislosti na existenci nastavení aplikace to mohou být dva panely. V případě, že soubor s nastavením neexistuje, bude prvním panelem právě *NastaveniPanel*, tato situace byla pojmenována: „První spuštěni“. Jak název napovídá, aplikace při absenci nastavení předpokládá právě, že se jedná o první spuštění. Bez nastavení není aplikace schopna správně fungovat, a proto pomocí třídy *Spravce_blokaci* znemožní uživateli aplikaci používat do doby, než vyplní všechny povinné údaje a řádně nastavení uloží. V druhém případě, tedy soubor nastavení existuje, prvním panelem se stane třída *Hlavni*.

Pro přepínání mezi jednotlivými panely slouží trojicí přetížených metod třídy *Spravce_panelu*, kde v základní verzi metody, níže uvedené, musí určit, kde jsem a kam chci jít.

```
public final void prohodPanely(Component a, Component b) {
    ini.remove(a);
    old = a;
    ini.add(b);
    actual = b;
    podminka();
    ini.getHlavni().pack();
    zmenPozici();
}
```

Metoda *podminka* zajišťuje provedení nezbytných akcí při přepnutí panelu, jakými jsou například naplnění komponent informacemi z nastavení aplikace nebo vyplněním listu zákazníků.

Při změně rozměrů aplikace se drží střed okna stále na původním místě, kde byla před danou změnou, tím se dosáhne v programu metodou *zmenPozici*.

```
private void zmenPozici() {
    Point tvelikost = new Point(ini.getHlavni().getSize().width,
            ini.getHlavni().getSize().height);
    int xx = (tvelikost.x - velikost.x) / 2;
    int yy = (tvelikost.y - velikost.y) / 2;
    Point oo = new Point(ini.getHlavni().getLocation().x - xx,
            ini.getHlavni().getLocation().y - yy);
    if (oo.x < 0) {
        oo.x = 0;
    }
    if (oo.y < 0) {
        oo.y = 0;
    }
    ini.getHlavni().setLocation(oo);
    velikost = tvelikost;
}
```

Dalším z řady metod třídy *Spravce_panelu* přetížené metody *prohodPanely* je pouze určením kam chci a tím spoléhám, že je uchována správná informace o tom, kde jsem byl. Třetím zástupcem přetížení je bezparametrická metoda, která se dá použít prvotně při inicializaci, nebo výhradně pro přechod typu „zpět“, kde se vracíme například z vybírání zákazníka zpátky na panel nové faktury.

Protože není přípustné, aby jednotlivé panely měli mezi sebou přímé vazby, byla nahrazena komponenta určující kam se má přepnout enumem *Panely*. Proto došlo k dalším přetížením již přetížené metody *prohodPanely* a vložen mezičlánek, který identifikuje podle enum *Panely* volaný panel, v případě, že není držen, jej inicializuje a pokračuje se v původní trojici.

```
public final void prohodPanely(Panely a) {
    prohodPanely(EnumToCompo(a));
}
```

Z důvodu nutnosti oznámení o stylu pracování volaného panelu, bylo přidáno šestého přetížení metody *prohodPanely*. Současné verzi je vyjádřen enumem *Mody*, s jediným prvkem. Tohoto oznámení bylo v aplikaci využito u panelů *NovaFaktura* a *NovyZakaznik*, kde místo vytváření nového bude načten vybraný a pouze upravován.

```
public enum Mody {
UPRAVIT;
}
public final void prohodPanely(Component a, Panely b, Mody c) {
    this.mod = c;
    prohodPanely(a, EnumToCompo(b));
}
```

Přepínání panelů je v předem známém cyklu návazností, kde je odkudkoli známé, kam se může pokračovat a kam vrátit. V práci jsou ale použity ještě panely, které do tohoto cyklu nepatří a mohou být volány nezávisle na tom, kde se právě uživatel nachází. Původní koncept na takové výjimky nebyl stavěn a v případě nevhodného volání panelů nepatřících do smyčky dojde k lokálnímu zacyklení přepínací smyčky, ze které se dá dostat jedině ukončením aplikace. Proto bylo přidáno „speciální volání“ (metoda `zadrzSpecialCall`) pro panely nepatřící do standardního přepínacího cyklu. Toto speciální volání zadrží komponentu cyklu, která by při jejím zahození znamenala zacyklení přepínací smyčky

a uloží ji mimo standardní úložiště. Po dalším přepnutí se speciálním voláním identifikuje zadrženou komponentu a vrátí potřebnou část řetězu zpět.

```
    private boolean isSpecialCall() {
        return this.specialCallContejnerB;
    }

    private Component getSpecialCall() {
        Component tmp = this.specialCallContejner;
        this.specialCallContejner = null;
        this.specialCallContejnerB = this.specialCallContejner != null;
        return tmp;
    }

    private void zadrzSpecialCall() {
        if (!this.isSpecialCall()) {
            this.specialCallContejner = this.old;
            this.specialCallContejnerB = true;
        } else {
            throw new ArrayStoreException("Nesmím zahodit zadrzený
panel!");
        }
    }

    private boolean specialExpresion(Component a) {
        return a instanceof NastaveniPanel
                || a instanceof XMLImportExport;
    }

    private void SpecialCall() {
        if (this.specialExpresion(actual)) {
            if (this.isSpecialCall()) {
                Component tmp = this.getSpecialCall();
                ini.remove(actual);
                this.actual = tmp;
            } else {
                throw new IllegalAccessError("Ztratil se kontejner!");
            }
        } else {
            this.zadrzSpecialCall();
        }
    }
```

O jednotlivé stavové informace poskytující aplikace za svého chodu se stará třída *Spravce_vypisu*. Identifikuje jednotlivé aktivní panely zobrazené uživateli v okně aplikace, kde následně vypíše název panelu do názvu aplikace. Reaguje na události správce *Spravce_vlaken* a výsledné, spolu s informací od *Spravce_dat* o stavu otevření projektového souboru, zapíše do stavového řádku aplikace.

Aplikace pracuje více vláknově, kromě spouštěného vlákna metody *main* a anonymního vlákna, ve kterém je inicializováno spuštění *Init*, má na starosti ve své režii provoz a synchronizaci dalších vláken třída *Spravce_vlaken*. Konkrétně spravuje v současné verzi jediné vlákno, pro které byla přidělena třída s implementací *Runnable BlikaniUpozorneni*, jež zajišťuje upozornění uživatele na stav neuložených záznamů

v paměti střídavým blikáním nápisu pro upoutání pozornosti. Vlákno je inicializováno a spuštěno teprve ve chvíli, kdy je ho poprvé od spuštění nutné, místo zastavení vlákna ve chvíli, kdy už není třeba upozorňovat, bude vlákno zastaveno pasivním čekáním za pomoci synchronizovaného objektu.

Účelem aplikace je i shromážděné informace vytisknout a tím vytvořit účetní doklady. Proto je součásti aplikace i třída *Spravce_tisku*, jež tuto problematiku spravuje. Ta však samotný tisk pouze připraví, využívá k tomu tříd knihovny *java.awt.print* a těmi jsou *PageFormat*, *Paper* a *PrinterJob*. O konkrétní vzhled jednotlivých stran a zpracování údajů se stará třída z pracovních kontejnerů *TiskKontejner* [5].

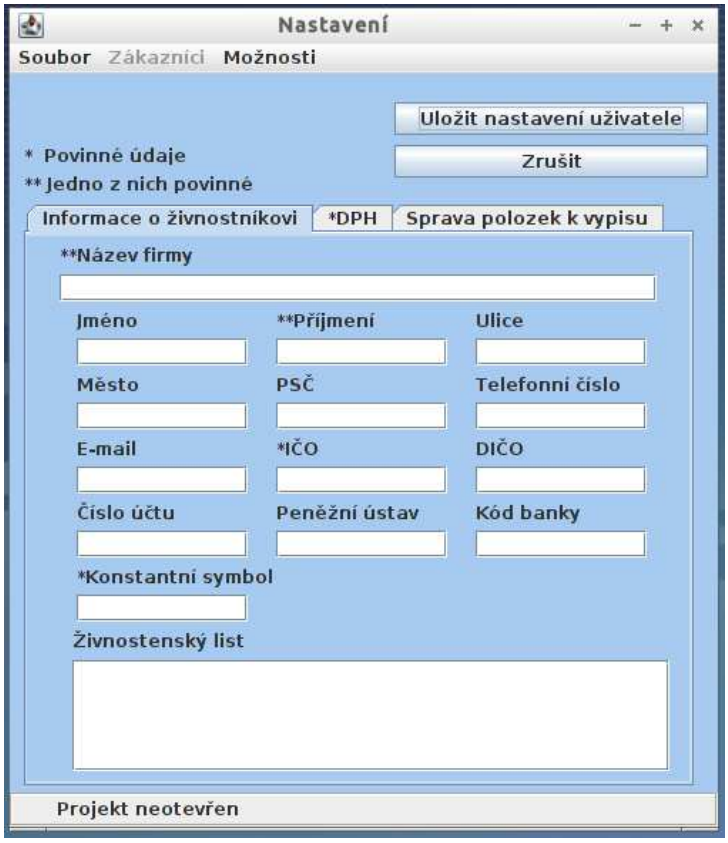
Pro účely zálohování, nebo přenášení nashromážděných informací pro jiné algoritmy, bylo využito formátu XML. O konkrétní manipulaci s XML dokumentem se stará třída *Spravce_expimpo*. Ta umožňuje exportovat nebo importovat celý seznam zákazníku, nebo faktur. Tento správce pro svoji funkci využívá knihoven *javax.xml* a *org.w3c.dom* [6][10].

Podle struktury jednotlivých dat byly vytvořeny funkce pro parsování objektu kontejnerů do elementu a zpětné elementu do objektu kontejneru.

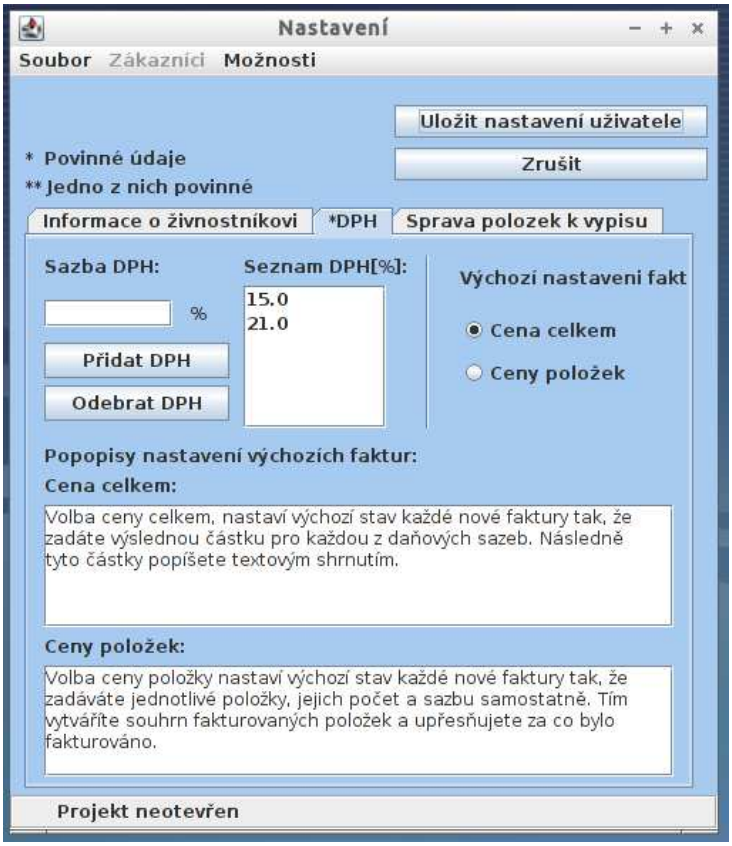
### 3.1.3 Panely

Všechny zde zmíněné třídy jsou potomky *javax.swing.JPanel* a jsou v podstatě grafickým rozhraním komunikujícím s uživatelem aplikace. Jsou vykreslovány v hlavním okně třídy *Fakturolam* a přepínaný pomocí třídy správce *Spravce_panelu*. Zde je jejich seznam a podrobnosti o některých z nich:

V případě prvního spuštění aplikace nás uvítá panel *NastaveniPanel*, panel je rozdělen do tří záložek, kde každá spravuje rozdílný typ informací. Je zapotřebí, aby uživatel sdělil aplikaci informace o sobě, svém sídle a živnosti, pro vybavení těmito informacemi výstupní dokumenty aplikace, toho uživatel dosáhne v záložce „Informace o živnostníkovi“. Aplikace také zpracovává daňové sazby DPH, ty se ale v rámci vývoje ekonomiky mění v záložce „DPH“, proto musí uživatel stanovit právě aktuální.




**Obrázek 2 – Ukázka panelu nastavení aplikace – Informace o živnostníkovy**




**Obrázek 3 – Ukázka panelu nastavení aplikace – DPH**

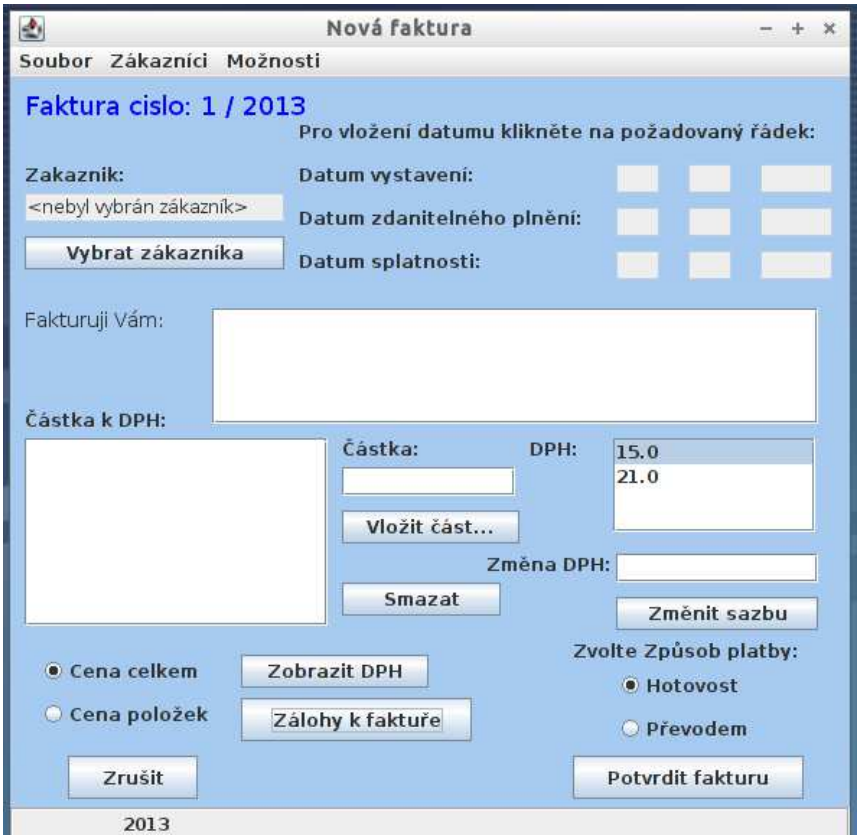
Kromě prvního spuštění se aplikace vždy spustí s prvotním panelem *Hlavni*. Tento panel je hlavní nabídkou aplikace, kde může uživatel přistoupit k jednotlivým částem aplikace. Z důvodu zaměření účetních dokladů na jeden rok, je projekt aplikace také právě na jeden rok zaměřen, v panelu *Hlavni* je nutné před otevřením projektu vybrat správný účetní rok.


**Obrázek 4 – Ukázka hlavní panelu aplikace**

Samotná správa faktur a zákazníků je obsáhnuta v nabídce panelu *HlavniPanelFakturace*, kde podle závislosti na existenci seznamů a otevřeného projektu máme v paměti přístup ke konečným seznamům nebo vytvoření nové položky.

Pro vytvoření nového nebo úpravu již existujícího zákazníka bylo použito panelu *NovyZakaznik*. Pro nového zákazníka je povinné alespoň pojmenování, kde musí být zadán jeden z údajů „Název firmy“, „Jméno“, nebo „Příjmení“, ostatní informace o zákazníkovi jsou dobrovolné.

Panel, který slouží k vytvoření nové faktury nebo její úpravě, či jenom pro prohlížení podrobností faktury, slouží třída *NovaFaktura*. Je nutné, aby faktura obsahovala údaje o zákazníkovi, pro kterého je faktura vystavována, datum, kdy měla být faktura vystavena a zároveň, kdy je závazek splatnosti a alespoň jediná částka pro fakturaci.




**Obrázek 5 – Ukázka panelu nové faktury**

V aplikaci je nutné pro některé z formulářů určit konkrétního zákazníka, pro kterého jsou sepisovány účetní údaje. K tomuto účelu byl vytvořen panel *VyberZakaznici*, který předává ostatním panelům potřebnou informaci a zároveň umožňuje zákazníka dodatečně před vybráním upravit.

K přímému procházení zákazníků s požadovanými detaily, bylo umožněno v panelu *ZobrazZakaznika*, který využívá posouvání v seřazeném pořadí zákazníků a posouvá se skrz metody třídy správce *Spravce_dat*.

Pro zobrazení seznamu zákazníku a další odkazy na manipulaci se zákazníky, kterými mohou být prohlížení, úprava, přidání, nebo smazání, mohou být docíleny prostřednictvím panelu *SeznamZakaznikuPanel*. Podobným způsobem pracuje se seznamem faktur panel *SeznamFakturPanel*.

Tvorba a správa smluv o dílo, kniha jízd, celkové vyúčtování, dokonce i deník živnostníka, je závislá na tzv. „Akcí“, jež hromadně určují pojmenování a zákazníka, se kterým se bude v těchto kontejnerech pracovat. Proto je nutné zmíněné „Akce“ spravovat a toho lze dosáhnout skrz panel *PanelAkci*.

V panelu *PanelCelkoveVyuctovani* bylo dosaženo kompletní správy náležitostí týkající se tvorby, správy a úpravy celkového vyúčtováni. Panel byl rozdělen do jednotlivých záložek, první z nich: „Seznam vyúčtování“, po vybrání konkrétní účetní položky slouží k zobrazení účetních souhrnů, exportování do faktury, tisku, nebo smazání.

Druhá ze záložek má za účel správu konkrétní účetní položky a zobrazení detailů. V případě, že je účetní položka vybrána, slouží právě pro zobrazení jejich detailů a úpravě, v opačném případě je vytvářena nová.

V aplikaci má živnostník možnost vytvářet na každý den podrobnosti o vykonané činosti právě v panelu *PanelDenik*.

Kniha jízd vedená seznamem pro každý den určeném kalendářem, je úkolem panelu *PanelKnihaJizd*. Ten je rozdělen do dvou záložek. V záložce „Výběr dne“ v tabulce kalendáře, snadno živnostník zjistí, jednotlivé zapsané dny a po výběru zkoumaného dne si může živnostník v záložce „Podrobnosti dne“ spravovat jednotlivé jízdy nebo se pouze informovat o statistikách daného dne.

Kompletní správa veškerých smluv o dílo bylo vměstnáno do jediné záložky panelu *SmlouvaODiloPanel* „Správa smluv“. Při vytváření má živnostník možnost inspirovat vzorovou ukázkou takové smlouvy v záložce „Vzor“, ten byl načerpán z programu uvedeného portálu a volně k dispozici veřejnosti.

Panel *Tisk* má v aplikaci jediný cíl, čímž je uvést do chodu algoritmus tisku vybraných dokumentů s dodržením zvolených kopii.

Jelikož bylo nutné pro účel dalšího zálohování nebo přenosu informací aplikace právě prostřednictvím XML, byl připraven jednoduchý panel *XMLImportExport*, ve kterém se jednotlivé kontejnery oddělují opticky pro srozumitelnost a aktivují se pouze tlačítka, pro které je možné v závislosti na aktuálním stavu kontejnerů v paměti vykonat požadovanou činnost.

### 3.1.4 Podpanely

Pod panely jsou třídami typu *JPanel* samostatně nečinné a součástí jednoho nebo více panelu nadřazených. Konkrétně zde zmíněné pod panely jsou využité právě v panelu *NovaFaktura* a s ním tvoří celek. Většinou jsou umístěné do komponenty *JInternalFrame* s možností pohybovat se po panelu, nebo zbavené okrajů a pohybových hendlerů, pevně uzamčené na pozici, kde uživatel ani nepostřehne existenci interního okna.

Rozlišením faktury zdali se jedná o kompletní účtovanou částku k jednotlivým daním nebo kompletní výčet jednotlivých účetních položek, se zde stávají panely *CelkemCenaPanel* a *PolozkaCenaPanel*, kde se konkrétně slovně shrne fakturovaný účel a přidá se částka k vybrané dani v podpanelu *CelkemCenaPanel* a kompletní výčet zase v podpanelu *PolozkaCenaPanel*.

Živnostník k započaté práci může od zákazníka požadovat platbu předem, tzv. „zálohu“, například pro nakoupení nadstandardního materiálu, ale k výsledné fakturované částce musí tyto zálohy připočíst, to má na starosti podpanel *VlozZalohyPanel*

Během vytváření faktury živnostník uvítá kompletní přehled částek včetně DPH a záloh se současným účetním stavem. To bylo vypracováno v podpanelu *VypisDPHPanel*.

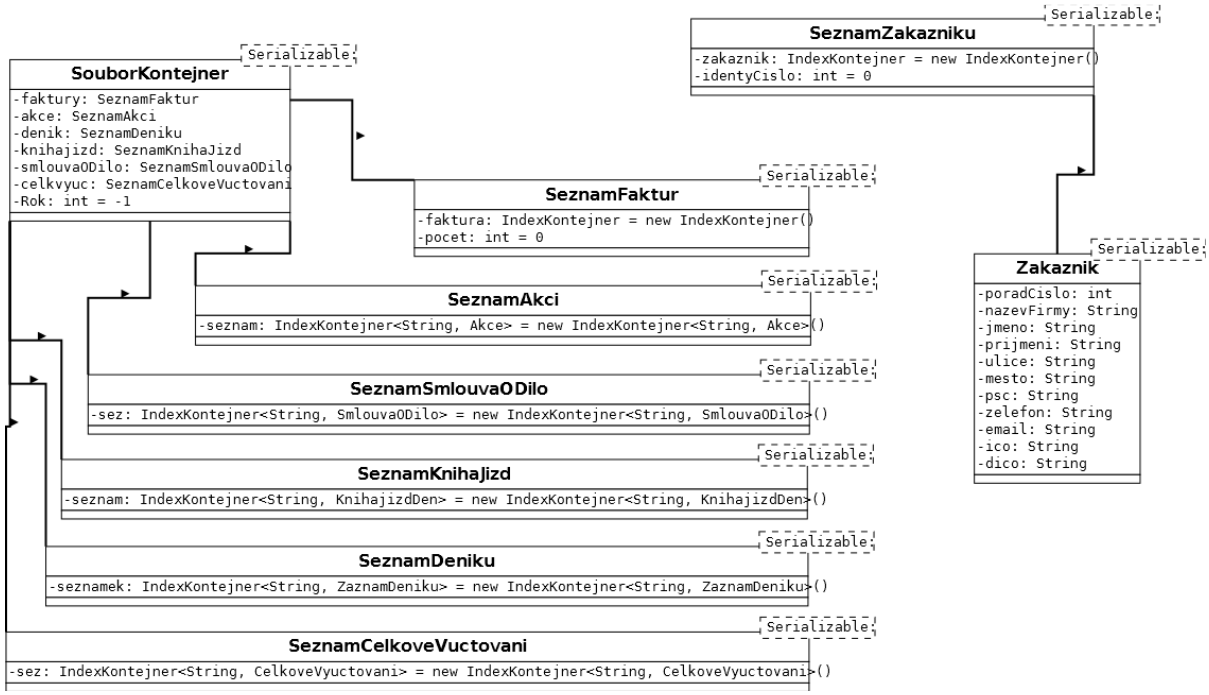
### 3.1.5 Statické funkce

Bylo využito statických metod pro aplikaci kdekoli v kódu bez inicializace, obecných pro použití a společných pro všechny. Informační zpráva o výzvě nebo varování s možností na událost reagovat, bylo použito ve statické třídě *Zprava*, kde byla využita komponenta *javax.swing.JOptionPane* pro vyvolání zprávy pomocí metody *showMessageDialog* a zpětného dotazu prostřednictvím metody *showOptionDialog*.

Práce s vypisováním jednotlivých čísel do určitých formátů byla zahrnuta v následujících statických třídách, rozdělením jednotlivých řádu tisíců hodnoty v rámci zobrazováno v podobě textového výstupu bylo umožněno v statické třídě *Oddel1000*, kdežto přímému slovnímu vyjádření konkrétní částky bylo vypracováno parsováním čísla v statické třídě *FloatTostringKC*.

### 3.1.6 Úložné kontejnery

Všechny tyto kontejnery jsou přímo ukládané do uložených souborů na disku, pro zjednodušení bylo využito implementace Serializable, která umožní pomocí knihovny *java.io* přímo serializovat do souborů kontejnery v paměti a zpětně je zase deserializovat při načítání přímo do paměti.




**Obrázek 6 – UML úložných kontejnerů**

### 3.1.7 Pracovní kontejnery

V práci bylo použito i kontejnerů, které nemají za účelem vždy jenom uchovávání a indexování informace, ale i její zpracování pro další využití.

Jak již bylo zmíněno v předešlých kapitolách, v práci je využito datových struktur Javy, z důvodu možnosti v budoucnu se vyskytujících nových datových struktur a možnosti jejich snadné výměny, byly současně využity za pomocí zapouzdření, konkrétně třída *IndexKontejner* v sobě zapouzdřuje datovou strukturu *HashMap*, dále pro datová struktura *LinkedList* byla zapouzdřena v třídě *Kontejner*. Tyto dva kontejnery slouží zároveň k ukládání informace do souboru, a proto na nich byla implementována interface *Serializable*.

Třída PredavaciKontejner byla využita k jedinému účelu v aplikaci, a to k předávání informace napříč panely skrz třídu Spravce_dat.

Komponenty *JList* a *JComboBox* pro zobrazení svých seznamů ve výchozím stavu využívají modelů, které zpracovávají pouze textové výpisy s indexačním pořadníkem. V případě potřeby zobrazit prvky na zmíněných komponentách seřazené, například podle abecedy, již nebude souhlasit pořadí jednotlivých záznamů s kontejnerem v paměti. Tato problematika byla vyřešena za pomoci třídy *SeznamPanelu*, kde uchová v sobě identifikační hodnotu původního prvku a jeho textové vyjádření, následně v sobě seznam seřadí a vygeneruje model pro zmíněné komponenty. Tím odpovídá pořadí zobrazené v komponentě se seznamem v třídě *SeznamPanelu*, díky uchovaným informacím, podle identifikační hodnoty identifikuje původní záznam a aplikace s touto informací může dále pracovat.

Pro zapouzdření, snadnou manipulaci a přehlednosti v kódu, byla oddělena z třídy *Spravce_dat* manipulace a uchování informací o stavu rozdílu uchovaných dat mezi pamětí a diskem. Následně byla přesunuta do třídy *StavUlozeni*.

Důležitou roli při tisku jednotlivých dokumentů hraje třída *TiskKontejner*, kde jednotlivé tiskové sestavy byly realizovány právě v ní.

# 4 Způsoby ukládání a uchování dat

## 4.1 Implementování rozhraní Serializable

V jednotlivých kontejnerech shromažďujících informace, které je třeba dlouhodoběji uchovávat, je třeba implementovat rozhraní *Serializable*, aby bylo možné dané kontejnery přímo z paměti aplikace zapsat do binárního souboru objektů za pomocí nástroje z knihovny *java.io*. Pro ukládání třídou *ObjectOutputStream* prostřednictvím metody *writeObject* a naopak k načítání pomocí třídy *ObjectInputStream* metodou *readObjec*t, kde se musí dále přetypovat na svoji instanci. Díky této implementaci dojde k automatické serializaci a zpětné deserializaci. Jelikož jsou tyto soubory serializované a binární, jdou přečíst jedině pomocí instancí souhlasných kontejnerů, kterými byly uloženy, proto je může přečíst jenom vypracovaná aplikace požadované verze. Koncovky souborů nesou číslici, ta označuje verzi serializovaných kontejnerů. Zkratka v koncovce „*zam*“ znamená záznam, „*P*“ projekt a „*bak*“ backup neboli záloha.

## 4.2 Soubor sdílených zákazníků

Stejně jako u souboru nastavení, i zde bylo využito konstant, jméno souboru bylo určeno „*DataPak.zam1*“, v aplikaci je navíc správa záloh zákazníků, která zálohu tohoto souboru uloží s rozdílnou koncovkou do souboru „*DataPak.bak*“. I zde, stejně jako u souboru nastavení, je pracováno se složkou pracovního adresáře aplikace. Soubor je sdílený nejenom s jediným projektem, ale se všemi, které uživatel vytvoří. Soubor zákazníků obsahuje serializovanou třídu *SeznamZakazniku*.

## 4.3 Kontejner projektu

Na rozdíl od souboru nastavení a seznamu zákazníků se soubor kontejneru projektu neukládá do předem stanoveného adresáře, ale je uživateli nabídnuto podle libosti si adresář vybrat. Celé jméno tohoto souboru také není předem známé, pouze jeho koncovka „*.Pzam1*“, zbytek názvu souboru je určeno podle libosti uživatele. Ale i kontejner projektu má možnost vytvoření záložního souboru, tentokrát s koncovkou „*.Pbak*“, záloha souboru bude nést název původního, ukládána a hledána bude vždy v adresáři s původním souborem. Soubor kontejner projektu obsahuje serializovanou třídu *SouborKontejner*, kde jsou zahrnuté kontejnery *SeznamFaktur*, *SeznamAkci*, *SeznamDeniku*, *SeznamKnihaJizd*, *SeznamSmlouvaODilo* a *SeznamCelkoveVuctovani*.

## 4.4 Export a import XML souborů

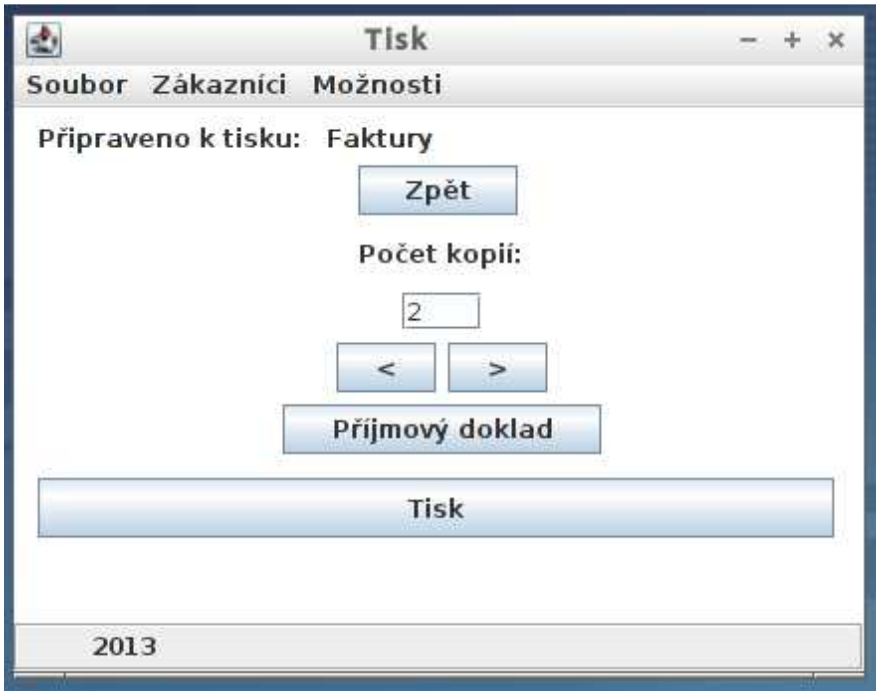
Jednotlivé soubory mají koncovku „*.xml*“ odpovídající formátu v rámci importu i exportu, uživatel může pracovat s jakýmkoli adresářem, který si prostřednictvím třídy *JFileChooser* v okně metody *showSaveDialog* nebo *showOpenDialog* vybere. Soubor je čistě textový a lze číst jakýmkoli prohlížečem textu[10].

## 4.5 Soubor nastavení

K sjednocení jména a přípony souboru bylo využito konstant uložených v „*Konstanty.Kons*“, konkrétně pro nastavení pro současnou verzi byly nastaveny na „*Settings.zam1*“. Tento soubor bude vždy ukládán do složky pracovního adresáře aplikace, kde jej při spuštění také bude hledat. Soubor nastavení obsahuje serializovanou třídu *Nastaveni*. Ta obsahuje kontejner s jednotlivými sazbami DPH, kontejner položek s výchozí cenou, dále informaci o preferovaném druhu faktury v logické proměnné typu *boolean* a patnáct proměnných typu *String*, které obsahují informace o živnostníkovi.

# 5 Tisk dokumentů

Jak již v této práci bylo zmíněné, o tisk v aplikaci se starají třídy *Spravce_tisku*, *TiskKontejner* a *Tisk*. Právě panel *Tisk* reprezentuje pouze komunikační článek mezi uživatelem a tiskovou částí aplikace, třída *Spravce_tisku* připraví náležité prostředky před zahájením tisku od interakce uživatele a to formát papíru s jeho okraji i nástroj k tisku. Ke konkrétním podobám výsledného dokumentu bylo zpracováno teprve až ve třídě *TiskKontejner*. Díky tomu pro změnu tiskové sestavy nemusí dojít ke změnám v celém prostředí tisku, ale pouze změně kontejneru, který zajišťuje konečnou podobu dokumentů [5].




**Obrázek 7 – Ukázka panelu tisku**

Aplikace v současné verzi tiskne pět druhů dokumentů, některé jsou si ale velice podobné. Prvním z pěti je vytvořená smlouva o dílo, ta se skládá převážně z řádků textů nadpisů a odstavců. Nadpisy jsou centrované na střed s tučnějším písmem než zbytek dokumentu.

Dalším v pořadí je faktura, ty jsou ale dvojího druhu. I tak některé náležitosti mají společné. Oba dokumenty vlastní centrovaný název „Faktura - Daňový doklad“ s číslem faktury, informace o živnostníkovi a zákazníkovi v oddělených rámcích. Mezi předešlými rámci a následujícím je informace o živnostenském listě uživatele, který aplikaci zadal v nastavení. Následuje samostatný rámec, který nese vyjádření o formě úhrady fakturované částky, datum vystavení zdanitelného plnění a samozřejmě datum splatnosti. Nechybí zde ani konstantní symbol. Nyní se ale typy faktury rozcházejí, faktura typu celkové ceny zde disponuje slovním vyjádřením o vyúčtované částce „Fakturuji Vám:“. Konečný rozdíl je ve výpisu sloupců, kde faktura typu celkové ceny vypisuje pouze zadané částky pro

jednotlivé daně, zatímco faktura typu jednotlivých položek zde vypisuje každou účetní položku s popisem a dosaženým množstvím. Následuje souhrnná částka proplacených záloh, která se přímo odečte od výsledné ceny, která je následně vypsána do rámečku. Následuje již jen místo pro podpis jednotlivých účastníků.

Čtvrtým z pěti je celkové vyúčtování, které vlastní podobnou hlavičku jako dvě předchozí faktury. Liší se ale obsahem, jako nadpis používá „Celkové vyúčtování akce“ a místo přehledu o formě úhrady a splatnosti nabízí celkový cenový přehled včetně DPH. Do sloupců k jednotlivým položkám navíc přibyl údaj o měrné jednotce účetní položky.

Je-li faktura, proplacena v hotovosti, živnostník musí doložit doklad o zaplacení částky, toho dosáhne posledním dokumentem „Příjmový pokladní doklad“. Nese stejné číslo jako vystavená faktura, jsou zde uvedeny podrobné údaje o živnostníkovi, který aplikaci zadal v rámci bloku „Dodavatel“, zadané údaje o zákazníkovi, kterému byla faktura vystavena v bloku „Zákazník“, dále číselné a slovní vyjádření částky a nakonec místo pro podpis jak dodavatele, tak i zákazníka.

Pro sestavení jednotlivých dokumentu bylo využito grafického vykreslení pomocí nástrojů *Graphics* a *Graphics2D*. Jednotlivé texty jsou vypisovány pomocí metody *drawString*, v dalších částech byly vykreslovány čáry metodou *drawLine* a obdélníky *drawRect*. Před samotným vykreslením dokumentu není předem známý počet stran, proto před vykreslováním jsou údaje k tisku předem připravovány.

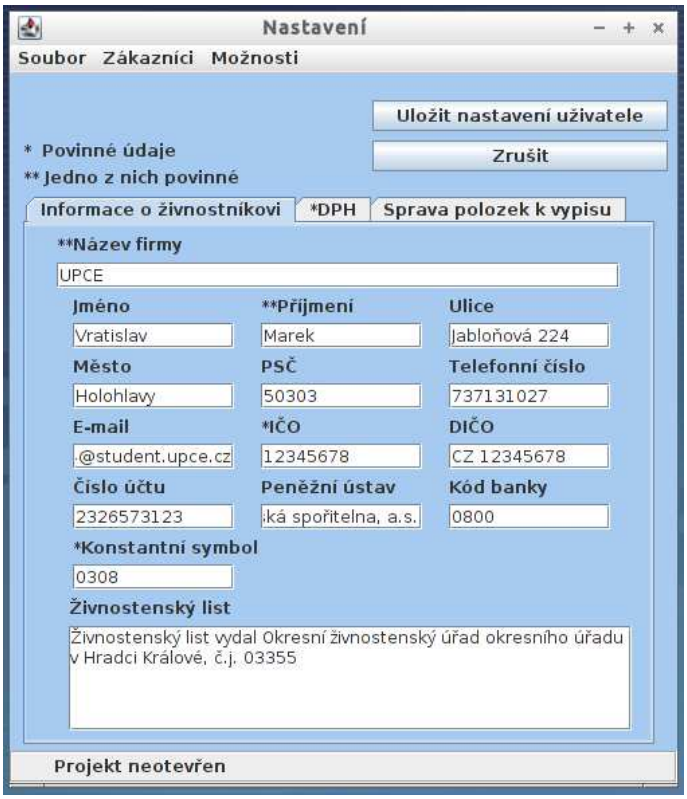
U čistě textových dokumentu, jako je smlouva o dílo, se shromažďují počty použitých řádku do pole stringů, pomocí třídy *FontMetrics* se vypočítá výška jednotlivých řádků a délka znaku či řetězců znaků. Pokud délka textu přesahuje šířku samotného řádku, je metodou *linewrap* rozdělena na další řádek. Polem stringů se ale nedá přenést údaj o vycentrování, proto se do pole centrované texty přidávají přes metodu *center*, ta vypočítá pomocí metriky odpovídající počet mezer z leva, jež text vycentruje. Po připravení tištěného textu dojde k výpočtu, zdali se výsledný obrazec vejde na jedinou stránku, pokud ne, doplní se do textu údaj o identifikaci, současné číslo stránky a celkem tištěných stran.

Dokument „Příjmový pokladní doklad“ je grafický a k předchozímu připravování nedojde právě proto, že je přímo dimenzován na jednu stránku papíru, na rozdíl od zbylých tří, kde se skládá z grafické části statické a dynamické. Statická část se vždy na první stránku vejde, ale dynamické součásti se musí dopočítat. Každá ze tří má předpřipravený počet řádků, které určují staticky vykreslovanou část, od této části je počítán každý dynamický řádek. Konkrétně jsou tyto dokumenty statické po záhlaví tabulek nad částkami, a jednotlivé částky jsou už přidávány do kontejneru ke zpracování.

# 6 Ovládání programu

V této kapitole bude popsána ukázková práce s výslednou aplikací krok po kroku, jak postupovat od prvního spuštění až k vytištění faktury.

Protože při získání aplikace se ve většině případů nepřenáší konfigurační soubory, z důvodu potřeby vytvořit si vlastní odpovídající právě údajům živnostníka, absence souboru nastavení uživatele odkáže na stav „První spuštění“. Aplikace v tomto stavu uživatelovi znemožní jakoukoli další aktivitu, než je zadání hlavně povinných součástí nastavení. Jak se v aplikaci může uživatel dočíst, povinné údaje jsou označené „*“, mezi některými si v zadání může uživatel vybrat, ty jsou označeny „**“. Uživatel tedy zadá údaje o sobě, které se rozhodl zadat. Dále uživatel musí přejít na záložku „DPH“, kde by měl zkontrolovat, zda li výchozí nastavení sazby daní odpovídá současné situaci a v případě potřeby je upravit.




**Obrázek 8 – Ukázková práce s aplikací - nastavení**

Po uložení se vytvoří na disku soubor, s uživatelem doplněnými údaji. Následně uživateli aplikace nabídne hlavní panel výběru, kde mu oznamuje skutečnost, že není projekt aplikace, a proto mu nejsou některé funkce dostupné. Nejprve na rolovací nabídce vybere uživatel účetní ročník, pro který chce s projektem pracovat, ve výchozím stavu je vybrán právě aktuální. Uživatel v nabídce „Soubor“ vybere položku „Nový projekt“, tím se mu otevře okno s výběrem adresáře a názvu nového souboru, výběr započne právě v pracovním adresáři aplikace. V ukázce uživatel vybere soubor „2013“ v aktuálním adresáři. Tím se uživateli otevřou všechny možnosti aplikace, kromě těch, které jsou

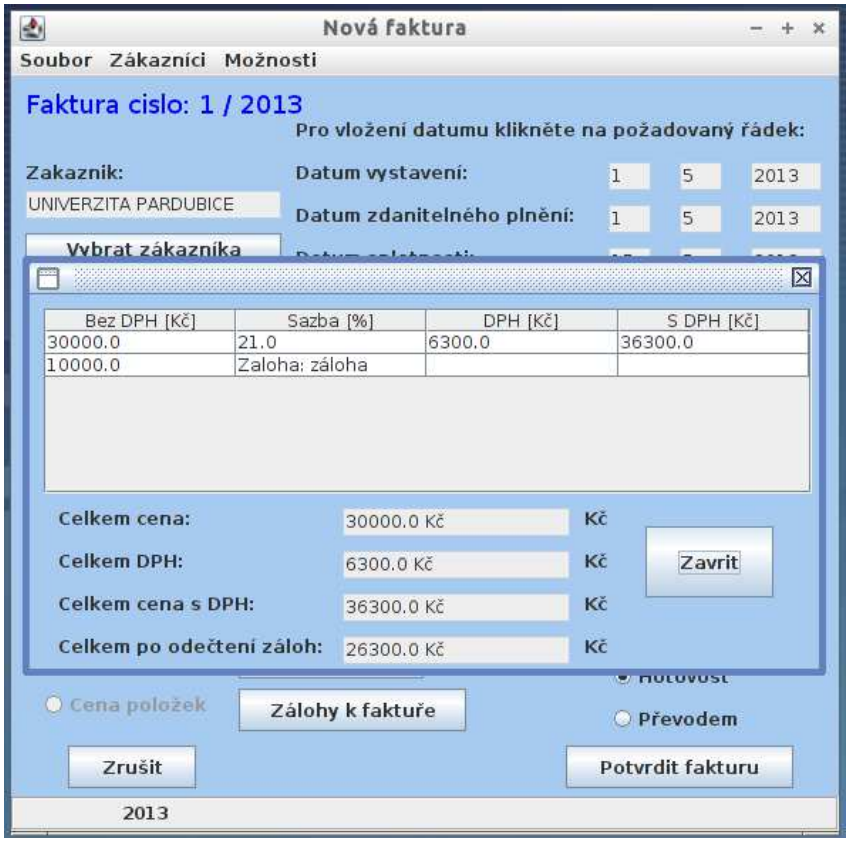
přístupné výhradně po naplnění první položkou. Soubor projektu se ale na disku nevytvořil, ten se vytvoří teprve ve chvíli, kdy projekt uživatel uloží. Aplikace uživateli však nedovolí uložit projekt do doby, než jej naplní první položkou.


**Obrázek 9 – Ukázková práce s aplikací - hlavní panel**

V rámci ukázky uživatel přejde tlačítkem „Faktury“ do části zabývajícím se právě problematikou vytváření a správou. Posléze uživatel bude panelem „Správa faktur“ informován nepřístupným seznamem zákazníků o jeho prázdnosti. Proto uživatel vybere možnost „Nový zákazník“, kde zadá potřebné povinné údaje označené „**“ se stejným významem jako v nastavení. Z toho pro uživatele vyplývá, že je alespoň jeden ze tří údajů o zákazníkovi povinný a to Název firmy, Jméno, nebo Příjmení. V rámci ukázky tedy uživatel zadá všechny údaje, které o zákazníkovi potřebuje uchovat a tabulku potvrdí tlačítkem „Potvrdit zákazníka“. Aplikace uživatele přesměruje do panelu se seznamem zákazníků, kde je jeho nový zákazník vyobrazen a ve stavové liště zahájí blikavé upozornění o neuložených nově zadaných údajích. Uživatel chce zákazníka uložit, a tak použije tlačítko odpovídajícího jména, jeho funkci na panelu „Seznam zákazníků“, kde se právě nachází, nebo může v nabídce aplikace „Zákazníci“ vybrat možnost „Uložit zákazníky“, která má stejnou funkci. Následně potom aplikace přestane upozorňovat na neuložený stav.

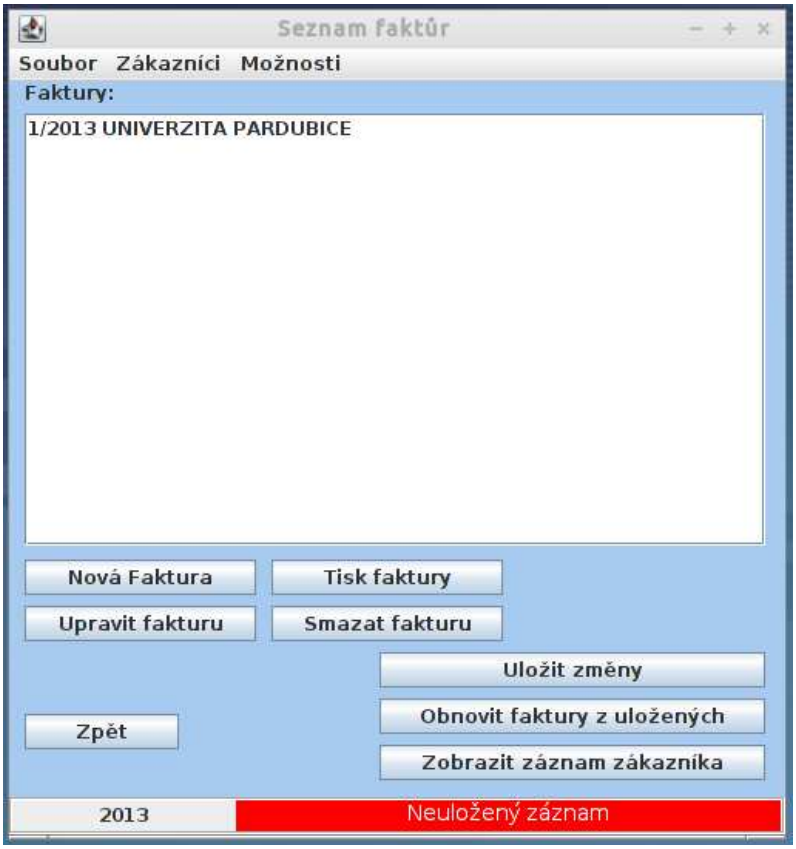

**Obrázek 10 – Ukázková práce s aplikací - nový zákazník**

Nyní má uživatel zadané* potřebné informace před tvorbou samotné faktury. Uživatel se tlačítkem „Zpět“ přesune na zprávu faktur, kde následně vybere možnost „Nová faktura“. V panelu „Nová faktura“ uživatel prve chce vybrat zákazníka, použije tlačítko „Vybrat zákazníka“, které ho přesune do panelu „Výběr zákazníků“ se seznamem vložených zákazníků a po označení jednoho z nich se naplní podrobnosti právě vybraným zákazníkem. Po označení uživatelem zvoleného zákazníka, uživatel potvrdí svůj výběr tlačítkem „Vybrat“ a vrátí se do panelu „Nová faktura“. Následně po kliknutí na stanovený řádek pro vybrané datum se zobrazí v interním panelu kalendář, který uživateli nabízí snadný výběr požadovaného data a zbylé podle něho dopočítá. Pokud se uživateli nebudou dopočítaná data líbit, na příslušném řádku je stejným způsobem má možnost změnit. Uživatel do formuláře pro text „Fakturuji Vám:“ zadal text „Vzorová faktura: vypracováni bakalářské práce“. Uživatel dále vybere daňovou sazbu 21% v seznamu DPH a následně zadá částku 30000 do kolonky Částka, zadávání potvrdí tlačítkem „Vložit částku“. Bezprostředně po potvrzení se částka vypíše s určenou sazbou do seznamu „Částka k DPH“. Uživateli v rámci ukázky byla zaplacena záloha, a tak vybere možnost „Zálohy k faktuře“, změní se interní panel panelu a nabídne uživateli seznam právě vložených záloh a nabídku pro vložení nové. Uživatel do kolonky částka sepíše 10000 a do popisu zálohy si připíše „záloha“, potvrdí tlačítkem „Vložit zálohu“ a ihned se to projeví v seznamu. Znovu stiskne volbu „Zálohy k faktuře“ a panel se vrátí do původního stavu. Nyní chce uživatel vědět podrobnosti o částkách a dani, proto vybere možnost „Zobrazit DPH“, uživateli se zobrazí okno s výčtem všech zadaných prvků a požadovaná statistika. Uživatel spokojen tlačítkem „Zavřít“ tabulku zavře, zkontroluje způsob platby, zda je v hotovosti a využije možnosti „Potvrdit fakturu“.




**Obrázek 11 – Ukázková práce s aplikací – nová faktura**

Uživatel je přesměrován po vložení nové faktury na panel „Seznam faktur“, kde ho hned aplikace upozorňuje, že nový záznam nebyl uložen. Uživatel potřebuje fakturu vytisknout a tak v seznamu požadovanou vybere a stiskne tlačítko „Tisk faktury“. To uživatele přesměruje na panel tisku, zde uživatel ponechá dvě kopie a stiskem tlačítka tisk vyvolá nabídku tiskáren a podrobnosti tisku. Uživatel vše podle svého systému vybere a potvrdí. Dojde k tisku dvou faktur, uživatel k tomu potřebuje ještě příjmový doklad, a tak stiskne tlačítko shodného jména. Situace se opakuje a uživateli se vytiskne příjmový pokladní doklad. Uživatel další pohledávky nemá a tak aplikaci zavírá. Aplikace stále nemá všechny zadané údaje uložené a před vlastním zavřením se uživatele dotáže, zda li si přeje všechny své změny uložit. Uživatel vybere možnost „Ano“, aplikace všechny kontejnery uloží a ukončí se.




**Obrázek 12 – Ukázková práce s aplikací - seznam faktur**

# Závěr

Při vývoji aplikace jsem směřoval jednotlivé kroky pro její další možný vývoj a rozšíření. Zdrojové kódy obsahují odkazy na metody nebo třídy, které dosud nebyly plně, nebo vůbec implementovány, nicméně běžnému uživateli jsou komponenty na neimplementované odkazy a třídy skryté. Ale počítá se s jejich připojením a doplnění v dalším vývoji aplikace.

Během vývoje své práce jsem učinil dvojici zásadních chyb, ze kterých jsem se poučil. První zásadní chyby jsem se dopustil hned u konceptu jádra aplikace. Původní návrh jádra byl v konceptu jedné třídy, která zajišťovala hlavní chod aplikace. Tento návrh byl funkční, ale pro rozsah mé práce byl neudržitelný a nepřehledný. Druhé chyby jsem se dopustil v okamžiku, kdy jsem se rozhodl k nápravě chyby první. Z práce jsem odstranil pouze jádro a to, co na něj navazovalo, jsem ponechal. Tím jsem si zastavil vývoj práce a zahájil pouze její opravu, jež mě stála příliš času. Pro příští práce jsem poučen a u další práce již předem navrhnu dostatečně dimenzované jádro pro daný rozsah práce. V případě kdy narazím na zásadní chybu provedení, sepíšu chyby, které jsem objevil a začnu práci od samého začátku.

V práci se mi povedlo dosáhnout všech mnou předpokládaných cílů. V průběhu vypracování práce jsem se nesetkal s neřešitelnými problémy a na všechny jsem našel kompromisní řešení. Největší překážkou mi byla nezkušenost s vývojem tak rozsáhlé aplikace a nespolupráce některých komponent. Postupem vývoje jsem nabíral zkušenosti, jež pro mě jsou velice cenné.

## Literatura


[1] SEDLÁČEK, Jaroslav a Petr VALOUCH. *Jednoduché účetnictví: vzorové příklady s řešením*. Vyd. 1. Brno: Computer Press, 2003, 173 s. Daně a účetnictví (Computer Press). ISBN 80-722-6967-4.

[2] ZEMÁNEK, Josef a Jiří LACINA. *Příručka pro začínající podnikatele*. Vyd. 1. Kralice na Hané: Computer Media, 2011, 96 s. ISBN 978-80-7402-109-1.

[3] MCLAUGHLIN, Brett. *Java*. 3rd ed. Sebastopol, CA: O'Reilly, c2007, 465 p. ISBN 05-961-0149-X.

[4] ECKEL, Bruce. Myslíme v jazyku Java: knihovna programátora. Praha: Grada, 2000, 431 s. ISBN 80-247-9010-6.

[5] Lesson: Printing. ORACLE. The Java Tutorials [online]. 1995 [cit. 2013-05-05]. Dostupné z: http://docs.oracle.com/javase/tutorial/2d/printing/

[6] HÉGARET, Philippe Le. W3C DOM IG. Document Object Model (DOM) [online]. January 19, 2005 [cit. 2013-05-05]. Dostupné z: http://www.w3.org/DOM/

[7] JCalendar Java Bean. TÖDTER, Kai. Toedter.com [online]. 2012 [cit. 2013-05-05]. Dostupné z: http://www.toedter.com/

[8] Vedení jednoduchého účetnictví. MF SERVIS. MS - SERVIS [online]. 2009 [cit. 2013-05-05]. Dostupné z: http://www.mf-servis.eu/vedeni-ucetnictvi/vedeni-jednoduch-ucetnictvi/

[9] JANOUŠEK, Ing. Karel. AUTOPLAN. KROBSOFTWARE S.R.O. KNIHA JÍZD [online]. 21.10.2009 [cit. 2013-05-05]. Dostupné z: http://www.autoplan.cz /upload/images/file/zrusknjiz.pdf

[10] KOSEK, Jiří. XML: eXtensible Markup Language. Kosek.cz [online]. 1999, 28. května 1999 [cit. 2013-05-05]. Dostupné z: http://www.kosek.cz/clanky/xml/

[11] SVÁK, Karel. Faktury z pohledu obchodního zákoníku, zákona o účetnictví a zákona o DPH. SG-SOFT, spol. s r. o. SG-SOFT [online]. 14.01.2007 [cit. 2013-05-05]. Dostupné z: http://www.sg-soft.cz/zzz_popisy/zzz_f%5Co_fakturach.htm

[12] ZEMÁNEK, Josef. Faktura. Euroekonom.cz [online]. 15.4.2011 [cit. 2013-05-05]. Dostupné z: http://www.euroekonom.cz/podnikani-faktura.php

# Příloha A – Vzorový výtisk příjmového pokladního dokladu

**Příjmový pokladní doklad č. 1/2013**

| **Dodavatel :** | **Zakazník :** | **Dnes: 29.04.2013** |
|---|---|---|
| UPCE Vratislav Marek<br>Jabloňová 224<br>Holohlavy 50303<br>IČO: 12345678<br>DIČ: CZ 12345678<br>Peněžní ústav: Česká spořitelna, a.s.<br>Číslo účtu: 2326573123/0800<br>Telefoní číslo: 737131027<br>E-Mail: st28594@student.upce.cz | UNIVERZITA PARDUBICE<br>Studentská 95<br>Pardubice 2 532 10<br>IČO: 00216275<br>DIČ: CZ00216275<br>Telefoní číslo: +420 466 036 111<br>E-Mail: webmaster@upce.cz | |
| Částka: 26300.0Kč | Slovy: dvacetšesttisictřistakorun | |
| **Podpis dodavatele:** | **Podpis zakazníka:** | |

**Příjmový pokladní doklad č. 1/2013**

| **Dodavatel :** | **Zakazník :** | **Dnes: 29.04.2013** |
|---|---|---|
| UPCE Vratislav Marek<br>Jabloňová 224<br>Holohlavy 50303<br>IČO: 12345678<br>DIČ: CZ 12345678<br>Peněžní ústav: Česká spořitelna, a.s.<br>Číslo účtu: 2326573123/0800<br>Telefoní číslo: 737131027<br>E-Mail: st28594@student.upce.cz | UNIVERZITA PARDUBICE<br>Studentská 95<br>Pardubice 2 532 10<br>IČO: 00216275<br>DIČ: CZ00216275<br>Telefoní číslo: +420 466 036 111<br>E-Mail: webmaster@upce.cz | |
| Částka: 26300.0Kč | Slovy: dvacetšesttisictřistakorun | |
| **Podpis dodavatele:** | **Podpis zakazníka:** | |

# Příloha B – Vzorový výtisk Faktura – Daňový doklad

## Faktura - Daňový doklad č. 1/2013

**Dodavatel :**

UPCE Vratislav Marek
Jabloňová 224
Holohlavy 50303
IČO: 12345678
DIČ: CZ 12345678
Peněžní ústav: Česká spořitelna, a.s.
Číslo účtu: 2326573123/0800
Telefoní číslo: 737131027
E-Mail: st28594@student.upce.cz

**Zakazník :**

UNIVERZITA PARDUBICE
Studentská 95
Pardubice 2 532 10
IČO: 00216275
DIČ: CZ00216275
Telefoní číslo: +420 466 036 111
E-Mail: webmaster@upce.cz

Živnostenský list vydal Okresní živnostenský úřad okresního úřadu v Hradci Králové, č.j. 03355

**Dodací a platební podmínky:**

| | |
|---|---|
| Forma úhrady: | v hotovosti |
| Datum vystavení: | 11.4.2013 |
| Datum zdanitelného plnění: | 11.4.2013 |
| Datum splatnosti: | 25.4.2013 |
| Konstantní symbol: | 0308 |

**Fakturuji Vám:** Vzorová faktura: Vypracování bakalářské práce

| Cena Bez DPH [Kč] | Sazba DPH | Částka DPH [Kč] | Cena s DPH [Kč] |
|---|---|---|---|
| 30000.0 | 21.0 | 6300.0 | 36300.0 |

Záloha: -10000.0 Kč

Celkem: 26300.0 Kč

Podpis Odběratel

Podpis Dodavatel

## Příloha C – Zdrojový kód třídy BlikaniUpozorneni

```
public class BlikaniUpozorneni implements Runnable {
    private Init init = null;
    private boolean bli = false;
    private final Object zamek = new Object();
    private volatile boolean pozastavit = false;
    private volatile boolean zastavit = false;
    private boolean blikProjekt;
    private boolean blikZakaznici;

    public BlikaniUpozorneni(Init init) {
        this.init = init;
    }

    public void setBlik(boolean Z, boolean P) {
        blikProjekt = !P;
        blikZakaznici = !Z;
    }

    @Override
    public void run() {
        while (!zastavit) {
            while (!pozastavit) {
                init.Spravce_vypisu().setStatus("", "");
                String temp = "";
                String temp0 = "";
                if (blikZakaznici || blikProjekt) {
                    temp += "Neuložený záznam ";
                    temp0 += "Neuložený záznam";
                }
                if (blikZakaznici && blikProjekt) {
                    temp += "Zakazniků a Projektu";
                } else if (blikZakaznici) {
                    temp += "Zakazniků";
                } else if (blikProjekt) {
                    temp += "Projektu";
                }
                //System.err.println("Projekt " + blikProjekt + "
Zakaznici " + blikZakaznici);
Logger.getLogger(BlikaniUpozorneni.class.getName()).log(Level.SEVERE,
null, ex);
                init.Spravce_vypisu().setStatus(temp0, temp);
                if (bli) {
                    bli = false;
                } else {
                    bli = true;
                }
                init.Spravce_vypisu().setStatus(bli);
                try {
                    if (bli) {
                        Thread.sleep(1500);
                    } else {
                        Thread.sleep(800);
                    }
                } catch (InterruptedException ex) {

Logger.getLogger(BlikaniUpozorneni.class.getName()).log(Level.SEVERE,
null, ex);
                }
```

```
            }
            init.Spravce_vypisu().resetStatus();
            synchronized (zamek) {
                try {
                    zamek.wait();
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    return;
                }
            }
        }
    }

    public void suspend() {
        pozastavit = true;
    }

    public void stop() {
        pozastavit = true;
        zastavit = true;
        synchronized (zamek) {
            zamek.notifyAll();
        }
    }

    public void resume() {
        pozastavit = false;
        synchronized (zamek) {
            zamek.notifyAll();
        }
    }
}
```

## Příloha D – Vzorový export XML faktury

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SeznamFaktur>
    <Faktura id="1">
        <ROK>2013</ROK>
        <CISLOZAKAZNIKA>0</CISLOZAKAZNIKA>
        <DEN>11</DEN>
        <MESIC>4</MESIC>
        <FAKTURUJIVAM>Vzorová faktura: Vypracování bakalářské
práce</FAKTURUJIVAM>
        <SEZNAMDPH>
            <DPH>15.0</DPH>
            <DPH>21.0</DPH>
        </SEZNAMDPH>
        <HOTOVOST bool="1"/>
        <SEZNAMZALOHPOLOZKA>
            <ZalohaPolozka>
                <CASTKA>10000.0</CASTKA>
                <POPISZALOHY>prvotní</POPISZALOHY>
            </ZalohaPolozka>
        </SEZNAMZALOHPOLOZKA>
        <SEZNAMCASTEK isCastkaPolozka="1">
            <CASTKADPH>
                <CASTKA>30000.0</CASTKA>
                <SAZBA>21.0</SAZBA>
            </CASTKADPH>
        </SEZNAMCASTEK>
        <Zakaznik id="0">
            <DICO>CZ00216275 </DICO>
            <EMAIL>webmaster@upce.cz</EMAIL>
            <ICO>00216275</ICO>
            <JMENO/>
            <MESTO>Pardubice 2</MESTO>
            <NAZEVFIRMY>UNIVERZITA PARDUBICE</NAZEVFIRMY>
            <PRIJMENI/>
            <PSC>532 10</PSC>
            <ULICE>Studentská 95</ULICE>
            <ZELEFON>+420 466 036 111</ZELEFON>
        </Zakaznik>
        <SDEN>25</SDEN>
        <SMESIC>4</SMESIC>
        <SROK>2013</SROK>
    </Faktura>
</SeznamFaktur>
```

## Příloha E – Vzorový export XML seznamu zákazníků

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SeznamZakazniku>
    <Zakaznik id="0">
        <DICO>CZ00216275 </DICO>
        <EMAIL>webmaster@upce.cz</EMAIL>
        <ICO>00216275</ICO>
        <JMENO/>
        <MESTO>Pardubice 2</MESTO>
        <NAZEVFIRMY>UNIVERZITA PARDUBICE</NAZEVFIRMY>
        <PRIJMENI/>
        <PSC>532 10</PSC>
        <ULICE>Studentská 95</ULICE>
        <ZELEFON>+420 466 036 111</ZELEFON>
    </Zakaznik>
    <Zakaznik id="1">
        <DICO/>
        <EMAIL/>
        <ICO/>
        <JMENO>Jakub</JMENO>
        <MESTO>Ebenovice</MESTO>
        <NAZEVFIRMY>Vzorový</NAZEVFIRMY>
        <PRIJMENI>Monich</PRIJMENI>
        <PSC>705 88</PSC>
        <ULICE>Medova 308</ULICE>
        <ZELEFON/>
    </Zakaznik>
    <Zakaznik id="2">
        <DICO/>
        <EMAIL/>
        <ICO/>
        <JMENO>Josef</JMENO>
        <MESTO>Pribislavice</MESTO>
        <NAZEVFIRMY>Vzorový</NAZEVFIRMY>
        <PRIJMENI>Matejka</PRIJMENI>
        <PSC>608 50</PSC>
        <ULICE>Janovická 851</ULICE>
        <ZELEFON/>
    </Zakaznik>
    <Zakaznik id="3">
        <DICO/>
        <EMAIL/>
        <ICO/>
        <JMENO>Matejka</JMENO>
        <MESTO>Sokolov</MESTO>
        <NAZEVFIRMY>Vzorový</NAZEVFIRMY>
        <PRIJMENI>Přibislav</PRIJMENI>
        <PSC>186 87</PSC>
        <ULICE>Mohutná 76</ULICE>
        <ZELEFON/>
    </Zakaznik>
</SeznamZakazniku>
```