Rozšíření programového systému NAHOS pro řízení nástrojového hospodářství

Bc. Marek Hlaváček

Diplomová práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav řízení procesů
akademický rok: 2006/2007

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Marek HLAVÁČEK**
Studijní program:	**N 2807 Chemické a procesní inženýrství**
Studijní obor:	**Automatizace a řídicí technika**
Téma práce:	**Rozšíření programového systému NAHOS pro řízení nástrojového hospodářství**

Zásady pro vypracování:

1. Rozšíření programových modulů pro tisk o možnosti tisku formulářů a tisku vícestránkových sestav.
2. Začlenění prvku typu "menu" do formulářů NAHOS.

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

[1] Šimůnek, Milan: SQL Kompletní kapesní průvodce, Grada, Praha1999, ISBN 80-7169-692-7
[2] Spell,B.: Java Programujeme profesionálně, Computer Press, Praha 2002, ISBN 80-7226-667-5
[3] Bradley,M.: XML Kompletní průvodce, Grada Publishing, Praha 2000, ISBN 80-7169-949-7
[4] Vařecha, Martin: Diplomová práce "Programové vybavení pro evidenci nářadí ve strojírenské výrobě - vzdálená správa databáze nářadí", 2003
[5] Verbovský, Jakub: Diplomová práce "Software for tools records an Engineering industry - communication with user", 2003
[6] Svoboda, Michal: Diplomová práce "Rozšíření programového systému pro správu nástrojového hospodářství ve strojírenské výrobě", 2004
[7] Hulman, Martin: rozpracovaná disertační práce "Návrh programového subsystému pro řízení nástrojového hospodářství"

Vedoucí diplomové práce: **doc. Ing. Lubomír Vašek, CSc.**
Ústav aplikované informatiky

Datum zadání diplomové práce: **13. února 2007**

Termín odevzdání diplomové práce: **24. května 2007**

Ve Zlíně dne 13. února 2007

prof. Ing. Vladimír Vašek, CSc.
děkan

L.S.

prof. Ing. Petr Dostál, CSc.
ředitel ústavu

ABSTRAKT

Tato práce pojednává o návrhu a implementaci programového vybavení rozšiřujícího informační systém nástrojového hospodářství NAHOS ve strojírenském podniku Tajmac-ZPS Zlín, a.s. o možnosti tisku formulářů a vícestránkových sestav a o využívání prvku typu „menu“. Distribuovaný systém je navržen ve třívrstvé architektuře klient/server s tlustým klientem a je implementován v jazyce Java. Komunikace klient/server je implementována RMI modelem. Manipulace s databází nástrojů se provádí pomocí SQL dotazů prostřednictvím univerzálně definovatelných formulářů. K tvorbě a editaci těchto formulářů se využívá editoru který je součástí systému.

Práce zahrnuje část teoretickou shrnující popis informačních systémů, systému NAHOS a část experimentální, obsahující popis rozšíření systému s uživatelskou a řešitelskou dokumentací.

Klíčová slova: Informační systém, Databázový systém, Nástrojové hospodářství, Strojírenský podnik

ABSTRACT

This dissertation deals with proposal and programme equipment implementation for form printing and multi-pages compositions and integration of factor “menu” type into information system NAHOS in engineering company Tajmac-ZPS Zlín, a.s.. Distributed system is proposed in three-ply architecture client/server with thick client and is implemented in Java language. Communication client/server is implemented with RMI model. Tools database manipulation is done with the assistance of SQL (structured query language) inquiries through universally definable forms. Editor, which is the part of the system is used for building and edition of these forms.

The work contains the theoretical part, which summarizes description of information systems, NAHOS system and the experimental part, which describes expanding of the system with users and solvers documentation.

Keywords: Information system, Database system, Instrumental management, Engineering company

Děkuji vedoucímu diplomové práce panu doc. Ing. Lubomíru Vaškovi za trpělivost a odborné vedení, rady a připomínky, které mi poskytoval při řešení diplomové práce.

Prohlašuji, že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD

Hlavním cílem této práce je návrh a realizace programových modulů pro tisk formulářů a vícestránkových sestav systému NAHOS pro nástrojové hospodářství. Systém byl navržen a vytvořen ve spolupráci strojírenského podniku Tajmac-ZPS Zlín a Univerzity Tomáše Bati ve Zlíně. Je výsledkem práce několika diplomantů a jejich jednotlivých prací. Ing. Martin Vařecha [1] navrhl a implementoval základ celé aplikace: práci s databází, síťovou komunikaci, autentifikaci uživatelů a zobrazení formulářů z definovaných XML dokumentů. Důležitou součástí celého systému je editor formulářů zpracovaný samostatně v diplomové práci Ing. Jakuba Verbovského [2]. Tento editor slouží k naprogramování uživatelských formulářů informačního systému pomocí jazyka XML. Ing. Michal Svoboda [3] provedl vyhodnocení požadavků zadavatele na informační systém NAHOS, které vycházely ze zjištěných nedostatků navrženého systému při jeho zavádění do praxe a na jejich základě navrhl rozšíření celého systému. Jeho bratr Ing. Petr Svoboda [4] upravil stávající editor o nové prvky (jako např skupina „radiobuttony“ či prvek „menu“) a zjednodušil práci s těmito prvky.

Mým druhým úkolem je realizovat programovou část zpracování prvku „menu“, navrženého Petrem Svobodou a začlenit ji do systému.

V závěrečné části zpracuji referenční příručku tiskových formulářů pro snadnější orientaci uživatele při jejich tvorbě.

I. TEORETICKÁ ČÁST

1 INFORMAČNÍ SYSTÉMY VE VÝROBNÍCH PODNICÍCH

Základním článkem organizace výroby je závod, v něm se uzavírá celý cyklus od objednání výrobku, technického, hmotného i organizačního zabezpečení jeho výroby, přes vlastní výrobu až po předání finálního výrobku odběrateli. Výrobní činnost závodu je rozdělena do řady provozů. Strojírenský výrobní cyklus představuje řadu procesů, např. přípravné procesy prováděné většinou na úrovni závodu, výrobní procesy probíhající na úrovni dílen a provozů, technologické procesy na pracovištích. Těmto úrovním odpovídají v zásadě i tři základní úrovně řízení: *závodní, provozní a technická*.

Informatika postupně proniká do nových oblastí našeho každodenního života. Rychlost vývoje informačních technologií společně se stávající dynamikou vývoje ekonomiky dnes staví společnosti před nové možnosti, jak optimalizovat některé tyto procesy a jejich řízení. Tlak trhu nepřetržitě nutí produkovat výrobky levněji, kvalitněji, efektivněji a rychleji. Jednou z cest, jak tomuto tlaku v tržní ekonomice čelit, je zavádění rozsáhlých informačních systémů do jednotlivých vrstev řízení podniku nebo rozšiřování již zavedených podnikových IS o nové segmenty odpovídající moderním trendům. Moderní výrobní podniky se bez podpory informačních a komunikačních systémů (IT systémů) neobejdou.

1.1 Informační systém

Jsou to systémy pro sběr, udržování, zpracování a poskytování informací a dat. Příkladem informačního systému může být kartotéka, telefonní seznam, kniha došlé pošty a nebo účetnictví. Systém nemusí být nutně automatizovaný pomocí počítačů a může být i v papírové podobě. Informacemi míníme sdělení, které odstraňuje nejistotu nebo nevědomost, daty míníme jakékoli zaznamenané poznatky či fakta. Jako zvláštní pojem zde vystupuje také znalost představující zobecnění poznání určité části reality. Informaci je možno také chápat jako data s nějakým přidaným významem (data + význam).

- Informace - [Latina], původní sdělení, zpráva, podávaná ústně, písemně nebo jiným způsobem (s pomocí signálů, technických prostředků); od poloviny 20. století obecně vědecký pojem, užívaný v řadě oborů, který znamená: a) zprávu o nějaké věci nebo situaci, jež se někomu předává; b)

zmenšení, snížení neurčitosti v důsledku získaného sdělení; c) sdělení těsně spojené s řízením, signály spojené se syntaktickými, sémantickými a pragmatickými charakteristikami; d) odraz různotvárnosti mezi libovolnými objekty a procesy v živé i neživé přírodě. [9]

- Data - existuje více definic, např.:
    - je výraz pro údaje, používané pro popis nějakého jevu nebo vlastnosti pozorovaného objektu. Data se získávají měřením nebo pozorováním, a lze je dělit na data spojitá a data nespojitá. Data spojitá se přitom vztahují k nějaké spojité stupnici, zatímco data nespojitá nikoliv.

Již dlouho je jasné, že hospodářství různých zemí netáhnou jen hmotné výrobky, ale také informace, znalosti a nové technologie. To si uvědomují i podniky a instituce, což napomáhá k rozvoji IS.

Data a informace IS jsou uloženy v databázových systémech viz. kap. 1.2., které však samy o sobě nejsou ve většině případů schopny poskytnout pohodlný, uživatelsky příjemný a současně distribuovaný přístup k datům. Pro prezentaci těchto dat uživateli je tedy vhodné využít distribuované aplikace, samotná struktura aplikace (jednotlivé vrstvy) je pak dána technologií řešení.

1.1.1 Důvody zavádění informačních systémů

Důvodem k zavádění IS do výrobních podniků je potřeba informací, které mají pro daný podnik přínos především z hlediska strategických výhod vůči konkurenci. Rostoucí složitost rozhodovacích procesů vyžaduje zavedení vhodně nastaveného IS, který svými funkcemi umožní získat co nejúplnější potřebné informace a tím přispěje ke zkracování doby potřebné pro přijetí rozhodnutí při rozhodovacích procesech a též k omezení rizik chybného rozhodování. Nasazení takového systému umožňuje výrobnímu podniku pružněji reagovat na potřeby inovace výrobků a služeb, dává lepší informace o využití výrobních kapacit vede k jejich efektivnějšímu využití, sledování toku materiálu a podobně. Všechny tyto činnosti umožňují růst podniku, zlepšení konkurence-schopnosti na současných trzích a mnohdy ušetření ne malých finančních prostředků.

1.1.2 Cíle, úlohy IS

Jednotlivé cíle, úlohy IS lze odvodit ze samotné struktury podniku a jeho jednotlivých procesů řízení a výroby na:

- strategické (plánování investic…)
- taktické (vedení, kontrola rozpočtů, návrh výroby,…)
- operační (každodenní rutina)

IS používané ve výrobních podnicích dělíme podle použití v jednotlivých úrovních řídicí struktury podniku do následujících skupin [8]:

- manažerské (EIS – Executive IS)
- taktické (DSS – Decision Support Systém)
- vedení (MIS – Management IS)
- expertní (KWS – Knowledge Work Systém)
- kancelářské (OIS – Office IS)
- operativní
    - TPS - transakční (banky…)
    - CRM - péče o zákazníky
    - RIS - rezervační systémy
    - CAM - konstrukční (CAD…)
    - GIS - geografické systémy

1.1.3 Projektování IS

Prvotním úkolem při projektování nového IS je zjišťování požadavků a hledání odpovědi na otázku, čemu má projektovaný IS sloužit, jaké funkce má plnit a jakých cílů má být jeho realizací dosaženo. Na základě zjištěných specifikací projektu je realizován systém tak, aby zajišťoval veškeré požadované funkce zadání.

Organizace řízení tvorby a návrhu systému má dnes tyto fáze [4, 8]:

- úvodní studie - specifikace požadavků na činnost (funkcionalitu) systému, studie proveditelnosti…
- rozbor zadání - podrobná formulace ve tvaru dokladu poskytující definitivní popis systému z pohledu uživatele pro potřeby jeho vývoje a ocenění jeho správnosti
- analytické modelování - modelování budoucího systému na konceptuální úrovni
- systémový design - modelování budoucího systému na technologické úrovni, převedení specifikací na hierarchii stále detailnějších schémat
- objektový design - definice požadovaných dat, procesů, jež se mají s daty provádět, až na úroveň, kdy mohou být vyjádřeny v podobě instrukcí pro počítačový program
- implementace - uvedení systému (hardwaru i softwaru) do provozu – instalace fyzického systému a aktivity s tím související, jako je školení operátorů a uživatelů
- zkušební provoz - nasazení systému v režimu testování, odzkoušení základních funkcí, ladící režimy
- nasazení - uvedení systému do provozu

Existuje spousta studií a metodik používaných pří návrhu IS, jako jsou např. následující [8]:

- procesně orientované přístupy (DeMarco, Gane/Sarson - velký důraz na DFD)
- datově orientované přístupy (Warnier/Orr - rozšíření o stavové diagramy)
- kombinace obou metod (tzv. Yourdonova metoda)
- strukturované metody (STC, JSP, JSD)

1.1.4 Architektura IS

Velmi důležitým hlediskem je volba architektury. Téměř výhradně se používá 3-vrstvá architektura. Jednotlivé vrstvy jsou:

- *prezentační (interakce s uživatelem)* - má za úlohu poskytovat uživatelské rozhraní pro uživatele a komunikovat s aplikační vrstvou
- *funkční (vlastní aplikace, bezpečnost, propojení se světem, kontrola…)* - komunikuje s prezentační vrstvou i s datovou vrstvou. Její úlohou je zpracovávat požadavky z prezentační vrstvy a odesílat jí požadované údaje, které se získají z datové vrstvy
- *datová (vlastní data)* - představuje určitý unifikovaný způsob ukládání údajů, aniž by aplikační vrstva věděla, kde a jak se ve skutečnosti údaje ukládají. Mohou to být jednoduché soubory na disku, většinou se ale jedná o relační databázi, ovládanou pomocí dotazovacího jazyka viz. kap. 1.2.1

Důležitá je i bezproblémová integrace IS, která má dvě hlediska: vnitřní, kde jde o proškolení pracovníků, nastavení prostředí a podobně, a vnější, kde se jedná zejména o zákazníky a dodavatele. Při volbě architektury bereme ohled také na:

- základní údaje (nejen samotného IS, ale také dodavatele, cenu)
- architekturu (zda-li mu bude vyhovovat)
- reference
- provozní prostředí (databázová platforma)
- vývojové prostředí (CASE nástroje)
- dokumentace, jazyková podpora
- doplňující služby (podpora, školení)
- standardy, specifikace, certifikace (audity, ISO-9001)
- flexibilita (možnost přizpůsobení)

1.2 Databázové systémy

Tvůrce informačního systému má za úkol zvolit vhodný databázový systém s ohledem na množství dat zpracovávaných systémem, rychlost zpracovávání, schopnost obsloužit požadované množství uživatelů, možnosti transakčního zpracování, uživatelské hledisko apod.

Databáze je určitá uspořádaná množina informací (dat) uložená na paměťovém médiu. V širším smyslu jsou součástí databáze i softwarové prostředky, které umožňují manipulaci s uloženými daty a přístup k nim. Tento systém se v české odborné literatuře nazývá systém řízení báze dat (SŘBD). Běžně se označením databáze - v závislosti na kontextu - myslí jak uložená data, tak i software (SŘBD).

1.2.1 Systém řízení báze dat (SŘBD)

Obstarává přístup k údajům uloženým v databázi. Tento název vznikl z původního anglického termínu DBMS (DataBase Management System). Na současném databázovém trhu existují tři základní typy SŘBD: *relační, objektově-relační a objektové* [1].

- *relační* - většina dnes používaných SŘBD při uspořádání údajů v databázi vychází z relačního modelu dat. V tomto modelu jsou údaje uspořádány do tabulek, které zpravidla shromažďují údaje o jednom druhu objektů. Relační SŘBD využívá jazyk SQL (*Structured Query Language*), jazyk obsahující příkazy pro definici dat, jejich úpravu a manipulaci s nimi

- *objektové* - pro objektové databáze neexistuje žádný oficiální standard. Kombinují se prvky objektově orientovaného programování s databázovými schopnostmi. Pro přístup k datům se používá objektově orientovaný jazyk(C++, Smalltalk, Java)

- *objektově-relační* - objektově-relační databázové systémy se snaží sjednotit rysy jak relačních, tak objektových databází. Do této kategorie patří, např. Informix, IBM, Oracle a Unisys, Postgre SQL. Pro přístup k datům se používá rozšíření jazyka SQL – SQL3.

1.2.2 SQL

Aby mohly být údaje z databáze přístupné ostatním aplikacím, musí SŘBD nabízet rozhraní, pomocí kterého s ním mohou spolupracovat ostatní programy . Aplikace používají jazyk SQL - Structured Query Language (strukturovaný dotazovací jazyk) pro zadávání požadavků na relační SŘBD. SQL je obecný nástroj pro manipulaci, správu a organizování dat uložených v relační databázi.

SQL není pouze dotazovací jazyk, ale s jeho pomocí je možno také definovat strukturu ukládaných dat, tedy strukturu tabulky, naplňovat sloupce tabulky (pole záznamů) daty a definovat vztahy a vazby mezi položkami dat. Dále umožňuje řízení přístupu k datům, tedy udělování a odebírání přístupových oprávnění na různých úrovních, čímž chrání data před náhodným nebo úmyslným poškozením, neautorizovaným čtením nebo manipulací s nimi, dále také umožňuje sdílené využívání dat. SQL není však plnohodnotným samostatným programovacím jazykem, např. proto, že se v něm ve většině implementací nenachází řídicí programové konstrukce a další požadované prvky, které by měl obsahovat každý obecný programovací jazyk. SQL je tedy standardizovaný nástroj pro práci s relačními databázemi [5].

1.2.2.1 MS Access

MS Access je program patřící do rodiny softwarových produktů MS Office, z toho vyplývá jeho zaměření na běžného uživatele. MS Access se nevyužívá pro složité databázové a internetové aplikace, nabízí jen malý výkon a nehodní se pro zpracování velkého množství dat. MS Access podporuje transakční zpracování SQL dotazů. K databázi lze přistupovat přes ODBC rozhraní. MS Access je závislý na platformě a operačním systému MS Windows.

1.2.2.2 MS SQL

Je databázový systém společnosti Microsoft, který funguje pouze na operačních systémech MS Windows (2000, XP, 2003 Server). Obsahuje funkce pro správu databází, tabulek, databázových relací apod. Integruje konfiguraci v grafickém uživatelském prostředí s možnostmi pro přístup více uživatelů. Systém podporuje transakční zpracování SQL a nabízí bezpečnost dat. Nevýhodou systému je vysoká cena a vyšší hardwarová náročnost, a také závislost na operačních systémech Windows.

1.2.2.3 MySQL

Databázový systém vyvinutý švédskou firmou MySQL AB, která se v současně době zabývá poskytováním profesionálních řešení na bázi MySQL nabízeného k volnému využívání. MySQL je multiplatformní s API pro celou řadu programovacích jazyků. Má stejně jako jiná SŘBD rozsáhlý repertoár datových typů a funkcí. V současnosti začalo MySQL podporovat i transakce a zamykání tabulek. Naopak tzv. uložené procedury MySQL nepodporuje. Pro zajištění bezpečnosti má v sobě MySQL zabudovaný systém uživatelů a práv. Nevýhodou MySQL je omezená kontrola konzistence dat, největší předností MySQL je rychlost.

1.2.2.4 Oracle

Oracle je velký, velmi výkonný komerční databázový systém, který integruje moderní bezpečnostní prvky pro ochranu dat, přístupy do databází přes uživatelská jména a hesla, podporuje transakční zpracování SQL. Nová verze (Oracle 10g) dokonce podporuje funkci - tzv. Grid computing, která umožňuje spojení více serverů tvořících dohromady jeden velký počítač, který dynamicky rozděluje výpočetní výkon podle aktuální potřeby jednotlivých aplikací. Oracle je systém nezávislý na platformě, podobně jako MySQL. Při tvorbě aplikací lze využívat nejrůznějších programovacích jazyků: C/C++, Perl, Java, JSP, PL/SQL, FastCGI.

1.2.2.5 Postgre SQL

PostgreSQL je objektově-relačním systémem řízení báze dat, založeným na projektu POSTGRES verze 4.2 Berkeleyské katedry počítačových věd Kalifornské univerzity v USA. PostgreSQL je přímý open-source následník původního kódu z Berkeley. PostgreSQL je považován za nejpropracovanější SŘBD s volně dostupným zdrojovým kódem. Také on je multiplatformní a poskytuje API rozličným programovacím jazykům. PostgreSQL je vhodný pro transakce, nahradil klasické zamykání tabulek systémem spravujícím více verzí dat. V praxi to znamená, že zatímco jedno vlákno nějaké záznamy přepisuje, jiné může bez přerušení ze stejné tabulky číst konzistentní data.

Bezpečnostní politika PostgreSQL je na mnohem vyšší úrovni než u MySQL. Kromě standardního přístupu chráněným heslem umožňuje PostgreSQL použití autentifikace pomocí Kerberosu nebo tzv. "Identifikačního protokolu" (ident-based authentication).

Víceuživatelský systém Postgre umožňuje slučovat různé uživatele do skupin s různými právy [1].

Jednotlivé databázové systémy mají odlišné parametry: množství ukládaných dat, rychlost prováděných operací, zabezpečení dat, uživatelská práva pro přístup k datům a operacím s daty, včetně podpory transakcí.

2 IS NÁSTROJOVÉHO HOSPODÁŘSTVÍ NAHOS

Programový subsystém zajišťuje zpracování dat nutných pro dynamický popis vztahů subsystému hospodaření s nářadím. Je zpracován ve formě programových modulů, které operují nad zvolenou databází, v nichž jsou uložena potřebná data. Programový subsystém je nezávislý na platformě, a pracuje v různých operačních systémech (Windows, Linux).

2.1 Funkce programového subsystému

V systému hospodaření s nářadím se provádějí tyto hlavní činnosti[1]:

1. Zadávání výroby
2. Sestavování nástrojových sestav
3. Seřizování nástrojových sestav
4. Doprava
5. Rozebírání nástrojových sestav
6. Správa nástrojů
7. Evidence

Zadávání výroby - zadání NC programů, podle nichž se bude vyrábět. Na základě těchto údajů se vytvoří seznam potřebných nástrojů, který bude předán do seřizovny

Sestavování nástrojových sestav - podle požadavků na vytvoření fyzických sestav nástrojů se podle tabulek vyberou potřebné komponenty. Tyto komponenty se odeberou ze skladu a vytvoří se z nich fyzické nástroje, které se umístí do skladu, odkud dochází k jejich expedici

Seřizování nástrojových sestav - na seřizovacím přístroji se seřídí nástroj na požadované rozměry, údaje o skutečných rozměrech se přesunou ze seřizovacího přístroje do databáze

Doprava - vytvoření požadovaných sestav nástrojů, tabulek korekcí k nim a jejich přemístění do příslušných zásobníků. V opačném směru - přemístění nepotřebných nástrojů ze zásobníku do skladu

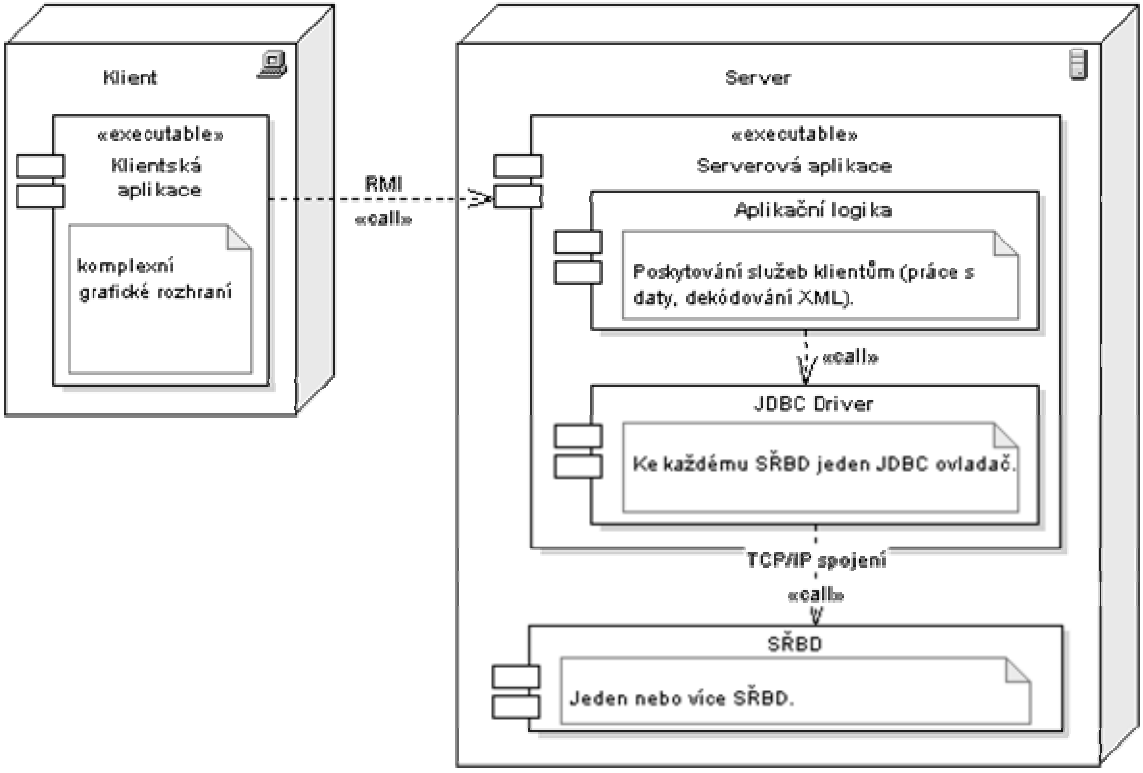
Rozebírání nástrojových sestav - z nástrojové sestavy, která se již nebude používat, se sestaví seznam nástrojů k rozebrání. Komponenty těchto nástrojů se po rozebrání vracejí do skladu

Správa nástrojů - zadávání a činnosti spojené s výdejem, příjmem a správou jednotlivých komponent na jednotlivých pracovištích (výdejnách) - činnosti spojené se sledováním spotřeby nástrojů, s jejich nákupem

Evidence - vytváření požadovaných evidenčních počítačových sestav, např. přehled stavu skladu, finanční přehled skladu, přehledy dostupných nástrojů

2.2 Celková struktura systému

Aplikace je rozdělena na dvě části - serverovou část a klientskou část. Třívrstvá architektura je přitom zachována, protože datová vrstva je tvořena jedním nebo i více SŘBD. Serverovou aplikací je využíván vždy jeden JDBC ovladač pro každý z použitých SŘBD jako univerzální rozhraní pro práci s datovou vrstvou. Současně serverová aplikace očekává volání klientů a vyřizuje jejich požadavky. Klientská aplikace provádí interakci s uživateli pomocí editorem definovaných formulářů. Použitím speciálních pluginů dokáže klientská aplikace pracovat i se specifickým strojním zařízením. Pro síťovou komunikaci klientských aplikací a serverové aplikace je použit mechanismus RMI viz. kap. 2.3.3. Celkovou strukturu systému ukazuje následující obrázek [3]:



Obr. 1 Celková struktura systému [3]

2.3 Použité technologie

Vývoj aplikace probíhal převážně na operačním systému MS Windows 2000, další část vývoje a testování pak probíhalo na operačním systému Suse Linux 7.1. Jako databázový server sloužil z počátku vývoje na platformě Windows zdroj dat ODBC propojený na databázový soubor MS Access 2000. V současné době na platformě Linux je využito databázového serveru PostgreSQL. Jako vývojového prostředí bylo využito prostředí Forte Java, v současné době prostředí Netbeans 3.6 s podporou Javy 1.4, či Netbeans 5.0 s Java 1.5. Popis jednotlivých technologií následuje.

2.3.1 Java

Java je objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems a představila 23. května 1995. Java je jedním z nejpoužívanějších programovacích jazyků na světě. Díky své přenositelnosti je používán pro programy, které mají pracovat na různých systémech počínaje čipovými kartami (platforma JavaCard), přes

mobilní telefony a různá zabudovaná zařízení (platforma Java ME), aplikace pro desktopové počítače (platforma Java SE) až po rozsáhlé distribuované systémy pracující na řadě spolupracujících počítačů rozprostřené po celém světě (platforma Java EE). Tyto technologie se jako celek nazývají platforma Java [6].

Vlastnosti Javy:

- *jednoduchý* - jeho syntaxe je zjednodušenou (a drobně upravenou) verzí syntaxe jazyka C a C++. Odpadla většina konstrukcí, které způsobovaly programátorům problémy a na druhou stranu přibyla řada užitečných rozšíření.
- *objektově orientovaný* - s výjimkou osmi primitivních datových typů jsou všechny ostatní datové typy objektové.
- *distribuovaný* - je navržen pro podporu aplikací v síti (podporuje různé úrovně síťového spojení, práce se vzdálenými soubory, umožňuje vytvářet distribuované klientské aplikace a servery).
- *interpretovaný* - místo skutečného strojového kódu se vytváří pouze tzv. mezikód (bajtkód). Tento formát je nezávislý na architektuře počítače nebo zařízení. Program pak může pracovat na libovolném počítači nebo zařízení, který má k dispozici interpret Javy, tzv. virtuální stroj Javy - Java Virtual Machine (JVM).

V pozdějších verzích Javy nebyl mezikód přímo interpretován, ale před prvním svým provedením dynamicky zkompilovány do strojového kódu daného počítače (tzv. just in time compilation - JIT). Tato vlastnost zásadním způsobem zrychlila provádění programů v Javě, ale výrazně zpomalila start programů.

V současnosti se převážně používají technologie zvané HotSpot compiler, které mezikód zpočátku interpretují a na základě statistik získaných z této interpretace později provedou překlad často používaných částí do strojového kódu včetně dalších dynamických optimalizací (jako je např. inlining krátkých metod atp.).

- *robustní* - je určen pro psaní vysoce spolehlivého softwaru – z tohoto důvodu neumožňuje některé programátorské konstrukce, které bývají častou příčinou chyb (např. správa paměti, příkaz „goto“, používání ukazatelů).

Používá tzv. silnou typovou kontrolu – veškeré používané proměnné musí mít definovaný svůj datový typ.

Správa paměti je realizována pomocí automatického Garbage collectoru který automaticky vyhledává již nepoužívané části paměti a uvolňuje je pro další použití. To bylo v prvních verzích opět příčinou pomalejšího běhu programů. V posledních verzích běhových prostředí je díky novým algoritmům pro garbage collection a tzv. generační správě paměti (paměť je rozdělena na více částí, v každé se používá jiný algoritmus pro garbage collection a objekty jsou mezi těmito částmi přesunovány podle délky svého života) tento problém ze značné části eliminován.

- *bezpečný* - má vlastnosti, které chrání počítač v síťovém prostředí, na kterém je program zpracováván, před nebezpečnými operacemi nebo napadením vlastního operačního systému nepřátelským kódem.
- *nezávislý na architektuře* - vytvořená aplikace běží na libovolném operačním systému nebo libovolné architektuře. Ke spuštění programu je potřeba pouze to, aby byl na dané platformě instalován správný virtuální stroj. Podle konkrétní platformy se může přizpůsobit vzhled a chování aplikace.
- *přenositelný* - vedle zmíněné nezávislosti na architektuře je jazyk nezávislý i co se týká vlastností základních datových typů (je například explicitně určena vlastnost a velikost každého z primitivních datových typů). Přenositelností se však myslí pouze přenášení v rámci jedné platformy Javy (např. J2SE). Při přenášení mezi platformami Javy je třeba dát pozor na to, že platforma určená pro jednodušší zařízení nemusí podporovat všechny funkce dostupné na platformě pro složitější zařízení a kromě toho může definovat některé vlastní třídy doplňující nějakou speciální funkčnost nebo nahrazující třídy vyšší platformy, které jsou pro nižší platformu příliš komplikované.
- *výkonný* - přestože se jedná o jazyk interpretovaný, není ztráta výkonu významná, neboť překladače pracují v režimu „právě včas“ a do strojového kódu se překládá jen ten kód, který je opravdu zapotřebí.
- *víceúlohový* - podporuje zpracování vícevláknových aplikací

- *dynamický* - Java byla navržena pro nasazení ve vyvíjejícím se prostředí. Knihovna může být dynamicky za chodu rozšiřována o nové třídy a funkce, a to jak z externích zdrojů, tak vlastním programem.
- *elegantní* - velice pěkně se v něm pracuje, je snadno čitelný (např. i pro publikaci algoritmů), přímo vyžaduje ošetření výjimek a typovou kontrolu

2.3.2 Xml

XML (eXtensible Markup Language, česky rozšiřitelný značkovací jazyk) je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat. Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Jazyk umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí. Prezentace dokumentu (vzhled) se potom definuje připojeným stylem. Další možností je pomocí různých stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury XML. Původní jazyk pro publikování HTML již přestal vyhovovat především pro svou složitost, která vznikla jeho postupným (a svévolným) rozšiřováním. Jazyk XML nemá žádné předdefinované značky (tagy, názvy jednotlivých elementů) a také jeho syntaxe je podstatně přísnější než HTML[7].

V aplikaci NAHOS hraje značkovací jazyk XML velmi významnou roli. Je využit k uložení struktury uživatelských formulářů aplikace do databáze. Odtud jsou pomocí zpětné analýzy zdrojového kódu XML formuláře systémem načteny a zobrazeny do vizuální podoby. Naopak v editorové části aplikace NAHOS je vizuálním a uživatelsky příjemným způsobem XML struktura formuláře vytvořena a uložena.

2.3.3 Síťová komunikace RMI

Vzdálené volání metod poskytuje jednoduchý, ale výkonný mechanismus pro tvorbu aplikací typu klient/server. Technologie byla vyvinuta firmou Sun Microsystems a je používána v řadě běžných aplikací. Server poskytuje své služby formou metod, které může klient volat tak, že pošle serveru požadavek RMI spolu s jeho parametry . Server provede příslušnou metodu a pošle zpět klientovi případnou návratovou hodnotu. Výhodou

RMI je jeho jednoduchost a široká dostupnost v rámci Javy. Nevýhodou je omezená flexibilita, pramenící právě z omezení na implementaci v jazyce Java [1]

2.4 Základní prvek – Formulář

Formulář je základním (nadřazeným) ovládacím prvkem v programu, obsahuje formulářové prvky (tlačítka, tabulky, popisky atd.). Každý formulář je popsán v tabulce forms XML dokumentem. V současné verzi aplikace jsou implementovány základní prvky, které umožňují všechny běžné způsoby zobrazení a všechny běžné operace. Jsou to: tabulka, malé textové pole, velké textové pole, rozbalovací seznam, tlačítko, textový popisek, rámeček s popiskem, zatrhávací pole a listovací lišta, a dále pak rozšiřující prvky přidané Ing. Petrem Svobodou jako je prvek RadioGroup (skupina radiobuttonů), prvek Template a prvek menu[4]. Kromě formulářových prvků může mít formulář nadefinované vlastní proměnné, datové relace (queries) atd. Formuláře jsou hlavním nástrojem IS NAHOS sloužící ke správě a řízení jeho dat.

2.5 Rozšíření systému

Dalším využíváním IS NAHOS v praxi vznikl požadavek tiskových formulářů a tisku vícestránkových sestav. Aby bylo také dále možno zjednodušit práci se systémem, jelikož u formulářů s větším počtem tlačítek může docházet k horší orientaci uživatele, formuláře se stávají nepřehlednými, je potřeba realizovat programovou část zpracování prvku menu.

Oba výše zmíněné požadavky jsem zpracoval a popsal v následujících částech této diplomové práce.

II. PRAKTICKÁ ČÁST

3 TISKOVÉ MODULY PRO TISK FORMULÁŘŮ A VÍCESTRÁNKOVÝCH SESTAV

V původním stavu systému byl tisk umožněn pouze volbou tlačítka „Tisk“ které se zobrazovalo v Popupmenu jako reakce na stisk pravého tlačítka myši v tabulce formuláře s daty. Při tomto tisku nelze nijak ovlivnit tiskové vlastnosti formuláře, ani jednotlivých prvků popř. jejich vlastností. Formulář je vytisknut tak, jak je zobrazen uživateli na obrazovce. Z důvodu potřeby tisku vícestránkových sestav vznikl požadavek na úpravu těchto tiskových možností. Pro tyto potřeby byly navrženy a přidány nové tiskové tagy do struktur xml a vznikly nové tiskové moduly pro zpracování formulářů definovaných pro tisk. Zpracování i návrh jsem se snažil vytvořit co nejobecněji pro volbu libovolných formulářů. Výstupem tiskových modulů je tiskový formulář ve formátu dokumentu pdf. Tento dokument lze archivovat pro pozdější účely kontroly, či s ním libovolně dále pracovat, poslat jako přílohu, vytisknout atd. Ke zpracování dat, vytvoření dokumentů pdf bylo využito knihoven iText viz. kap. 3.3.

3.1 Tiskové formuláře

Do současné chvíle byly vytvořeno pět, tiskových formulářů a k nim související formuláře předvoleb. Formálně lze tyto formuláře rozdělit na dvě skupiny:

1. přehledy o stavu a pohybech nástrojů uvnitř podniku

   $tisk_soupis

   $tisk_regleta

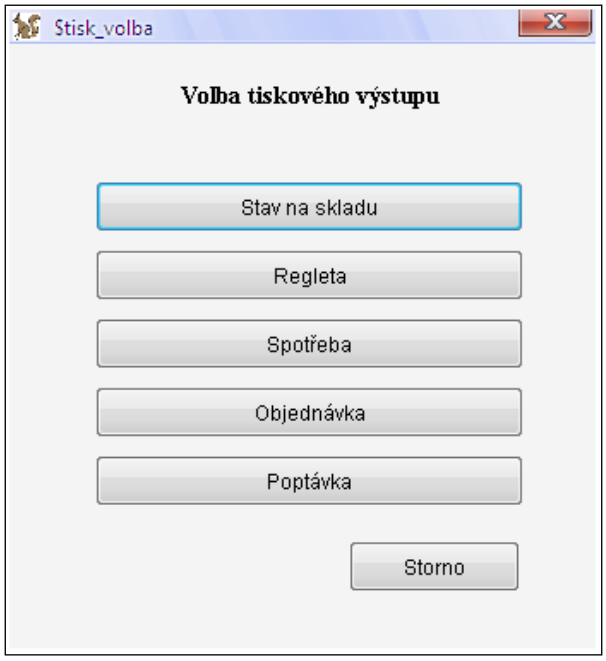
   $tisk_spotreba

2. formuláře související s nákupem komponent

   $tisk_objednavka

   $tisk_poptavka

Pro začlenění těchto formulářů do systému byl vytvořen jednoduchý formulář *$tisk_volba* s nabídkou jejich volby.



Obr. 2 Formulář $tisk_volba

Formulář obsahuje pouze tlačítka pro volbu formulářů, a tlačítko zavření formuláře.

Formuláře přehledů o stavu a pohybech nástrojů uvnitř podniku

3.1.1 $tisk_soupis

Formulář slouží k zobrazení a posléze tisku stavu nástrojů ve skladu, k nastavení parametrů slouží formulář *$tisk_volba_stavSkladu.*

Obr. 3 Formulář $tisk_volba_stavSkladu

Ve formuláři předvoleb volíme autora soupisu a jméno osoby která bude následně provádět kontrolu. Dále typ skladového místa a skladové místo samotné. Jako poslední volíme jméno a místo uložení dokumentu. Po výběru jednotlivých předvoleb a stisku tlačítka „Náhled tisku“ se zobrazí náhled samotného tiskové formuláře. Dokument ve formátu pdf je generován po stisku tlačítka PDF.

3.1.2 $tisk_regleta

Formulář slouží k zobrazení a posléze tisku sestav nástrojových komponent, které jsou „v pohybu“, tj. komponenty, které do dané výdejny přicházejí, resp.z ní odcházejí (tyto sestavy se nazývají reglety). Zajímavé jsou také sestavy nástrojových komponent, u nichž žádný pohyb nenastal. Rozlišujeme tři typy reglet.

- Komponenty bez pohybu.
- Komponenty přicházející/odcházející do/z výdejny převodem.
- Komponenty přicházející z objednávky.

K výběru typu reglety a nastavení jejích parametrů slouží formulář *$tisk_volba_regleta*

Obr. 4 Formulář $tisk_volba_regleta

Ve formuláři předvoleb jako první volíme období, za které se regleta sestavuje, jméno autora a osoby která bude následně provádět kontrolu. Dále typ skladového místa, skladové místo samotné a typ reglety. Jako poslední volíme jméno a místo uložení dokumentu. Po výběru jednotlivých předvoleb a stisku tlačítka „Náhled tisku“ se zobrazí

náhled samotného tiskové formuláře. Dokument ve formátu pdf je generován po stisku tlačítka PDF.

3.1.3 $tisk_spotreba

Formulář slouží k zobrazení a posléze tisku spotřebovaných nástrojů za určité období, toto období je voleno ve formuláři předvoleb *$tisk_volba_spotreba.*

Obr. 5 Formulář $tisk_volba_spotreba

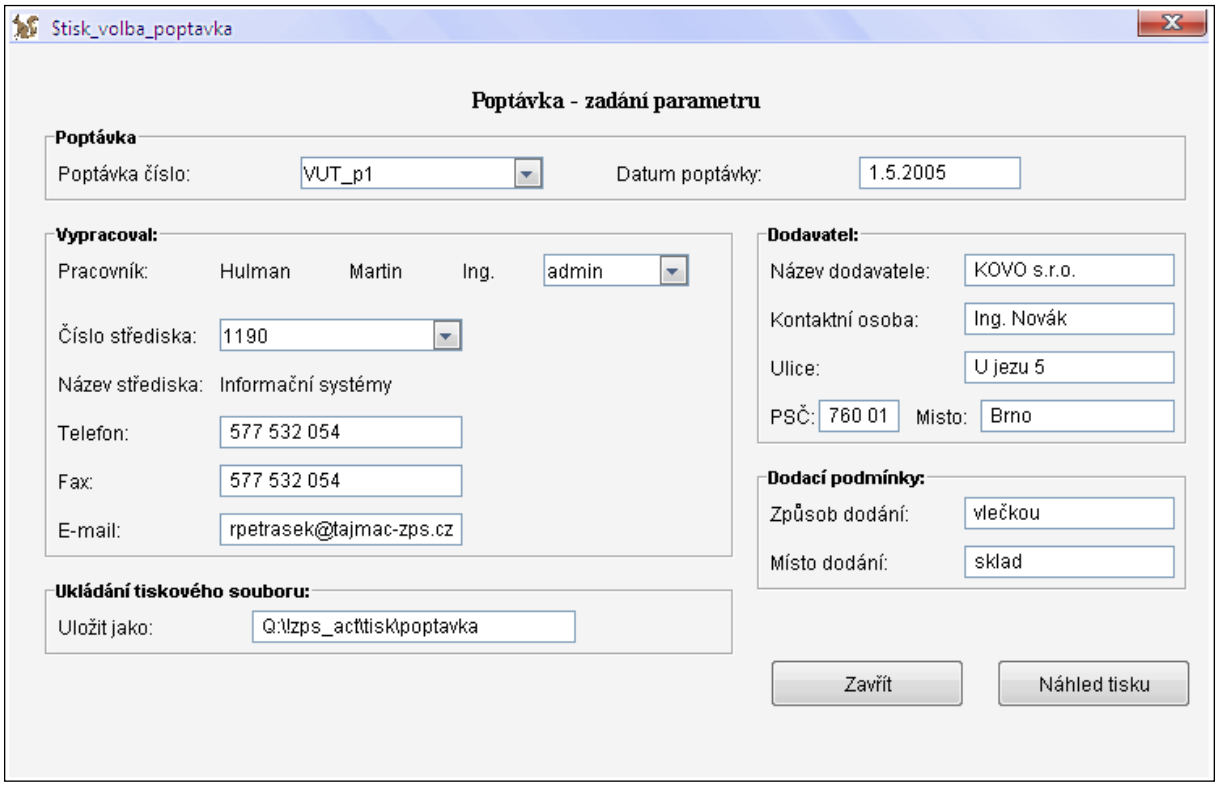
Jako druhý parametr volíme jméno a místo uložení dokumentu. Po výběru jednotlivých předvoleb a stisku tlačítka „Náhled tisku“ se zobrazí náhled samotného tiskové formuláře. Dokument ve formátu pdf je generován po stisku tlačítka PDF.

Formuláře související s nákupem komponent

V praxi je obvyklým postupem poptávka – nabídka –objednávka), ve stejném pořadí jsou vytvářeny i data formulářů poptávek a objednávek systému NAHOS.

3.1.4 $tisk_poptavka

Formulář slouží k zobrazení a posléze tisku údajů poptávek. Nastavení dodavatele, způsobu, místa dodání a ostatních podrobností je součástí formuláře $tisk_volba_poptávka.

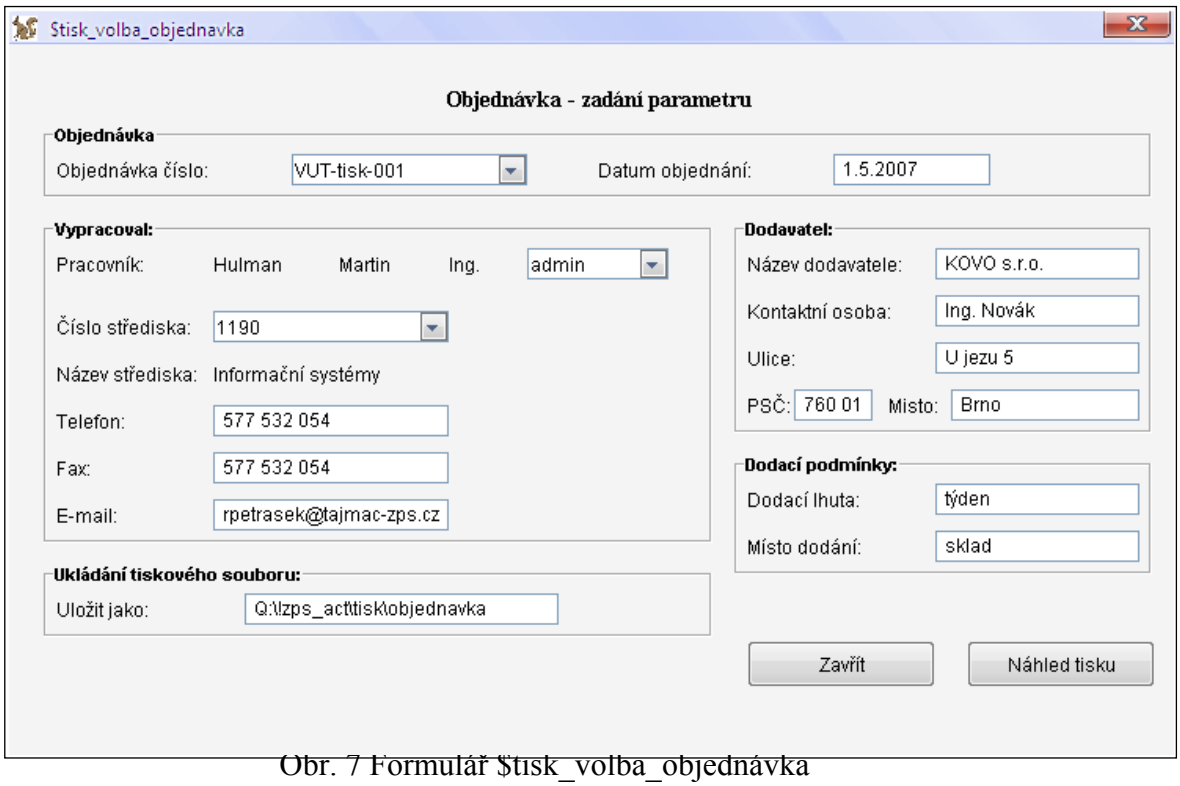


Obr. 6 Formulář $tisk_volba_poptávka

Formulář je svou strukturou obdobou formuláře objednávky, má však definován pouze jednu tabulku dat. Stejně tak jako u objednávky volíme pracovníka tvořícího poptávku, dodavatele a na závěr cíl a jméno výsledného dokumentu. K zobrazení náhledu dojde po stisku tlačítka „Náhled tisku“. Dokument ve formátu Pdf je generován po stisku tlačítka PDF.

3.1.5 $tisk_objednavka

Formulář slouží k zobrazení a posléze tisku objednávek nástrojů. Nastavení dodavatele, termínů a ostatních podrobností je součástí formuláře *$tisk_volba_objednavka.*



Obr. 7 Formulář $tisk_volba_objednávka

Formulář objednávky je v systému nejsložitější, obsahuje pět předdefinovaných tabulek, které se plní v závislosti na datech uživatele. Z těchto pěti nabídek jsou čtyři tabulky hodnot podle nabídek a jedna bez nabídek. Po volbě čísla objednávky nastavuje osobu která objednávku vytváří, kontakty na ni, dodavatele a v neposlední řadě termíny a místo dodání. Stejně jako u ostatních formulářů je posledním parametrem cíl a místo uložení výsledného dokumentu. K zobrazení náhledu samotného formuláře dochází po stisku tlačítka „Náhled tisku“. Dokument ve formátu pdf je generován po stisku tlačítka PDF.

3.2 Požadavky tiskových formulářů

Hlavními požadavky na tiskové formuláře byli a jsou:

1. Tisk vícestránkových sestav
2. Obsah jednotlivých stránek bude uživatelsky definovatelný (umístění prvků v záhlaví, zápatí, výskyt prvků na jednotlivých stránkách,...)
3. Jednotlivé stránky u vícestránkových sestav budou umožňovat zobrazení postupných součtů dat za stránku

4. Pro různé druhy tisku dat, bude umožněna volba velikosti výstupu (formát stránek, orientace)
5. Formuláře mohou obsahovat více než jednu tabulku
6. Tisk základních prvků - label, table

Postupem času a dle jednotlivých připomínek pracovníků firmy Tajmac ZPS,a.s. zejména Ing. Janáta, vznikly posléze i požadavky další, především pak na grafický vzhled dokumentů pdf a to:

7. Volba ohraničení tabulek (je / není) - jelikož výstupy sestav i reglet, mohou mít až stovky stránek, vznikl tento požadavek z důvodu šetření toneru, popř. inkoustu tiskáren
8. Uživatelem definovaný font textů (prvky label, table) - u tabulek odlišení fontu záhlaví a datové části
9. Tisk rámečku (prvek border)
10. Tisk obrázku (prvek iconLabel)

Jelikož je systém dále vyvíjen a inovován, jsou jednotlivé požadavky i v současné chvíli postupně řešeny a zapracovávány do systému.

3.3 iText

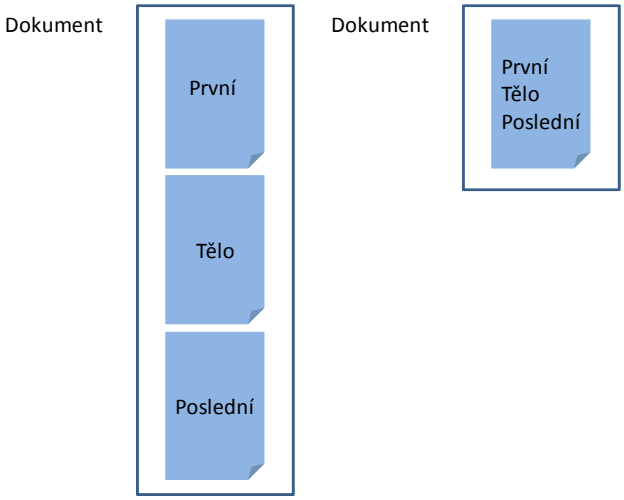
Je knihovna umožňující generování dokumentů ve formátu pdf. Jednotlivé třídy knihovny iText jsou velmi užitečným nástrojem pro ty, kteří potřebují generovat jednoduché, na platformě nezávislé dokumenty obsahující prvky jako text, lists, tabulky a obrázky. Speciálně pak v kombinaci s technologií Java(TM) technology-based Servlets. [10].

Knihovny vyžadují JDK 1.4. a jsou volně šířitelné pod multilicencí MPL a LGPL.

Ve své diplomové práci jsem využil knihoven iText k tvorbě výstupu tiskových formulářů ve formě dokumentu pdf. Podrobnější popis použitých metod, resp. jejich metod je uveden v příloze PI (Komentáře ze zdrojových kodů).

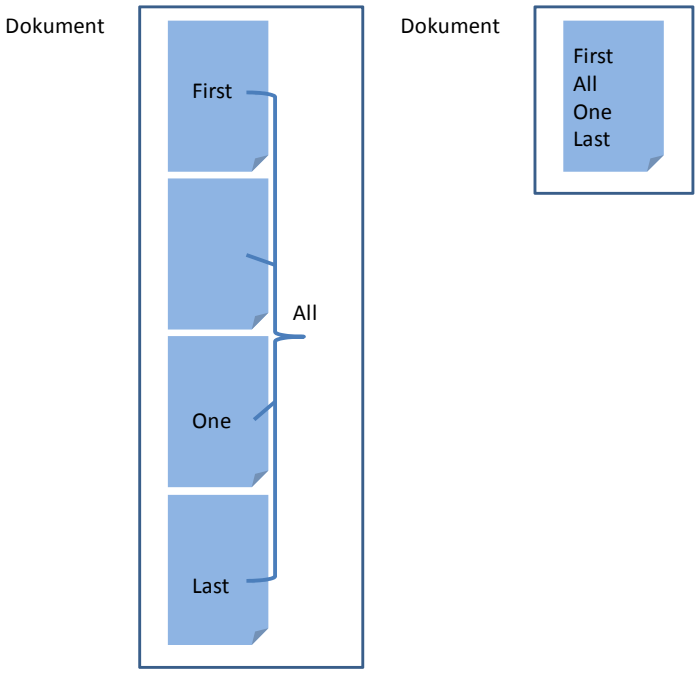
3.4 Forma zpracování, hlavní myšlenka, jádro

Hlavní myšlenkou návrhu tiskových tagů a jejich pozdějšího zpracování, je rozdělení dokumentu na určité typově rozdílné stránky. Jednotlivé typy stránek byly odvozeny ze struktury dokumentu. Tedy každý dokument může obsahovat, a také obsahuje stránku: první, poslední a stránky mezi nimi. Pokud je tímto dokumentem jedna jediná stránka, považujeme ji za první, tělo, i poslední viz. Obr. 8.



Obr. 8 Stránky dokumentu

Od této myšlenky odvozujeme možné umístění jednotlivých prvků (labelů, tabulek, obrázků,…) v dokumentu, následně tedy typy stránek na kterých se budou nacházet.



Obr. 9 Typy stránek dokumentu

Pro další zpracování dochází k menší formální úpravě, kdy tělem dokumentu rozumíme zároveň stránku první i poslední a vzniká také označení, hodnota parametru pro „jednu určitou stránku“ viz. Obr. 9. Stále platí, pokud existuje jedna jediná stránka, splňuje všechny typy stránek.

Jednotlivé typy stránek jsou hodnoty parametru `type` tiskového tagu `page`. Následuje jejich podrobnější popis:

1. first - prvek je umístěn jen na první stránce dokumentu
2. all - prvek se nachází na všech stránkách dokumentu
3. one - prvek se nachází jen na jedné konkrétní stránce dokumentu
4. last - prvek se nachází na poslední stránce dokumentu

Zápis těchto vlastností v souboru xml je následující:

```
<print>
      <page type="first">
      …
      </page>
</print>
```

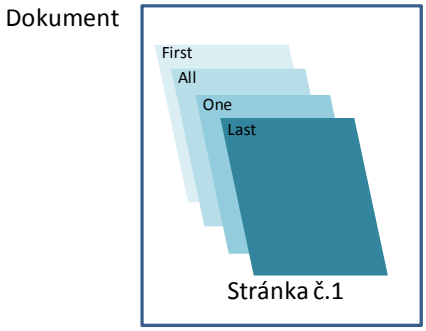
Pozn.: číslo stránky, umístění prvku u hodnoty `one` může být určeno více způsoby, v současné verzi se určuje podle tagu `location` a jeho parametrů `<location type="relative" relatedID="$tbl_objNab_1"…>` Je tedy odvozeno od umístění prvku, na který se odkazujeme. Tohoto je využito např. u umístění nadpisů tabulek ve formuláři objednávky.

Zápis těchto vlastností v souboru xml je následující:

```
<print>
      <page type="one">
          …
          <location type="relative" relatedID="$tbl_objNab_1"
                    unit="pixel" yValue="15"/>…
      </page>
</print>
```

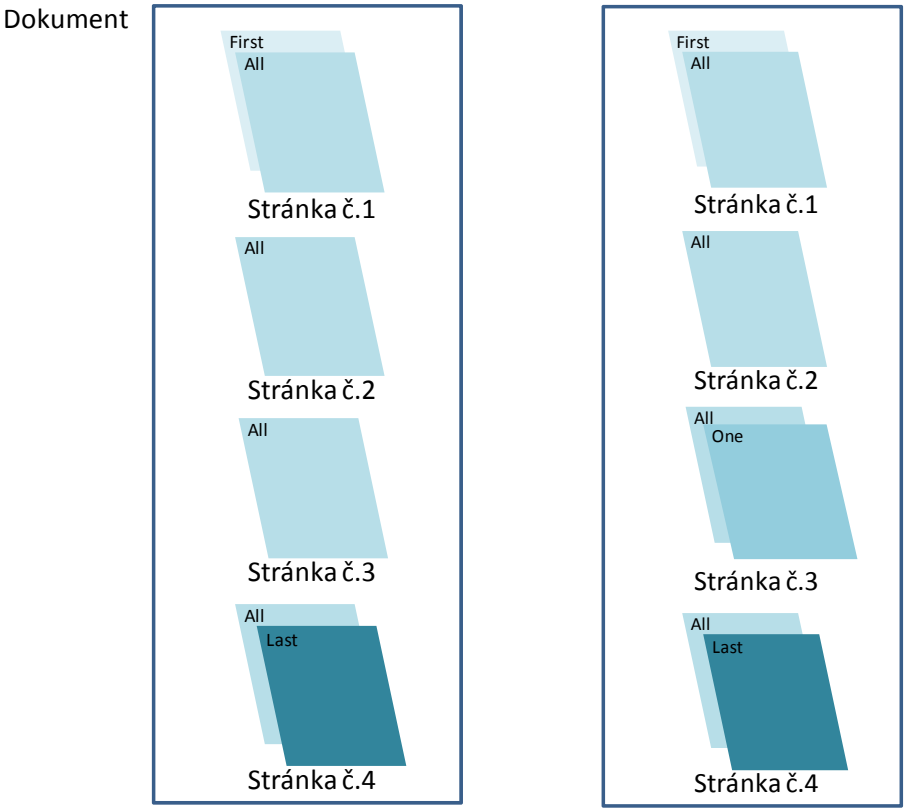
Při samotné tvorbě dokumentu je tedy každá jednotlivá stránka jakousi kombinací jednotlivých typů stránek a jejich prvků (vektorů dat) podle umístění v dokumentu viz. Obr. 10 a Obr.11.

Dokument s jednou stránkou:



Obr. 10 Dokument s jednou stránku

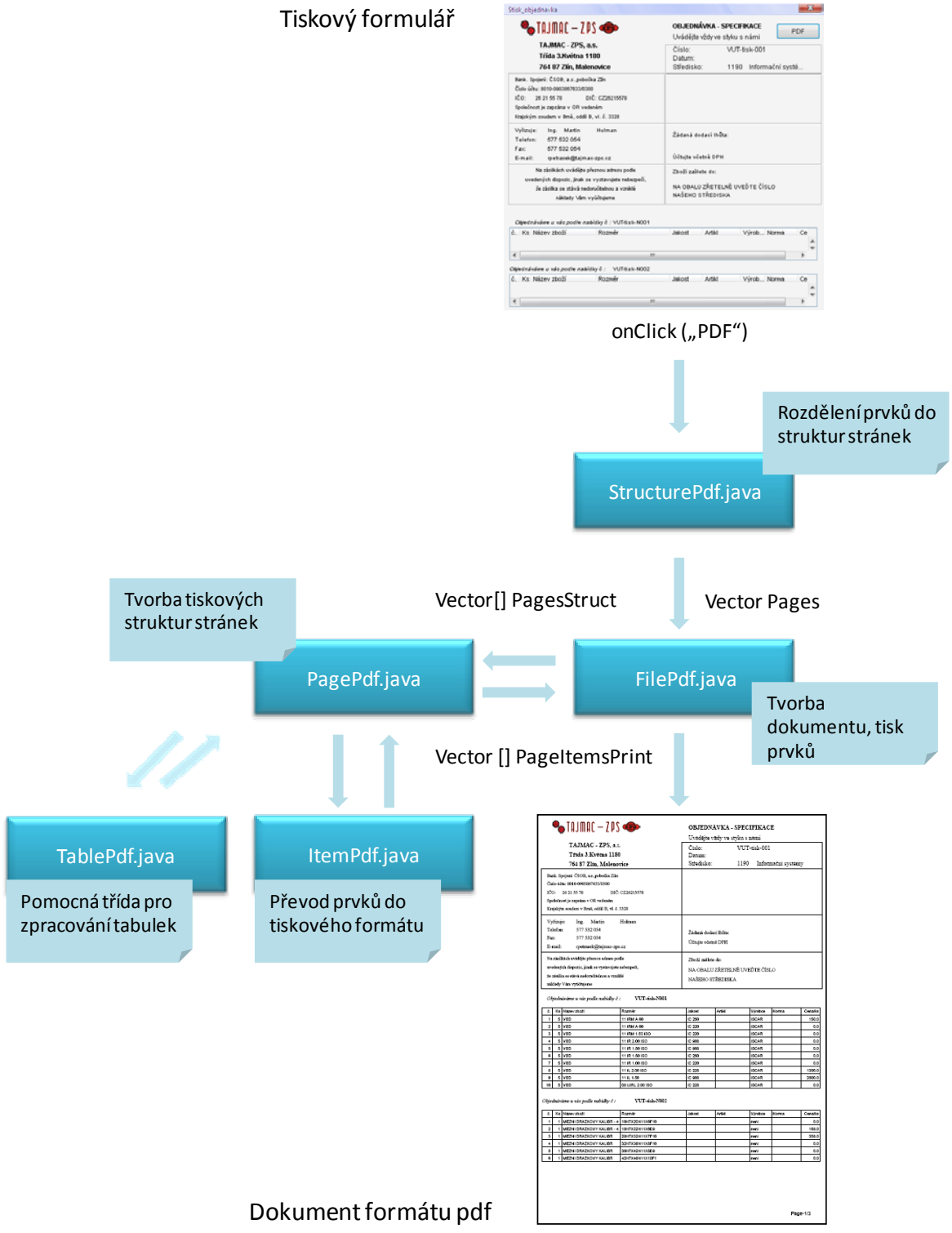
Dokument se čtyřmi stránkami:



Obr. 11 Dokument se čtyřmi stránkami

Pozn.: pro každý prvek umístěný dle hodnoty `one` může být číslo stránky jiné.

3.5 Programové zpracování tiskových formulářů



Obr. 12 Programové schéma tiskových modulů

Všechny třídy jsou součástí `package xerus.client.forms.pdf`. Stručnější popis jednotlivých z nich, jejich metod a proměnných následuje.

3.5.1 StructurePdf.java

Tato třída má za úkol projít formulář a vytvořit jednotlivé struktury stránek, resp. datové struktury s prvky na nich umístěnými. Vstupem je struktura formuláře, jako instance třídy `FormStruct balíčku client.structures.form`. Výstupem vektor polí Pages, kde jednotlivé pole jsou data odpovídajících typů stránek. Každé pole má v sobě mimo formulářových prvků na nultém místě připojeny informace o vytvářeném dokumentu, resp. o jeho stránkách - formát, orientace…viz následující popis a schéma:

Datová struktura

Tab. 1 Datová struktura tiskových formulářů - Vector Pages

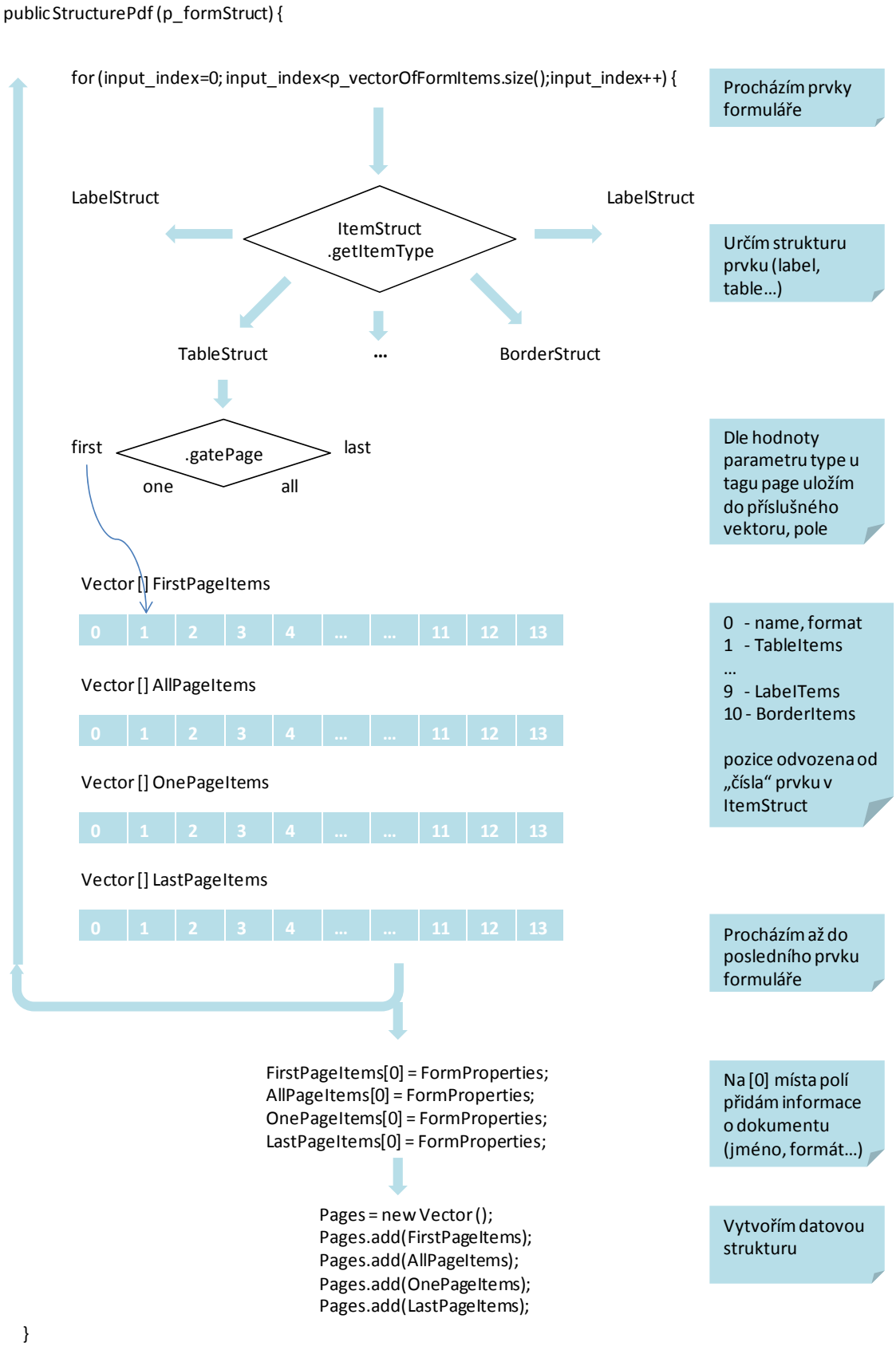
Vector Pages

Vector [] FirstPageItems	Vector [] AllPageItems	Vector [] OnePageItems	Vector [] LastPageItems

Vector [] First PageITems

0	1	2	3	…	…	13
FormProperties	TableITems					ImageItems

0	FormProperties – jméno dokumentu, formát stránek (size, formát…)
1	TableITems – vektor tabulek
2	null
3	null
4	null
5	null
6	null
7	null
8	null
9	LabelItems – vektor labelů
10	BorderItems – vektor rámečků
11	null
12	null
13	ImageItems – vektor obrázků



Obr. 13 Schéma vytvoření datové struktury tiskového formuláře

Struktura polí byla odvozena v souvislosti se třídou ItemStruct. Každý zpracovávaný prvek formuláře má své jedinečné číslo kterým je identifikovatelný:

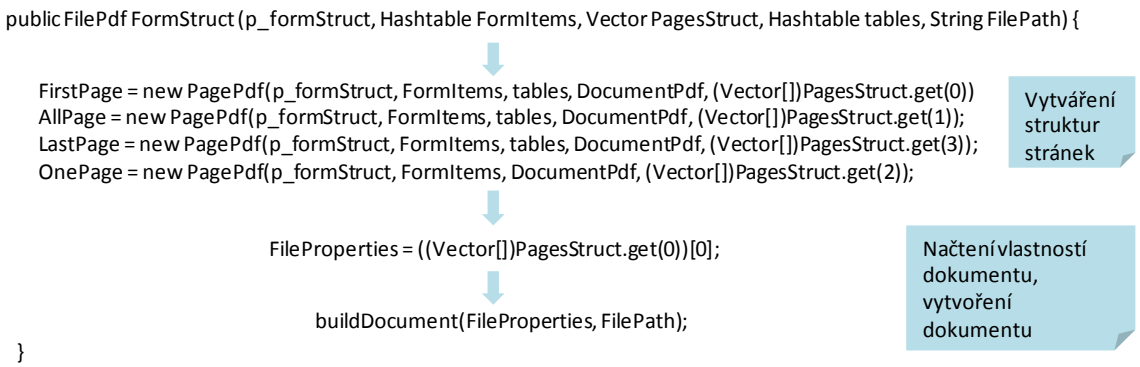
Tab. 2 Tabulka aktivních prvků třídy ItemStruct

```
1  public final static int TABLE = 1;
2  public final static int BUTTON = 2;
3  public final static int TEXTINPUT = 3;
4  public final static int TEXTAREA = 4;
5  public final static int COMBOBOX = 5;
6  public final static int LISTER = 6;
7  public final static int CHECKBOX = 7;
8  public final static int EMBEDDEDFORM = 8;
9  public final static int LABEL = 9;
10 public final static int BORDER = 10;
11 public final static int BUTTONGROUP = 11;
12 public final static int RADIOBUTTON = 12;
13 public final static int ICONLABEL = 13;
// typy aktivnich prvku
```

Z tohoto číslování byly tedy odvozeny pozice prvků v tiskových polích pro větší přehlednost a orientaci. Stejné pozice jsou dodrženy i v tiskových polích se souřadnicemi a daty určenými pro tisk, v instancích odvozených ze třídy PagePdf. Datová struktura vytvořená instancí třídy StructurePdf je předána ke zpracování třídě FilePdf která má za úkol vytvoření dokumentu pdf a jeho uložení na disk.

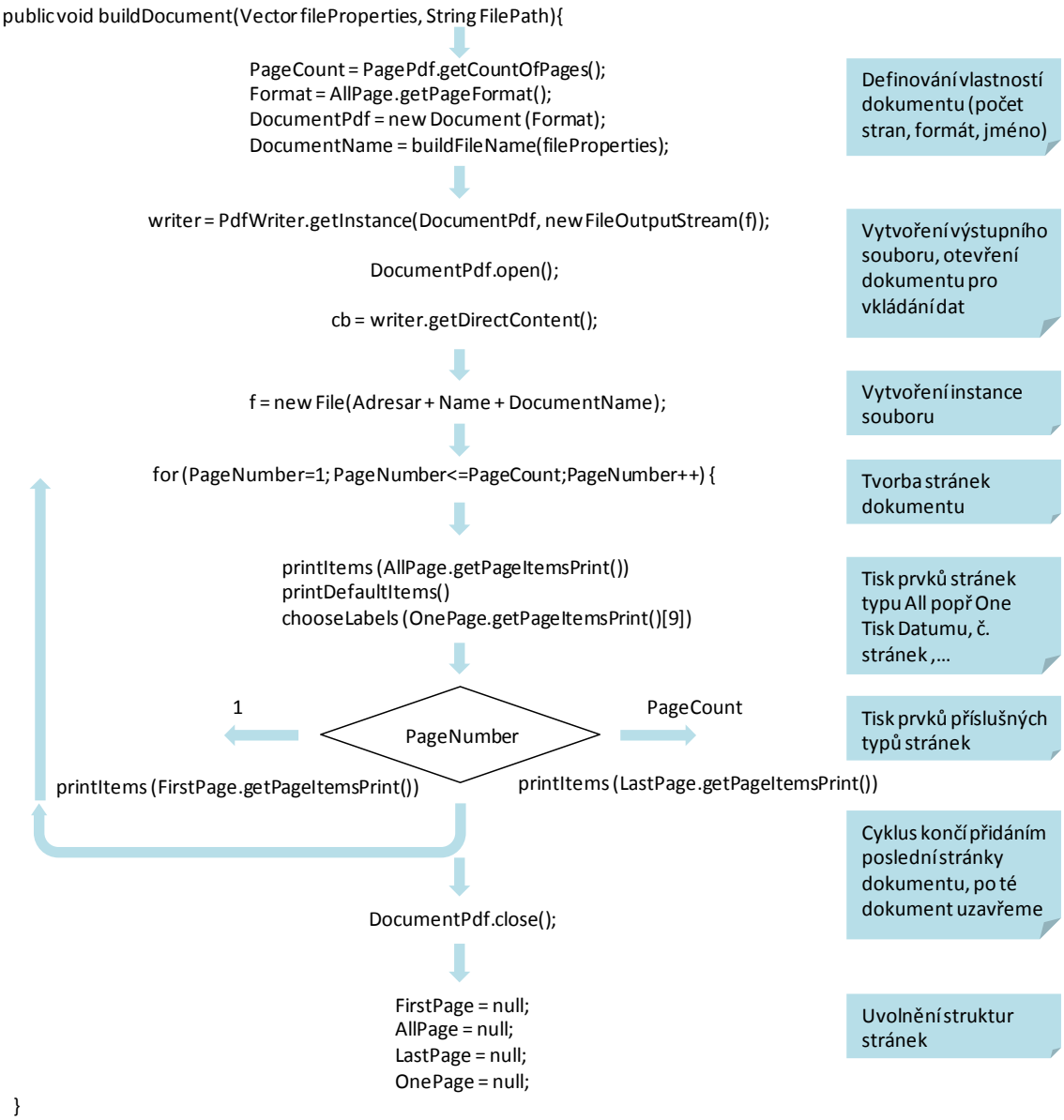
3.5.2 FilePdf.java

Vstupem této třídy je tedy datová struktura tiskového formuláře, tabulka aktivních prvků formuláře, hashtable tabulek formuláře a parametr s cestou kam uložit vytvořený dokument ve formátu pdf, který je výstupem třídy. Třída je jádrem tiskových modulů, v hlavním cyklu jsou vytvářeny jednotlivé stránky dokumentu, jejichž struktury jsou vytvořeny před samotným začátkem jako instance třídy PagePdf. Tato třída má na starost převod dat jednotlivých prvků do tiskového formátu, má za úkol vytvořit jejich souřadnice a ty pak zpět předat v co nejjednodušší podobě k výslednému zpracování metodami třídy FilePdf. Konstruktor třídy FilePdfa:



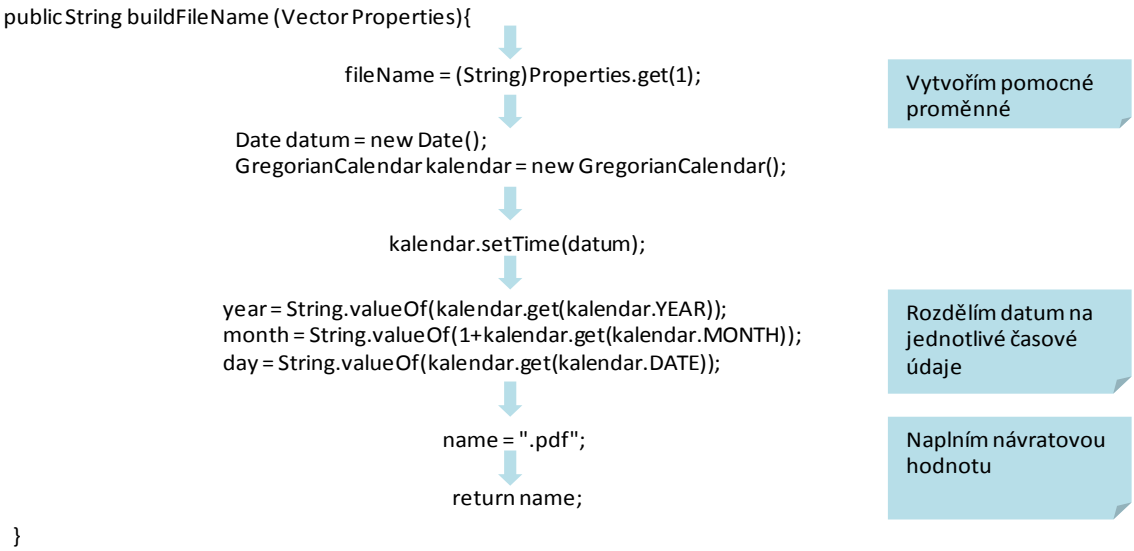
Obr. 14 Konstruktor třídy FilePdf.java

Po vytvoření tiskových struktur voláme metodu buildDocument která již vytvoří konkrétní dokument, viz následující schéma:



Obr. 15 Metoda buildDocument (Vector fileProperties, String FilePath)

V prvních krocích načteme počet stránek, formát a pomocí těchto údajů vytvoříme prázdný dokument. Nastavíme jméno výstupního souboru a cestu adresáři kde má být výsledný dokument uložen pro archivaci.

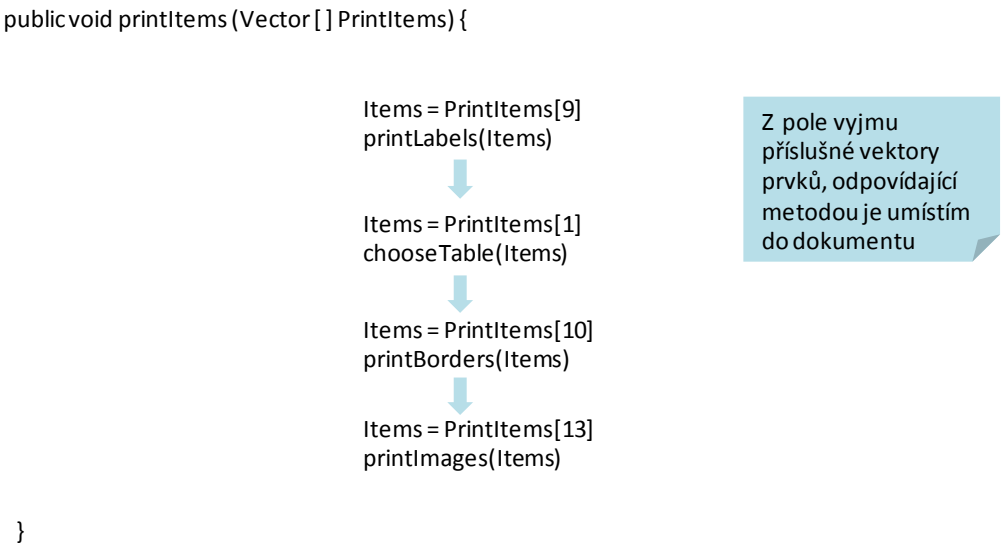


Obr. 16 Metoda String buildFileName (Vector Properties)

Metoda vrací jako hodnotu řetězec se jménem dokumentu. Jméno dokumentu může být vytvořeno kombinací časových údajů, popř. jména uloženého ve struktuře xml nezávisle na uživateli.

```
<form>
...
<print>
      ...
      <fileName>Obj</fileName>
      ...
</print>
</form>
```

V současné verzi vrací metoda pouze koncovku souboru (.pdf) a jméno samotné si volí uživatel před náhledem tisku ve formuláři předvoleb, viz. kap. 3.1. Po tomto kroku nastavíme určené jméno souboru a cestu k němu. Připojíme odkaz na soubor a otevřeme dokument pro vkládání dat. Při vkládání dat procházíme jednotlivé stránky dokumentu a podle čísla stránky vybíráme struktury stránek a jejich tiskových polí pro vložení, tyto data předáme metodě printItems, která data rozdělý ke zpracování již konkrétním metodám iTextu k tisku (umístění) prvků na stránky dokumentu.



Obr. 17 Metoda printItems (Vector [] PrintItems)

Jednotlivé metody mají za úkol převzít vektory dat a za použití metod knihoven iText je umístit na stránky dokumentu.

- *printLabels (Items)*

  metoda předává data labelů k umístění, data jsou vektorem instancí třídy ItemPdf, datovou částí je Vektor s textem k zobrazení a uživatelem definovaným fontem, před samotným tiskem je vytvořen nejdříve font pro zobrazení a po té jsou data předána k vykreslení

  programová část umístění labelu:

```
cb.beginText();
cb.setTextMatrix((float)pointX,(float)pointY);
cb.showText(text);
cb.endText();
```

- *chooseTable (Items)*

  z předaného vektoru dat je podle čísla stránky vybrána konkrétní tabulka s daty, tabulka je po té předána metodě PrintTable která tabulku umístí podle příslušných souřadnic na stránce dokumentu, samostatná část věnovaná tabulkám je popsána v kapitole 3.6.1.2.

  programová část umístění tabulky:

```
tablepdf.writeSelectedRows(0, -1, x, y, cb);
```

- *printBorders (Items)*

  metoda připravuje rámečky k umístění na stránku, před samotným umístěním je ještě nastavena tloušťka, popř. zvolená barva, tloušťka je zadávána parametrem ve struktuře souboru xml

  programová část umístění rámečku:

```
cb.setLineWidth(0.5f);
cb.setRGBColorStrokeF(0f, 0f, 0f);
cb.rectangle(pointX, pointY, width, -height);
cb.stroke();
```

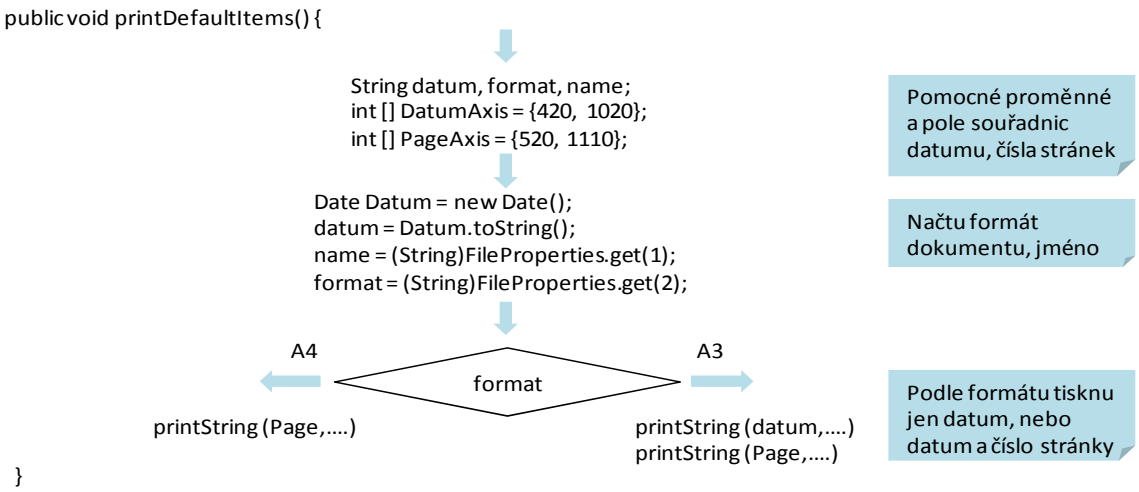
- *printImages*

  metoda připravuje obrázky, datová část obsahuje tříprvkový vektor který má ve své první části data obrázku, a na zbylých jeho šířku a výšku

  programová část umístění obrázku:

```
cb.addImage(image,width,0,0,height,pointX,pointY);
```

Mimo prvků vyčítaných ze struktur stránek jsou na stránky umístěny ještě hodnoty jako datum a čísla stránek. Formát, tvary těchto dat nejsou ovlivněny uživatelem ale jsou definovány v programu. Popis metod umístění následuje.



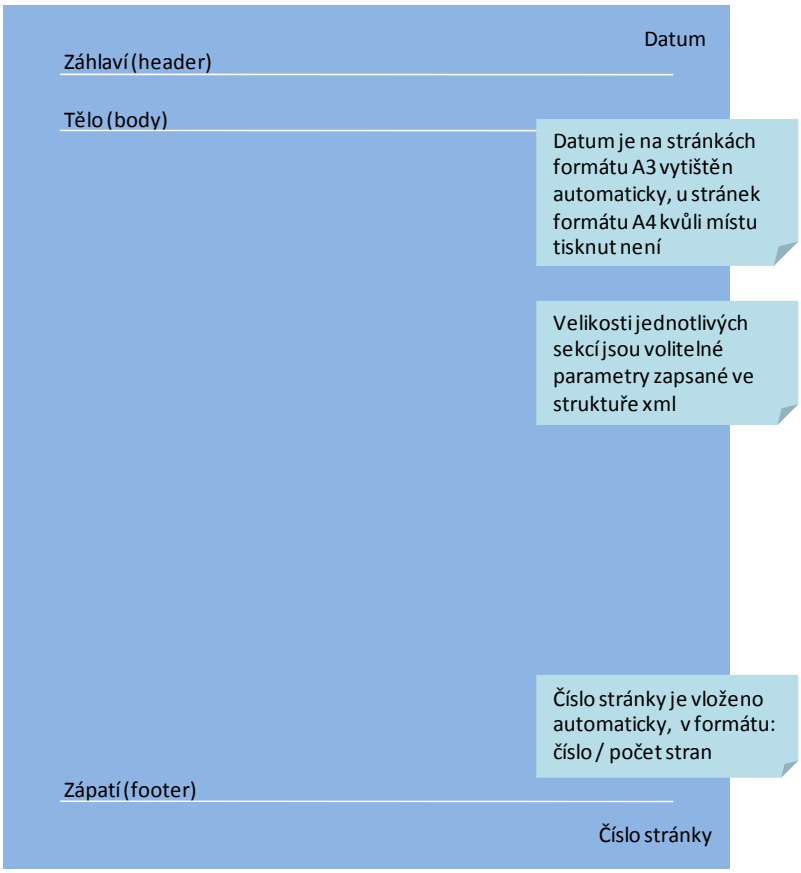
Obr. 18 Metoda printDefaultItems()

Pokud je nastaven formát stránky A4, datum tisknut není z důvodů úspory místa v hlavičce dokumentů. Čísla stránek jsou ve formátu: číslo stránky / počet stran.

3.6 Vytváření struktur stránek

Tak jako byly jednotlivé typy stránek odvozeny od struktury dokumentu, je stejně odvozena i struktura stránky a z ní odvozeny další tiskové tagy. Stránku rozdělíme na jednotlivé sekce:

Struktura stránky



Obr. 19 Struktura stránky

Jednotlivé sekce definují další možnost, vlastnost umístění prvku již na jednotlivé stránce. Tato vlastnost je zapsána jako hodnota parametru `type` tiskového tagu `pageSection`. Hodnoty parametru tedy jsou:

1. header - prvek je umístěn v záhlaví stránky
2. body - prvek je umístěn v těle stránky
3. footer - prvek patří do zápatí

Zápis těchto vlastností v souboru xml je následující:

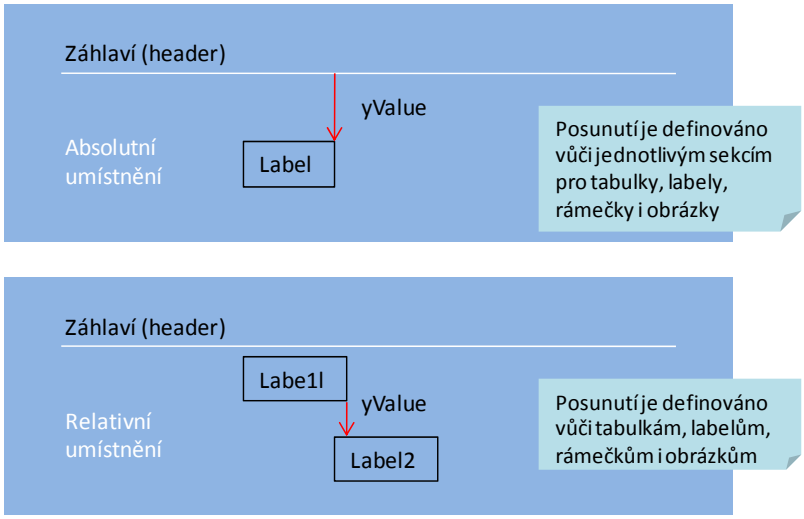
```
<print>
      <page type="first">
            <pageSection type="body">
            …
            </pageSection>
      </page>
</print>
```

Poslední volbou umístění, je volba zda bude prvek na stránce umístěn absolutními souřadnicemi, nebo relativně vůči prvku jinému. Tato vlastnost je hodnota parametru `type` tagu `location`. Hodnoty parametru tedy jsou:

1. absolute
2. relative

Pozn.: pro objednávkový formulář nabývá parametr ještě také hodnoty `relatid`, tato hodnota parametru je popsána v samotné kapitole tisku tabulek viz. kap. 3.6.1.2.

Poslední vlastností která již definuje konkrétní souřadnicové umístění na stránce je hodnota parametru posunutí `yValue`. Parametr definuje umístění v ose y, x-ová souřadnice je přebírána ze náhledového zobrazení formuláře, nikoliv z tiskových vlastností.



Obr. 20 Popis parametru `type`, `yValue` tiskového tagu `location`

Hodnota parametru yValue nám tedy definuje vzdálenost posunutí od jednotlivých sekcí stránky při absolutním umístění nebo od jiného prvku při relativním umístění. Při relativním umístění je prvek na který se odkazujeme definován hodnotou parametru `relatedID`.

Zápis těchto vlastností v souboru xml je následující:

absolutní umístění

```
<print>
      <page type="first">
            <pageSection type="body">
                  <location type="absolute" yValue="30">
            </pageSection>
      </page>
</print>
```

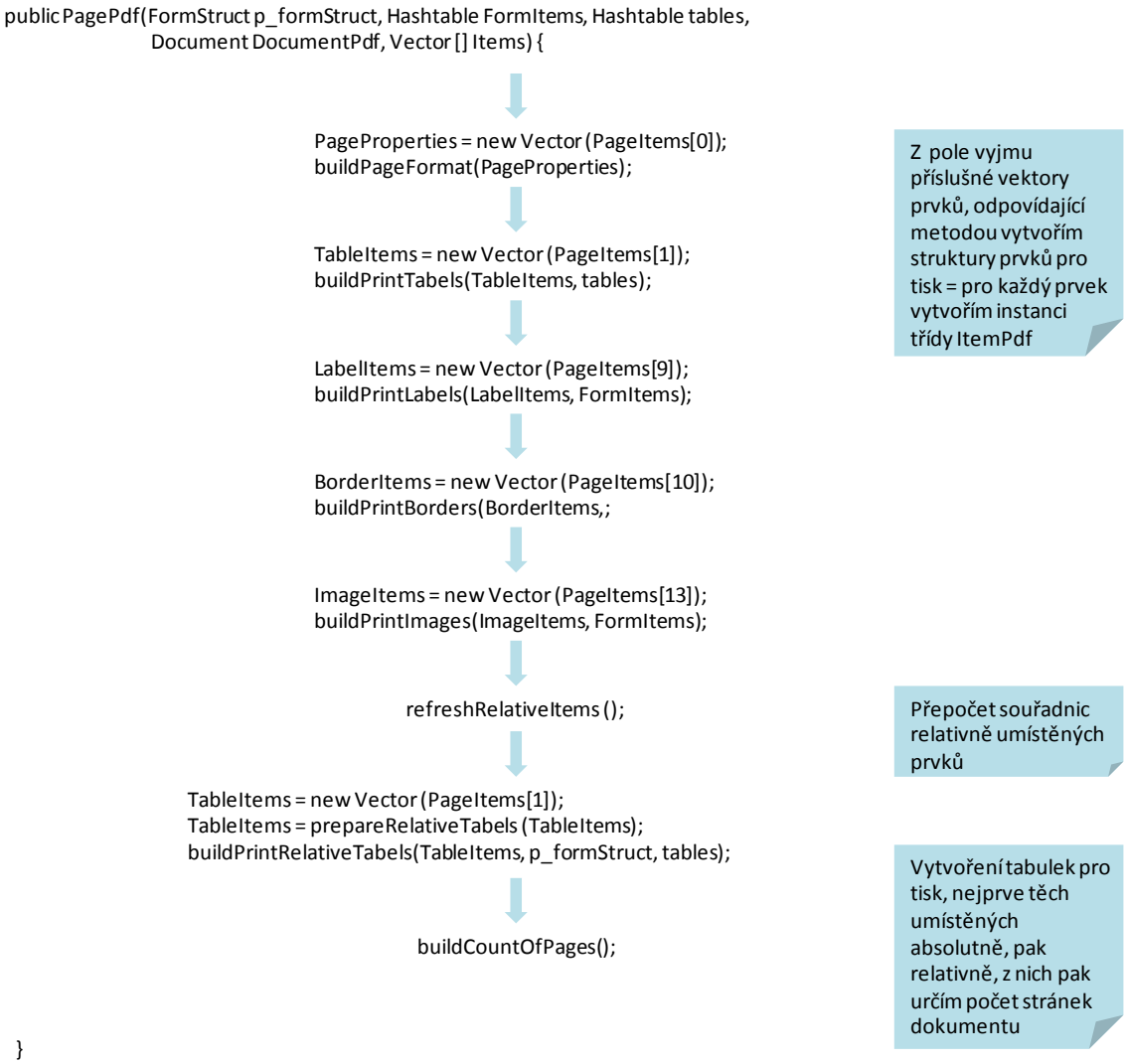
relativní umístění

```
<print>
      <page type="first">
            <pageSection type="body">
                  <location type="relative" relatedID="$label1"
                              yValue="30">
            </pageSection>
      </page>
</print>
```

Zpracování prvků formuláře xml, výpočet jejich umístění a souřadnic na stránce má na starost třída PagePdf popsaná níže.

3.6.1 PagePdf.java

Vstupem třídy jsou veškeré dostupné informace o struktuře formuláře, hashtable aktivních prvků, vytvářený dokument, hastable tabulek a struktura prvků pro konkretní typ vytvářené stránky (First, All, Last) platí pro první konstruktor třídy, pokud vytváříme stránku ze struktury typu One jsou vstupem stejná data, avšak bez hashtable tabulek. Výstupem je datová struktura, pole vektorů jednotlivých prvků ve formátu pro tisk. Formát těchto dat je popsán v kap. 3.6.1.1.



Obr. 21 Schéma tvorby struktur stránek, souřadnic prvků

Postupně zpracováváme datovou strukturu jednotlivých prvků vytvořenou třídou StructurePdf a pro jednotlivé prvky formuláře vytváříme jejich datovou podobu ve formátu pro tisk, jako instance třídy ItemPdf. Hlavní myšlenkou tiskových modulů bylo co nejmožnější oddělení jednotlivých částí zpracování dat. Máme tedy část přípravy dat, kde rozlišujeme která data zpracovat (třída StructurePdf), pak část která má na starost vytváření dokumentu (třída FilePdf). Pro tuto část jsou data připravena třídou PagePdf. Samotný tisk (umístění) prvků spočívá v předání dat ve třídě FilePdf metodám knihoven iText které byly využitu pro tvorbu dokumentu. Tyto metody mají ve své podstatě velmi podobný formát, vždy přiřazují určitá data (label, tabulku,…) na konkrétní souřadnice x,y, viz následující příklady:

jedna z metod pro umístění labelu:

```
cb.setTextMatrix((float)pointX,(float)pointY);
```

metoda pro umístění tabuly:

```
tablepdf.writeSelectedRows(0, -1, x, y, cb);
```

metoda pro umístění rámečku:

```
cb.rectangle(pointX, pointY, width, -height);
```

Aby tedy předání dat ve třídě FilePdf bylo co nejjednodušší, byla vytvořena třída popisující tuto *„tiskovou podobu“* dat.

3.6.1.1 ItemPdf.java

Tab. 3 Obecná struktura tisknutých prvků (ItemPdf)

ItemPdf

0	1	2
x	y	DATA

Po výpočtu konkrétních souřadnic jednotlivých prvků je pro každý z nich vytvořena instance této třídy a uložena do příslušného vektoru a nakonec do příslušného pole dle typu stránky. Zároveň jsou všechny prvky uloženy v „mapě dokumentu“ = GridItems = hashtable tisknutých prvků, proměnná třídy ItemPdf.

Následující tabulka popisuje jednotlivé datové části konkrétních prvků. Podle tohoto „klíče“ jsou pak data zpět čtena při předávání metodám knihoven iTextu.

Tab. 4 Datové části tisknutých prvků

prvek	x souřadnice	y souřadnice	formát
label	float	float	Vector
table	float	float	PdfPTable
border	float	float	Rectangle
image	float	float	Vector

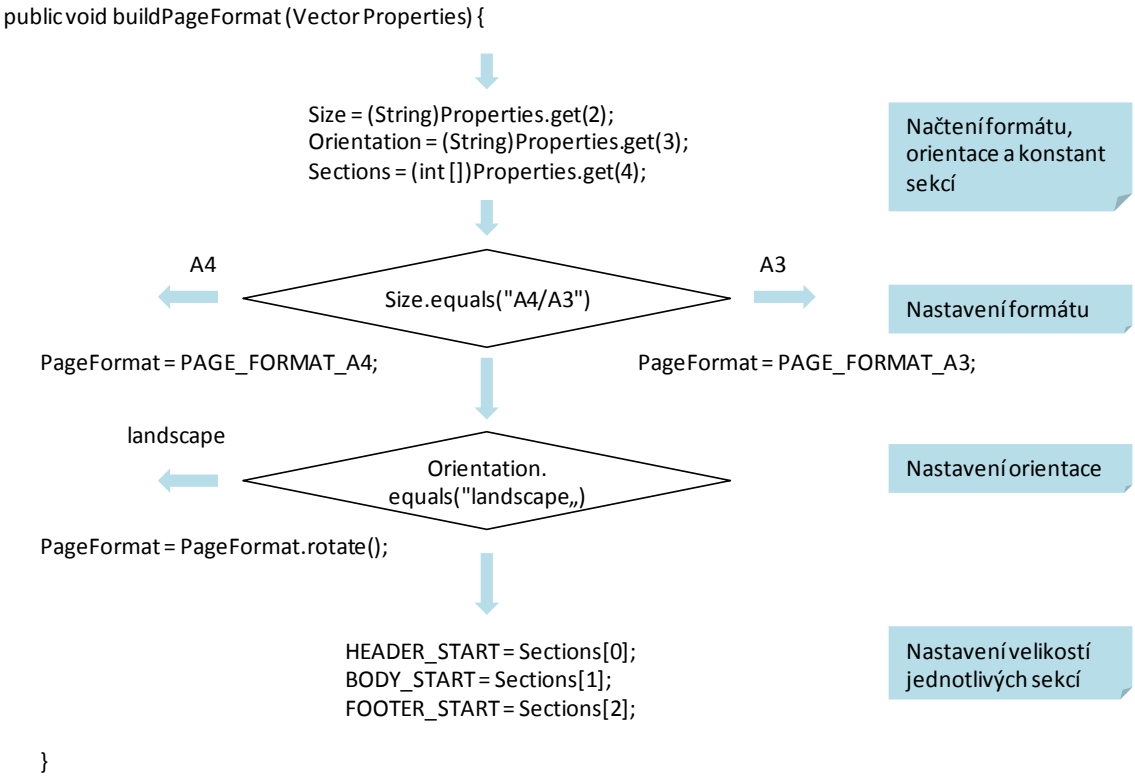
String	Font

Image	int	int

Pro případné uložení, definování nových vlastností tedy nemusíme měnit žádné struktury dat, pouze před vytvořením instance konkrétnímu prvku vlastnosti nastavíme.

buildPageFormat (Vector Properties)

metoda má za úkol nastavení formátu stránky, orientace a nastavení velikostí jednotlivých sekcí, tyto data jsou v jednotlivých polích struktur stánek uloženy na prvním místě [0], viz. Tab. 1.



Obr. 22 Metoda buildPageFormat (Vector Properties)

V souboru xml jsou zpracovávané informace o formuláři zapsané následovně:

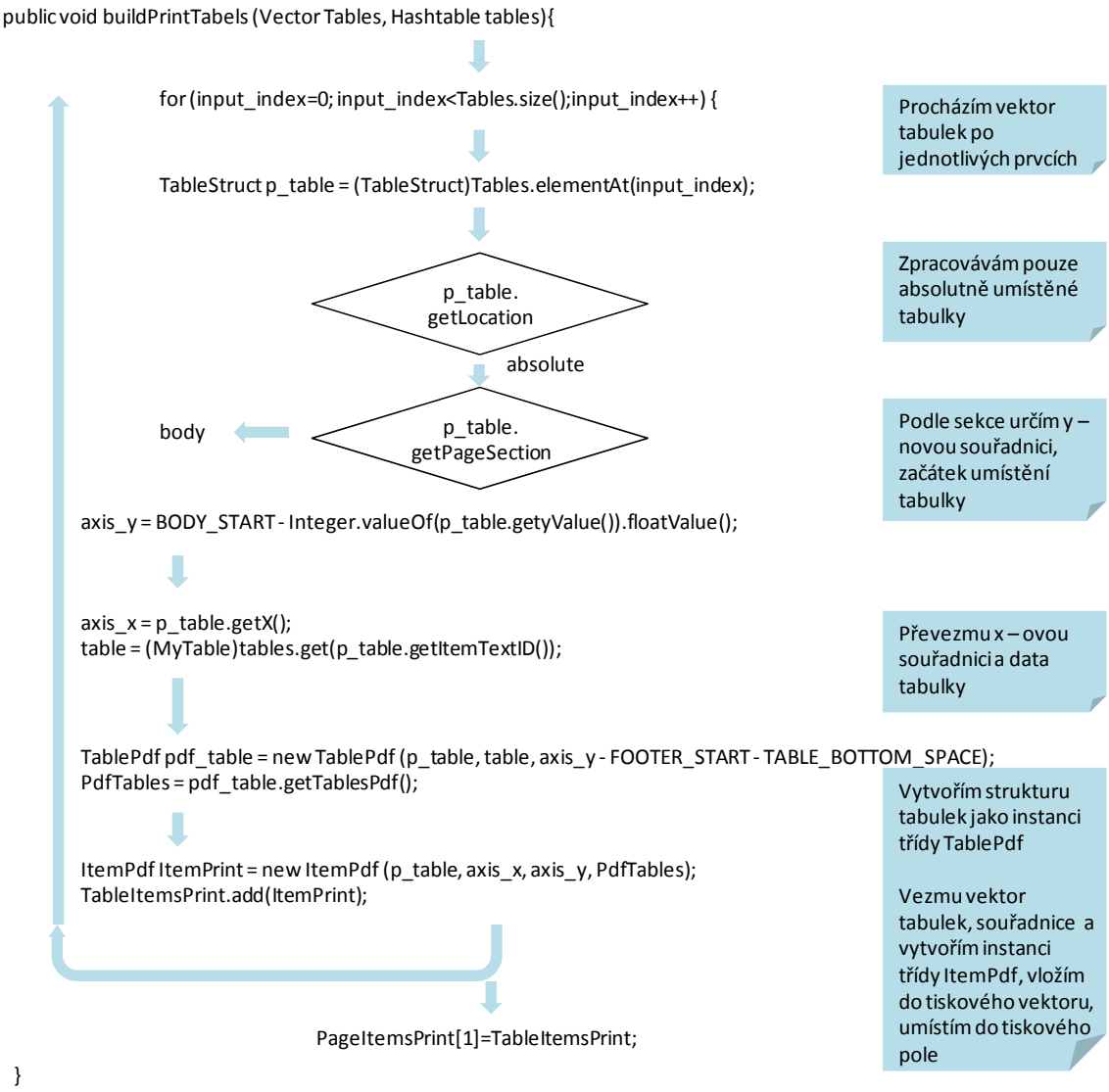
```
<print>
      <fileName>Obj</fileName>
      <pageSize>A4</pageSize>
      <pageOrientation>portrait</pageOrientation>
      <pageSection header="840" body="730" footer="50"/>
</print>
```

Jméno může a nemusí být využito pro pojmenování výsledného tiskového dokumentu, v současné verzi využito není a jméno je přebíráno jako parametr volby tiskového formuláře předcházejícího před formulářem tiskovým, viz. kap. 3.1.

Následující metody mají podobnou strukturu, jejich úkolem je přiřadit prvkům obsažených v příslušných vektorech konkrétní souřadnice na stránce a tyto souřadnice spolu s daty uložit v tiskovém formátu jako instance třídy ItemPdf do tiskových vektorů, polí.

buildPrintTabels (Vector Tables, Hashtable tables)

připravuje tiskovou podobu tabulek formuláře, projde vektor tabulek a pro tabulky které existují, obsahují data, určí nejprve souřadnice a pomocí nich vytvoří tiskové tabulky s daty. Tiskové tabulky jsou instancemi třídy PdfPTable knihoven iText. Pro tvorbu těchto tabulek byla vytvořena třída TablePdf. Schéma metody a popis třídy následuje:

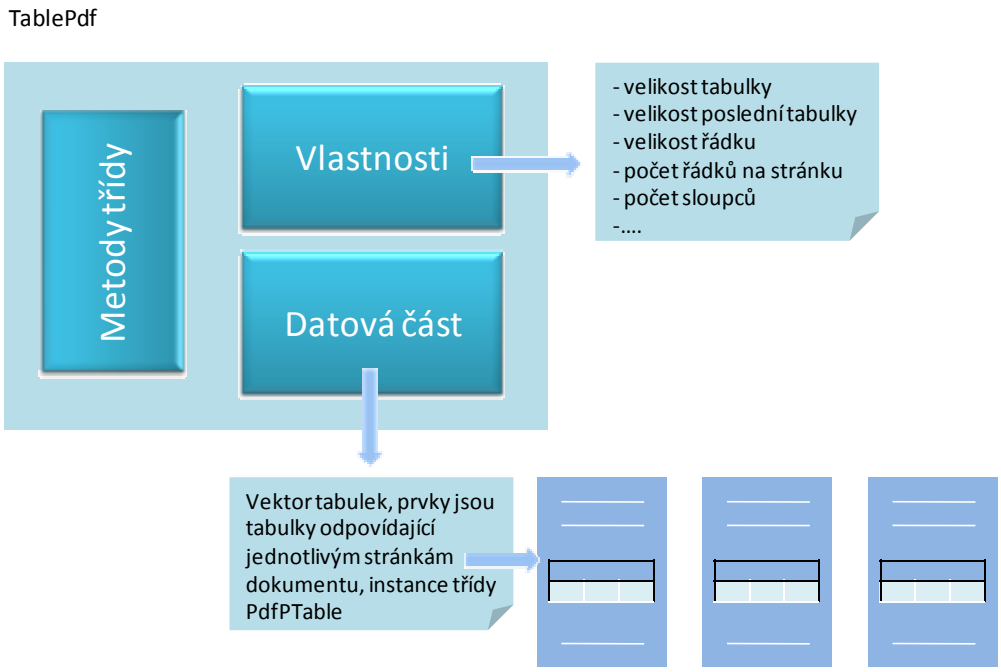


Obr. 23 Metoda buildPrintTabels (Vector Tables, Hashtable tables)

Metoda slouží k obsloužení tabulek které jsou umístěny absolutně na stránce dokumentu a navíc jen pro tabulky umístěné v sekci `body`, s umístěním tabulek v jiných sekcích není počítáno. Popis tabulek, jejich umístění, výpočet dat atd. následuje:

3.6.1.2 Tabulky, TablePdf.java

Obsloužení tohoto prvku je v systému nejsložitější, avšak záleží na něm samotná struktura dokumentu, zejména počet stránek. Hlavní rozdíl mezi tabulkou a ostatními prvky je ten, že tabulka jako taková má na každé stránce dokumentu, jinou datovou část. Ostatní prvky svou hodnotu nemění, tisknemeli-je na jakékoliv stránce. Proto, mluvíme-li o tabulce, chápeme tím i jeji „podtabulky“ připadající jednotlivým stránkám dokumentu. Tisková forma prvku tabulky je instancí třídy TablePdf a má následující podobu:



Obr. 24 Obecná struktura třídy TablePdf.java

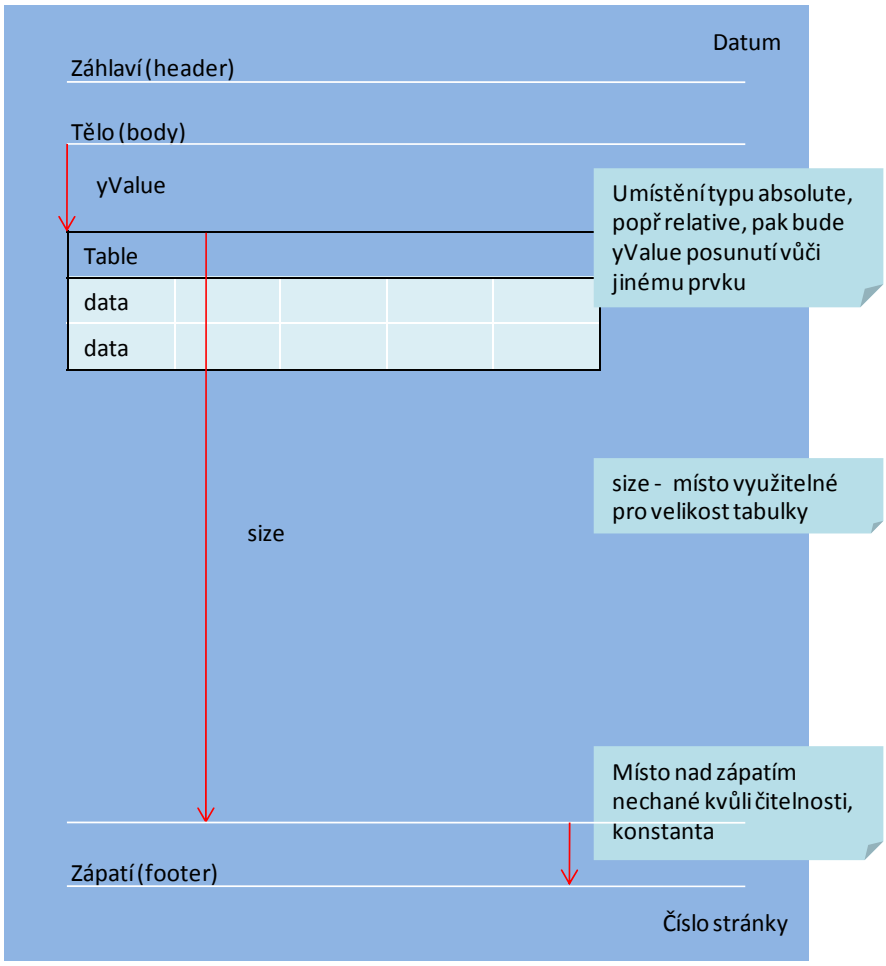
Výpočet jednotlivých podtabulek závisí na volbě umístění. Prvek tabulka rozšiřuje hodnotu parametru type tagu `location`, popsaného v kap. 3.6.o hodnotu `relatid`. Výpočet souřadnic je pak pro hodnoty parametrů následovný:

- *absolute* - tabulka je umístěna absolutními souřadnicemi, hodnota parametru `yValue` udává její y-novou vzdálenost od sekce body
- *relative* - tabulka je umístěna absolutními souřadnicemi, hodnota parametru `yValue` udává jeji y-novou vzdálenost od jiného prvku v sekci body
- *relatid* - tabulka je umístěna relativně vůči jiné tabulce

Zápis těchto vlastností v souboru xml je následující:

```
<print>
      …
      <location type="absolute" yValue="45"/>
      nebo
      <location type="relative" relatedID="$lbZasilka4" yValue="45"/>
      nebo
      <location type="relatid" relatedID="$tbl_objNab_1" yValue="45"/>
      …
</print>
```

Princip zpracování a výpočtu souřadnic tabulky na stránce ukazují následující obrázky:



Obr. 25 Umístění tabulky na stránce dokumentu

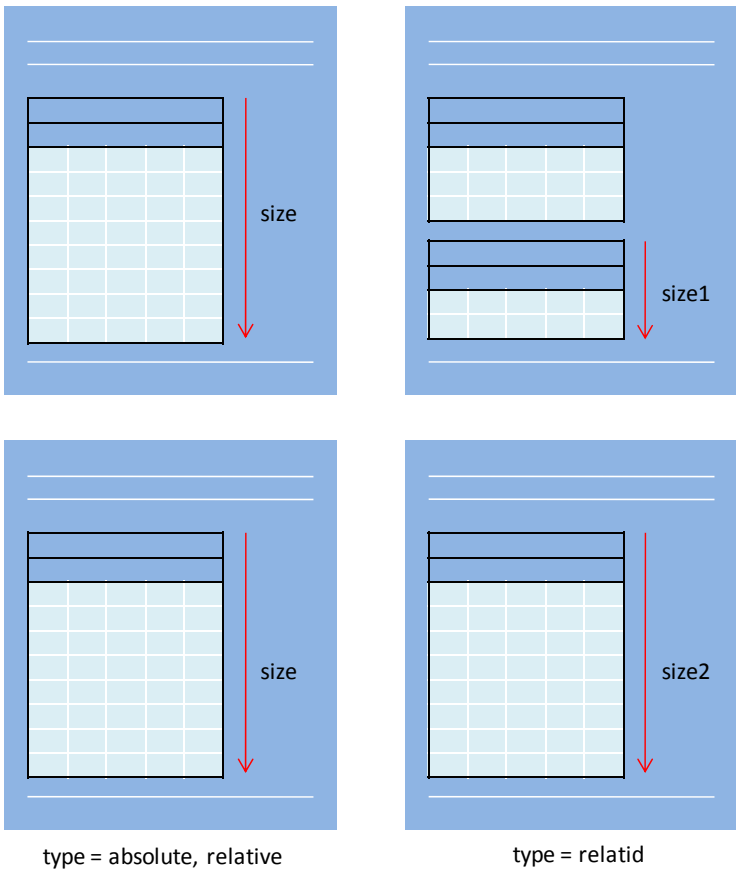
Údaje se kterými pracujeme při tvorbě tabulek jsou hodnoty parametru `yValue` a využitelná velikost místa na stránce *size*. Pro výpočet této velikost platí následující vztah:

```
size = axis_y - FOOTER_START - TABLE_BOTTOM_SPACE
```

kde

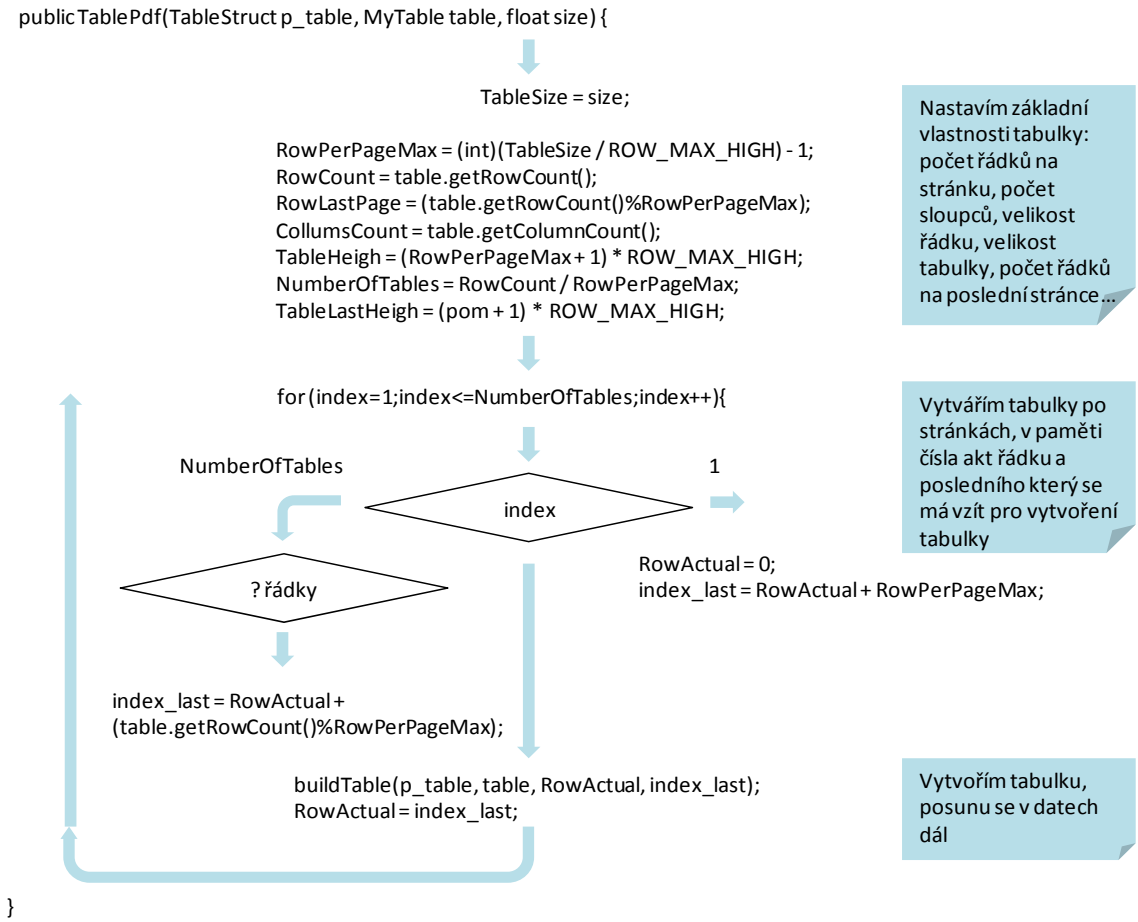
```
axis_y = BODY_START - yValue        type = „absolute“
axis_y = y_relative - yValue        type = „relative“
FOOTER_START                        začátek sekce zápatí
TABLE_BOTTOM_SPACE                  konstanta pro volné místo
```

Tyto výpočty a údaje platí pro tabulku umístěnou absolutně, či relativně, jsou vstupními hodnotami prvního konstruktoru třídy TablePdf. Srovnání umístění s umístěním typu relatid ukazuje následující obrázek:



Obr. 26 Porovnání umístění tabulek dle parametru `type` tagu `location`

Pokud umísťujeme tabulku absolutně, či relativně, začíná vždy na každé stránce stejně, a má stejnou velikost(mimo stránku poslední), naopak při umístění typu `relatid` je zpracování složitější a v paměti musíme udržovat velikosti dvě, a dvě rozdílné souřadnice umístění. Tabulka je pak tvořena druhým konstruktorem třídy TablePdf.



Obr. 27 Konstruktor třídy TablePdf pro hodnoty `absolute, relative`

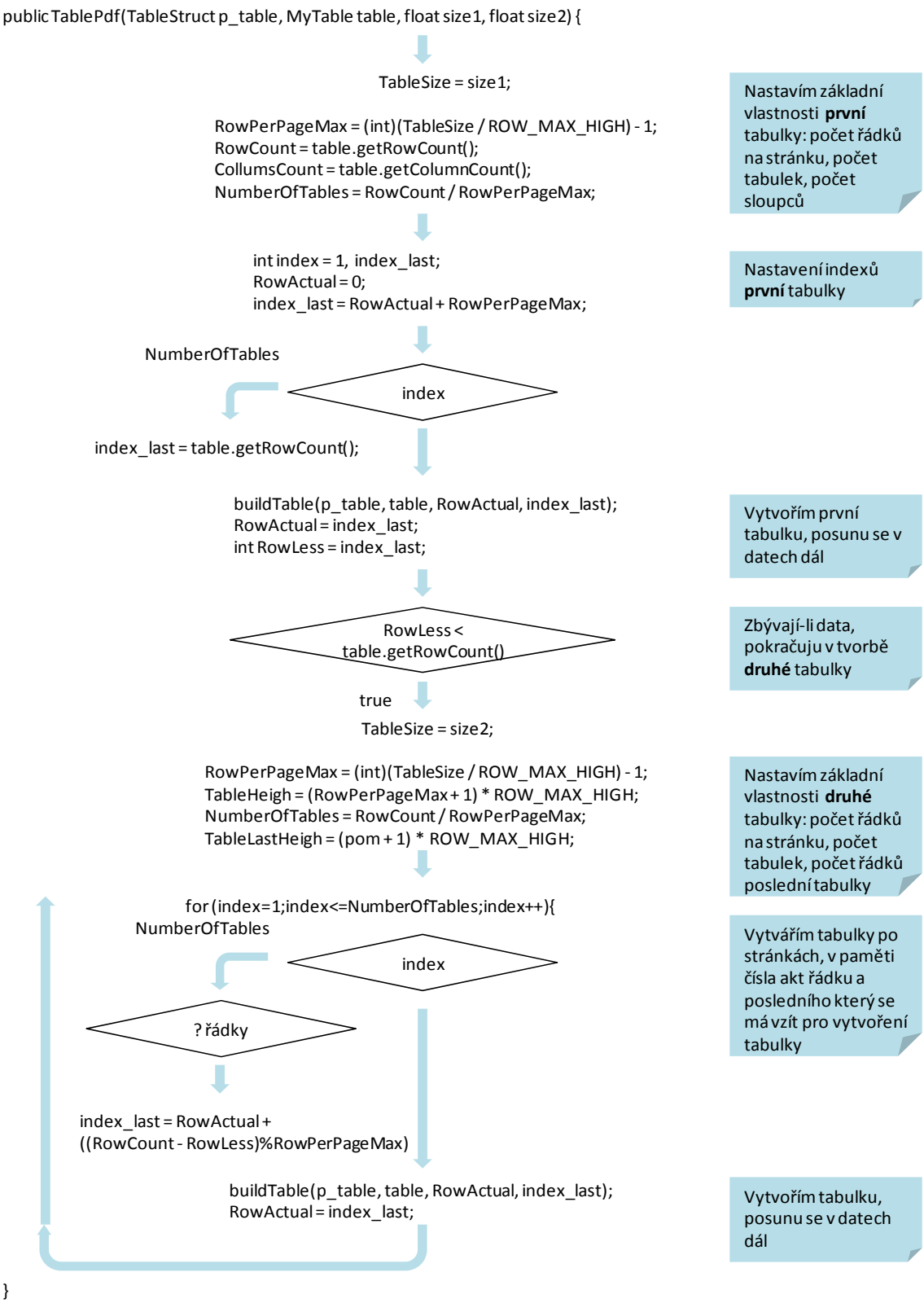
Z využitelného místa pro velikost tabulky na stránce *size* odvozujeme většinu proměnných třídy popisujích vlastnosti tabulky:

- *RowPerPageMax* - maximální počet řádků tabulky které se vlezou na stránku dokumentu
- *RowLastPage* - počet řádků „poslední“ tabulky
- *NumberOfTables* - počet tabulek které jsem schopen vytvořit z max. počtu řádků a velikosti dat tabulky
- *TableHeigh* - velikost tabulky na stránce

Ostatní proměnné odvozujeme z vlastností tabulky zobrazené v náhledu tiskového formuláře, jsou jimi:

- *RowCount* - počet řádků dat
- *CollumsCount* - počet sloupců tabulky

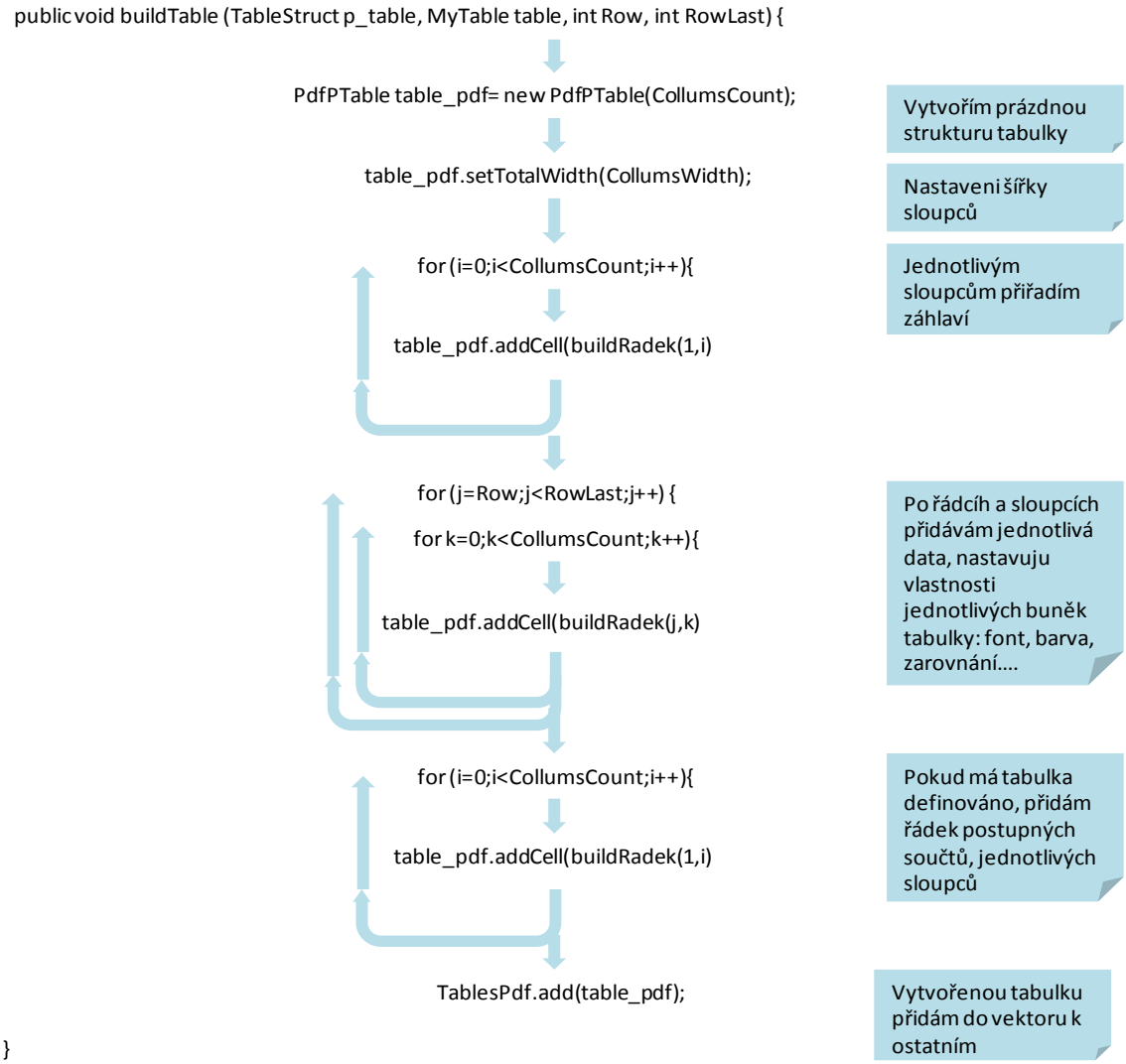
Se stejnými proměnnými pracuje i druhý konstruktor třídy pro tabulky umístěné hodnotou `relatid` parametru type tagu `location`.



Obr. 28 Konstruktor třídy TablePdf pro hodnotu `relatid`

Rozdíl oproti prvnímu konstruktoru je jen v rozdělení výpočtu na dvě části, protože první podtabulka má jinou využitelnou velikost než ty na následujících stránkách a jinou y-novou souřadnici. Jinak se zpracování dat neliší. Samotné datové části tabulek jsou vytvářeny pomocí následující metody třídy:

metoda buildTable (TableStruct p_table, MyTable table, int Row, int RowLast)



Obr. 29 Metoda buildTable (TableStruct p_table, MyTable table, int Row, int RowLast)

Po té co je tabulce přiřazena šířka jednotlivých sloupců, jako první vložím záhlaví tabulek. Postupem času vznikl u tabulek požadavek na odlišení záhlaví tabulky a datové části, toho je docíleno nastavením různých fontů těchto částí. Parametry jsou přebírány ze souboru xml a jejich zápis má následující formát:

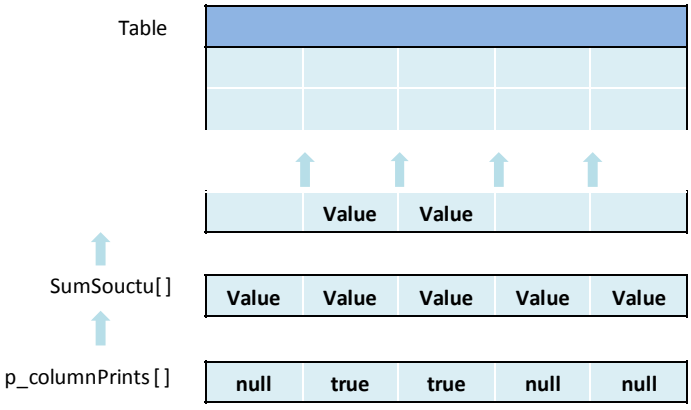
```
<print>
      ...
      <fontp face1="Arial" size1="8" face2="Arial" size2="7"/>
      ...
</print>
```

Parametry `face1` a `size1` definují font záhlaví tabulky, `face2` a `size2` font pro datovou část.

Po přidání posledního řádku dat, kontrolujeme, zda nemá některý ze sloupců nastavenou vlastnost pro zobrazení postupného součtu za stránku, tato vlastnost je uložena v parametru `printSuma` tagu `column`. Pokud se má součet sloupce zobrazovat je potřeba uvést hodnotu `true`.

```
<table>
...
      <columns>
      <column title="Kód" width="105" align="left" format=""
       printSuma="true"/>
      </columns>
...
</table>
```

Postupné součty jsou vypočítávány zároveň s přidáváním dat do tabulky, hodnoty jsou uloženy v poli které svou velikostí odpovídá počtu sloupců v tabulce. Pokud tedy přidáváme řádek postupných součtů, procházíme tento vektor a dle něj plníme příslušné buňky pod sloupci viz následující obr.



Obr. 30 Přidání řádku postupných součtů do tabulky

Samotné obsloužení přidání jednotlivých buněk tabulky má na starost přetížená metoda buildCell.

metoda buildCell (int FontSize, float High, boolean Grid, Color BackGround, String Align, BaseFont bf, String Data)

vstupní parametry metody:

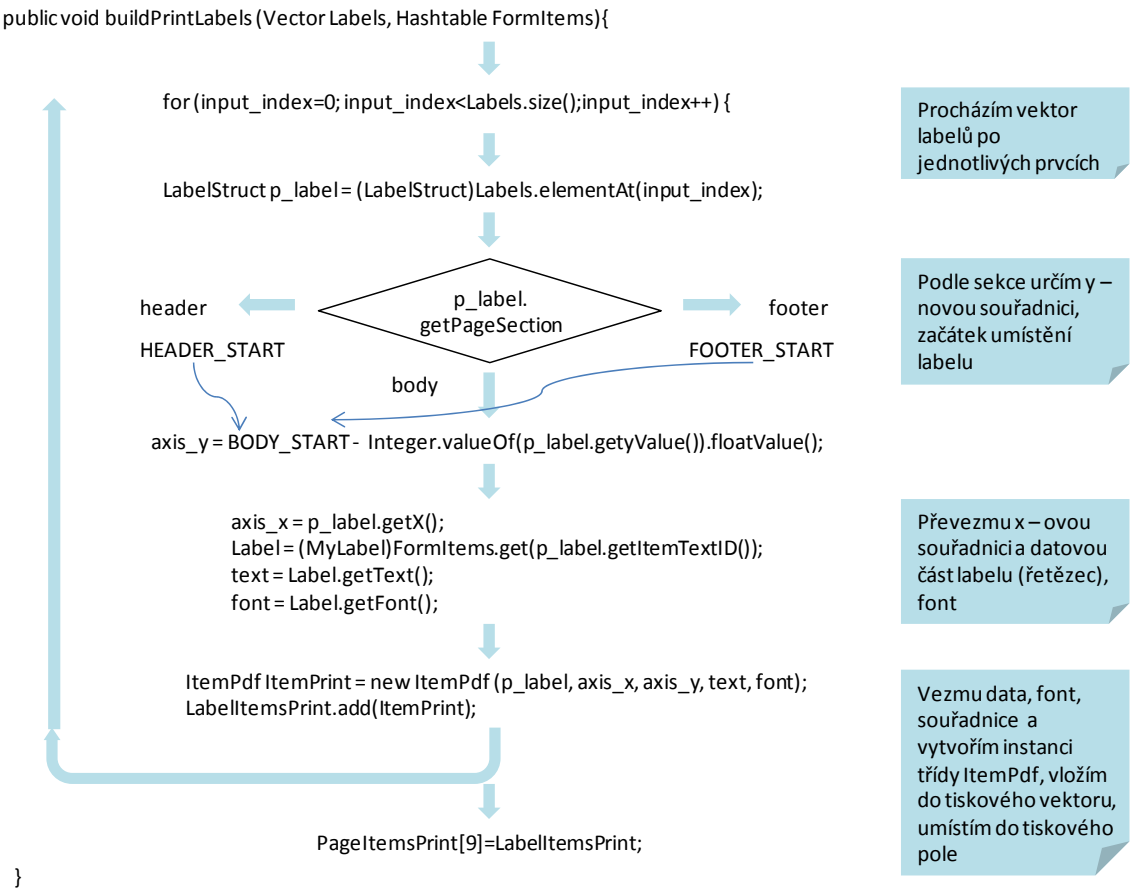
- *FontSize* - velikost fontu textu buňky
- *High* - velikost buňky
- *Grid* - rámeček buňky
- *BackGround* - barva pozadí buňky
- *Align* - zarovnání textu buňky
- *bf* - font buňky
- *Data* – vkládaný řetězec

Metody se liší parametrem `grid`, druhá metoda má parametr typu `int`. Tento návrh je kvůli oddělení ohraničení, první metoda s parametrem typu `boolean`, vypíná / zapíná ohraničení celé buňky, to je využito u vykreslení ohraničení celé tabulky. Druhá metoda je použita k oddělení řádku součtů čárou od celé tabulky, hodnota typu `int` nám tedy udává pouze jednu stranu obdélníku která se má vykreslit. V souboru xml je parametr ohraničení zapsán jako:

```
<print>
      ...
      <gride type="true"/>
      ...
</print>
```

buildPrintLabels (Vector Labels, Hashtable FormItems)

Připravuje tiskovou podobu prvků typu label formuláře, projde vektor labelů a podle umístění v jednotlivých sekcích vypočítá konkrétní souřadnice, pomocí souřadnic a datové části vytvoří jednotlivé instance třídy ItemPdf a ty uloží do tiskových vektorů a posléze na příslušné místo v tiskovém poli.



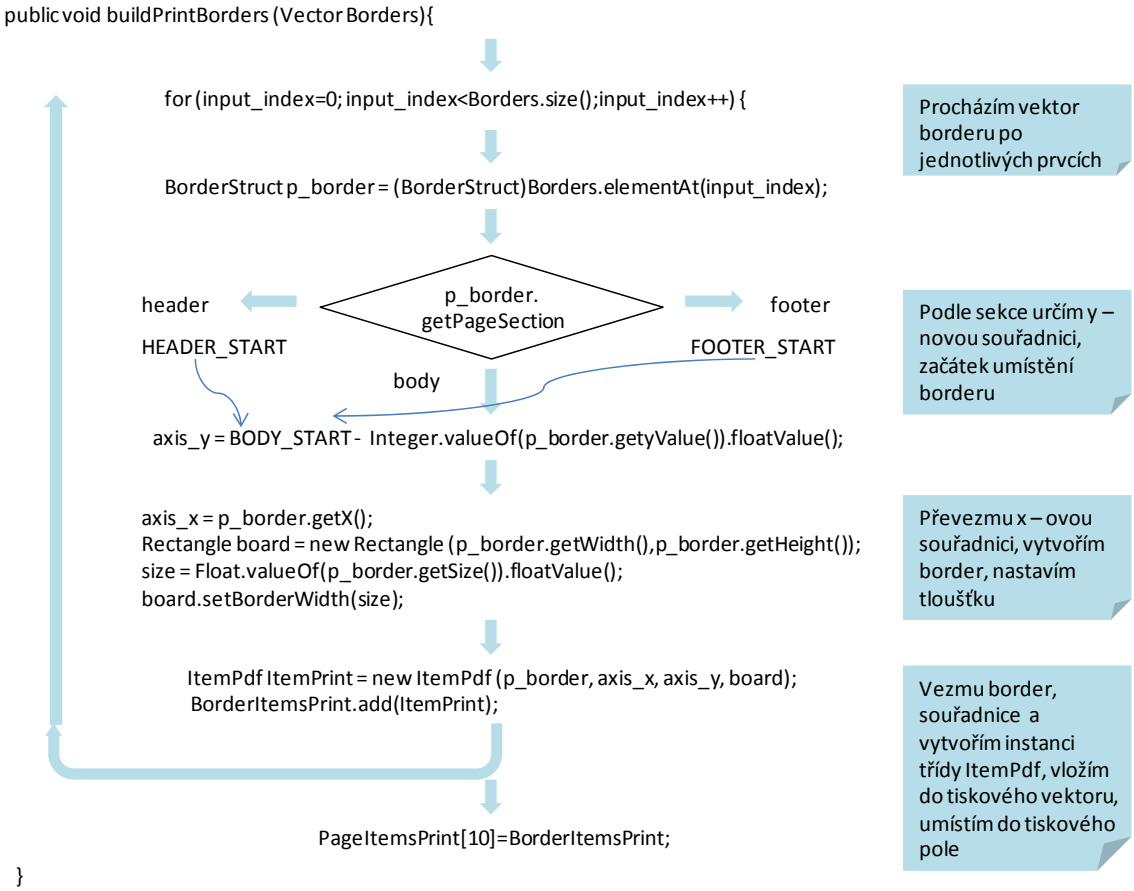
Obr. 31 Metoda buildPrintLabels (Vector Labels, Hashtable FormItems)

Metoda zpracovává labely umístěné absolutně i relativně na stránce, souřadnice labelů umístěných relativně jsou následovně přepočteny metodou RefreshRelativeItems () uvedenou níže. Datovou část instance třídy ItemPdf tvoří Vektor jehož prvky jsou řetězec a jeho font, viz. také Tab. 4. Font není definován zvlášť v tiskových tazích, ale je převzat z parametrů pro náhledové zobrazení tiskového formuláře v systému. V souboru xml zapsáno následovně:

```
<label>
      ...
      <font face="Arial" size="10" style="bold"/>
      ...
      <print>
            ...
      </print>
</label>
```

buildPrintBorders (Vector Borders)

Připravuje tiskovou podobu prvků typu border formuláře, projde vektor borderů a podle umístění v jednotlivých sekcích vypočítá konkretní souřadnice, pomocí souřadnic a datové části vytvoří jednotlivé instance třídy ItemPdf a ty uloží do tiskových vektorů a na příslušné místo v tiskovém poli.



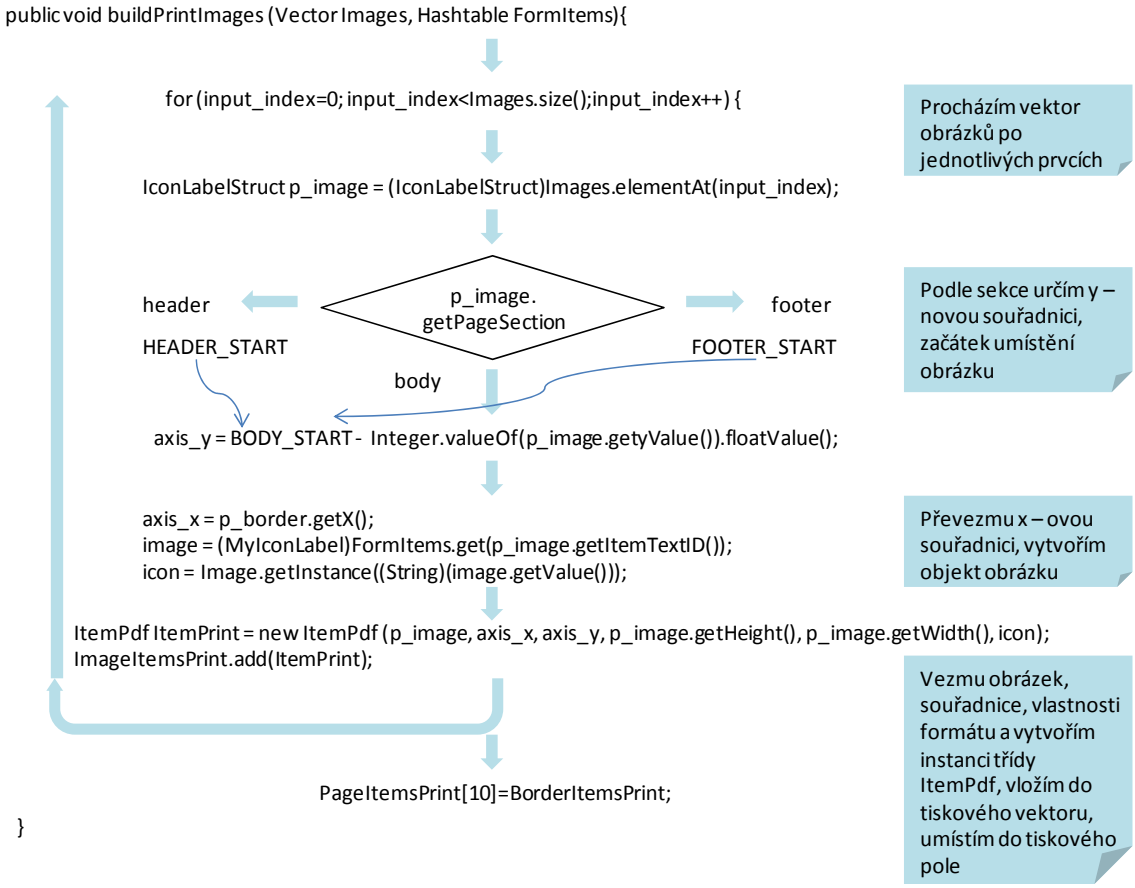
Obr. 32 Metoda buildPrintBorders (Vector Borders)

Metoda zpracovává bordery umístěné absolutně i relativně na stránce, souřadnice borderů umístěných relativně jsou následovně přepočteny metodou RefreshRelativeItems () uvedenou níže. Datovou část instance třídy ItemPdf tvoří řámeček (Rectangle) viz. také Tab. 4. Tloušťka rámečku je definována hodnotou parametru `lineWidth` tagu `properties`. V souboru xml zapsáno následovně:

```
<border>
      ...
      <properties lineWidth="0.5"/>
      ...
</border>
```

buildPrintImages (Vector Images, Hashtable FormItems)

Připravuje tiskovou podobu prvků typu iconLabel formuláře, projde vektor prvků a podle umístění v jednotlivých sekcích vypočítá konkrétní souřadnice, pomocí souřadnic a datové části vytvoří jednotlivé instance třídy ItemPdf a ty uloží do tiskových vektorů a na příslušné místo v tiskovém poli.

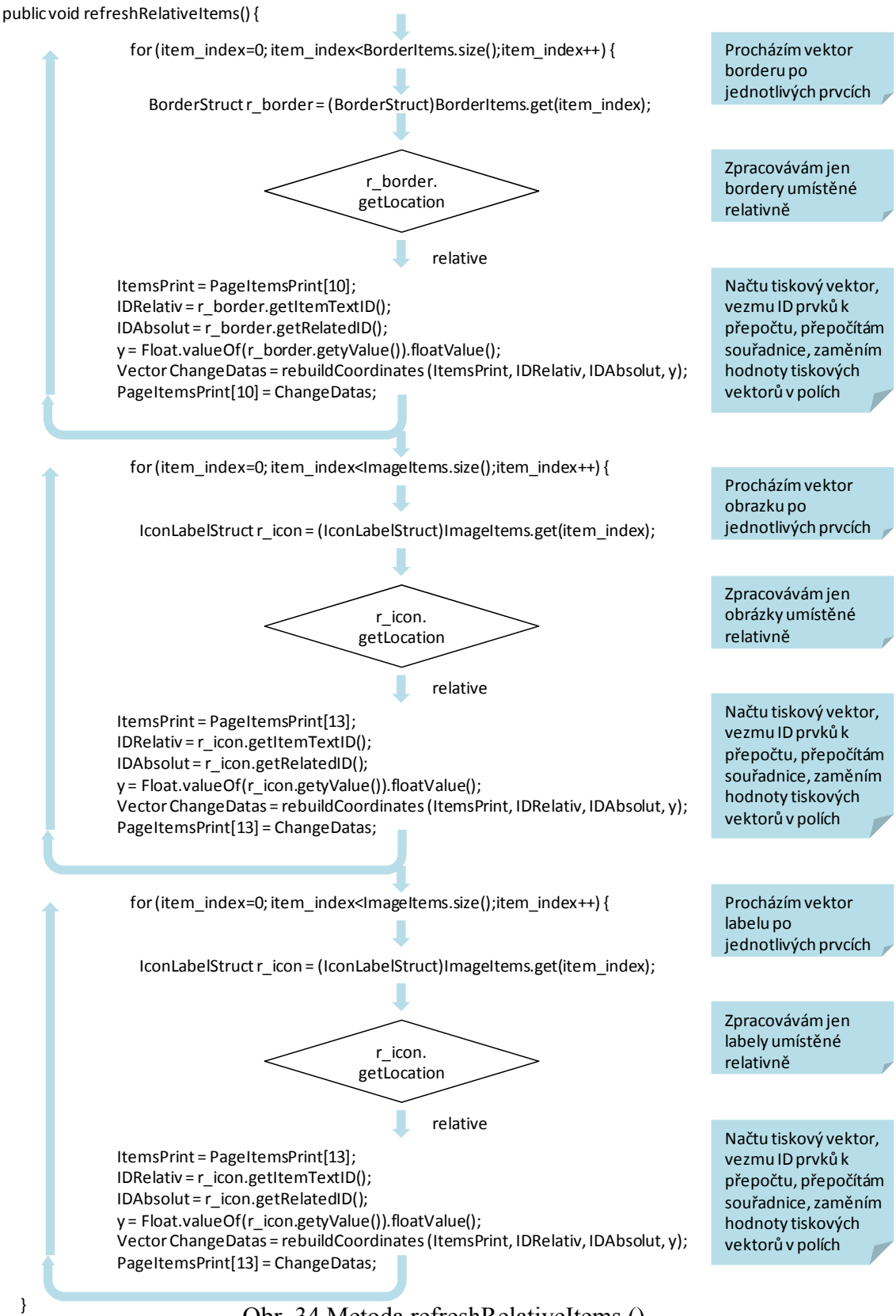


Obr. 33 Metoda buildPrintImages (Vector Images, Hashtable FormItems)

Metoda zpracovává obrázky umístěné absolutně i relativně na stránce, souřadnice obrázků umístěných relativně jsou následovně přepočteny metodou RefreshRelativeItems () uvedenou níže. Datovou část instance třídy ItemPdf tvoří Vektor jehož prvky jsou Image, výška, šířka. viz. také Tab. 4. Cesta k souboru ve kterém je uložen obrázek je definována hodnotou tagu `iconPath`. V souboru xml zapsáno následovně:

```
<iconLabel>
        ...
        <iconPath>Q:/!zps_act/images/ZPS_logo.gif</iconPath>
        ...
</iconLabel>
```

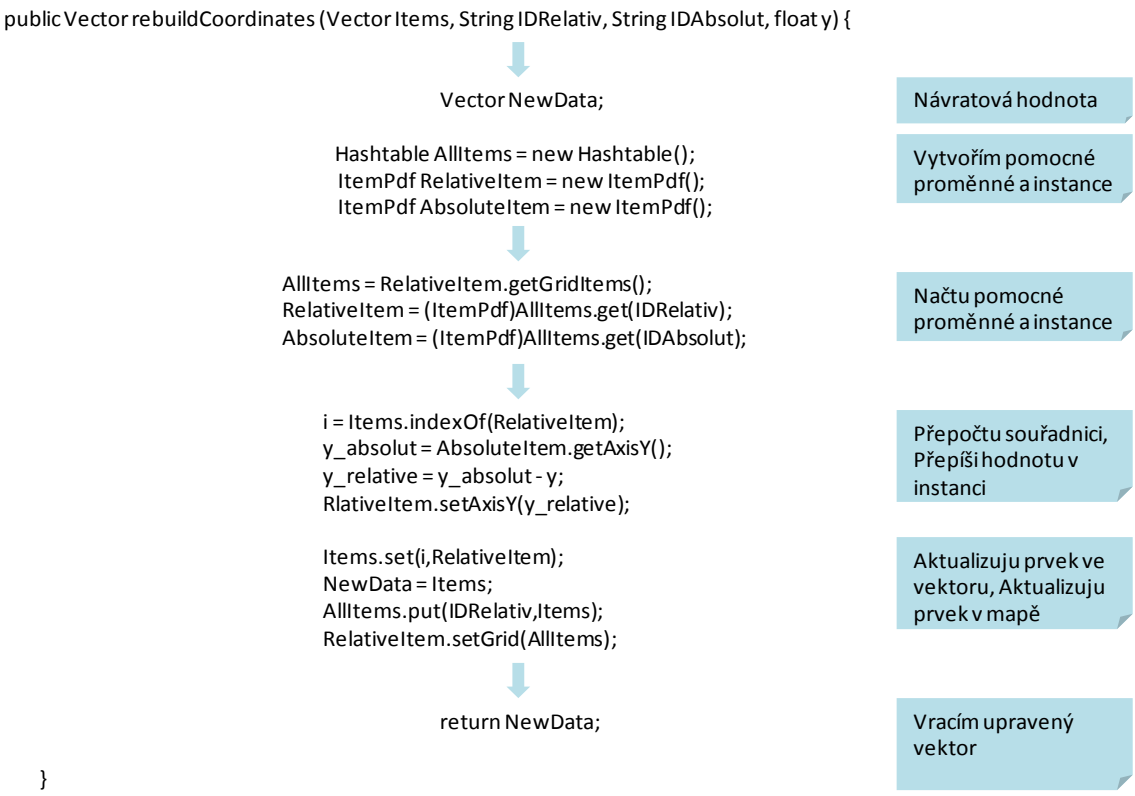
refresh RelativeItems ()



Obr. 34 Metoda refreshRelativeItems ()

Metoda upravuje y-nové souřadnice prvků formuláře umístěných relativně, hlavní částí je jiná metoda – rebuildCoordinates (ItemsPrint, IDRelativ, IDAbsolut, y). Po přepočtení souřadnic, vrací zamněněný tiskový vektor dat, ten je pak znova uložen v tiskovém poli, viz metoda refreshRelativeItems (). S relativně umístěnými prvky tedy napřed pracujeme jakoby byly umístěné absolutně viz metody buildPrintLabels, buildPrintBorders,…, teprve pak jsou jejich souřadnice přepočteny, vše z praktického důvodu, napřed je potřeba mít absolutně umístěné prvky abychom mohli umístit ty relativní k nim.

Vector rebuildCoordinates (ItemsPrint, IDRelativ, IDAbsolut, y)



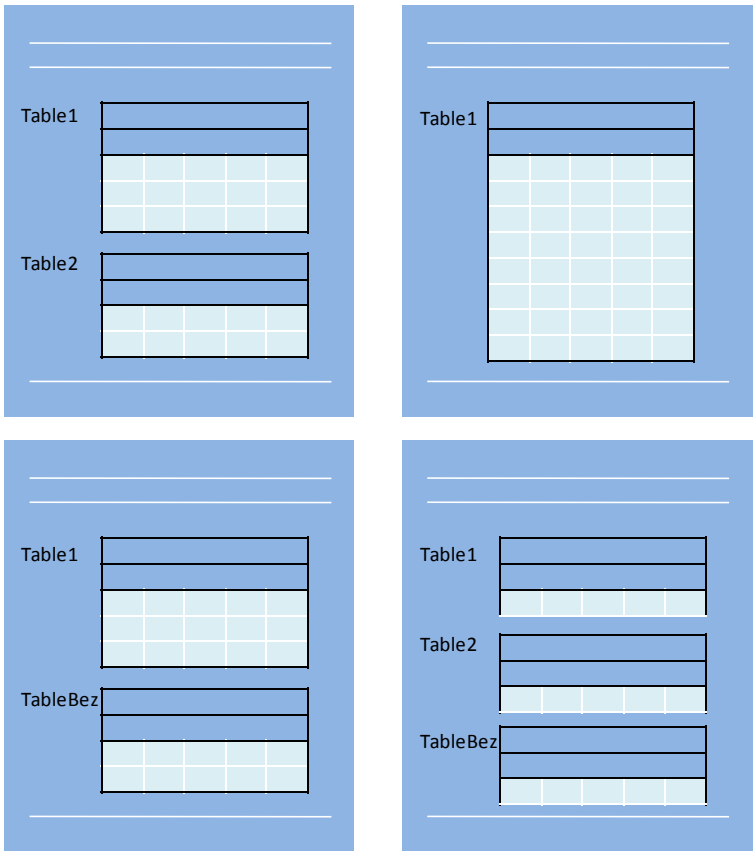
Obr. 35 Metoda Vector rebuildCoordinates (ItemsPrint, IDRelativ, IDAbsolut,y)

Jednotlivé proměnné metody:

- *ItemsPrint* - tiskový vektor dat (prvky instance třídy ItemPdf)
- *IDRelativ* - textové ID prvku u kterého přepočítáváme souřadnice
- *IDAbsolut* - textové ID prvku ke kterému se vztahuje prvek umístěný relativně
- *y* - hodnota yValue, posunutí

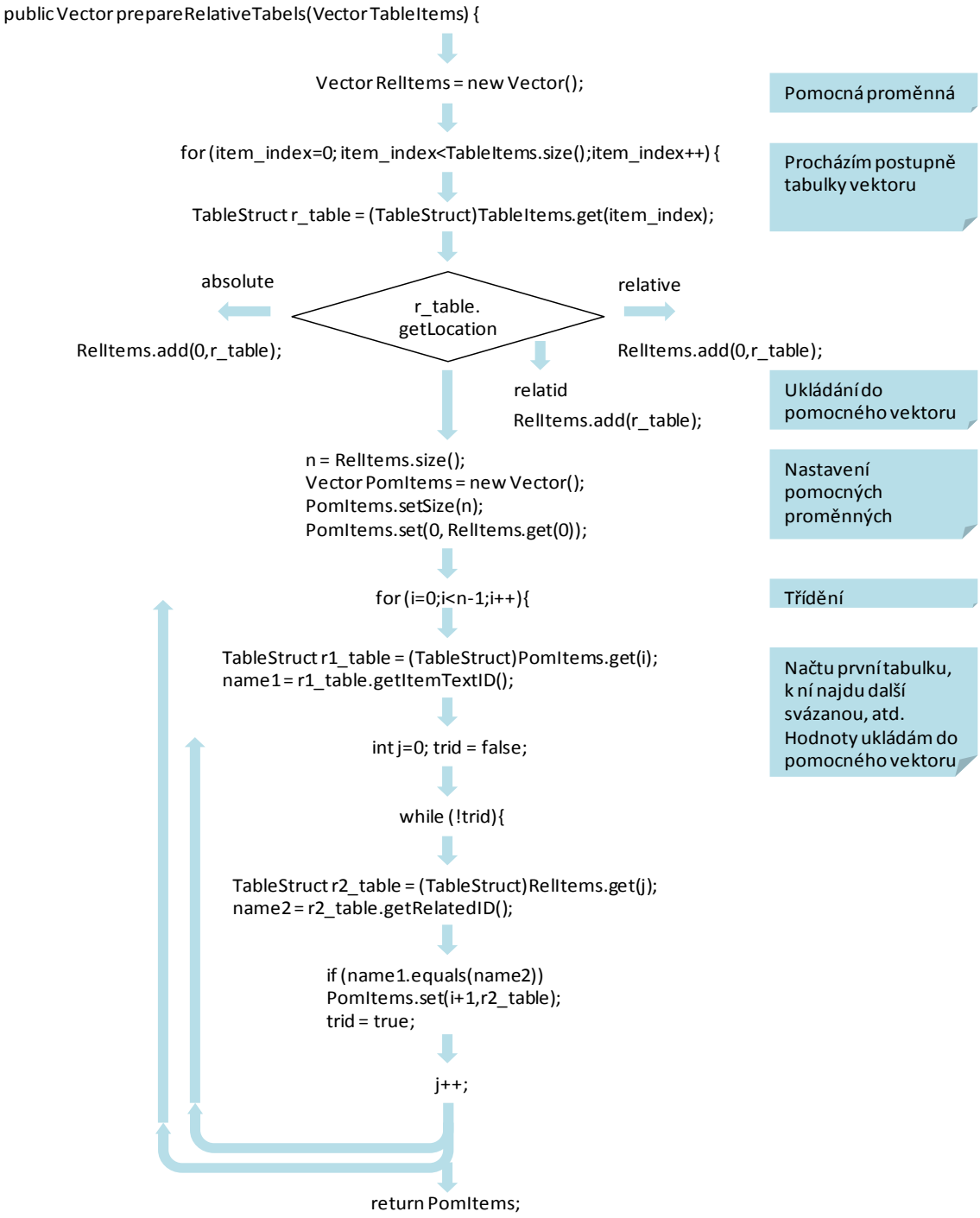
Jako poslední určujeme souřadnice tabulek, umístěných relativně na stánkách dokumentu. Těchto tabulek je využito při tvorbě tiskového formuláře objednávky. Formulář obsahuje 5 před-definovaných struktur tabulek, které se plní daty v závislosti na uživateli a jeho požadavcích. Před samotnou tvorbou tabulek, je nutné setřídit vektor tabulek podle pořadí ve kterém na sebe navazují. Ošetření vzniklo z důvodu jiného zpracování prvků formuláře na platformě Linux a jiného na platformě Windows.

```
<form $Objednavka>
...
<Tabulka s nabídkou č.1>
Table1
...
...
<Tabulka s nabídkou č.2>
Table2
...
...
<Tabulka s nabídkou č.3>
Table3
...
...
<Tabulka s nabídkou č.4>
Table4
...
...
<Tabulka bez nabídky>
TableBez
...
...
</form>
```



Obr. 36 Formulář objednávky a jeho tabulky

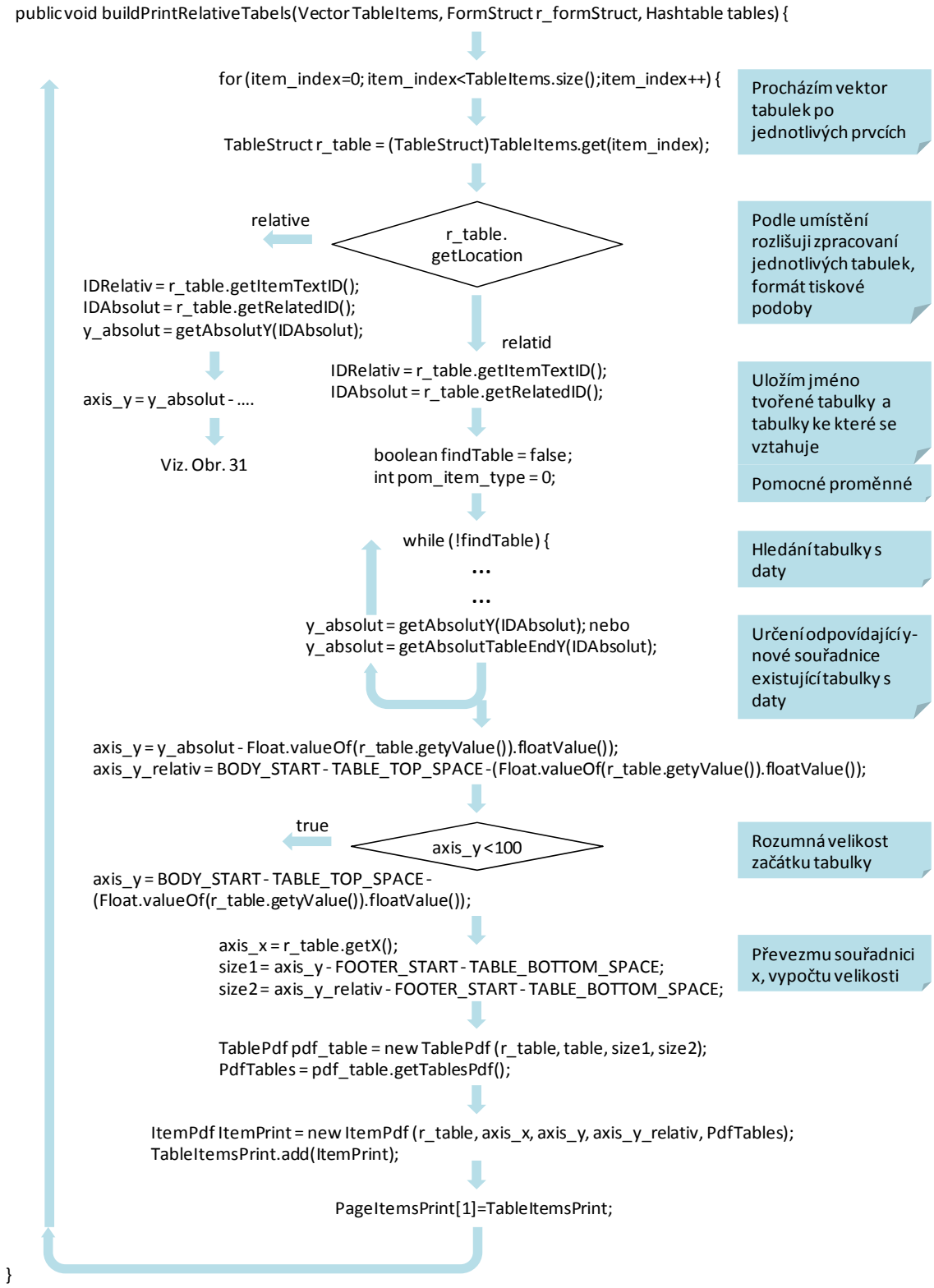
Ve výsledném tiskovém formuláři tedy může být tabulka jedna, či více, jak ukazuje předcházející obrázek. Existuje také klíč – nelze dosáhnout stavu aby vzniklo: Table2 – TableBez, či Table3 – TableBez. Tabulky Table1, TableBez mohou existovat samostatně, ale tabulky Table2, Table3, Table4 nemohou být bez svých předchůdců. Při tvorbě takovéhoto formuláře je nutno dodržet pravidlo – první tabulka musí být definována hodnotou parametru `location` jako `absolute` nebo `relative`, od této tabulky, odvozujeme posléze souřadnice ostatních. Algoritmus třídění obsahuje následující metoda:



Obr. 37 Metoda Vector prepareRelativeTabels (Vector TableItems)

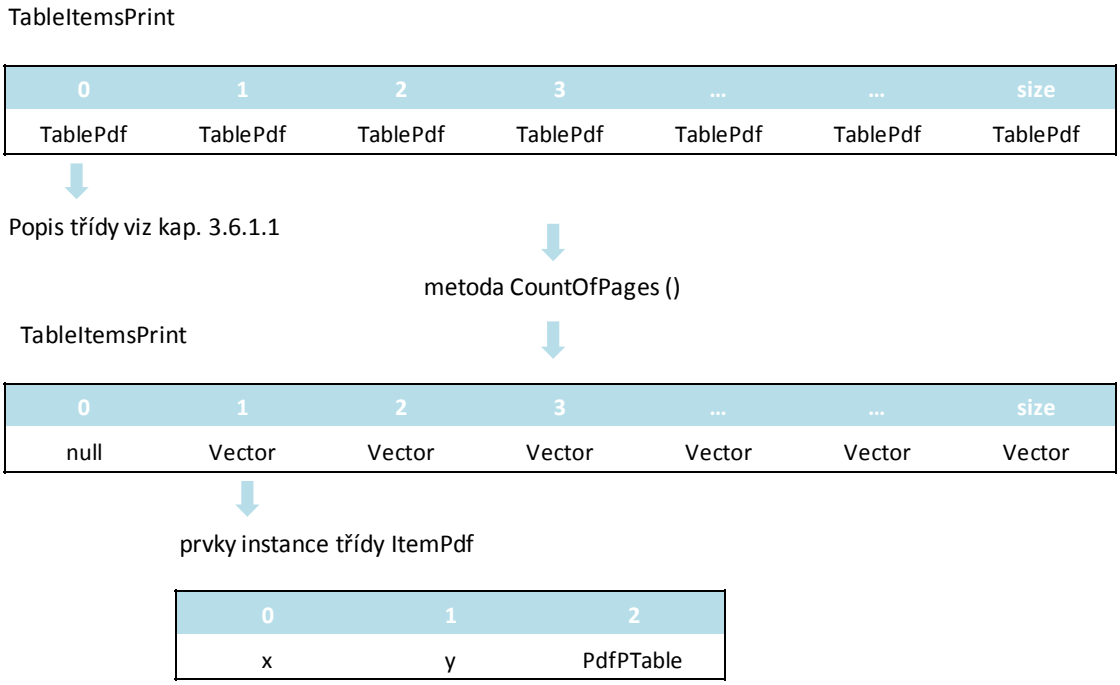
Princip zjednodušeně: na první místo vektoru uložím tabulku s hodnotou parametru `location` - `absolute` nebo `relative`. V následném cyklu za ni řadím všechny ostatní tak jak na sebe formálně navazují. Pokud by tabulky nebyly takto setříděny, mohlo by se stát že budou jejich souřadnice vypočítávány v nesprávném pořadí, tedy bude vytvářena Table3 před Table2 což povede k logické chybě – nemohu umístit tabulku k ještě

neexistujícímu předchůdci. S takto upraveným vektorem tabulek vstupuji do následující metody:



Obr. 38 Metoda buildPrintRelativeTabels (Vector TableItems, FormStruct r_formStruct, Hashtable tables)

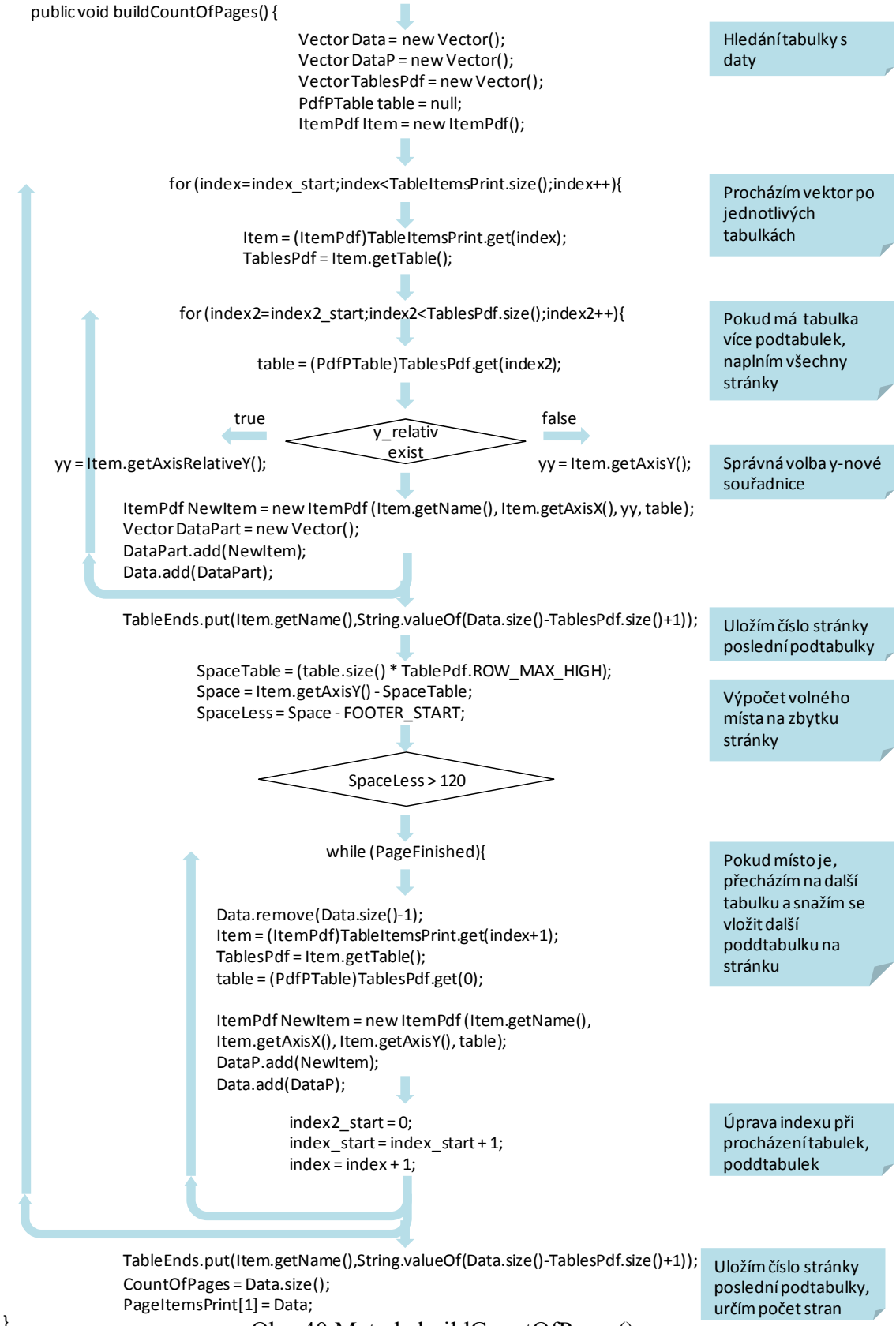
Nejdůležitější částí tvorby dokumentu je správné určení počtu stran. Pokud určíme menší počet stran, nebudou zobrazena všechna data, pokud větší, vzniknou nám prázdné stránky atd. Počet stran odvozujeme z celkového umístění tabulek na stránkách dokumentu, následující metoda přiřadí konkrétní tabulky jednotlivým stránkám dokumentu a vytvoří z nich konečnou podobu tiskového vektoru tabulek. Podobu vektoru před a po zpracování metodou ukazuje následující obrázek:



Obr. 39 Tisková forma tiskového vektoru tabulek TableItemsPrint

Výsledný tiskový vektor je tedy nachystán stejně jako ostatní prvky v podobě pouze převzetí dat a předání příslušným metodám knihoven iText k vykreslení, takto připravená data již nejsou nijak upravována. Velikost vektoru udává celkový počet stránek dokumentu. Popis metody následuje:

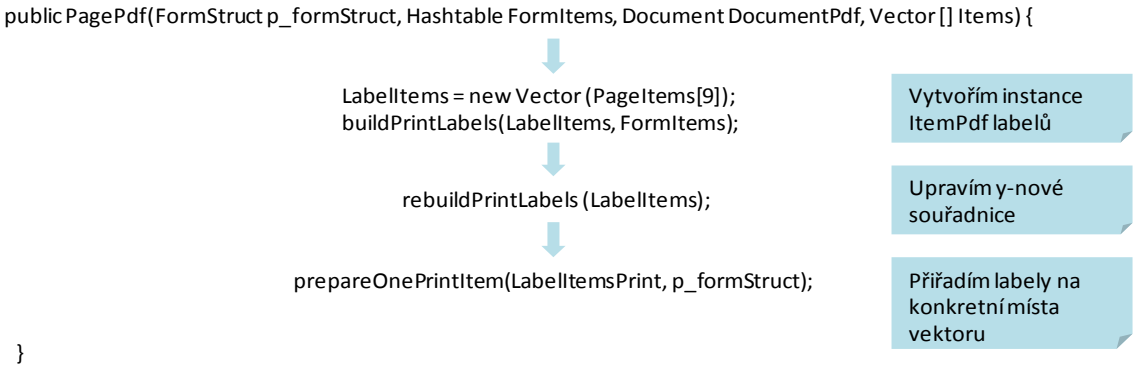
buildCountOfPages()



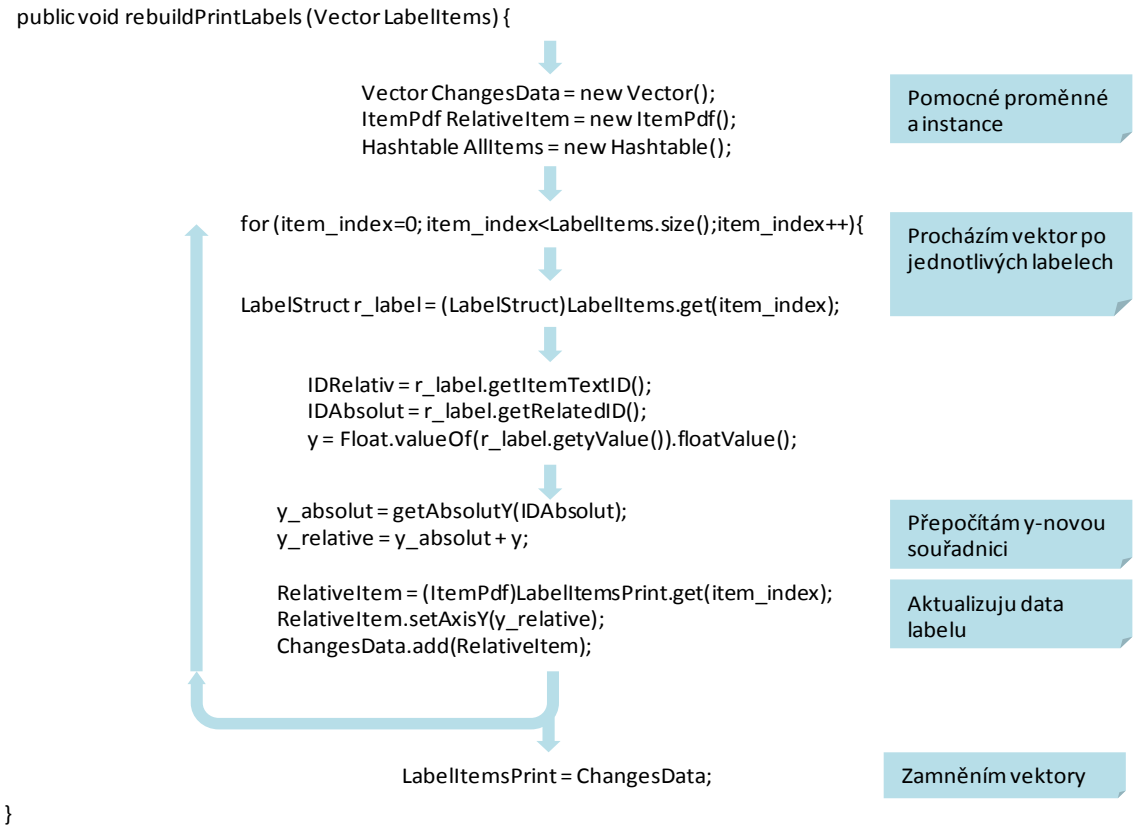
Obr. 40 Metoda buildCountOfPages()

Druhý konstruktor třídy PagePdf slouží pro vytvoření struktury stránky typu `one`. Z této struktury je v současné verzi využita pouze část tiskových vektorů labelů a to pro zobrazení nadpisů tabulek u formuláře Objednávky. Struktura tohoto tiskového vektoru byla vytvořena tak, aby jeho prvky odpovídaly jednotlivým stránkám dokumentu s formálním posunutím indexu o jeden, tak aby prvek(1) odpovídal první stránce atd.

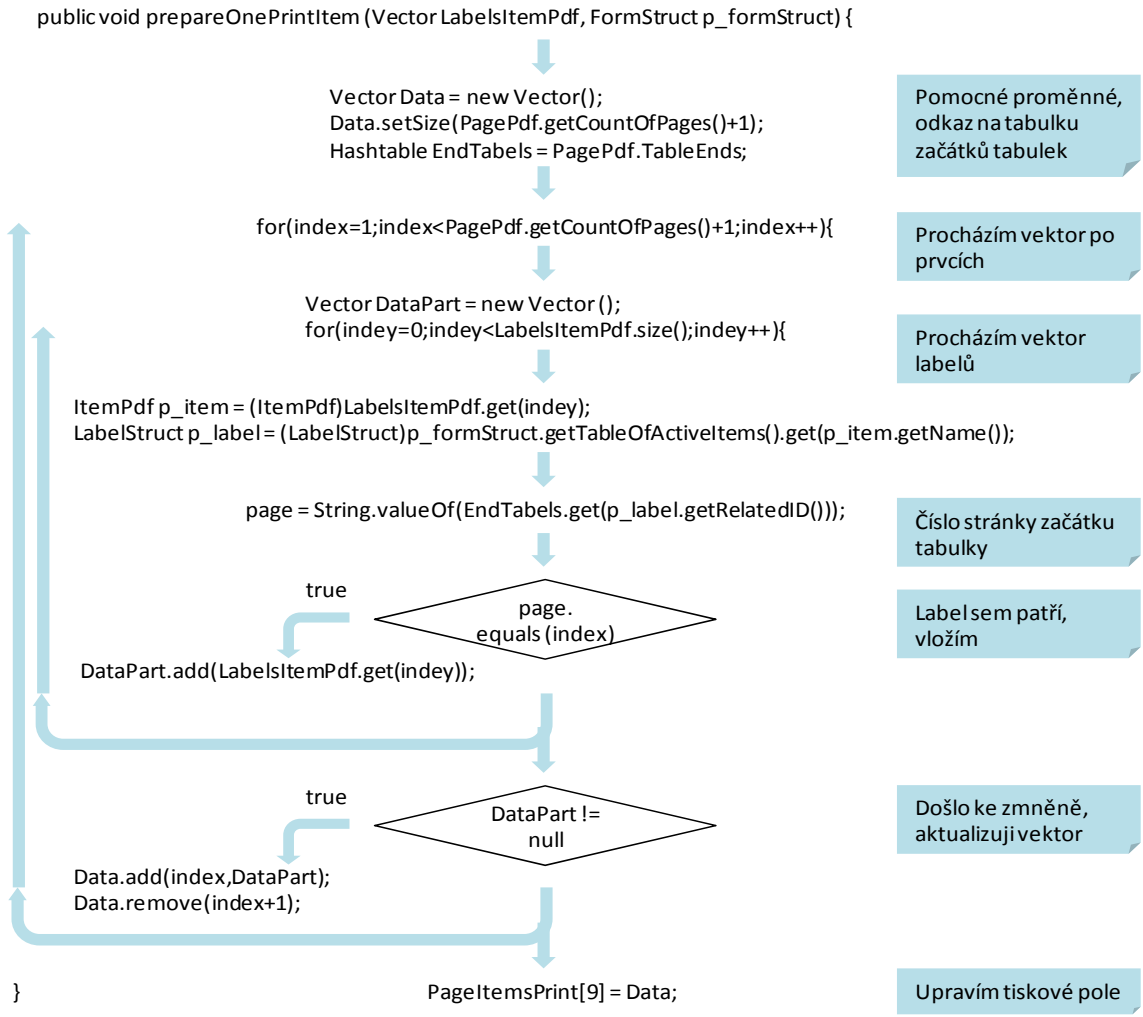
Konstruktor a jeho metody:



Obr. 41 Konstruktor třídy PagePdf.java



Obr. 42 Metoda rebuildPrintLabels (Vector LabelItems)



Obr. 43 Metoda prepareOnePrintItem (Vector LabelsItemPdf, FormStruct p_formStruct)

Výsledný tiskový vektor labelů struktury stránky má následující podobu:

LabelItemsPrint

PageNumber	1	2	3	...	...	LastPage
null	Vector	Vector	Vector	Vector	Vector	Vector

prvky instance třídy ItemPdf

0	1	2
x	y	String

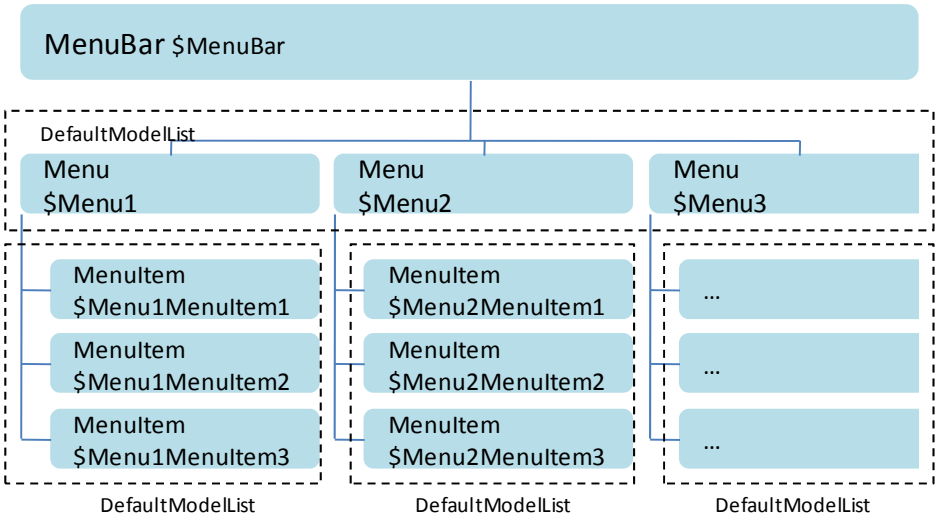
Obr. 44 Struktura tiskového vektoru LabelItemsPrint stránky typu `one`

Při samotném tisku, resp. umísťování těchto prvků na stránky dokumentu, vybíráme podle čísla stránky konkrétní prvky vektoru, pokud existují.

Pozn.: Při výpočtu y souřadnice těchto nadpisů tabulek, byl obrácen výpočet relativní souřadnice. U všech ostatních prvků, pokud má prvek hodnotu parametru type tagu `location` rovnu `relative`, je y = y_absolut – yValue. V tomto případě je to y = y_absolut + yValue a prvky(nadpisy) se tedy odkazují umístěním na své tabulky kterým náleží.

4 PRVEK MENU

Formuláře které obsahují velké množství funkcí a jsou ovládány pouze tlačítky se mohou stát nepřehlednými. Proto je vhodnější některé z těchto funkcí přiřadit prvku MenuBar (panel menu) a jeho přídavným položkám. Struktura prvku MenuBar odpovídá struktuře komponenty JMenuBar do níž jsou přidávány hlavní položky JMenu a položky roletového menu při události onClick na položku menu – JMenuItem. Část přidání prvku MenuBar a jeho součástí do formuláře zpracoval ve své diplomové práci Ing. Petr Svoboda. Obecné schéma stromové struktury menu vypadá následovně:



Obr. 45 Obecné schéma stromové struktury prvku Menu

Mým úkolem v této diplomové práci bylo vytvoření programových struktur prvků a jejich zahrnutí do systému. Třídy popisující tyto struktury jsou:

- *MenuBarStruct.java* - pro popis prvku MenuBar
- *MenuStruct* - pro popis hlavních položek Menu
- *MenuItemStruct* - pro popis jednotlivých položek Menu

Všechny tyto třídy byly přidány k ostatním již existujícím do balíčku `xerus.structures.form`. Programové zpracování obsluhujících tříd:

- *MyMenuBar* - programová obsluha struktury MenuBar
- *MyMenu* - programová obsluha struktury MenuStruct
- *MyMenuItem* - programová obsluha struktury MenuItemStruct

Příklad zápisu prvku „menu“ v souboru xml je následující:

```
<form>
...
<items>
    <menubar>
    <bounds x="0" y="0" width="350" height="20"/>
    <itemTextID>$menubar</itemTextID>
      <description>Menu</description>
      <menu>
            <itemTextID>$menu0</itemTextID>
            <text>menu0</text>
            <menuitem type="none">
                  <description>
                  </description>
                  <itemTextID>
                  $podmenu0
                  </itemTextID>
                  <text>
                  podmenu0
                  </text>
                  <onLoad>
                  </onLoad>
                  <onClick>
                  </onClick>
            </menuitem>
      </menu>
      </menubar>
</items>
...
</form>
```

V současné chvíli dosahují položky menu první úrovně, tak jak znázorňuje stromová struktura viz. Obr. 45. Nelze tedy obsluhovat menu kde položka menu má svou další položku.

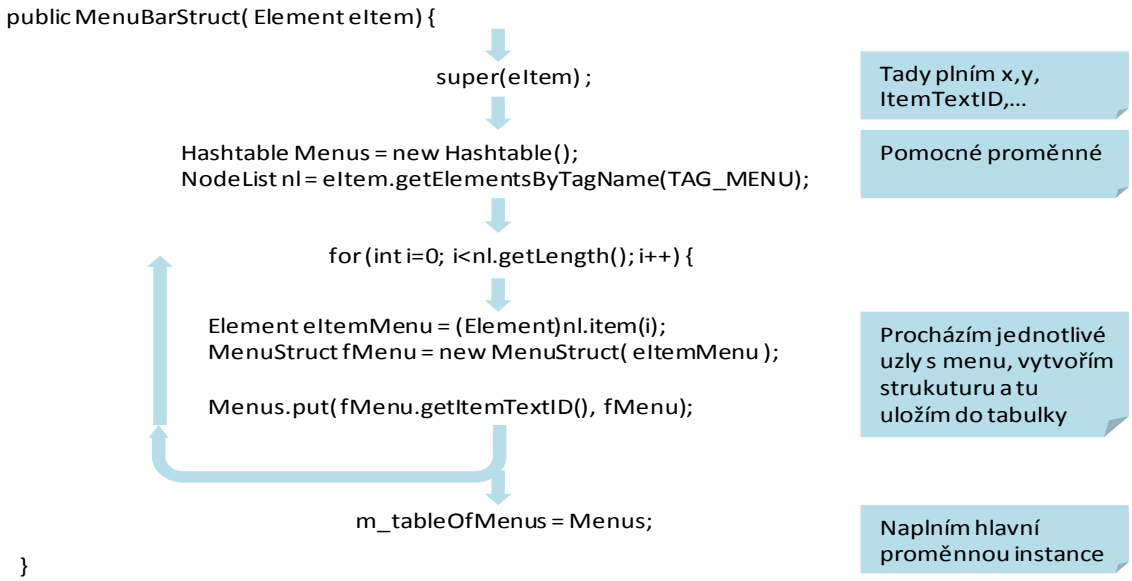
U popisu jednotlivých tříd struktur uvedu pouze schémata nejdůležitějších metod, či konstruktorů které vytváří hlavní proměnné. Vytváření ostatních proměnných instancí pomocí rozhraní `java.io.Serializable` již bylo mnohokrát popsáno v jiných diplomových pracích, uvedu jen příklad načtení hodnoty tagu `text`:

```
m_text = MyXMLFactory.getTagStringValue(eMenu, this.TAG_TEXT);
```

4.1 Třídy popisující strukturu prvku

MenuBarStruct.java

Hlavní proměnnou je hashtable m_tableOfMenus, tabulka která v sobě uchovává struktury jednotlivých menu, klíčem je textové ID menu, hodnotou jeho struktura MenuStruct. Třída je potomkem třídy ItemStruct a implementuje rozhraní `java.io.Serializable`. Tabulka se plní v konstruktoru třídy.

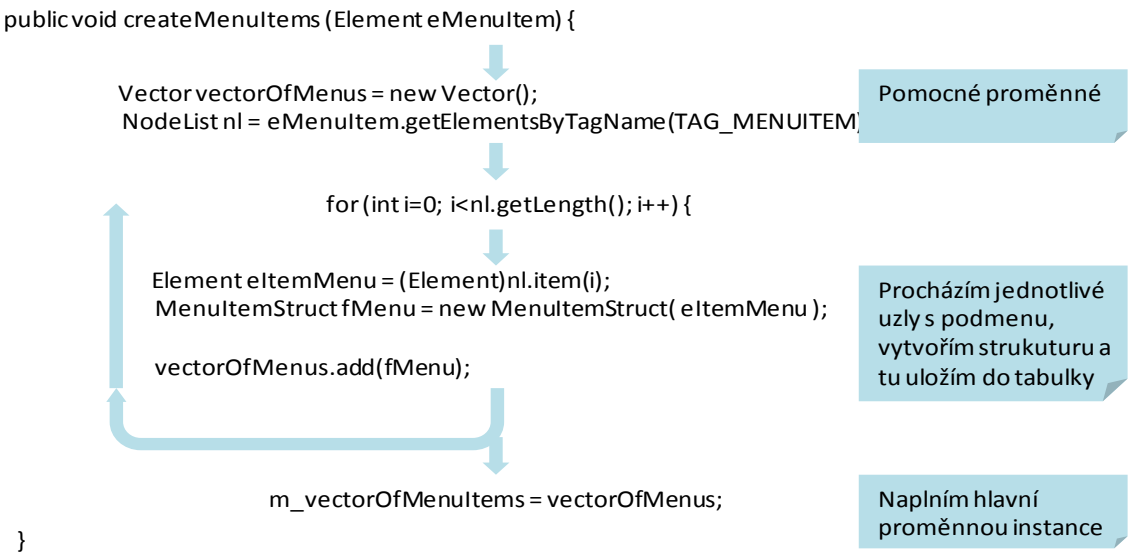


Obr. 46 Konstruktor třídy MenuBarStruct.java

V prvním kroku vytvořím pomocnou hashtable pro ukládání průběžných hodnot prvků menu a určím počet položek menu. Postupně pak procházím jednotlivé uzly menu a vytvářím jejich struktury. Výsledky ukládám a na konci pak pouze přiřadím proměnné instance hodnotu pomocné hashtable.

MenuStruct.java

Hlavní proměnnou je m_vectorOfMenuItems vektor jednotlivých položek menu (MenuItemStruct) patřících vytvářenému menu (MenuStruct). Vektor se plní v metodě createMenuItems () (Element eMenuItem). Ostatní proměnné instance jsou naplněny v konstruktoru třídy. Tato třída je také potomkem třídy ItemStruct a implementuje rozhraní `java.io.Serializable`.



Obr. 47 Metoda createMenuItems (Element eMenuItem)

Pro vytvoření pomocných proměnných procházím uzly jednotlivých položek menu a vytvářím jejich instance třídy MenuItemStruct, hodnoty ukládám do pomocného vektoru který na závěr předám proměnné instance.

MenuItemStruct.java

Je z výše uvedených tříd nejjednodušší, v konstruktoru třídy pouze plním jednotlivé proměnné instance (události, ID, text…). Příklad takového zpracování:

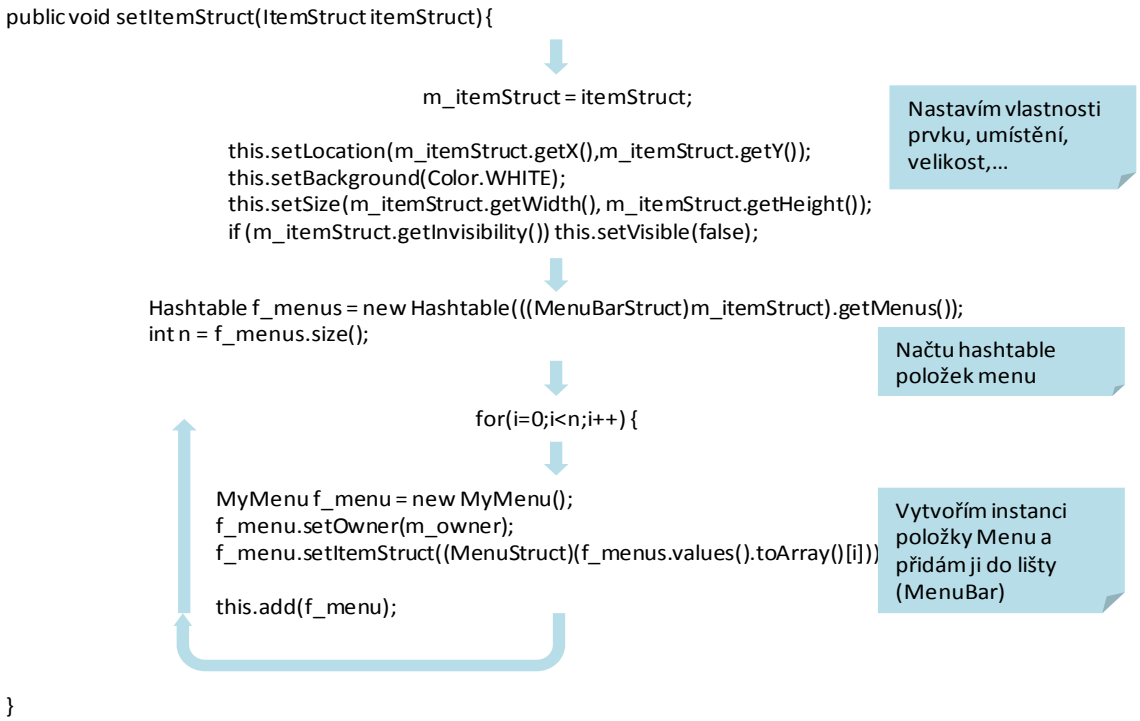
```
m_text = MyXMLFactory.getTagStringValue(eMenu, this.TAG_TEXT);
```

Stejně tak jako u předchozích tříd, nebudu popisovat jednotlivé třídy podrobně, jen uvedenu nejdůležitější metody, popř. konstruktory a hlavní proměnné.

4.2 Třídy popisující programovou obsluhu

MyMenuBar.java

Třída je potomkem třídy JMenuBar knihovny swing a implementuje rozhraní IFormActiveItem. Má za úkol z hodnot vytvořených třídou MenuBarStruct vytvořit aktivní prvek formuláře a nastavit jeho vlastnosti. Toto vše je věcí konstruktoru a metody setItemStruct () která naplní MenuBar položkami menu, viz. následující schéma a popis:

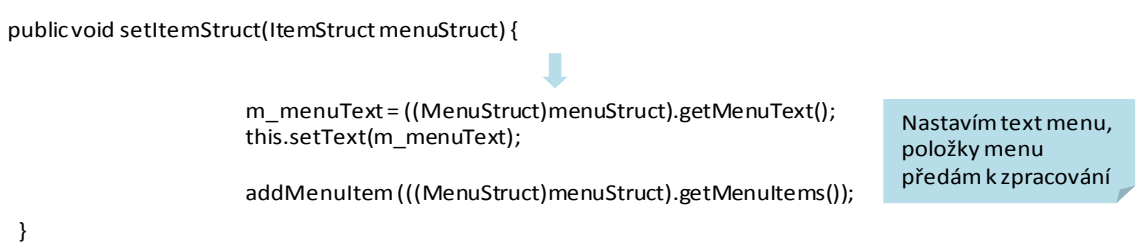


Obr. 48 Metoda setItemStruct (ItemStruct itemStruct)

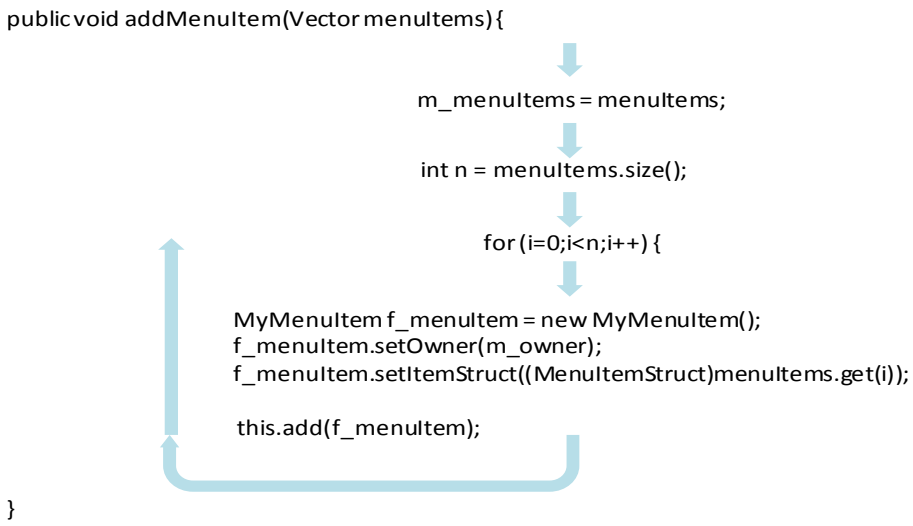
Většina nastavovaných vlastností prvku jsou hodnoty převzaté z formuláře xml, mimo barvu. Tato položka nebyla ještě uvolněna pro možnost volby uživatelem.

MyMenu.java

Třída je potomkem třídy JMenu knihovny swing a implementuje rozhraní IFormActiveItem. Instance této třídy tvoří jednotlivé položky menu třídy MyMenuBar. Položky menu samotného, jsou instance třídy MyMenuItem popsané v další části. Tyto položky jsou vytvářeny v metodě addMenuItem (), která naplní Menu položkami MenuItem, viz následující schéma a popis:



Obr. 49 Metoda setItemStruct (ItemStruct menuStruct)

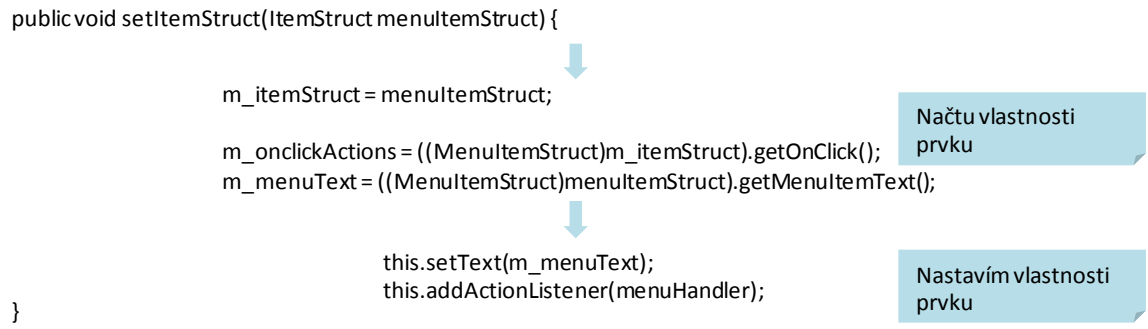


Obr. 50 Metoda addMenuItem (Vector menuItems)

Položkám menu je v současné chvíli nastavována pouze vlastnost text, vyčítaná ze struktury xml.

MyMenuItem.java

Třída je potomkem třídy JMenuItem knihovny swing a implementuje rozhraní IFormActiveItem. Instance této třídy tvoří jednotlivé položky menu třídy MyMenu. Hlavní částí je metoda setItemStruct (ItemStruct menuItemStruct).



Obr. 51 Metoda setItemStruct (ItemStruct menuItemStruct)

Stejně jako u rodičovského menu, i jednotlivým položkám je v současné chvíli přiřazována pouze vlastnost `text`.

Pozn.: Jelikož prvky Menu, MenuItem nejsou zahrnuty v tabulce aktivních prvků formuláře i když implementují rozhraní IFormActiveItem, musel být vlastník (owner) těchto prvků nastaven samostatně a také byl vytvořen samostatný „poslouchač“ (Listener) položek menu k obsloužení údálostí onClick.

Prvek „menu“ zatím nebyl implementován v žádném z formulářů, mimo formuláře testovacího k ověřená funkčnosti. Také některé tagy popisující tento prvek v souboru xml nelze definovat v editoru formulářů. Rozvoj tohoto prvku, definování nových vlastností či použití může být předmětem dalších diplomových prací.

ZÁVĚR

Hlavním cílem této diplomové práce byl návrh a implementace programového vybavení rozšiřujícího informační systém nástrojového hospodářství ve strojírenském podniku Tajmac-ZPS Zlín, a.s. o možnosti tisku formulářů a vícestránkových sestav a využívání prvku typu „menu“. Všechny uvedené požadavky zde byly vyřešeny.

Programové vybavení spolu s tiskovými strukturami dat jsem se snažil vytvořit co nejobecněji aby jej bylo dále možné jednoduše rozšířit dle nových požadavků a připomínek pracovníků firmy Tajmac-ZPS Zlín, a.s. Je podporován tisk všech základních prvků které byly požadovány, jako label, table, border a iconLabel a lze vytvářet více stránkové sestavy či formuláře s více než jednou tabulkou. Bylo také vytvořeno a odzkoušeno několik tiskových formulářů. Po implementaci těchto formulářů a programového vybavení byl do systému zahrnut a odzkoušen prvek typu „menu“.

V závěrečné části jsem zpracoval jednoduchou příručku pro snadnou orientaci uživatele při tvorbě těchto formulářů, s vysvětlením jednotlivých tiskových tagů, jejich možných hodnot a funkcí v systému.

SEZNAM POUŽITÉ LITERATURY

[1] VAŘECHA, M. *Programové vybavení pro evidenci nářadí ve strojírenské výrob*ě – *vzdálená správa databáze nářadí*. Zlín, 2003. 68 s. Diplomová práce na IIT FT UTB

[2] VERBOVSKÝ, J. Programové vybavení pro evidenci nářadí ve strojírenské výrobě – komunikace uživatele se systémem. Zlín, 2003. 58 s. Diplomová práce na IIT FT UTB

[3] SVOBODA, M. *Rozšíření programového systému pro správu nástrojového hospodářství ve strojírenské výrobě.* Zlín, 2004. 93 s. Diplomová práce na IIT FT UTB

[4] SVOBODA, P. *Rozšíření programového systému pro správu nástrojového hospodářství ve strojírenské výrobě.* Zlín, 2006. 82 s. Diplomová práce na IIT FAI UTB

[5] STEPHENS, R. K, PLEW R. R. *Naučte se SQL za 21 dní.* Praha: Computer Press, 2004. 581s. ISBN 80-722-6870-8

[6] SPELL, B. *Java Programujeme profesionálně.* Praha: Computer Press, 2002. ISBN 80-72266-67-5

[7] GRUSOVÁ, L. *XML pro úplné začátečníky.* Praha. Computer Press, 2002. ISBN 80-7226-697-7

[8] *Informační systémy.* URL: <http://en.wikipedia.org/wiki/Information_systems>

[9] *Informace.* URL: <http://www.cojeco.cz/informace>

[10] *iText.* URL: <http://www.lowagie.com/iText>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

IS	Informační systém
JDBC	Java Database Connectivity
JVM	Java Virtual Machine, virtuální stroj jazyka Java
NAHOS	Nástrojové hospodářství
OHN	Oblast hospodaření s nářadím
RMI	Remote Method Invocation, vzdálené volání metod
SDK	Standard Development Kit, standardní vývojový balík
SQL	Structured Query Language, strukturovaný dotazovací jazyk
SŘBD	Systém Řízení Báze Dat
XML	eXtensible Markup Language, rozšiřitelný značkovací jazyk

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I METODY KNIHOVNY I TEXT PRO TISK PRVKŮ

Hlavní částí tvorby dokumentů je objekt: `cb` jako instance třídy `public class PdfContentByte`. Pomocí tohoto objektu a jeho metod umísťujeme jednotlivé prvky na stránky dokumentu pdf.

```
/**
 * <CODE>PdfContentByte</CODE> is an object containing the user
 * positioned text and graphic contents of a page. It knows how to apply
 * the proper font encoding.
 */
public class PdfContentByte
```

metody pro tisk (umístění) labelů:

```
/**
 * Starts the writing of text.
 */
public void beginText()

/**
 * Changes the text matrix. The first four parameters are {1,0,0,1}.
 * <P>
 * Remark: this operation also initializes the current point
 * position.</P>
 * @param        x                operand 3,1 in the matrix
 * @param        y                operand 3,2 in the matrix
 */
public void setTextMatrix(float x, float y)

/**
 * Shows the <CODE>text</CODE>.
 *
 * @param text the text to write
 */
public void showText(String text)

/**
 * Ends the writing of text and makes the current font invalid.
 */
public void endText()
```

metody pro tisk (umístění) tabulek, tabulka je instancí třídy PdfPTable

```
/** This is a table that can be put at an absolute position but can also
 * be added to the document as the class <CODE>Table</CODE>.
 * In the last case when crossing pages the table always break at full
 * rows; if a row is bigger than the page it is dropped silently to avoid
 * infinite loops. PdfPTableEvent can be associated to the table to do
 * custom drawing when the table is rendered. @author Paulo Soares
 */
public class PdfPTable implements Element

/**
 * Writes the selected rows to the document.
 * @param rowStart the first row to be written, zero index
 * @param rowEnd the last row to be written + 1. If it is -1 all the
 * rows to the end are written
 * @param xPos the x write coodinate
 * @param yPos the y write coodinate
 * @param canvas the <CODE>PdfContentByte</CODE> where the rows will
 * be written to
 * @return the y coordinate position of the bottom of the last row
 */
public float writeSelectedRows(int rowStart, int rowEnd, float xPos,
float yPos, PdfContentByte canvas)
```

metody pro tisk (umístění) rámečků

```
/**
 * Changes the <VAR>line width</VAR>.
 * The line width specifies the thickness of the line used to stroke
 * a path and is measured in used space units.
 * @param   w     a width
 */
public void setLineWidth(float w)


/**
 * Adds a rectangle to the current path.
 * @param         x           x-coordinate of the starting point
 * @param         y           y-coordinate of the starting point
 * @param         w           width
 * @param         h           height
*/
```

```
public void rectangle(float x, float y, float w, float h)
/**
 * Strokes the path.
 */
public void stroke()
```

metody pro tisk (umístění) obrázků

```
/**
  * Adds an <CODE>Image</CODE> to the page. The positioning of the
  * <CODE>Image</CODE>
  * is done with the transformation matrix. To position an
  * <CODE>image</CODE> at (x,y)
  * use addImage(image, image_width, 0, 0, image_height, x, y).
  * @param image the <CODE>Image</CODE> object
  * @param a an element of the transformation matrix
  * @param b an element of the transformation matrix
  * @param c an element of the transformation matrix
  * @param d an element of the transformation matrix
  * @param e an element of the transformation matrix
  * @param f an element of the transformation matrix
  * @throws DocumentException on error
  */
public void addImage(Image image, float a, float b, float c, float d,
float e, float f) throws DocumentException
```

PŘÍLOHA P II: SOUHRNNÝ POPIS TISKOVÝCH TAGŮ FORMULÁŘE

Popis slouží jako příručka pro snadnou orientaci navrhovatele tiskových formulářů a jako návod k jejich tvorbě. U popisu každého prvku či formuláře samotného nejdříve uvedu strukturu tiskových tagů v xml a pak její popis a možné hodnoty jednotlivých parametrů. V současné verzi jsou v tiskových formulářích využity tyto prvky: label, table, border, iconLabel (obrázek).

Formulář

```
<print>
    <fileName>Obj</fileName>
    <pageSize>A4</pageSize>
    <pageOrientation>portrait</pageOrientation>
    <pageSection header="840" body="730" footer="50"/>
</print>
```

<*fileName*> tiskové jméno formuláře, může se přidat k pojmenování dokumentu ve formátu pdf, pokud se tato položka uvolní v metodě buildFileName(…)

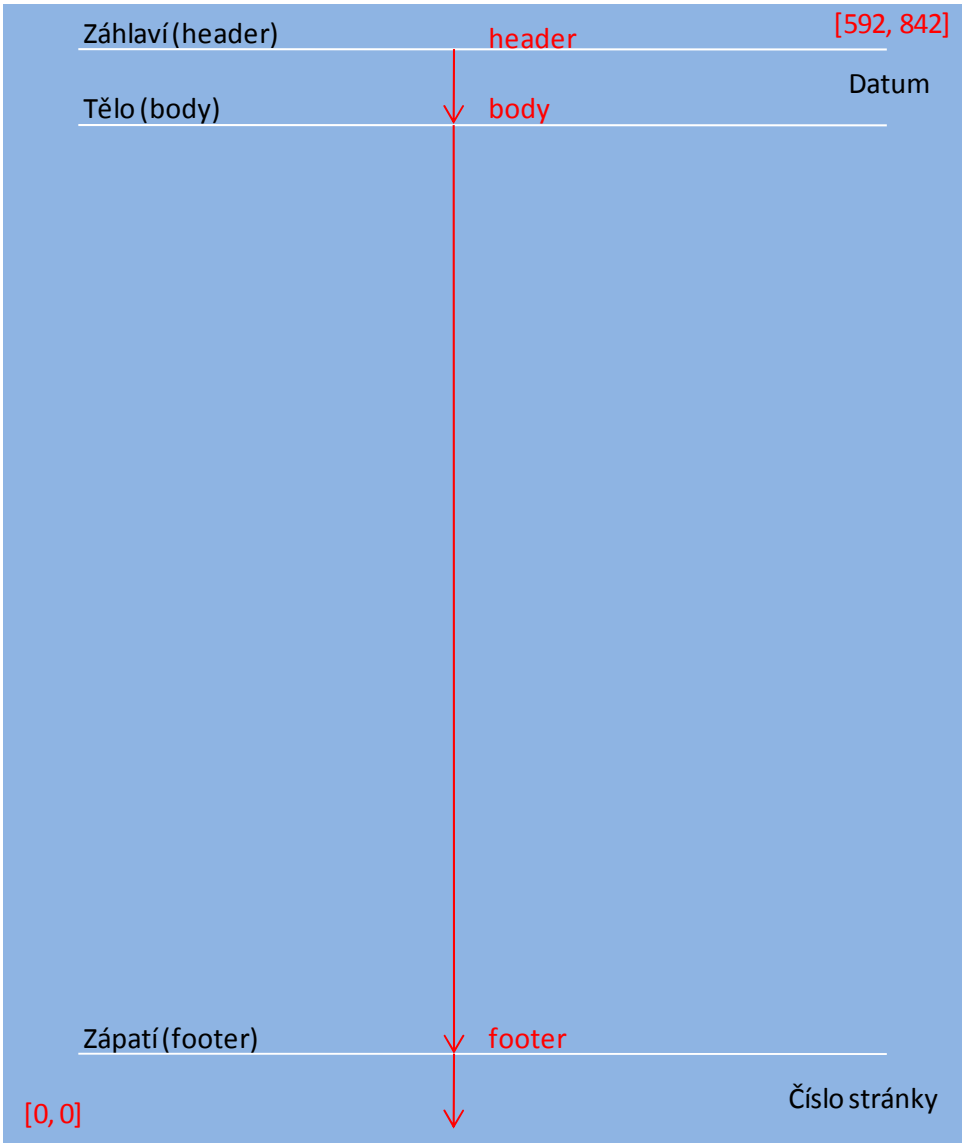
<*pageSize*> formát stránek dokumentu, možné hodnoty:

- A4
- A3

<*pageOrientation*> orientace stránek v dokumentu, možné hodnoty:

- portrait
- landscape

<*pageSection*> hodnoty začátku jednotlivých sekcí stránky dokumentu, viz následující obrázek:



Z praktického použití formulářů a jejich návrhu jsou v současné chvíli tyto začátky sekcí nastaveny následovně:

- pro formát stránky A4: header="840" body="730" footer="50"
- pro formát stránky A3: header="840" body="780" footer="50"

tyto hodnoty byly také odvozeny z velikostí jednotlivých formátů stránek (v pixelech)

- formát A4: 592 x 842
- formát A3: 1192 x 842 (orientace landscape)

Společné tiskové tagy pro prvky label, table, border, iconLabel

prvek umístěný absolutně

```
print>
    <page type="all">
        <pageSection type="header">
            <location type="absolute" unit="pixel"
            yValue="30" />
        </pageSection>
    </page>
</print>
```

prvek umístěný relativně

```
<print>
    <page type="one">
        <pageSection type="body">
            <location type="relative" relatedID="$tbl_objNab_1"
            yValue="15"/>
        </pageSection>
    </page
</print>
```

<page type=“ “> typ stránky na které bude prvek umístěn

- first - prvek bude umístěn jen na první stránce dokumentu
- all - na všech
- last - na poslední
- one - na jedné jediné

<pageSection type=""> sekce ve které se bude prvek nacházet

- header - v záhlaví
- body - v těle
- footer - v zápatí

<*location type=""*> typ umístění

- absolute - yValue je posunutí od začátku zvolené sekce
- relative - yValue je posunutí od jiného prvku
- relatid - speciální parametr pro umístění tabulek k jiné tabulce

<*location relatedID=""*> textové ID prvku ke kterému prvek umísťujeme

<*location yValue=""*> hodnota posunutí y-nové souřadnice

Specifické tiskové tagy

Table

```
<columns>
    <column title="Kód" width="105" align="left" format="" printSuma=""/>
    ...
</columns>
<print>
    <page grid ="false">
        <pageSection type="body">
            <location type="absolute" relatedID="$tbl_objNab_1" unit="pixel"
            yValue="0"/>
             <fontp face1="Arial" size1="8" face2="Arial" size2="7"/>
        </pageSection>
    </page>
</print>
```

<*column printSuma=""/>* zobrazení součtu hodnot řádků sloupce tabulky za stránku

- true
- false - platí i při neuvedení hodnoty parametru

<*page grid =""*> zapnutí, vypnutí ohraničení tabulky

- true
- false

<*fontp face1="" size1=""*> volba fontu a velikosti písma záhlaví tabulky

<*fontp face2="" size2=""*> volba fontu a velikosti písma dat tabulky

Border

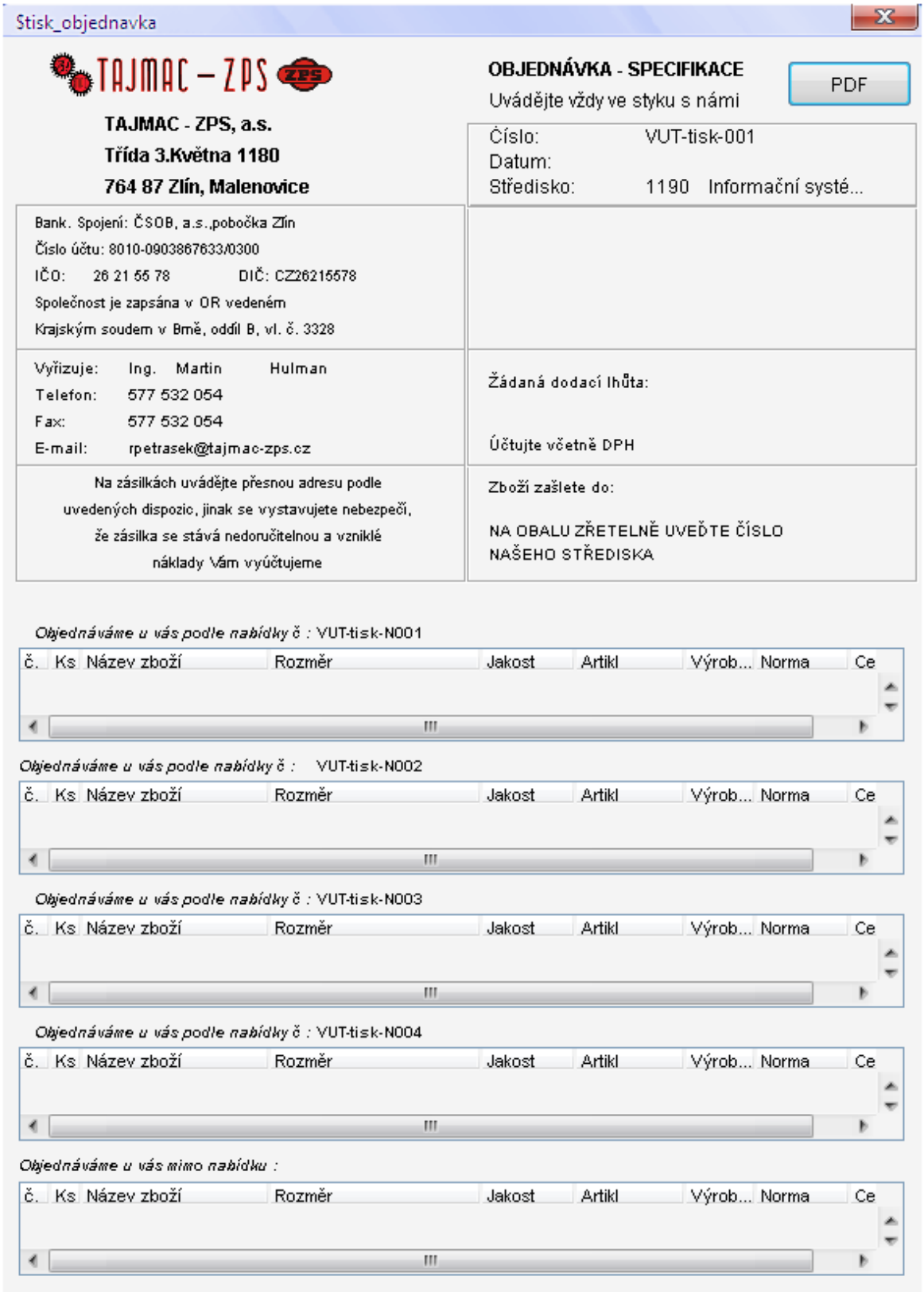
<*properties lineWidth*> hodnota tloušťky rámečku, parametr je umístěn mimo ostatní tiskové tagy

IconLabel

<*iconPath*> cesta k umístění obrázku, např.:

- Q:/!zps_act/images/ZPS_logo.gif

PŘÍLOHA P III: NÁHLED TISKOVÉHO FORMULÁŘE



PŘÍLOHA P IV: NÁHLED DOKUMENTU VE FORMÁTU PDF

TAJMAC – ZPS

OBJEDNÁVKA - SPECIFIKACE

Uvádějte vždy ve styku s námi

TAJMAC - ZPS, a.s.
Třída 3.Května 1180
764 87 Zlín, Malenovice

Číslo: VUT-tisk-001
Datum:
Středisko: 1190 Informační systémy

Bank. Spojení: ČSOB, a.s.,pobočka Zlín
Číslo účtu: 8010-0903867633/0300
IČO: 26 21 55 78 DIČ: CZ26215578
Společnost je zapsána v OR vedeném
Krajským soudem v Brně, oddíl B, vl. č. 3328

Vyřizuje: Ing. Martin Hulman
Telefon: 577 532 054
Fax: 577 532 054
E-mail: rpetrasek@tajmac-zps.cz

Žádaná dodací lhůta:
Účtujte včetně DPH

Na zásilkách uvádějte přesnou adresu podle uvedených dispozic, jinak se vystavujete nebezpečí, že zásilka se stává nedoručitelnou a vzniklé náklady Vám vyúčtujeme

Zboží zašlete do:
NA OBALU ZŘETELNĚ UVEĎTE ČÍSLO NAŠEHO STŘEDISKA

Objednáváme u vás podle nabídky č : **VUT-tisk-N001**

č.	Ks	Název zboží	Rozměr	Jakost	Artikl	Výrobce	Norma	Cena/ks
1	5	VBD	11 IRM A 60	IC 250		ISCAR		150.0
2	5	VBD	11 IRM A 60	IC 220		ISCAR		0.0
3	5	VBD	11 IRM 1.50 ISO	IC 220		ISCAR		0.0
4	5	VBD	11 IR 2.00 ISO	IC 908		ISCAR		0.0
5	5	VBD	11 IR 1.50 ISO	IC 908		ISCAR		0.0
6	5	VBD	11 IR 1.50 ISO	IC 250		ISCAR		0.0
7	5	VBD	11 IR 1.00 ISO	IC 220		ISCAR		0.0
8	5	VBD	11 IL 2.00 ISO	IC 228		ISCAR		1300.0
9	5	VBD	11 IL 1.50	IC 908		ISCAR		2600.0
10	5	VBD	08 UIRL 2.00 ISO	IC 228		ISCAR		0.0

Objednáváme u vás podle nabídky č : **VUT-tisk-N002**

č.	Ks	Název zboží	Rozměr	Jakost	Artikl	Výrobce	Norma	Cena/ks
1	1	MEZNI DRAZKOVY KALIBR - 4	16H7X20H11X6F10			není		0.0
2	1	MEZNI DRAZKOVY KALIBR - 4	18H7X22H11X6D9			není		168.0
3	1	MEZNI DRAZKOVY KALIBR	28H7X32H11X7F10			není		358.0
4	1	MEZNI DRAZKOVY KALIBR	32H7X38H11X8F10			není		0.0
5	1	MEZNI DRAZKOVY KALIBR	36H7X42H11X8D9			není		0.0
6	1	MEZNI DRAZKOVY KALIBR	42H7X48H11X10F1			není		0.0

Page-1/3