Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Konfigurovatelný systém pro ukládání a transformaci různorodých dat městské silniční dopravy

Plzeň, 2014 Jakub Zíka

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne … podpis …

Poděkování

Chtěl bych poděkovat především panu Ing. Tomáši Potužákovi, PhD., za jeho odborné rady, poskytnuté materiály a konstruktivní připomínky při vedení této práce.

Abstract

The submitting bachelor work deals with making configurable system for saving and transforming heterogeneous data of city traffic. The main goal was to make a program which will load a file (textual or XML) with data of city traffic. We suppose that files containing the same data can have, to some extent, different alignment and different key words, that initiate this data.

Abstrakt

Předkládaná bakalářská práce se zabývá vytvořením konfigurovatelného systému pro ukládání a transformaci různorodých dat městské silniční dopravy. Hlavním cílem práce bylo vytvoření programu, jenž načte soubor (textový nebo XML) s daty o silniční dopravě. Předpokládá se, že soubory obsahující stejná data mohou mít do jisté míry různé uspořádání a různá klíčová slova uvozující tato data.

Obsah

1 Úvod

Městská silniční doprava je zdrojem různorodých dat, která lze využít v detailní simulaci dopravy (např. množství vozidel, které projelo dopravním pruhem v křižovatce, množství chodců na přechodech, množství lidí na zastávkách MHD atd.) Ačkoliv lze některá tato data za určitých okolností získávat automaticky (např. počet vozidel, která projela dopravním pruhem), mnohé z nich je nutno zaznamenávat ručně přímým pozorováním, případně zpětným pozorováním obrazového záznamu. Výsledkem tohoto zaznamenávání mohou být záznamy na papíře, textové soubory, `xml` soubory, excelovské soubory a podobně. Pokud jsou navíc tato data získána z různých zdrojů a od různých pracovníků, je možné, že mají různorodou strukturu (různé pořadí informací, různá klíčová slova uvozující informace), i když obsahují podobné informace.

Hlavním cílem práce je vytvoření takového programu, který dokáže tato data číst, rozpoznat a správně roztřídit z textových a `xml` souborů s různou strukturou. Důraz je kladen na správné rozeznání dat a jejich bezproblémové načtení i v případě neúplnosti. Neúplností je myšleno, že není například zaznamenáno, zda silnice má semafor nebo autobusovou zastávku. To znamená, že program nesmí vyžadovat striktní posloupnost dat, kterou by musel autor souboru při záznamu dodržovat.

Program bude získaná data ukládat do vnořené databáze, která, jak napovídá název, je součástí programu. Z toho vyplývá, že načtená data musí být sjednocena do jednotného formátu a datových typů tak, aby došlo k bezproblémovému uložení do databáze. Jednotnost také slouží k přehlednosti dat při jejich pozdějším zobrazení. Program bude umět data i exportovat. Export bude do souboru `xml` a bude mít vždy stejnou strukturu.

Samotný text bude členěn do sedmi kapitol. Druhá se zabývá možností ukládání dat s využitím vnořených databází. Ve třetí kapitole jsou uvedena data, jež lze v silniční dopravě zjišťovat a měřit. Ve čtvrté a páté kapitole je popis vytvoření systému pro ukládání různorodých dat silniční dopravy. Šestá kapitola obsahuje výsledky testování systému testovacími daty. Sedmá kapitola je zhodnocením celé práce.

2 Vnořené databáze

2.1 Co jsou vnořené databázové systémy

Klasické systémy řízení báze dat (SŘBD), jak je známe, využívají architekturu *klient-server*. SŘBD, ve kterém jsou veškerá data uložena, se chová jako server a k němu se může vzdáleně připojit jedna či více klientských aplikací, které s daty pracují. Vnořené (embedded) databáze ovšem nabízejí architekturu *in-proces*. Ta nám dává možnost uložit data na disk, do paměti počítače nebo kombinovat dohromady předchozí dvě uložení [1] , [2].

Jak je z předchozího odstavce patrné, vnořená databáze může být přímou součástí programu, jemuž slouží jako strukturované úložiště dat. V další části se zaměříme pouze na vnořené databáze určené pro jazyk Java, kde je spojení mezi databází a programem zajištěno pomocí JDBC [1].

2.2 Důvody využití

Vnořené databáze jsou velice jednoduché, protože poskytují pouze to, co aplikace nezbytně potřebuje. Tím pádem bohužel neoplývají všemi možnostmi jako běžné SŘBD. To ovšem také snižuje jejich velikost. Jelikož jsou malé, snižuje se tím i možnost výskytu *bugů* v našem programu. Protože jsou data většinou uložena na disku nebo v paměti, je přístup k nim rychlejší, než kdybychom se museli připojovat ke vzdálenému serveru [2].

2.3 Odlišnosti od klasického SŘBD

Vnořené databáze nevyužívají žádná přístupová práva k autorizaci uživatele. Důvod je velmi jednoduchý. Uživatel má plný přístup k datům, není tedy důvod je šifrovat [2].

Dalším rozdílem jsou možnosti přístupu. Obvykle vnořené databáze obsahují pouze dva druhy přístupu. Buď *read-only* nebo *read-write*. Význam spočívá v tom, že z databáze může číst více aplikací najednou, ale zapisovat do ní může pouze jedna. Existují vnořené databáze, které využívají jemnější dělení zámků. Nezamykají celé databáze, ale buď uzamykají blok (*page-level lock*) nebo jednotlivé řádky dat (*row-level lock*). Ovšem čím jsou zamykané bloky menší, tím je přístup složitější a to vede k celkovému zpomalení enginu databáze [2].

Většina vnořených databází nenabízí všechny možnosti dotazování jako *jazyk SQL*. K datům se ve většině případů dostaneme přes spojení *klíč-hodnota*. Lze tedy využít pouze základních přístupů k datům jako je vyhledání dat, vložení nebo úprava dat, smazání či sekvenční procházení [2].

2.4 Přehled použitelných vnořených databází

Vybraná databáze musí splňovat následující požadavky. Musí být použitelná v jazyce Java. Měla by být rychlá a mít možnost ukládat data na disk. V rámci těchto parametrů byly nalezeny databáze H2, HSQLDB, Apache Derby (též nazývána JavaDB) a Firebird [1] , [3].

2.4.1 Firebird

Relační SŘBD vyvíjený vývojáři Firebird Project. Předchůdcem tohoto projektu je InterBase. Ta je nadále vyvíjena samostatně společností Borland. Výhodou Firebird databáze je velmi malá velikost a jednoduché nastavení. Pro připojení lze použít jak *ODBC,* tak JDBC knihovnu, dále jsou k dispozici moduly pro *Python*, *PHP*, *Perl* a další [4]. Firebird lze spustit ve třech módech, a to jako *SuperServer* (sdílí cache paměť mezi jednotlivá spojení a využívá vlákna pro obsluhu každého spojení), *Classic* (jeden nezávislý proces pro každé spojení) *a Embedded* (vnořená databáze v programu). Dále nabízí grafické uživatelské rozhraní pro snadnější manipulaci s daty. Součástí projektu je velké množství utilit, což konečnou velikost navýší na 5MB (bez dat) [5].

2.4.2 H2

H2 je relační databázový manažer, který je psaný v Javě. Název H2 vznikl zkrácením názvu Hypersonic 2. Podle názvu se může zdát, že databáze je rozšířením kódu HSQLDB. H2 je ovšem napsaná samostatně úplně od začátku, zůstává pouze stejný autor Thomas Mueller a podobná funkcionalita. Databáze podporuje jak knihovnu JDBC tak i ODBC [4]. Databáze nabízí podobně jako HSQLDB módy server-klient a in-proces. Je zde k dispozici jednoduché uživatelské prostředí. Na disku nám zabere pouhých 1.5MB (bez dat) [6] , [7]. Po stažení databáze máme k dispozici navíc i webový server [4].

2.4.3 HSQLDB (HyperSQL DataBase)

HSQLDB je další známý databázový manažer. Stejně jako H2 je i HSQLDB psaná v Javě. Je založena na projektu Hypersonic SQL Project, jenž se stále vyvíjí [8].

Databáze, opět jako H2, nabízí dva módy spuštění server-klient a in-proces [9] , [10]. Připojení lze realizovat přes JDBC knihovnu, která je součástí staženého `hsqldb.jar`. Data se v počítači mohou ukládat přímo na disk nebo do paměti RAM (doporučeno pro malá množství dat). Stažený JAR obsahuje HSQLDB Manager – grafické uživatelské rozhraní pro správu databází [11]. V paměti HSQLDB zabírá něco okolo 1.39 MB (bez dat) [9].

2.4.4 Apache Derby

Jako H2 a HSQLDB je i Apache Derby psán v Javě. Tento databázový manažer je velmi jednoduchý na instalaci i ovládání [12]. Při používání režimu embedded je připojení k databázi realizováno přes knihovnu JDBC. Derby můžeme spustit také v režimu *server*, ve kterém lze komunikovat jak přes JDBC, tak i pomocí *TCP/IP*. Databáze podporuje všechny 4 úrovně transakční izolace. Podobně jako všechny ostatní předvedené databáze i Derby nabízí nástroje pro správu databáze [4]. Jako jediný z představených databázových manažerů nenabízí grafické uživatelské rozhraní v základním balíku. Je svižný a v paměti počítače zabírá 2.6MB (bez dat). [12]

2.5 Výběr

Při shrnutí vlastností všech výše zmíněných databázových systému je zřejmé, že by se každá z nich dala v projektu použít. Pro potřeby této práce byla vybrána jedna vnořená databáze.

Módy	**H2**	**Derby**	**HSQLDB**
Čistá Java	Ano	Ano	Ano
Paměťový mód	Ano	Ano	Ano
Šifrovaná databáze	Ano	Ano	Ano
ODBC Driver	Ano	Ne	Ne
Fulltextové vyhledávání	Ano	Ne	Ne
Multi Version Concurrency	Ano	Ne	Ano
Footprint (jar/dll velikost)	1.5 MB	2.6 MB	1.4 MB

Tab. 1:Ukázka jednotlivých vlastností zkoumaných databázových manažerů [7]

Hlavními faktory v rozhodování bylo připojení a jednoduchost manipulace s databází. Všechny databázové systémy podporují připojení přes knihovnu JDBC. HSQLDB a

Derby nepodporují připojení přes ODBC (viz. Tab. 1), což nevadí - připojení k programu bude realizováno přes JDBC. Při měření velikosti místa na disku, které by databáze připojená na projekt zabírala, vyhrála HSQLDB. Firebird naopak zabíral nejvíce. Je to dáno množstvím utilit, které nabízí. H2 zabírá jen o něco málo více než HSQLDB, což je zapříčiněno mnohem větším výčtem funkcí, jež H2 nabízí. My ovšem tyto funkce potřebovat nebudeme [4] , [7] , [8] , [9] , [12].

Po pár zkušebních příkazech a lehkou manipulací s databázemi jsem se rozhodl v projektu použít databázi HSQLDB. Tuto volbu podpořila zejména jednoduchost a velmi dobrá ovladatelnost.

3 Měřitelné atributy pozemní komunikace

3.1 Základní výčet atributů

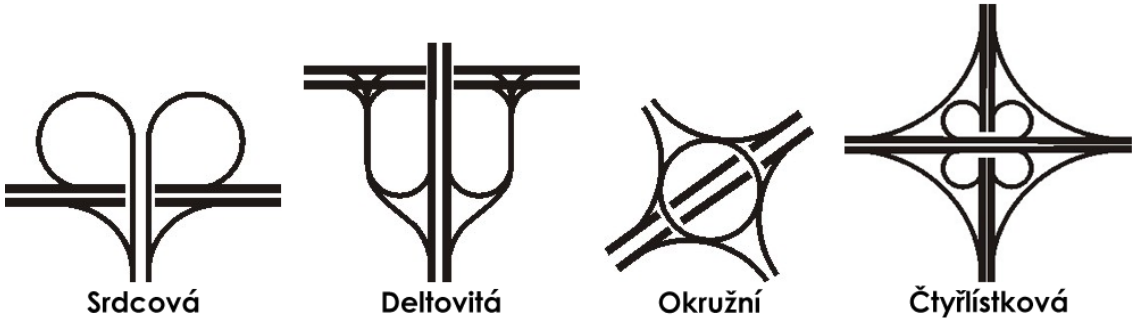
Základním prvkem pozemní komunikace je křižovatka, která se jako celek skládá z řady atributů. Tyto atributy lze měřit, nebo zaznamenávat události, které se na nich odehrávají.

3.1.1 Křižovatka a křížení pozemních komunikací

Křižovatka je propojení nebo setkání alespoň dvou pozemních komunikací. Za křižovatku nelze prohlásit sjezdy k nemovitostem nebo propojení dvou a více lesních cest či připojení obslužných zařízení [13].

Obr. 1:Prosté úrovňové křižovatky [13]

To jsou například čerpací stanice pohonných hmot, odpočívadla apod. Křižovatky se dělí do tří základních skupin: na úrovňové (viz. Obr. 1), mimoúrovňové (viz. Obr. 2) a kombinované (ty vzniknou spojením předchozích typů [14]). Každá křižovatka má své jméno a počet paprsků.



Obr. 2:Mimo úrovňové křižovatky [13]

Křížení je místo, kde se pozemní komunikace protínají pouze půdorysově (např.: most vedoucí přes dálnici) nebo se zde protíná například pozemní komunikace s drážní

komunikací. Lze tak označit i místo střetu pozemní komunikace s vodotečí[1]. Křížení se dělí na úrovňové a mimoúrovňové [14] , [15].

Mezi úrovňové křížení patří *přejezd*, jinak také železniční přejezd. Jde o křížení pozemní komunikace s drážní komunikací, či úrovňové křížení cyklistického pruhu s jiným dopravním pruhem pozemní komunikace [15]. Dalším úrovňovým křížením je *přechod*. Je to křížení komunikace pro chodce s pozemní komunikací [14]. Na přechodu lze vypozorovat, že v určitý moment stojí na levé či pravé straně určitý počet osob. Lze tedy zaznamenat počty osob na obou stranách nebo, při součtu levé a pravé strany, celkový počet osob na přechodu.

3.1.2 Části křižovatky

Křižovatka se skládá z *paprsků křižovatky*, což je část pozemní komunikace od hranice křižovatky k místu průsečíku os křižujících se komunikací. Paprsek se skládá z průběžných pruhů a větví křižovatky [15]. Paprsek nese jméno ulice, ve které se nachází. Může obsahovat světelné signalizační prostředky pro řízení dopravy nebo také zastávku MHD.

Větev křižovatky je pás propojující paprsky křižovatky mimo střed křižovatky. Bývá oddělen od paprsku ostrůvkem nebo dopravním stínem [15]. *Jízdní pruhy* jsou různé směry jízdy v paprsku vedoucí středem křižovatky. Mohou být průběžné (jízdní proud neodbočující vlevo ani vpravo), odbočovací (jízdní proud odbočující vpravo nebo vlevo) nebo připojovací (pro zařazení do průběžného pruhu) [15]. V určitém časovém úseku každým pruhem projede určitý počet vozidel.

3.1.3 Rozdělení křižovatek podle stupně usměrnění dopravy

Křižovatky lze dělit podle toho, jakými stavebními úpravami je na nich usměrněn pohyb dopravních prostředků. Stavebními úpravami je myšleno dopravní značení, dopravní ostrůvky nebo dokreslení různých druhů pruhů.

Prosté (neusměrněné) křižovatky – dopravní směry nejsou nijak členěny ani stavebními úpravami ani dopravními vodorovnými nebo svislými značkami [14].

[1] Vodní tok je koryto s vodou, která odtéká z povodí. Tok může být v celé délce nebo v části povrchový nebo podpovrchový, přirozený nebo umělý

Částečně usměrněné křižovatky – zde jsou směry členěny pomocí stavebních úprav (např. ostrůvků) a dopravními vodorovnými a svislými značkami. Obvykle je toto usměrnění použito na hlavní či dopravně významnější komunikaci [14].

Usměrněné křižovatky – na těchto křižovatkách je stavebními úpravami a dopravními značkami přesně vymezen možný pohyb po dopravních proudech (dopravní ostrůvky, řadící pruh pro odbočení vlevo či vpravo, připojovací pruh, se středním ostrůvkem) [14].

3.1.4 Rozdělení křižovatek podle možnosti řízení dopravy

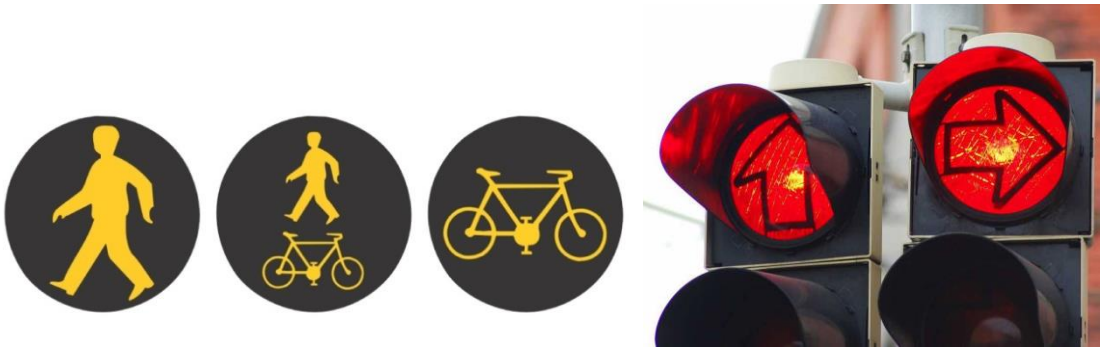
Křižovatky lze dělit podle toho, jakým způsobem je na nich řízen pohyb dopravních prostředků a osob na přechodech.

Křižovatky neřízené – přednost jízdy je zde dána pouze vyhláškou pravidel stanovených pozemní komunikací (hlavní a vedlejší směr, přednost zprava, pravidla pro chodce na přechodech aj.) [14].

Křižovatky řízené – doprava je řízena signálním plánem světelně signalizačního zařízení. Není tím myšleno řízení dopravy příslušníkem policie. Podmínkou je řízení pomocí světelné signalizace [14].

3.1.5 Světelně signalizační zařízení

Světelně signalizační zařízení slouží pro regulaci pohybu vozidel a chodců po pozemní komunikaci. Zařízení se skládá ze signálů.



Obr. 3:Ukázka obrázků na semaforech a jejich možná kombinace [17] , [18]

Každý z těchto signálů může být konstruován jako plný, čistý kruh nebo může obsahovat signalizační šipku, tvar jízdního chodce nebo jízdního kola, autobusu či jiného dopravního prostředku. Není vyloučena ani možnost kombinace těchto znaků

[16]. Pro vozidla platí signály *červená – žlutá – zelená*. Chodci mají na svém signalizačním zařízení pouze červenou a zelenou.

Pro regulaci pohybu tramvají se používají jiné typy světelně signalizačních zařízení. V Plzni se využívají dva prvky pro řízení dopravy tramvají. První je krabička se čtyřmi kuličkami (čočkami) a žlutým obdélníkem s černým P zvaná *tramvajový předsignál.* Ten slouží jako informace pro řidiče tramvaje, jaký signál může na křižovatce očekávat. Na křižovatce je poté předsignál samostatný nebo kombinovaný se signalizací *výzvové návěstidlo,* jenž se skládá ze skupiny diod [19].

Signalizace tramvajového předsignálu před křižovatkou:

- *dvě krajní kulatá* – očekávej stůj – na křižovatce může svítit signál „volno“
- *dvě nad sebou kulatá* – očekávej volno

Signalizace výzvového návěstidla je skupina diod. Čočky výzvového návěstidla na křižovatce mají trochu jiný význam. Dvě krajní kulatá znamenají stůj a dvě kulatá nad sebou jsou volno. Tento signál se pak spojí se signalizací diod:

- *směr vlevo* - (\ )
- *směr přímo* - (|)
- *směr vpravo* - (/)

Obr. 4:Vlevo tramvajový předsignál, vpravo kombinace tramvajového předsignálu a výzvového návěstidla [19]

Když si všechny informace o světelně signalizačních zařízeních shrneme, uvidíme, že informací je zde k zaznamenání opravdu mnoho. Nejprve rozlišíme, zda jde o zařízení pro regulaci chodců, vozidel či tramvají. Poté je dobré vědět, jakou kombinaci

signalizace zařízení používá. Dále se mění i výplně světel. Máme světla s obrázkem či bez obrázku. Když s obrázkem, tak jakým. Po pozorování více zařízení, si také všimneme, že ne každá barva svítí na každém zařízení stejně dlouho. Je to dáno hustotou provozu vozidel, jež místem se světelně signalizačním zařízením projíždí.

Proto se rozlišují dva druhy řízení dopravy z pohledu přepínání světel a jejich intervaly:

Statické řízení (pevné signální programy) – každý cyklus má stejné pořadí a stejnou délku intervalů. Nijak nereaguje na změny na křižovatce a stále opakuje stejný cyklus [19].

Dynamické řízení – pořadí cyklu a délka intervalů se mění v závislosti na dění na křižovatce. Pro zjištění dění na křižovatce se používají pro vozidla indukční smyčky, pro chodce tlačítka a pro tramvaje trolejové kontakty [19].

3.1.6 Městská hromadná doprava a její subsystémy

Městská hromadná doprava je vždy tvořena jedním nebo několika subsystémy. Mezi ty základní patří:

Tramvajový – kolejové vozidlo s přívodem elektrického trakčního proudu. Vozidlo je závislé na přívodu elektrického proudu a jeho rozsah pohybu je určen směrem a délkou kolejnic. Mezi výhody toho dopravního prostředku lze počítat provoz bez škodlivých exhalací, jednoduché řízení rozjezdu a brždění, vysokou životnost vozidel a velkou přepravní kapacitu. Nevýhodou je hlučnost a rozsah dopravy. Při poruše na kolejích nelze najít jinou cestu a celý směr je zablokován do doby, než je závada vyřešena. Zavedení dráhy pro tramvaje a nákup samotných tramvají také není levnou záležitostí [20].

Trolejbusový – vozidlo s trolejovým přívodem a odvodem trakčního proudu. Omezené vozidlo, závislé na poloze trolejových pásů. Tento subsystém se využívá jako integrovaná součást dopravních systémů. Výhodou je nižší zatěžování životního prostředí a velmi malá hlučnost. Jednoduché řízení rozjezdu a brždění. Mezi zápory by se dalo zařadit omezení pohybu a závislost provozu na dodávkách elektrické energie [20] , [21].

Autobusový – vozidlo určené pro přepravu osob, jež má více než 9 míst k sezení. Nepočítá se místo řidiče. Vozidlo s uzavřenou karoserií určeno pro nezávislou přepravu

osob hromadnou osobní dopravou. Subsystém autobusů se používá jako základní, doplňující, napájecí nebo překrývající dopravní síť. Podle konstrukce lze autobusy dělit na nízkopodlažní, jednopodlažní, dvoupodlažní a kloubové. Mezi silné stránky autobusové dopravy patří nezávislost na napájecím vedení. V případě poruchy na komunikaci, lze velmi rychle upravit trasu. Velkou nevýhodou je ovlivnitelnost ostatními vlivy silničního provozu, a také výrazné negativní dopady na životní prostředí [20] , [21].

Rychlodrážní – za subsystém rychlodrážní dopravy lze považovat například městskou nebo příměstskou rychlodráhu. Rychlodráha a jejich různé podoby se vyvinuly až rozvojem MHD ve velkých městech [20].

- *Tramvajová rychlodráha* – jde o systém umožňující napojení na nižší (městská tramvaj) a vyšší (příměstská regionální železnice) systém. Jeho výhodou je velká kapacita přepravy (5-30 tisíc osob za hodinu) v jednom směru a také rychlost (25-35km/h) [20].
- *Městská (příměstská) rychlodráha* – hlavním rysem tohoto subsystému je důkladná segregace od ostatní dopravy ve městě a okolí. Trať může být vedena v různých úrovních – pod zemí, na povrchu, nad zemí. Vozidla na těchto drahách jsou poháněna elektrickým proudem, který dostávají přes troleje nebo třetí kolejnicí. Velkým kladem je rychlost a objem osob, který dokáže vozidlo přepravit. Tato vysoká výkonnost také ovšem vyžaduje vyšší náklady na provoz, což je velké mínus [20].

Při bližším pozorování a znalosti jízdních řádů si můžeme všimnout hned několika atributů, jež lze zaznamenat. Každý dopravní prostředek městské hromadné dopravy má nějaké označení, většinou to bývají čísla kombinovaná se jménem konečné zastávky. Dopravní prostředek má také nějaký směr nebo zastávku, na kterou míří. Pro každou zastávku má dopravní prostředek jasně daný čas příjezdu a odjezdu. Lze si zaznamenat i počet osob, který je dopravním prostředkem přepravován.

V předchozím odstavci jsme se zmínili o *zastávce.* Je to místo předepsaným způsobem označené a určené pro výstup, nástup či přestup osob cestujících pomocí městské hromadné dopravy (MHD) [22]. Zastávka je se zaznamenáváním dat úzce spjata s MHD. O každé zastávce lze říci, na které křižovatce se nachází a na kterém z paprsků křižovatky je umístěna. V určitý moment je na zastávce přesný počet osob. V případě

příjezdu vozidla městské hromadné dopravy se tento počet může a nemusí změnit. Někdo nastoupí, někdo zase vystoupí z dopravního prostředku nebo se nic nestane, protože mohl přijet spoj, na který nikdo nečeká.

3.1.7 Typy zastávek podle druhu zastavujících vozidel

Zastávky lze rozdělit do několika skupin. Například podle toho, jaký dopravní prostředek MHD u dané zastávky zastavuje.

Tramvajová zastávka – místo určené pro nástup, výstup či přestup osob přepravujících se pomocí tramvaje. Zastávky bývají často umístěny přímo v dopravní komunikaci mezi pruhy. Propojení s komunikací pro chodce je realizováno pomocí přechodu pro chodce. Často bývá kombinován se světelně signalizačním zařízením.

Autobusová a trolejbusová zastávka – místo určené pro nástup, výstup či přestup osob přepravujících se pomocí autobusu či trolejbusu. Tyto dva dopravní prostředky sdílí stejný typ zastávky. Pokud na zastávce staví trolejbus, je velmi pravděpodobné, že zde bude mít svou zastávku i autobus. Opačným směrem tato věta ovšem neplatí. Lze to ovšem jednoduše zjistit při pohledu vzhůru, zda nad zastávkou vedou troleje.

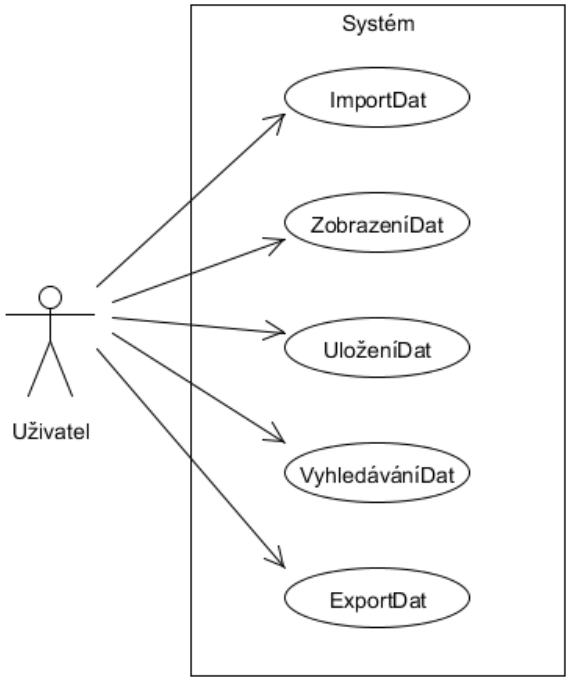
Rychlodrážní stanice – tento typ zastávky slouží pro nástup, výstup a přestup osob přepravujících se pomocí rychlodrážních vozidel. Tyto zastávky jsou velmi často, jako celé vedení této dopravy, vedeny mimo dopravní komunikaci. Záleží také na tom, zda je dráha vedena nad, pod nebo na povrchu.

Sdružené – kombinace předchozích typů.

4 Analýza

4.1 Diagram případů užití

Diagram případů užití zobrazuje volby, které může uživatel při práci s programem použít.



Obr. 5:Diagram případů užití

4.2 Specifikace požadavků

Program by měl být navržen tak, aby uměl zpracovávat soubory s koncovkou `txt` a `xml`. Soubory by měly obsahovat určitá data v určité struktuře. Tato struktura ale může být do jisté míry variabilní.

4.2.1 Rozbor TXT

Do `txt` se budou zapisovat data stylem „*klíč (oddělovač) hodnota“*, např.: „jméno křižovatky = Náměstí Míru“ (viz. Obr. 6). Oddělovač dat může každý používat jiný, proto bude jednodušší před každým načtením souboru vyzvat uživatele k zadání oddělovače, který použil. Protože někdo zaznamenává hodnoty a pak k nim dopisuje klíče, bude program umět rozeznat i přehozená data. Každá trojice klíč-oddělovač-

hodnota by měla být na samostatném řádku. Soubor bude přehlednější a lépe se bude i načítat.

```
Náměstí Míru = křižovatka
začátek měření = 2013-03-10 11:54
konec měření = 2013-03-10 12:09

rameno křižovatky = Klatovská jih
směr pruhů = PR
rovně = 2
vpravo = 5
```

Obr. 6: Ukázka načítaného souboru

Klíč „jméno křižovatky“ bude sloužit jako oddělovač záznamů o různých křižovatkách. Proto by tento atribut měl být vždy první v záznamu o křižovatce. Navíc je povinné mít tento atribut v každém souboru minimálně jednou, stejně jako čas měření, jinak bychom netušili, kde a kdy byla data zaznamenána. Některá data je nutno zapisovat v určitém pořadí. Například pokud má křižovatka 4 ramena, nemůže uživatel nejprve vyjmenovat ramena a poté psát, co na nich naměřil. Je nutné tedy skládat podřazené atributy pod nadřazené v takovém pořadí, v jakém odpovídají realitě. Důležité je, aby byly všude stejné druhy informací. Jakmile uživatel u jednoho pruhu v rameni vyplní typ pruhu, musí to udělat i u ostatních pruhů v rameni.

```
<krizovatka jmeno=”Náměstí Míru”>
   <cas casZacatku="2013-03-10 11:54" casKonce="2013-03-10
12:09"/>
   <rameno jmeno=”Klatovská-jih”>
      <pruhVjezd sipky=”PR”>
         <pocetPrujezdu smer=”R” rovne=”10” />
         <pocetPrujezdu smer=”P” vpravo=”15” />
      </pruhVjezd>
   </rameno>
</krizovatka>
```

Obr. 7:Ukázka zapouzdření elementů

Jelikož může program používat kdokoliv, je možné, že si uživatel nebude pamatovat jména atributů všech hodnot. Proto by měl být program schopen určit neznámé atributy. Tyto atributy poté nabídne k upřesnění, tehdy uživatel vybere jednu z variant názvů, ke

které se jméno jeho atributu nejvíce blíží. Po určení všech neznámých atributů si program nové názvy zapamatuje.

4.2.2 Rozbor XML

Při zápisu dat do `xml` by měly být elementy do sebe zapouzdřeny (viz. Obr. 7). Pomůže to jak přehlednosti, tak jednoduššímu parsování. Podobně jako u `txt` i zde je načtení dat podmíněno nutnosti výskytu záznamu o křižovatce. Zde ovšem není nutné jméno, postačí samotný element „křižovatka“. Pro názvy atributů a elementů zde platí stejný mechanizmus rozpoznávání a učení jako u `txt`.

Pro přesnější určení dat je výhodnější zadávat data jako hodnoty atributů elementu. To znamená, že by mezi elementy neměl být žádný text. Všechna data budou hodnotami atributů ve `start` elementu.

4.2.3 Rozpoznané atributy

Program bude umět rozpoznat základní atributy křižovatky a jejich vlastnosti. Jako první v souboru bude jméno křižovatky a časový interval, kdy měření probíhalo. Dále lze ke křižovatce doplnit počet paprsků, typ křižovatky a MHD, které se během měření na křižovatce pohybovalo. Časové údaje budou rozpoznatelné pouze ve formátu HH:MM RRRR-MM-DD, lze zadat i obráceně. Bude zadán vždy začátek a konec intervalu v daném formátu.

Každý jednotlivý paprsek bude popsán jménem. Tím je myšlena ulice, kterou paprsek prochází. Dále jsou zde informace o přechodu pro chodce, jízdních pruzích, které tvoří paprsek, semaforech a zastávkách. Všechny tyto informace musí být vždy uvedeny u jména paprsku, ke kterému patří. Pruh je vždy buď vjezd, nebo výjezd z křižovatky. Každý pruh má také určitý směr. Ten bude popsán následujícím způsobem: L – pruh zahýbá vlevo, P – pruh zahýbá vpravo, R – pruh vede křižovatkou rovně. Jednotlivé směry lze i kombinovat. Počty aut, které projely daným pruhem, budou popsány vždy dvojicí směr a počet vozidel. Pokud je semaforů na paprsku více, budou zadány vždy hned za sebou. Semafor bude určitého typu, je tím myšleno, zda je pro auta, tramvaje či chodce. Lze doplnit i údaj, zda se u semaforu nachází indukční smyčka nebo jiné funkcí podobné čidlo. Každá barva semaforu bude popsána barvou, dobou trvání (jak dlouho svítí – v sekundách) a časovým intervalem. Interval bude ve stejném formátu jako čas měření. Posledním atributem, jenž musí být zaznamenán u jména paprsku, ke kterému

patří, je zastávka. Zastávek bude moci být v souboru více. Každá bude označena jménem, tím bude název zastávky MHD. Dále zde budou údaje o osobách, které čekají na MHD. V případě, že na zastávce během měření zastaví vozidlo MHD, budou zaznamenány i údaje o počtu nastupujících a vystupujících osob.

4.2.4 Uložení, vyhledání a export dat

Data se budou ukládat do vnořené databáze. V uložených/načtených datech bude možno vyhledávat. Vyhledávání bude rozděleno na tři části. Bude možné vyhledávat křižovatky podle jejich jmen, názvů ulic a času, kdy byly změřeny. Druhá část umožňuje vyhledat MHD podle křižovatky, na níž bylo MHD zaznamenáno, čísla linky, směru linky a času, ve kterém bylo MHD zaregistrováno na pozemní komunikaci.

Vyhledávání podle času bude rozděleno na několik možností. Půjde zadat čas a datum začátku měření a také čas a datum konce měření. Při vyplnění celého formátu začátku měření, budou vyhledány záznamy měřené od této doby do budoucna. Naopak vyplněním celého formátu konce měření budou vyhledány záznamy měřené od této doby do minulosti. Vyplněním pouze času nebo data budou vyhledány záznamy, které přímo odpovídají zadaným časovým údajům.

Program umožní export dat do XML souboru s jednotnou strukturou. Bude možno exportovat, jak data načtená ze souboru, tak vyhledaná z databáze.

4.3 Použité technologie

4.3.1 Jazyk

K realizaci budou použity technologie, které jsou kompatibilní s jazykem Java. V tomto jazyce bude totiž aplikace vyvíjena.

4.3.2 Grafické uživatelské prostředí

K vytvoření grafického uživatelského prostředí bude použit JFC `Swing`, jenž je součástí Javy. Hlavní okno aplikace je potomek třídy `JFrame`. Pro další úrovně otevíraných oken je použita třída `JDialog`. Tento způsob je zvolen proto, aby uživatel nemohl volně klikat na 4 okna zároveň. Může komunikovat vždy jen s nejvrchnějším oknem.

4.3.3 Čtení a zápis (vstup a výstup) dat

Pro načtení `xml` souboru jsem se rozhodoval mezi technologií SAX a StAX. Nabízela se i možnost použít DOM, který načte celý soubor do paměti a vytvoří stromovou strukturu dat. Lze díky tomu procházet data opakovaně a v libovolném pořadí. My ale tuto možnost nepotřebujeme a v zájmu šetření paměti tento způsob načtení zvolen nebyl. Nakonec jsem zvolil technologii StAX. Výpis je realizován pomocí třídy *XMLStreamWriter*.

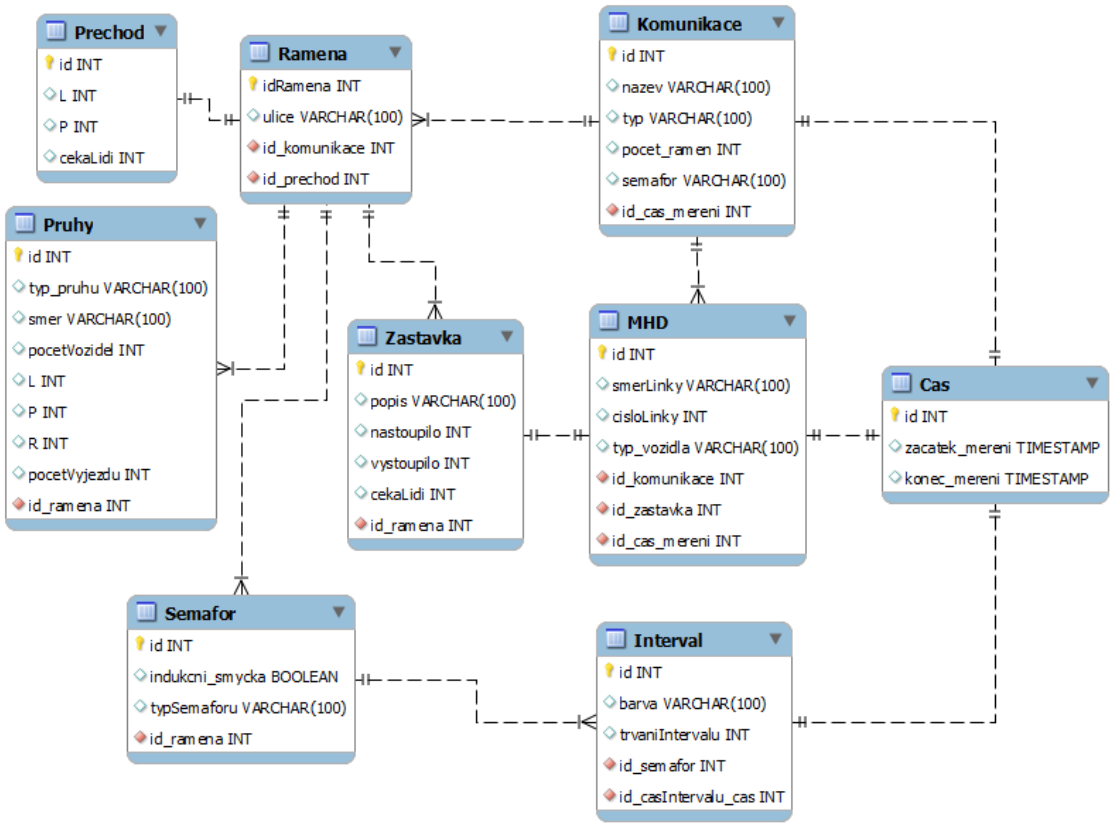
4.3.4 Ukládání dat

Ukládání dat probíhá do vnořené databáze, aby byl program snadno přenositelný z počítače na počítač a nebylo nutné řešit připojení na databázový server. Pro projekt byla určena HSQLDB. Důvody, které mě k tomuto závěru vedly, jsou popsány v teoretické části v kapitole 2.

5 Implementace

5.1 ERA-diagram

Data jsou ukládána do tabulek vnořené databáze. Každá tabulka je zároveň i třídou v balíku `aplika.tridyDB`. Každý sloupec (výjimkou jsou cizí klíče) tabulky je zároveň i atribut třídy, která se jmenuje stejně jako tabulka.



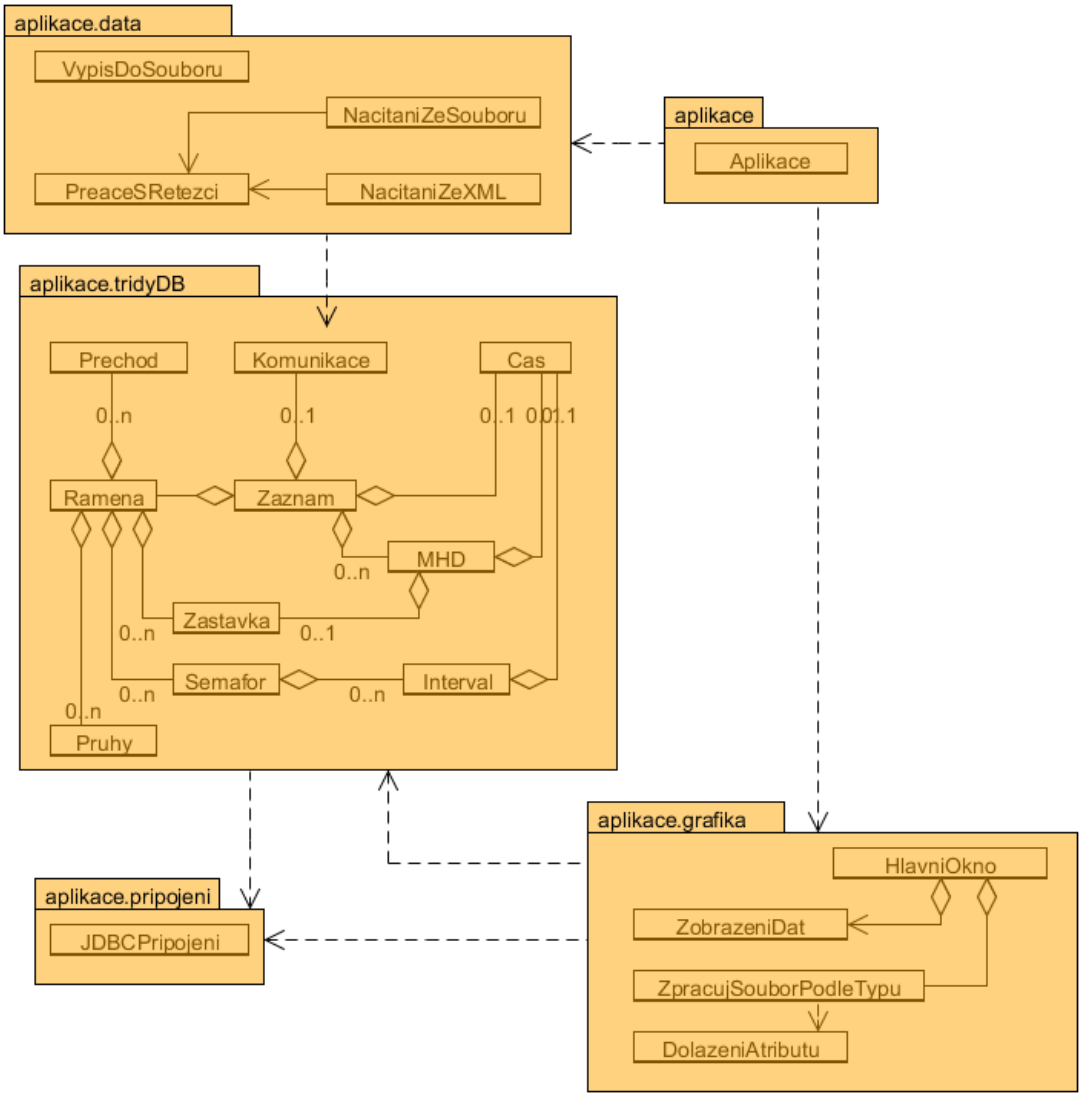
Obr. 8:ERA-diagram

5.2 Balík aplikace

Celý projekt je zastřešen balíkem `aplikace`. Ten obsahuje pouze jednu třídu `Aplikace` a 4 další balíky. Třída obsahuje jedinou metodu `main()`, která spouští celou aplikaci. Na obrázku (viz. Obr. 9) vidíme, jak zhruba vypadá celá struktura balíků a tříd.

5.3 Balík aplikace.data

Balík `aplikace.data` představuje *datovou vrstvu*. Obsahuje třídy `Nacitani-ZeSouboru`, `Nacitani-ZeXML`, `PraceSRetezci`, `VypisDoSouboru`.



Obr. 9:Struktura balíků a tříd (jsou uvedeny pouze asociace mezi balíky a třídami v balících)

5.3.1 Třída NacitaniZeSouboru

Načítání z `txt` je pomocí třídy `NacitaniZeSouboru`. Čte se po řádcích. Po načtení se řádek rozdělí do pole podle zadaného oddělovače. V poli je vyhledáno jméno atributu (klíč). Pokud není nalezeno, je celý řádek předán do seznamu neznámých `ArrayList<String>`. V opačném případě se přečte další řádek.

Pokud je po přečtení celého souboru seznam neznámých neprázdný, je odeslán do instance třídy `DolazeniAtributu` z balíku `aplikace.grafika`. Zde proběhne rozpoznání neznámých atributů (více v kapitole 5.4.4). Poté se program vrátí do `NacitaniZeSouboru` a zahájí vytváření instancí třídy `Zaznam`.

Program znovu prochází řádky, ale tentokrát s tím rozdílem, že už rozpozná všechny atributy. Každý řádek rozdělí podle oddělovače, najde jméno atributu, a to pošle do *switche*. Zde se podle atributu provede příslušná operace. Vytváří se tak seznam `ArrayList<Zaznam>` záznamů. Seznam je pak k dispozici aplikační vrstvě pro další manipulaci.

5.3.2 Třída NacitaniZeXML

Soubor s příponou `xml` se načte pomocí `NacitaniZeXML`. Čtení je proudové a je realizováno pomocí technologie StAX. Při čtení se vždy nejprve hledá `start` element. Po jeho nalezení proběhne test, zda je programu jméno `start` elementu známo. Pokud ho nezná, předá jej do seznamu `ArrayList<String>` neznámé.

Stejně jako u `txt` i zde, pokud je seznam neznámých neprázdný, je odeslán do instance třídy `DolazeniAtributu` z balíku `aplikace.grafika` a proběhne rozpoznání. Jakmile program zná všechny elementy, může začít vytváření záznamů.

Znovu se prochází celý soubor, ale teď jméno elementu putuje do *switche*. Zde se zařadí do odpovídající větve a provede příslušnou metodu. Po dokončení čtení je k dispozici seznam `ArrayList<Zaznam>` záznamů. Tento seznam je dále zpracováván aplikační vrstvou.

5.3.3 Třída PraceSRetezci

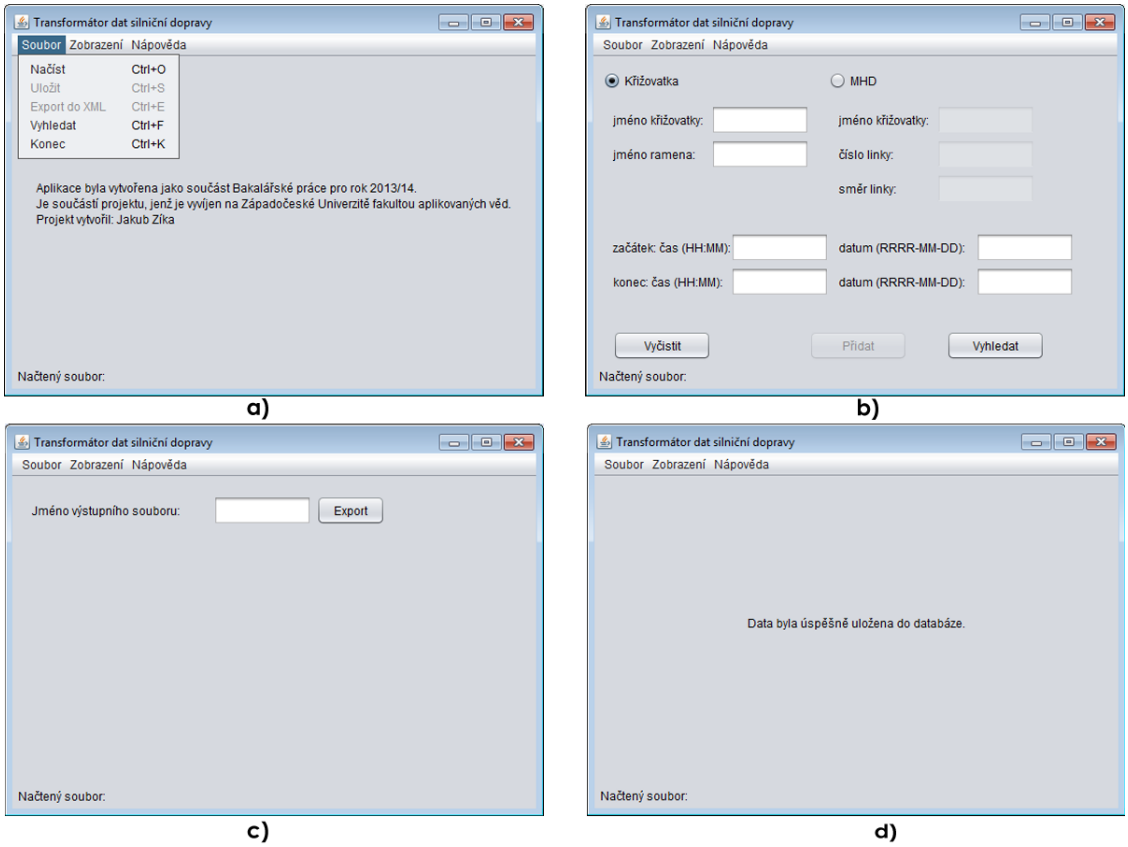
Ve třídě `PraceSRetezci` jsou jen *statické* metody pro různé úpravy řetězců. Mezi nejpoužívanější patří metody `rozpoznejATR()` a `nauceSeATR()`. Metoda `rozpoznejATR()` přijme slovo, o kterém bychom rádi věděli, zda patří mezi známá pojmenování atributů. Metoda projde soubor s ekvivalenty atributů, a pokud najde shodu, jako návratovou hodnotu použije *hlavní jméno* rozpoznaného atributu. Hlavní jména atributů jsou ta jména, která se nachází v souboru s ekvivalenty vždy na prvním místě každého řádku. Každé hlavní jméno atributu odpovídá jedné větvi ve *switchi,* který určuje metodu zpracování každého atributu. Druhá metoda `nauceSeATR()` zapisuje nově rozpoznané atributy do souboru s ekvivalenty.

5.3.4 Třída VypisDoSouboru

Poslední třídou tohoto balíku je `VypisDoSouboru`. Instance této třídy vytvoří `xml` soubor z dat, které dostane v seznamu záznamů. Zapisuje se pomocí třídy `XMLOutputFactory`.

5.4 Balík aplikace.grafika

Všechny třídy, které tvoří grafické uživatelské prostředí, jsou uloženy v balíku `aplikace.grafika`.

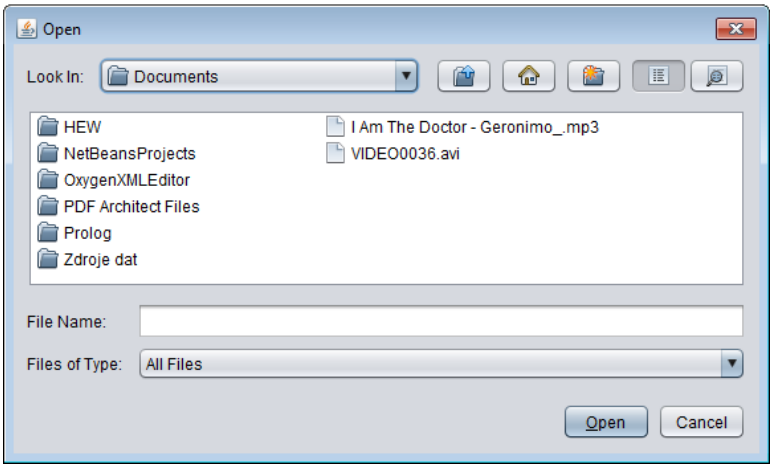


Obr. 10:Proměny hlavního okna

5.4.1 Třída HlavniOkno

Hlavní třídou je zde `HlavniOkno`, která zobrazí základní vzhled programu po spuštění (viz. Obr. 10a). Z tohoto okna lze provádět všechny úkony, které program nabízí. Okno je typu `JFrame` s lištou `JMenuBar` s nabídkou v horní polovině okna. Nabídka se skládá z instancí tříd `JMenu` a `JMenuItem`. Dále okno obsahuje dvě instance třídy `JLabel`. První label po spuštění zobrazuje informace o programu, druhý říká uživateli,

jaký soubor má aktuálně načtený. Prvků na ploše je mnohem více, ale mají vypnutou viditelnost.



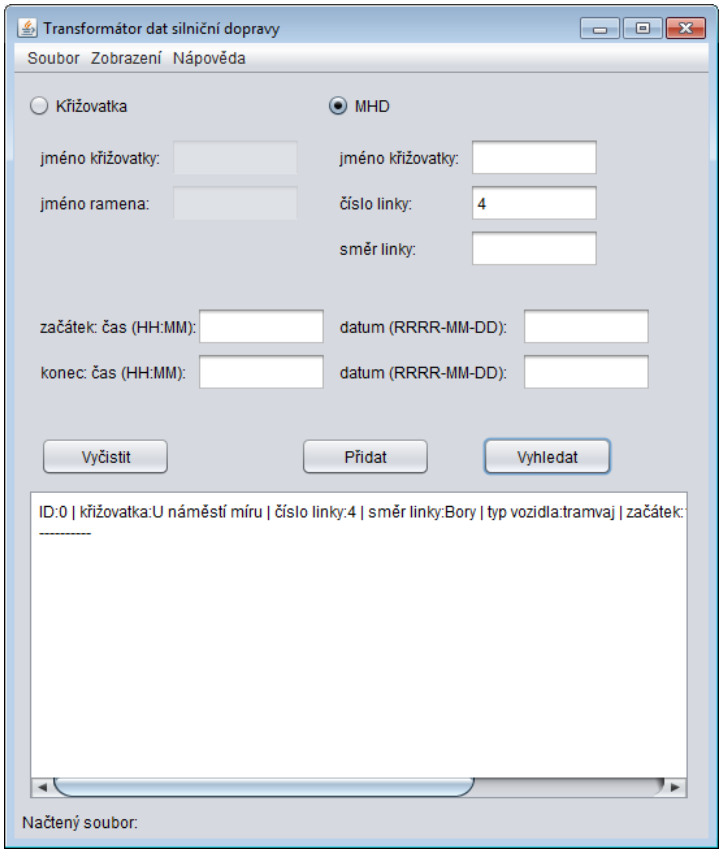
Obr. 11:Instance třídy JFileChooser

Menu *Soubor* nabízí možnosti *Načíst, Uložit, Exportovat do XML, Vyhledat* a *Konec*. Možnost *Načíst* po kliknutí vytvoří instanci třídy `JFileChooser`, pomocí které si vybereme soubor, který bychom chtěli načíst do programu ke zpracování (viz. Obr. 11). Úspěšné načtení souboru bude oznámeno hláškou, jež se zobrazí na místě, kde předtím byly informace o programu. Volbou *Uložit* se uloží záznamy do vnořené databáze. Po uložení se zobrazí hláška (viz. Obr. 10d). Při zvolení možnosti *Exportovat do XML* se vzhled okna změní (viz. Obr. 10c). Zmizí informace o programu. Uživatel bude vyzván, aby do textového okna `JTextField` zadal jméno souboru. Po potvrzení bude vytvořen soubor, jenž bude obsahovat data, která předtím byla v programu jako seznam záznamů.

Možnost *Vyhledat* přidá na okno nové ovládací prvky (viz. Obr. 10b). Dvojici přepínačů typu `JRadioButton`, jež jsou v jedné skupině `ButtonGroup`. Tím bude vždy označen pouze jeden přepínač. Těmito přepínači se vybírá oblast dat, ze které bude vyhledáváno. Při volbě *Křižovatka* se budou hledat záznamy o křižovatkách a jejich ramenech. Vyhledávání lze blíže specifikovat pomocí přídaných textových oken. Lze dovyplnit *jméno křižovatky* nebo *jméno ramena*, které hledáme. Je tu možnost i blíže specifikovat časové období. Může se hledat v určitém intervalu, konkrétní datum nebo od určitého data dále do minulosti či budoucnosti. Zvolením přepínače *MHD* se vyhledávají záznamy o dopravních prostředcích MHD. Vyhledávání lze opět specifikovat pomocí *směru linky* a *čísla linky*. Stejně jako pro křižovatku i zde je

možnost zadání časového intervalu. Jakmile jsou všechna důležitá data zadána, může se spustit vyhledávání. To se provede tlačítkem *Vyhledat*.

Výsledek úspěšného hledání (viz. Obr. 12). Pokud chceme s nalezenými záznamy dále pracovat, stiskne se *Přidat*. Tímto úkonem se vytvoří seznam instancí třídy Záznam. Data jsou tak připravena pro další použití a je možnost je opět uložit, exportovat, atd..

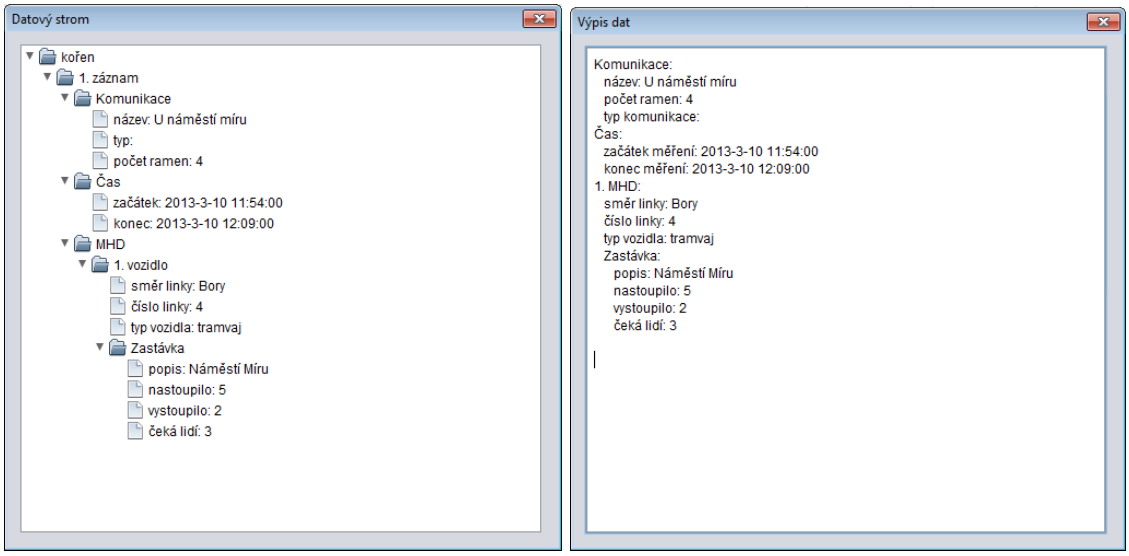


Obr. 12:Ukázka úspěšného vyhledávání v databázi dat

Pokud se uživateli výsledek vyhledávání nelíbí a již si uložil data do seznamu záznamů, stačí stisknout *Vyčistit*. Tím se vymaže seznam záznamů tak, že nepůjde nic zobrazit ani exportovat, a tyto možnosti se v nabídce zašediví. Poslední možností v menu *Soubor* je volba *Konec*. Ta celou aplikaci ukončí. Další záložkou je *Zobrazení*, zde je na výběr ze dvou možností *Strom* a *Výpis dat*. Obě vytvoří instanci třídy `JDialog`. Dále se už jen podle výběru, zda se jedná o strom nebo výpis, provede příslušná operace. Poslední záložkou je *Nápověda.* Název může možná trochu klamat, nejedná se zde o nápovědu k ovládání programu, nýbrž jen o informace o autorovi a účelu vytvoření této aplikace.

5.4.2 Třída ZobrazeniDat

Jak již bylo zmíněno v předchozí kapitole, pokud je seznam instancí třídy `Zaznam` neprázdný, program může tento seznam zobrazit. K tomuto účelu slouží instance třídy `ZobrazeniDat`. Záleží už jen na tom, zda se má zobrazit strom dat nebo jen jejich výpis.



Obr. 13:Vlevo zobrazení stromové struktury, vpravo zobrazení výpisu dat

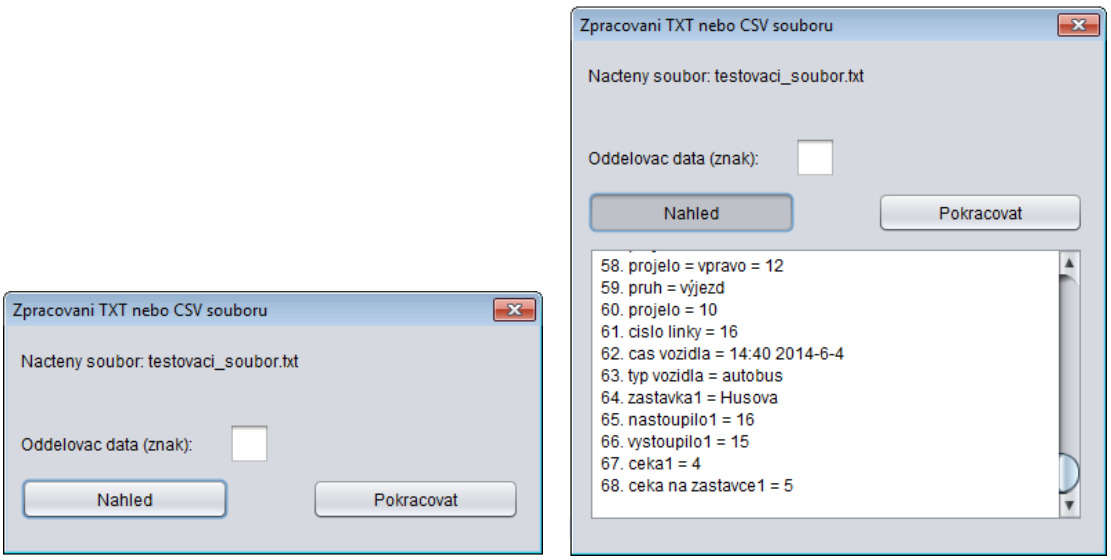
Pro vytvoření stromové struktury je potřeba určit strukturu zapouzdření dat a vytvořit tak model stromu. Modelje datového typu `DefaultTreeModel`. Do modelu se vkládají rodiče a potomci dat podle toho, jak to jak jsou do sebe zanořeni. Začne se u kořene stromu. Vytvoří se tedy instance třídy `DefaultMutableTreeNode`. Tento objekt obsahuje metodu na vytvoření uzlu (potomka) `add(DefaultMutableNode potomek)`. Pomocí této metody se určí potomek kořene. Dále už se jen uzly stromu přidávají podle toho, jak jsou data do sebe zanořená. Jakmile jsou všechny uzly hotové, vezme se kořen a vloží se do instance `DefaultTreeModel`. Strukturovaný výpis je vytvořen postupným voláním přepsané metody `toString()`. Ta je obsažena v každé třídě balíku `aplikace.tridyDB`.

5.4.3 Třída ZpracujSouborPodleTypu

Jakmile program zjistí cestu k načítanému souboru, zjistí se jeho přípona. Instance třídy `ZpracujSouborPodleTypu` na základě přípony spustí metody pro zpracování souboru. Pokud je načítaným souborem `xml`, nic nezobrazuje, pouze se vytvoří instance

třídy `NacitaniZeXML` a provede se načtení souboru. Pokud se jedná o soubor typu `txt`, zobrazí okno, ve kterém vyzve uživatele, aby zadal znak pro oddělování dat.

Pokud uživatel zapomněl, jaký znak použil, může si to zjistit pomocí tlačítka *Náhled* (viz. Obr. 14). Toto tlačítko vytvoří textové okno a zobrazí v něm obsah načítaného souboru. Po zadání oddělovacího znaku se pokračuje ke zpracování dat tlačítkem *Pokračovat*.



Obr. 14:Náhled načteného souboru vlevo vypnutý, vpravo zapnutý

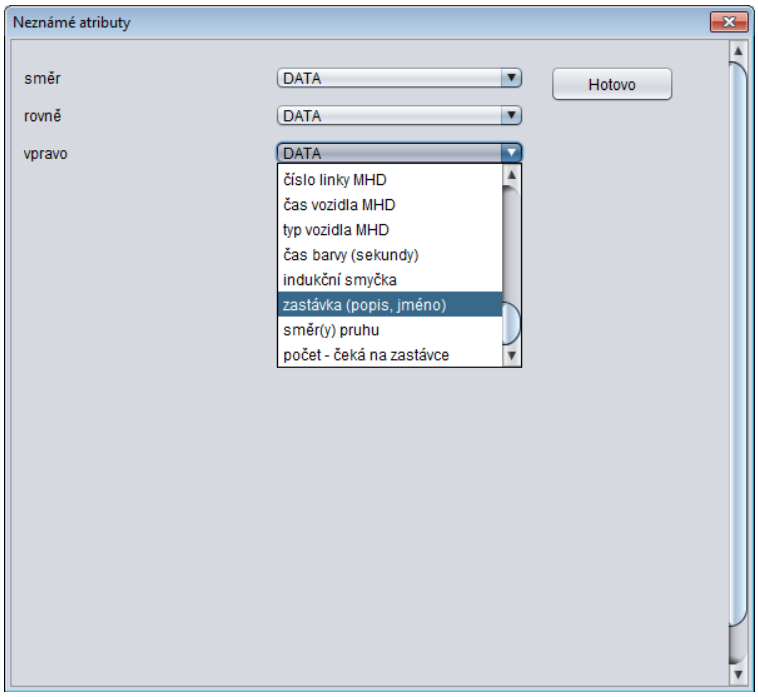
5.4.4 Třída DolazeniAtributu

Určení neznámých atributů probíhá ve třídě `DolazeniAtributu`, a to jen v případě, že je seznam neznámých atributů neprázdný. Po vytvoření instance se spustí okno, které obsahuje *n* dvojic (viz. Obr. 15). Každá dvojice je tvořena neznámým atributem a `JComboBoxem`, ten obsahuje seznam hlavních jmen atributů. Z tohoto seznamu si pak uživatel vybere ten atribut, který se neznámému atributu významově rovná. Pro případ, že je jako neznámý atribut uvedena hodnota atributu, je v seznamu zahrnuta položka *DATA*. Při této volbě se neznámý atribut nezapamatovává.

Jakmile jsou všechny atributy správně určeny, pokračuje se tlačítkem *Hotovo*. Po stisku se soubor zpracuje a vytvoří seznam instancí třídy `Zaznam`.

5.5 Balík aplikace.pripojeni

Připojení k databázi a veškerá manipulace s daty na trase program – databáze se odehrává v balíku `aplikace.pripojeni`. Ten obsahuje pouze jednu třídu `JDBCPripojeni`.



Obr. 15:Instance třídy DolazeniAtributu

5.5.1 Třída JDBCPripojeni

Metody této třídy zajišťují připojení a veškerou manipulaci s daty v databázi. Samozřejmostí jsou metody pro připojení a odpojení databáze. Data se do databáze vkládají pomocí metody `insert()`.

```
JDBCPripojeni db;
db = new
JDBCPripojeni("konfigurace/vytvor_tabulkySQL.txt");
String trida = "Cas";
String data = "3,'2012-4-10 11:50:00','2012-4-10
12:09:00'";
db.insert(trida, data);
```

Obr. 16:Ukázka užití metody insert

Parametr *trida* je jméno tabulky, do které bude zapisováno. Druhý parametr *data* představuje řetězec s daty (viz. Obr. 16). Pro výběr dat z databáze je metoda `select()`. Parametr *trida* je stejně jako u metody `insert()` jméno tabulky, *co*

reprezentuje sloupce, které budou vybrány, a *kriterium* představuje pravidlo, podle kterého budou data ze sloupců filtrována (viz. Obr. 17).

Třída při prvním spuštění aplikace načte ze souboru *vytvor_tabulkySQL.txt* potřebné příkazy pro vytvoření tabulek v databázi. Při dalším spuštění už jen kontroluje, zda jsou tabulky vytvořeny. Pokud by je uživatel smazal, znovu se vytvoří čistý základ bez dat.

```
JDBCPripojeni db;
db = new
JDBCPripojeni("konfigurace/vytvor_tabulkySQL.txt");
String trida = "Komunikace, Ramena, Cas";
String co = "*";
String kriterium = "Komunikace.nazev LIKE '%náměstí%' AND
Komunikace.id=Ramena.id_komunikace AND
Komunikace.id_cas_mereni=Cas.id";
db.select(trida, co, kriterium);
```

Obr. 17:Ukázka užití metody select

5.6 Balík aplikace.tridyDB

Všechny třídy, které představují elementy pozemní komunikace, jsou umístěny v balíku `aplikace.tridyDB`. Vše, co lze na pozemní komunikaci změřit, je poté vedeno jako atribut třídy. Všechny třídy obsahují metody na vytvoření uzlu datového stromu, výpisu do `xml` souboru, uložení do databáze a přepsanou metodu `toString()`.

5.6.1 Třída Záznam

Nejdůležitější třídou v balíku je třída `Zaznam`. Je totiž nositelem všech údajů o jednom měření křižovatky. Z instancí této třídy je složený seznam, který se dá ukládat do databáze, exportovat do `xml` nebo zobrazovat jako výpis či stromová struktura. Třída obsahuje instance tříd `Cas`, `Komunikace` a seznamy instancí tříd `Ramena` a `MHD`.

```
DateFormat format1 = new SimpleDateFormat("H:mm d.M.yyyy");
DateFormat format2 = new SimpleDateFormat("H:mm yyyy-M-d");
DateFormat format3 = new SimpleDateFormat("d.M.yyyy H:mm");
DateFormat format4 = new SimpleDateFormat("yyyy-M-d H:mm");
```

Obr. 18:Bezpečně rozpoznatelné formáty času

Třída `Komunikace` reprezentuje křižovatku. Obsahuje pouze jméno křižovatky a počet paprsků. Čas je zaznamenán do třídy `Cas`. Zaznamenává se zde čas měření nebo čas příjezdu a odjezdu dopravního prostředku MHD. Aby se dalo v časových

záznamech vyhledávat i na časové přímce, musí mít čas tvar, který lze převést na `TIMESTAMP`. Třída obsahuje několik tvarů zápisu, které je schopna převést na `TIMESTAMP`. Díky tomuto uložení lze vyhledávat záznamy v určitém časovém intervalu nebo také od určitého data do minulosti či budoucnosti. Další třídy už obsahují hodnoty buď jako `int`, `String`, nebo jako instanci jiné třídy z tohoto balíku.

Jednotlivé instance třídy `Ramena` obsahují informace o přechodech, pruzích, zastávkách a semaforech. Každá instance třídy `MHD` uchovává informace o směru linky, číslu linky a zastávce, u které vozidlo zastavilo.

```
DateFormat vystup = new SimpleDateFormat("yyyy-M-d
H:mm:ss");
```

Obr. 19:Formát uložený v databázi jako TIMESTAMP

6 Testování

Pro zjištění správné funkcionality se musí program otestovat. Bude testován především na načítání, a to na datech jedné křižovatky. Data mají reálný základ. Z důvodu otestování všech oblastí načítání jsou některé údaje fiktivní. Jedno měření bude zapsáno do dvou soborů. První bude typu `txt` a druhý `xml`. Zjistíme tím především, zda data ze dvou typově různých souborů se stejnými informacemi vytvoří totožné záznamy.

Důležité je, aby oba soubory měly nastavené kódování na UTF-8. Pokud tomu tak nebude, některé znaky se mohou chybně uložit nebo zobrazit. Také vyhledávání by nemuselo být tak přesné, protože by databáze neuměla některé znaky vyhledat.

6.1 Soubor typu TXT

```
křižovatka = U Práce
početRamen = 4
začátekMěření = 2014-06-20 12:50
konecMěření = 2014-06-20 13:00

ramenoKřižovatky = Klatovská třída - jih
typPruhu = vjezd
směrPruhu = RP
rovně = 43
vpravo = 21
typPruhu = výjezd
početVýjezdů = 2
projelo = 103

typVozidla = tramvaj
směrLinky = Bory
čísloLinky = 4
zastavkaMHD = U Práce
nastoupilo = 2
vystoupilo = 10
```

Obr. 20:Ukázka textového souboru

Když si rozebereme soubor typu `txt`, je zde několik pravidel, která se musí při vytváření dodržet. Každý záznam je na samostatném řádku. Informace jsou zapisovány stylem *klíč – hodnota*. Mezi klíčem a hodnotou je vždy oddělovací znak. Tento znak

musí být v celém souboru jednotný. Je to proto, že se při načítání program zeptá, jakým znakem má od sebe data oddělovat. Jako první informace v souboru musí být jméno křižovatky. Pokud tomu tak nebude, načtení neproběhne.

Data, která k sobě významově patří, musí být co nejblíže u sebe. V ukázkovém souboru je tomu například u ramena křižovatky. Jak víme, každé rameno se skládá z pruhů. V ukázce je vidět, že hned pod jménem ramena je popis pruhu. Druhý příklad je na konci ukázky. Každé vozidlo MHD je popsáno v samostatném odstavci. Důležité je, aby byly všude stejné druhy informací. Jakmile uživatel u jednoho pruhu v rameni vyplní typ pruhu, musí to udělat i u ostatních v rameni. Kdyby tomu tak nebylo, mohlo by to vést k chybnému zpracování souboru. Část ukázkového souboru (viz. Obr. 20). Celý soubor je k nalezení v kapitole B.

6.1.1 Načtení a zobrazení dat

Z výše uvedeného záznamu dat je patrné, že oddělovacím znakem zde bude „=“. Spustíme si tedy program, načteme soubor, zadáme oddělovací znak, určíme všechny neznámé atributy a necháme soubor zpracovat. Po úspěšném načtení si prohlédneme data. Vybereme si stromovou strukturu, bude to pro kontrolu dat přehlednější (viz. Obr. 14). Při postupné kontrole uzlů stromu a dat v souboru vidíme, že všechna data byla úspěšně načtena. Teď ovšem musíme stejná data nechat načíst ze souboru typu `xml`. Po načtení zkontrolujeme, zda se vytvořila totožná stromová struktura.

6.1.2 Ekvivalentní zápisy dat

Některá data lze zaznamenat různými způsoby a výsledek po zpracování bude stejný. Jak už bylo zmíněno, lze zaměnit pořadí *klíč – hodnota* na *hodnota – klíč*. Není tedy rozdíl v zápisu *křižovatka – U Práce* a *U Práce – křižovatka*.

```
směrPruhu = vpravo = rovně
vpravo = 12     < ekvivalentní zápis >     projelo = vpravo = 12
rovně = 10      < ekvivalentní zápis >     projelo = rovně = 12
```

Obr. 21:Ukázka ekvivalentního zápisu

Další výjimka pro zápis se týká hlavně atributů zpracovávající směry pruhu. Testovací soubor obsahuje zápis `směrPruhu = RP`. Tuto informaci lze zapsat i následovně `směrPruhu = vpravo = rovně`, výsledek načtení bude stejný. Program si zkratku RP po načtení sám sestaví. Hodnoty jednotlivých směrů se dají také zapsat

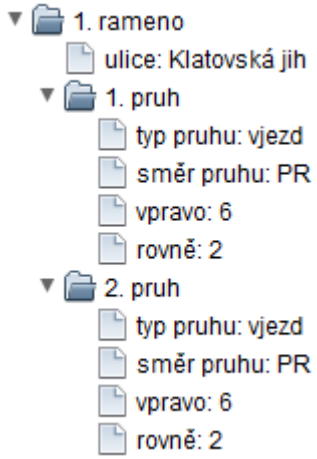
různě. Můžeme zapsat dvojici *směr = hodnota*, nebo *projelo = směr = hodnota* (viz. Obr. 21).

Směr vpravo, vlevo může být zapsán jako R a P nebo v angličtině. Všechny tyto typy zápisu program bezpečně rozezná. Složený směr z LPR nemusí mít při zápisu jasné pořadí písmen (viz. Obr. 22).

```
Náměstí Míru = křižovatka
rameno křižovatky = Klatovská jih
pruh = vjezd
směr pruhů = RP
rovně = 2
vpravo = 6
pruh = vjezd
směr pruhů = PR
rovně = 2
vpravo = 6
```

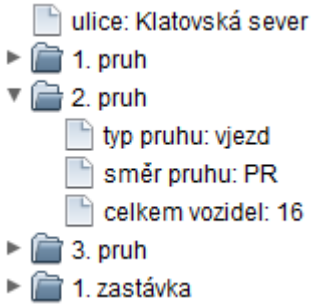
Obr. 22:Přehozené označení v zápise

Uživatel se nemusí bát, že pokud prohodí některé z písmen, data by se mu nenačetla. Program umí rozpoznat všechny kombinace těchto znaků (viz. Obr. 23).



Obr. 23:Výpis směru při zadání prohozeného tvaru (RP a PR)

Pokud uživatel zaznamenal vozidla pruhu RP, jak stojí v řadě, ale už neví, které vozidlo kam odbočilo, může zadat celkový počet. Například: `směrPruhu = vpravo = rovně = 16`. V zobrazení poté nebude určený každý směr s počty vozidel, zobrazí se pouze celkový počet (viz. Obr. 24). Tento údaj je možné zapsat ještě jedním způsobem (viz. Obr. 25).



Obr. 24:Načtení zápisu směrPruhu = vpravo = rovně = 16

6.2 Soubor typu XML

Soubor typu `xml` se skládá z elementů, které jsou do sebe postupně zapouzdřené. Zde tedy data zapisujeme do elementů a připisujeme k nim atributy. Atributy budou data, která popisují vlastnosti elementu. Pokud bych chtěl například popisovat dopravní vozidlo a jeho vlastnosti, jméno elementu bude vozidlo a jeho vlastnosti (typ vozidla, barva, počet kol, atd.) budou atributy elementu. Důležité je, aby podobně jako u `txt` byla v souboru informace křižovatce. Nemusí zde být jméno, postačí element pojmenovaný například *křižovatka*.

```
směrPruhu = vpravo = rovně
projelo = 16
```

Obr. 25:Ekvivalentní zápis

Data, která k sobě významově patří, musím být v sobě zapouzdřena. Na struktuře `xml` je to dobře vidět. Nadřazený element křižovatka a v něm zapouzdřeny všechny informace, které se k ní vztahují. Důležité je mít na paměti základní obecná pravidla při psaní `xml`. Elementy se nesmí křížit, pouze zanořovat. Každý element musí být řádně ukončen, řádkový element na konci lomítkem a párový koncovým elementem. Hodnoty atributů jsou vždy v uvozovkách. Ukázka z testovacího souboru (viz. Obr. 26). Celý soubor je k nalezení v příloze, kapitola B.

6.2.1 Načtení a zobrazení dat

Načtení souboru je zde jednodušší než u `txt`. Po zadání cesty k souboru se rovnou zobrazí okno pro určení neznámých elementů. Pokud žádné neznámé nejsou, soubor se po zadání cesty načte a můžeme rovnou pracovat s daty. Opět si data zobrazíme pomocí stromové struktury. Jak vidíme, strom je totožný s výpisem, který jsme provedli po načtení `txt` (viz. Obr. 14). Více v kapitole 6.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<krizovatka nazev="U Práce" pocetRamen="4">
     <cas zacatekMereni="2014-06-20 12:50"
konecMereni="2014-06-20 13:00" />
     <rameno jmenoUlice="Klatovská třída - jih">
          <pruhVjezd sipky="RP">
               <pocetPrujezdu smer="R" projelo="43" />
               <pocetPrujezdu smer="P" projelo="21" />
          </pruhVjezd>
          <pruhVyjezd pocet="2">
               <pocetPrujezdu projelo="103" />
          </pruhVyjezd>
     </rameno>
     <mhd typMHD="tramvaj" cisloLinky="4"
smerLinky="Košutka">
          <zastavka popis="U Práce" nastoupilo="6"
vystoupilo="4" />
     </mhd>
     <mhd typMHD="tramvaj" cisloLinky="4" smerLinky="Bory">
          <zastavka popis="U Práce" nastoupilo="2"
vystoupilo="10" />
     </mhd>
</krizovatka>
```

Obr. 26:Ukázka XML souboru

6.2.2 Ekvivalentní zápisy dat

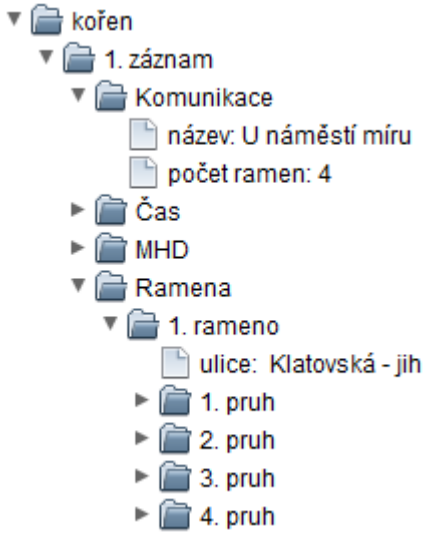
Stejně jako u `txt` se i v `xml` dají některé elementy či atributy zapsat jinak. Při zaznamenání jména křižovatky není potřeba toto jméno psát jako atribut. Můžeme ho zapsat jako párový element a mezi elementy vložit hodnotu (název křižovatky).

```
<krizovatka>
  <nazev>U náměstí míru</nazev>
  <pocetRamen>4</pocetRamen>
  <cas casZacatku="2013-03-10 11:54" casKonce="2013-03-10
12:09"/>
  <rameno>
    <jmenoUlice>Klatovská - jih</jmenoUlice>
```

Obr. 27:Přepsané atributy na elementy

Ekvivalentní zápis (viz. Obr. 27), ten obsahuje i ekvivalentní zápis počtu ramen a jména ulice. Atribut počet ramen je součástí elementu *krizovatka*. Atribut jméno ulice se

nachází v elementu *rameno*. Stejně jako u křižovatky lze tak název ulice a počet ramen vyjmout z atributů elementu a zapsat jako samostatný párový element. Struktura zápisu je stejná jako u názvu křižovatky.



Obr. 28:Zobrazení dat ekvivalentního zápisu názvu křižovatky, počtu ramen a jména ulice

Po načtení oba vzorové zápisy vytvoří totožná data (viz. Obr. 28). Atributy začátek a konec měření z elementu *cas*, lze také zapsat párovými elementy. Standardní zápis času je ve vzorovém souboru.

```
<zacatekMereni>2013-03-10 11:54</zacatekMereni>
<konecMereni>2013-03-10 12:09</konecMereni>
```

Obr. 29:Ekvivalentní zápis času

Po použití ekvivalentního zápisu se nám formát načteného času nezměnil. Je tedy vidět, že oba druhy zápisu vytvoří totožné záznamy.

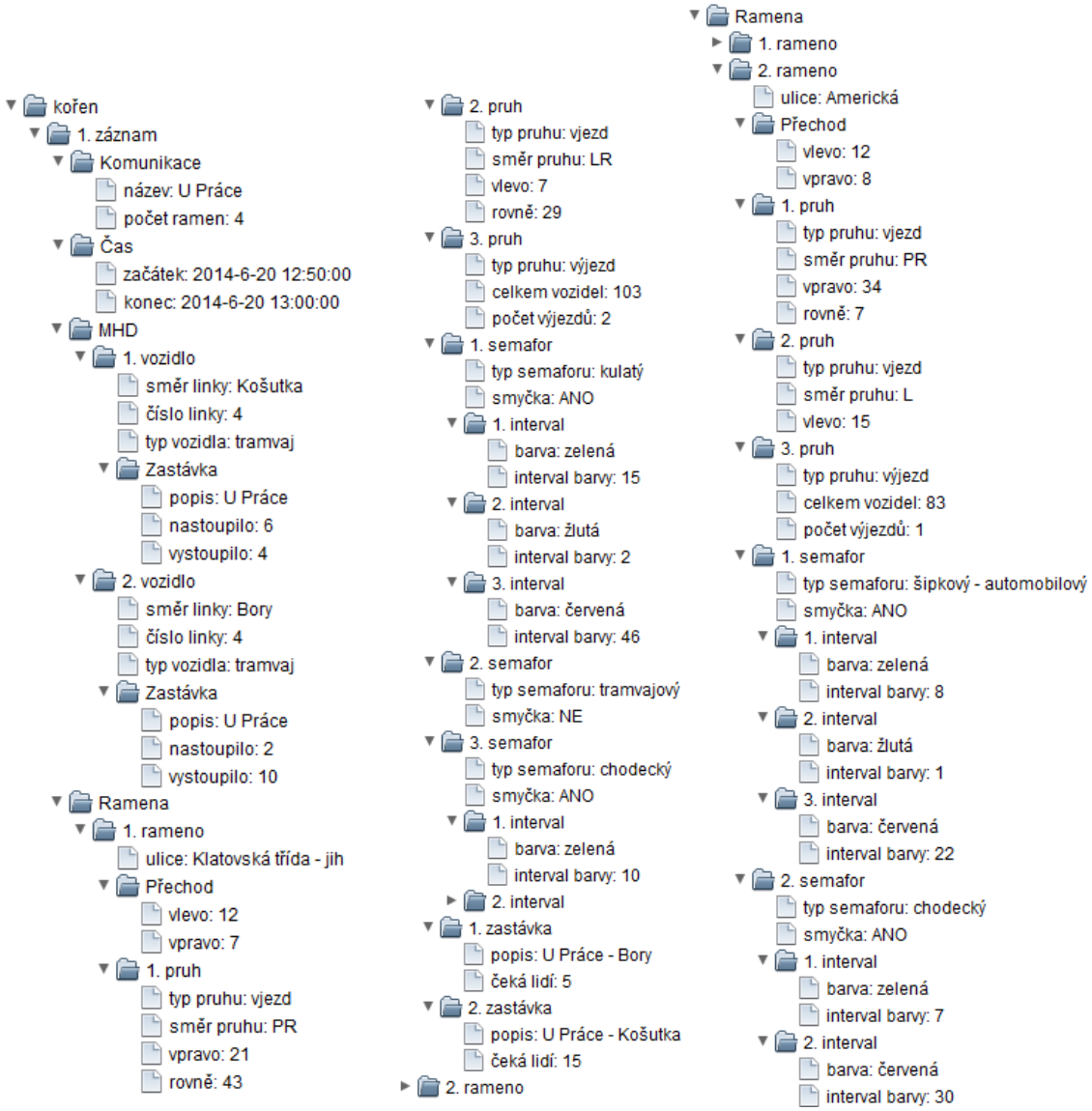
6.3 Kontrola výpisu

Při kontrole načtených vzorových souborů vidíme, že obě načtení vytvořila stejnou stromovou strukturu (viz. Obr. 30). Je tak jednoznačně vidět, že oba dva druhy načítání zpracovávají data stejně, a vytváří tak totožné záznamy.

6.4 Vyhledávání

Když si jeden z načtených vzorových souborů uložíme do databáze (je jedno jaký, oba vytvoří stejné záznamy), můžeme v datech potom vyhledávat. Vyhledat lze konkrétní křižovatku a její ramena nebo MHD na konkrétní křižovatce. Vyhledávač je *case*

sensitive, což znamená, že rozeznává velká a malá písmena. Je tedy rozdíl mezi „U náměstí míru“ a „u náměstí míru“. Jsou to dva odlišné řetězce.

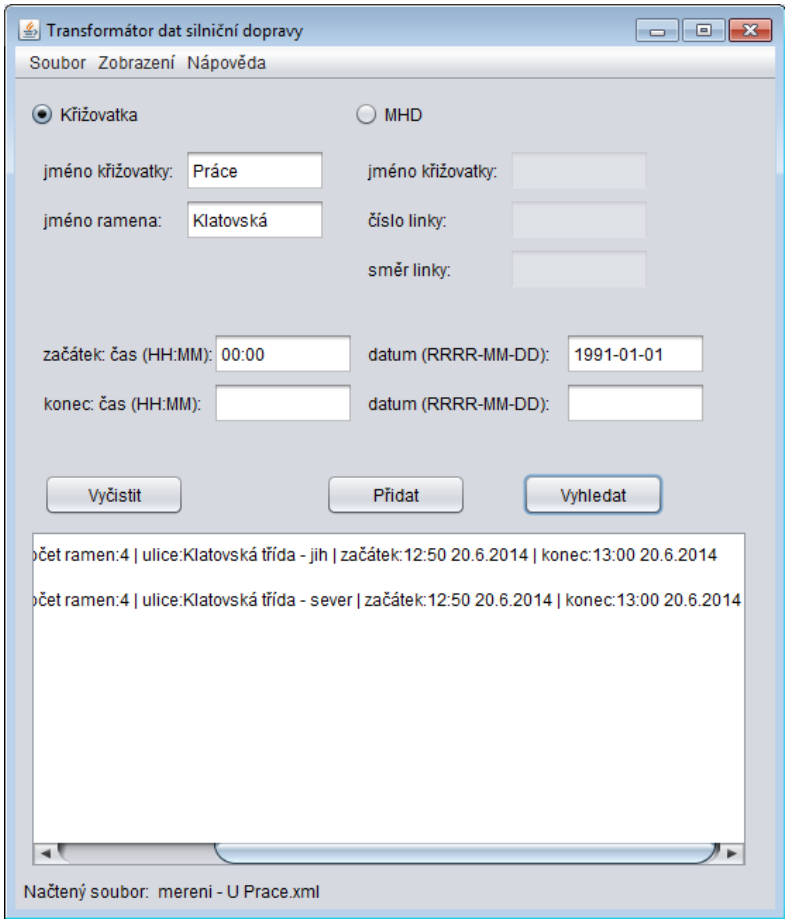


Obr. 30:Zobrazení načtených dat ve stromové struktuře

6.4.1 Vyhledání křižovatky

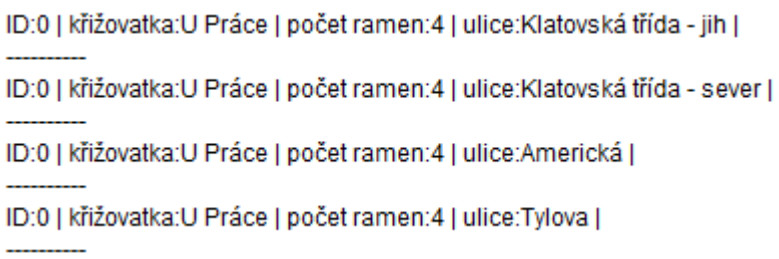
Křižovatka jde vyhledat podle názvu. Ten nemusí být úplný, program najde název i podle částečně zadaného vstupu. Při zadání pouze jména křižovatky nám program najde všechny shody. Navíc řekne, která ramena jsou ke křižovatce v databázi přiřazena. Pokud bychom tedy chtěli najít pouze jedno rameno, stačí vyplnit pole *jméno ramena* a dát *Vyhledat*. Zobrazí se nám výsledky křižovatek a ramen odpovídající zadanému názvu.

Pokud tedy chceme najít křižovatku „U Práce“, stačí zadat do okénka *jméno křižovatky* celé jméno nebo jen jeho část. Výsledkem bude výpis jména křižovatky a všech ramen, které v ní leží (viz. Obr. 32).



Obr. 31:Výsledek vyhledávání podle – jméno křižovatky, jméno ramena, začátek měření

Při vyplnění políčka *jméno ramena* například hodnotou „Klatovská“ se nám výběr omezí už jen na „Klatovská – jih“ a „Klatovská – sever“.



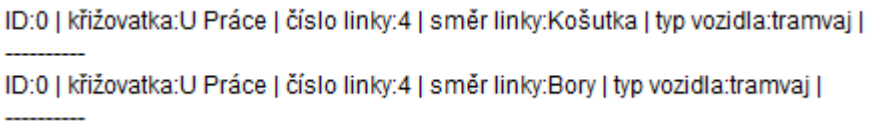
Obr. 32:Výsledek hledání podle - jméno křižovatky - Křižovatka

Při hledání podle času si musíme dávat pozor na formát, ve kterém čas zadáváme. Jak má daný tvar vstupu vypadat, je zobrazeno hned vedle políčka pro zadání vstupu. Při

vyplnění celého času a data začátku měření se hledají všechny záznamy s časem kalendářně vyšším (viz. Obr. 30). Pro stejný druh vyplnění, ale u konce měření se hledají data podle kalendáře s nižší hodnotou. Vyplněním jednoho políčka času ze čtyř se budou hledat záznamy přesně odpovídající zadané hodnotě. Při vyhledávání v určitém časovém rozmezí musí být vyplněna všechna 4 pole.

6.4.2 Vyhledání MHD

Pokud budeme chtít vyhledat prostředek MHD, musíme nejdříve přepnout radiobutton na MHD. Vyhledávání probíhá podobně jako u křižovatky. Vyplněním políčka *jméno křižovatky* se nám zobrazí všechna vozidla MHD, která byla zaznamenána na této křižovatce (viz. Obr. 33).



Obr. 33:Výsledek hledání podle - jméno křižovatky - MHD

Když vyplníme *směr linky* hodnotou Bory, výsledkem hledání bude pouze druhý záznam z obrázku 33. Hledání podle času zde funguje stejně jako u křižovatky. Platí zde stejný formát a funkcionalita různých druhů zadání času.

Při vyhledávání pomocí času lze údaje různě kombinovat. Může se zadat například čas a datum začátku měření a k tomu doplnit pouze čas konce měření. Dostaneme tak výsledky, které jsou od zadaného data do budoucnosti a mají zároveň určitý čas ukončení měření. Data nelze vyhledávat křižně. Je tím myšlen čas začátku měření a datum konce měření.

6.5 Export

Pro export dat stačí vybrat v menu Soubor a Export do XML. Budeme vyzváni k zadání jména souboru. Vyplňuje se pouze jméno bez koncovky. Tu si program při generování doplní sám. Necháme si vyexportovat některá data, která si vyhledáme. Otevřeme si vyhledávání a nejprve stiskneme *Vyčistit*. Vymažeme tak z paměti programu načtený soubor. Kdybychom to neudělali, vyhledaná data by se nám připojovala k datům z načítání ze souboru. Vyhledáme si ramena křižovatky, která mají v názvu „Americká“. Výsledek vyhledávání vložíme do paměti programu tlačítkem *Přidat*. Teď

už jen Soubor -> Export do XML a jméno souboru. Stiskneme Export a námi pojmenovaný soubor se teď nachází ve složce *export*. Struktura souboru (viz. Obr. 34).

```
<?xml version="1.0" encoding="UTF-8"?>
<!--generovany vystup-->
<seznamZaznamu>
    <zaznam>
        <krizovatka nazev="U Práce" pocetRamen="4"></krizovatka>
        <cas zacatekMereni="2014-6-20 12:50:00" konecMereni="2014-6-20 13:00:00"></cas>
        <seznamRamen>
            <rameno ulice="Americká">
                <prechod vlevo="12" vpravo="8"></prechod>
                <seznamPruhu>
                    <pruh typPruhu="vjezd" smerPruhu="PR" vpravo="34" rovne="7"></pruh>
                    <pruh typPruhu="vjezd" smerPruhu="L" vlevo="15"></pruh>
                    <pruh typPruhu="vyjezd" smerPruhu="" celkemProjelo="83" pocetVyjezdu="1"></pruh>
                </seznamPruhu>
                <seznamSemaforu>
                    <semafor typSemaforu="šipkový - automobilový" indukcniSmycka="ano">
                        <interval barva="zelená" trvaniIntervalu="8"></interval>
                        <interval barva="žlutá" trvaniIntervalu="1"></interval>
                        <interval barva="červená" trvaniIntervalu="22"></interval>
                    </semafor>
                    <semafor typSemaforu="chodecký" indukcniSmycka="ano">
                        <interval barva="zelená" trvaniIntervalu="7"></interval>
                        <interval barva="červená" trvaniIntervalu="30"></interval>
                    </semafor>
                </seznamSemaforu>
                <zastavka popis="U Práce - do křižovatky" cekaLidi="9"></zastavka>
                <zastavka popis="U Práce - od křižovatky" cekaLidi="12"></zastavka>
            </rameno>
        </seznamRamen>
    </zaznam>
</seznamZaznamu>
```

Obr. 34:Struktura vyexportovaného souboru

7 Závěr

Cílem práce bylo vytvořit program, který umí načítat soubory typu `txt` a `xml` s určitou strukturou dat. Tato data měl umět zpracovat do takové podoby, aby je bylo možné prohlížet, ukládat a exportovat. Uložená data navíc musí umět do určité míry procházet a umožňovat jejich export do `xml`.

Program v konečné fázi umí načítat oba druhy souborů. Je dána struktura souborů, podle které by se měly soubory vytvářet, aby je program dokázal bezpečně a správně načíst. Po načtení je možné data procházet. Na výběr je strukturovaný výpis a stromová struktura. Dále je možné data uložit do vnořené databáze nebo je nechat vyexportovat. Export má jednotnou strukturu pro všechna data. Ve vnořené databázi lze vyhledávat zvlášť pozemní komunikace a městskou hromadnou dopravu. Komunikace lze hledat podle jména křižovatky, jména ramena křižovatky nebo podle časového intervalu. MHD lze dohledat podle jména křižovatky, na kterém bylo zjištěno, směru linky (konečná stanice), čísla linky a časového intervalu, ve kterém bylo MHD změřeno.

Reference

1. **Graves, Steve.** *COTS database for embedded systems.* místo neznámé : Embedded Computing Design magazine, 2007.

2. *Root.* [Online] 3. 9 2004. [Citace: 7. 12 2013.] www.root.cz/clanky/embedded-database-uvod/.

3. *4ITdevelopers.* [Online] 3. 8 2006. [Citace: 7. 12 2013.] www.4itdevelopers.net/-embedded_databaze.aspx.

4. Malina, Martin. *Kompatibilita JBoss AS s open source databázemi.* Brno : Masarykova Univerzita, 2008.

5. Poznejte Firebird za 2 minuty. *Firebird News.* [Online] [Citace: 11. 6 2014.] http://www.firebirdnews.org/docs/fb2min_cz.html.

6. Presentation and use of H2 Database Engine. *Baptiste Wicht.* [Online] 06. 08 2010. [Citace: 8. 12 2013.] http://baptiste-wicht.com/posts/2010/08/presentation-usage-h2-database-engine.html.

7. *H2 Database Engine.* [Online] [Citace: 8. 12 2013.] www.h2database.-com/html/main.html.

8. 2.3.2 released! *HyperSQL.* [Online] 11. 3 2013. [Citace: 19. 12 2013.] http://hsqldb.-org/.

9. HyperSQL. *HSQLDB.* [Online] 8. 10 2013. [Citace: 19. 12 2013.] hsqldb.org.

10. Chapter 9. SQL Syntax. *HyperSQL.* [Online] 7. 11 2013. [Citace: 19. 12 2013.] http://hsqldb.org/doc/guide/ch09.html.

11. HyperSQL DataBase (HSQLDB). *ITnetwork.cz.* [Online] [Citace: 10. 6 2014.] http://www.itnetwork.cz/java-hsqldb-hypersql-database-tutorial.

12. *Apache Derby.* [Online] 5. 12 2013. [Citace: 8. 12 2013.] db.apache.org/derby/-index.html.

13. Křižovatky a křížení. *Katedra dopravního stavitelství, Fakulta stavební, VŠB-TU Ostrava.* [Online] [Citace: 2. 6 2014.] http://kds.vsb.cz/ord/krizovatky-pojmy.htm.

14. křižovatky. *Urbanistické středisko Ostrava, s.r.o.* [Online] [Citace: 31. 5 2014.] http://www1.uso.cz/public/SKV02/ARCH/5_KRIZOVATKY_UROVNOVE.pdf.

15. Vodní a dopravní stavby. *Vysoká škola báňská - Technická univerzita Ostrava - Fakulta stavební.* [Online] [Citace: 1. 6 2014.] http://fast10.vsb.cz/krajcovic/-!kombinovane/!dopravni_a_vodni_stavby/pomucky_k_reseni/pdf/KRIZOVATKY_PK_KOMBI.pdf.

16. Česká společnost pro stavební právo. *Česká společnost pro stavební právo.* [Online] [Citace: 8. 6 2014.] spolstavprav.cz/sts_notlib_docs/text_čnot%202013_0108_GR.docx.

17. V Lounech nastaví semafory tak, aby se chodcům lépe přecházelo. *Žatecký a Lounský deník.cz.* [Online] [Citace: 12. 6 2014.] http://zatecky.denik.cz/galerie/-foto.html?mm=semafor_silnice_cervena_sipka&s=45.

18. Cyklisté dostávají rovnoprávnost s chodci. *Horydoly.cz.* [Online] [Citace: 12. 6 2014.] http://www.horydoly.cz/foto/cyklozmeny_2011/ipage00005.htm.

19. Vysvětlení funkce předsignálů, výzvových signálů a preference tramvají. *Správa veřejného statku města Plzně.* [Online] [Citace: 12. 6 2014.] http://www.svsmp.cz/-svetelna-signalizace/vysvetleni-funkce-predsignalu-vyzvovych-signalu-a-preference-tramvaji.aspx.

20. Pavel Drdla Univerzita Pardubice / Dopravní fakulta Jana Pernera / Katedra technologie a řízení dopravy. [Online] [Citace: 9. 6 2014.] http://www.drdla.-wz.cz/skripta/4.pdf.

21. *Katedra dopravího stavitelství - Fakulta stavební - VŠB-TU Ostrava.* [Online] Technická základna MHD - stručné shrnutí. [Citace: 9. 6 2014.] http://kds.vsb.cz/mhd/-ostatni-tz.htm.

22. AUTOBUSOVÉ A TROLEJBUSOVÉ ZASTÁVKY. *Vysoká škola báňská - Technická Univerzita Ostrava.* [Online] [Citace: 10. 6 2014.] http://fast10.vsb.cz/-mahdalova/MHD/predna02.pdf.

Seznam obrázků

Přílohy

A Uživatelská příručka

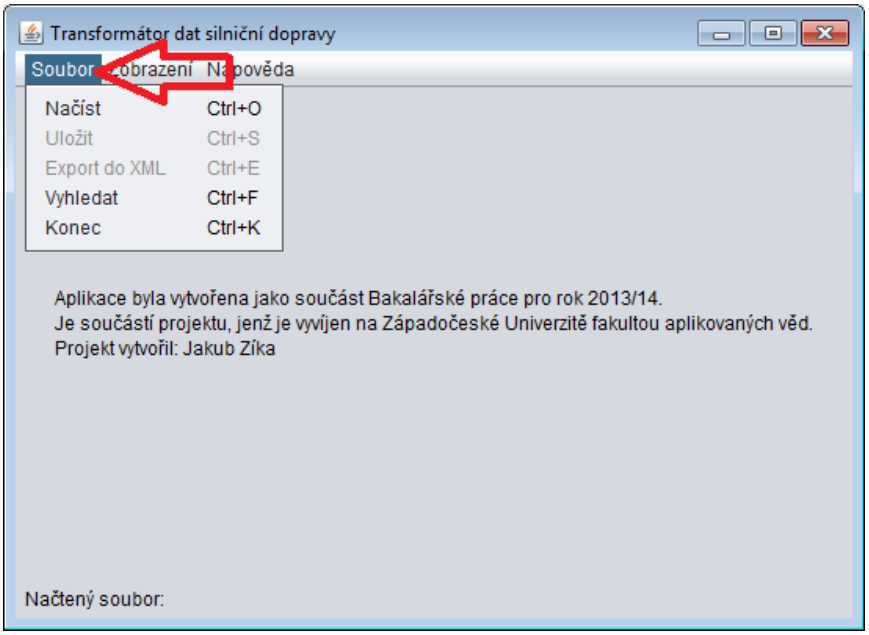
A.1 Umístění a nároky

Program je umístěn na CD, které je součástí dokumentu. Médium se nachází na zadní desce v přihrádce. Obsahuje readme.txt a složky Text_prace, Program a Testovaci_soubory. Readme.txt je stručný popis, co se kde nachází. Text_prace obsahuje elektronickou formu psané práce. Ve složce Program jsou umístěny všechny komponenty potřebné pro spuštění aplikace. Testovaci_soubory obsahuje soubory k procvičení ovládání programu. Je nutné, aby počítač, na kterém bude aplikace spuštěna, měl nainstalovanou Javu verze 7 nebo vyšší.

A.2 Konfigurace a spuštění

Abychom mohli aplikaci spustit, musíme ji nejdříve zkopírovat z CD do počítače. Otevřeme CD a složku Program si zkopírujeme kamkoliv do počítače. Pak už jen otevřeme složku a můžeme aplikaci spustit. Spuštění se provede poklikem na TransformatorDat(.jar).

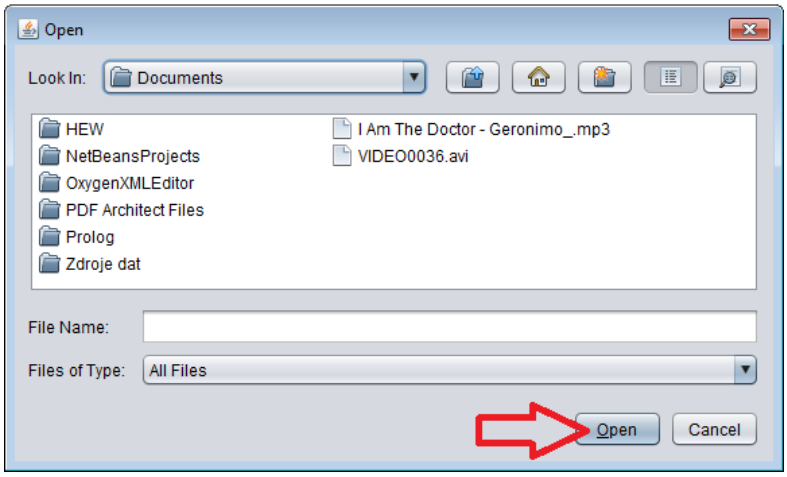
A.3 Načítání souboru



Obr. 35:Hlavní okno

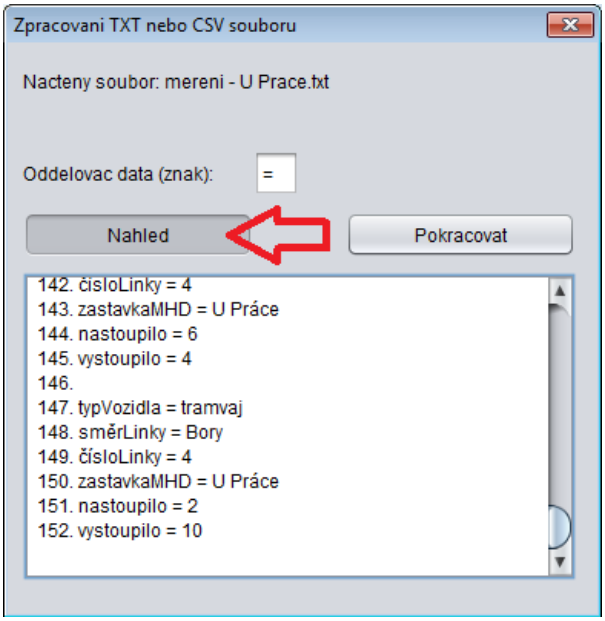
Zobrazí se hlavní okno programu, ve kterém je v horní polovině umístěno menu (viz. Obr. 35). Rozkliknutím nabídky *Soubor* máme možnosti *Načíst, Uložit, Exportovat do*

XML, *Vyhledat* a *Konec*. Uložit a Exportovat do XML jsou šedivé, protože po spuštění nemáme žádná data v programu, které bychom mohli uložit nebo exportovat.



Obr. 36:Okno pro výběr souboru k načtení

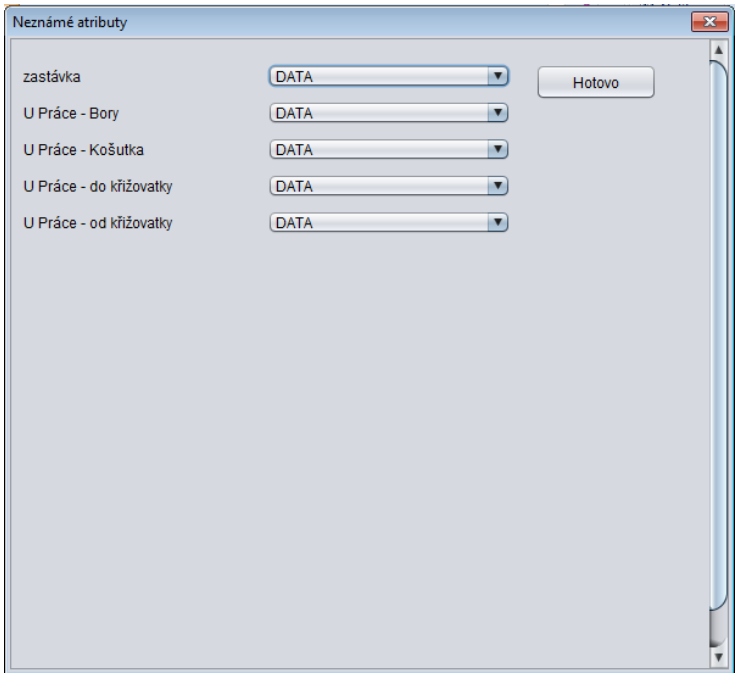
Zvolením možnosti Načíst se objeví okno pro zadání souboru k načtení do programu. Lze načíst soubory typu TXT a XML. Aby byl soubor správně načten, musí mít určitou strukturu zápisu dat (více v kapitolách A.7, A.8).



Obr. 37:Zobrazení náhledu souboru

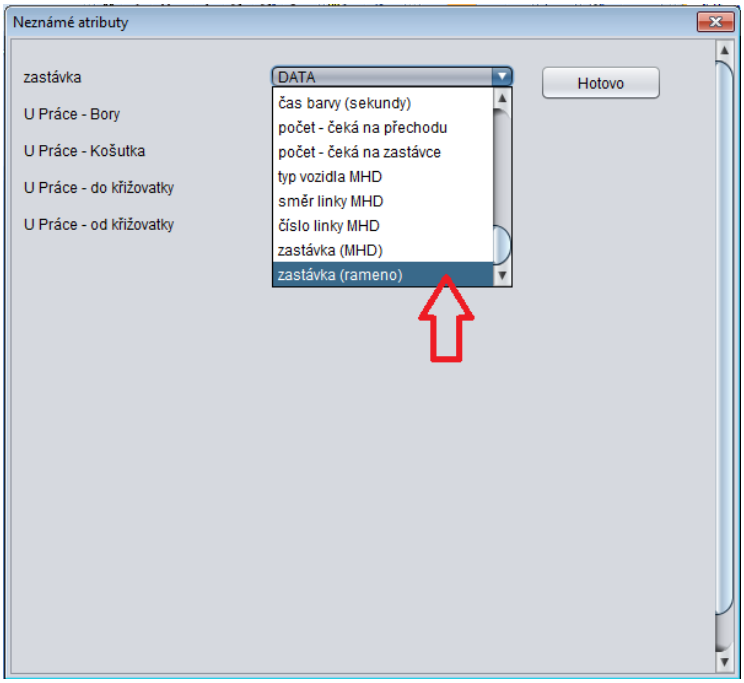
Pro rychlé vyzkoušení načítání jsou na CD ve složce Testovaci_soubory připraveny dva soubory. Jeden je typu TXT a druhý XML. Oba obsahují stejná data, ale každý je zapsán tak, aby splňoval strukturu načítání dat pro svůj formát.

Jako první si vybereme soubor *mereni - U Práce.txt*. Jakmile se k němu proklikáme, dáme *Open* a pokračujeme v načítání.



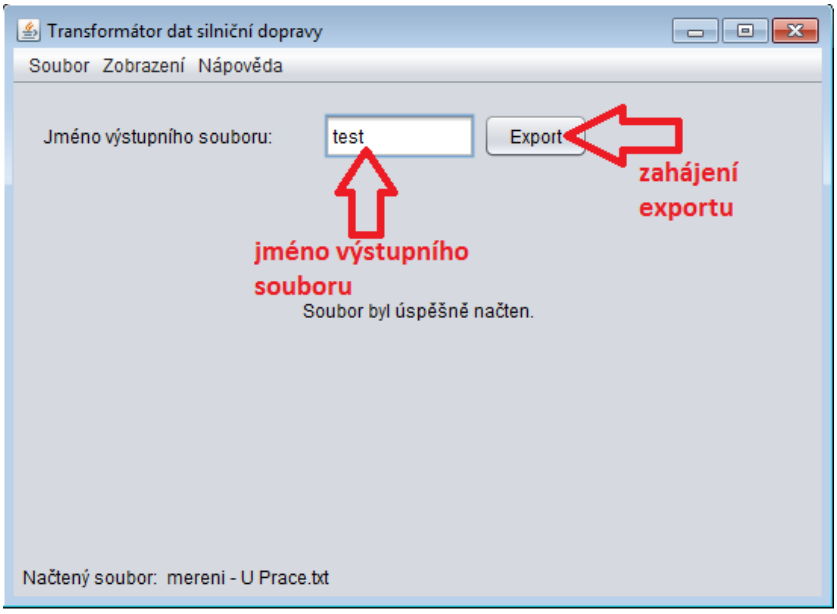
Obr. 38:Okno pro upřesnění názvů

Objeví se výzva pro zadání oddělovacího znak (znak oddělující klíč a hodnotu na řádku v souboru `txt`).



Obr. 39:Nastavení významu slova

Jak soubor uvnitř vypadá, si můžeme prohlédnout stisknutím tlačítka *Náhled* (viz. Obr. 37). Po vyplnění oddělovacího znaku, pokračujeme v načítaná tlačítkem *Pokračovat*. Dostaneme se tak k upřesnění názvů hodnot, které program nezná (viz. Obr. 38). Upřesňují se pouze ta slova, která v souboru popisují význam hodnot. Hodnoty se neupřesňují. Jak vidíme na obrázku (viz. Obr. 38), jediné slovo uvozující hodnotu, je *zastávka*.



Obr. 40:Export souboru

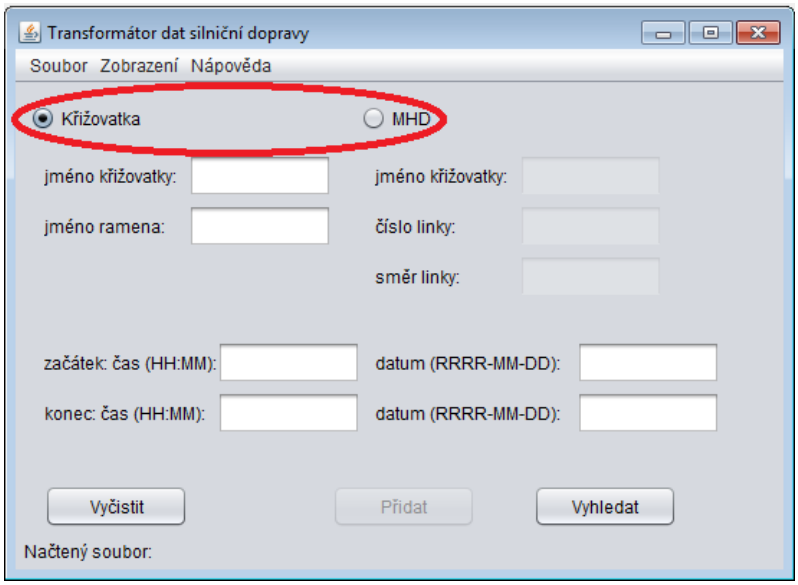
Proto pro ni rozbalíme nabídku a vybereme možnost *zastávka (rameno)* (viz. Obr. 39). Tím programu řekneme, jak má chápat význam tohoto slova. Ostatní slova jsou hodnotami. Pro hodnoty necháme nastavenou položku *DATA*. Ta programu říká, že tyto údaje jsou hodnoty a tím pádem si nemá jejich tvar zapamatovat. Jakmile jsou určeny všechny významy slov, dokončí se načtení souboru stiskem tlačítka *Hotovo*. Program se vrátí k úvodnímu oknu a uživateli hláškou na ploše oznámí, že byl soubor načten. Nyní můžeme s daty provádět další operace.

A.4 Export dat

Můžeme si také všimnout, že už lze použít tlačítka Uložit a Exportovat do XML. V tom případě si data uložíme do programu stisknutím Uložit. Po uložení nám program opět ohlásí, že se data uložila do programu. Stiskem tlačítka Exportovat do XML nás program vyzve zadat jméno výstupního souboru, pojmenujeme si ho třeba „test“ a zvolíme *export* (viz. Obr. 40). Soubor *test.xml* je nyní umístěn ve složce *export*.

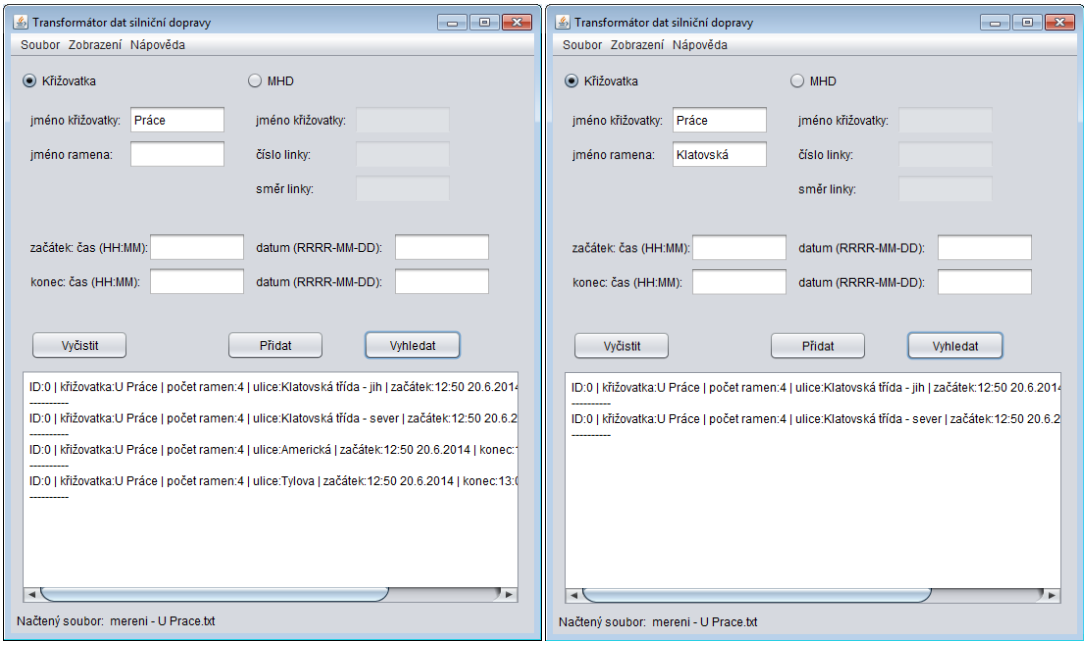
A.5 Vyhledávání dat

Jakmile jsou data uložená v programu, lze v nich vyhledávat. Vyhledávání spustíme přes Soubor->Vyhledávání. Lze vyhledávat dva druhy informací. Těmi jsou pozemní komunikace a všechny informace o nich nebo MHD a všechny informace o nich. Tento výběr je specifikován přepínačem v horní části okna (viz. Obr. 41).



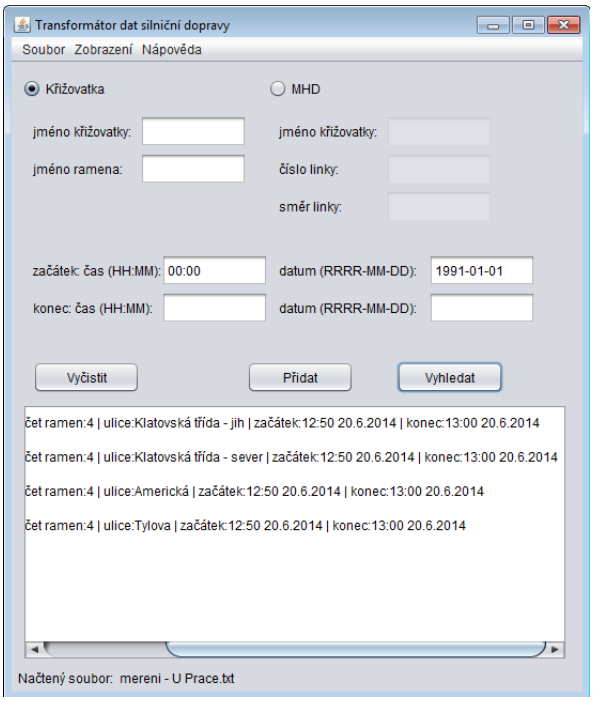
Obr. 41:Přepínač mezi křižovatkou a MHD

Vyplníme-li *jméno křižovatky*, vyhledáváme křižovatky podle jména. U jména křižovatky se nám objeví i všechna ramena, která k ní patří.



Obr. 42:Výsledky vyhledávání

Vyplněním *jméno ramena*, hledáme ramena křižovatek s tímto názvem. U MHD vyplnění *jméno křižovatky* vyhledáváme MHD měřené na této křižovatce. *Číslo linky* najde MHD s tímto číslem linky. Pro *směr linky* najde program MHD s cílovou stanicí rovnající se směru linky. Vyhledávání je *case sensitive* tzn., že „náměstí“ a „Náměstí“ jsou dvě úplně rozdílná slova. Do pole vyhledávače není potřeba zadávat celý název. Zobrazí se ty výsledky, kterým se zadaný řetězec rovná nebo je jejich částí (viz. Obr. 42). Toto neplatí jen pro čísla (číslo linky).



Obr. 43:Vyhledávání podle času

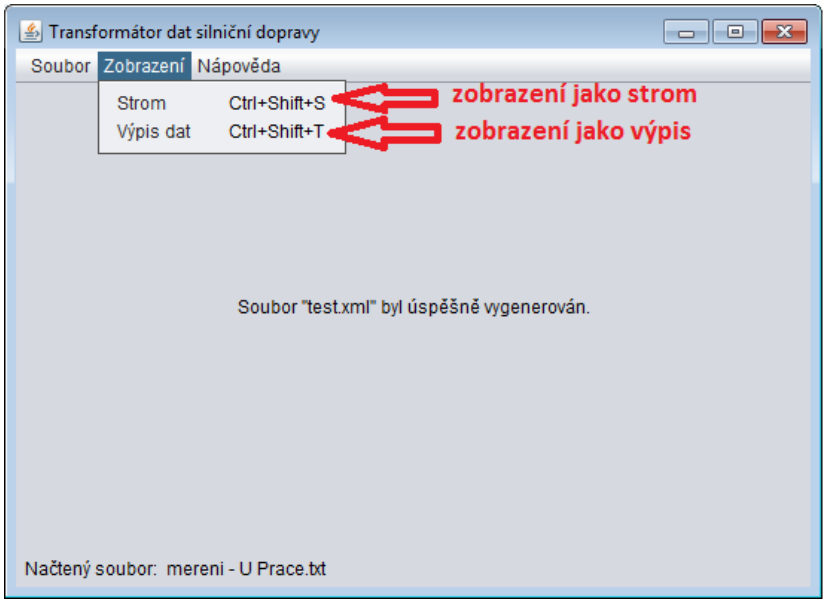
Při vyhledávání podle času se musí dodržet formát vstupu. Ten je vždy uveden nalevo od políčka pro zadání času. Při zadání času a data začátku měření, budou nalezeny všechny záznamy od tohoto data dále do budoucnosti (viz. Obr. 43). Pokud stejným stylem vyplníme konec měření, zobrazí se výsledky od tohoto data směrem do minulosti. Vyplněním celého začátku i konce měření se vytvoří časový interval. Všechny nalezené výsledky pak byly měřeny v tomto časovém intervalu.

Vyplněním pouze jednoho pole hledáme přímou shodu s datem nebo časem. Při vyhledávání pomocí času lze údaje různě kombinovat. Můžeme zadat například čas a datum začátku měření a k tomu doplnit pouze čas konce měření. Dostaneme tak výsledky, které jsou od zadaného data do budoucnosti a mají zároveň určitý čas

ukončení měření. Data nelze vyhledávat křižně. Je tím myšleno zadání času začátku měření a data konce měření najednou.

A.6 Prohlížení dat

Načtená data si můžeme v programu prohlížet. Rozkliknutím záložky *Zobrazení* dostaneme na výběr ze dvou možností. Můžeme data zobrazit jako výpis nebo jako stromovou strukturu.



Obr. 44:Možnosti prohlížení dat

V menu poslední záložka *Nápověda* pouze zobrazuje obecné informace o aplikaci. Tento výpis se zobrazuje při spuštění na ploše aplikace.

A.7 Struktura TXT

Při vytváření vlastního souboru typu `txt` se musí dodržet několik základních pravidel. Soubor musí být v kódování UTF-8. Pokud tomu tak nebude, česká interpunkce nebude správně rozpoznána.

```
typSemaforu = chodecký
indukčníSmyčka = ano
barva = zelená
trváníBarvy = 10
barva = červená
trváníBarvy = 70
```

Obr. 45:Zadání semaforu (zkopírováno z testovacího souboru)

Data jsou zadána ve formátu *klíč – hodnota*, například: „křižovatka = U Práce“. Na každém řádku je jedna trojice klíč-oddělovač-hodnota. Oddělovač je vždy mezi klíčem a hodnotou. Může být zvolen jakýkoliv znak, který se nevyskytuje v textu s jiným významem. Oddělovač musí být v celém souboru jednotný. Na začátku souboru musí být jméno křižovatky a čas měření. Pokud to tak nebude, načtení souboru se neprovede. Důležité informace musí být u sebe, například popisujeme-li semafor, musí hned pod první zmínkou o něm být i ostatní informace, které se k němu vztahují.

Celá struktura souboru se všemi atributy, které lze zapsat je k vidění v testovacím souboru *mereni – U Práce.txt*. Některé zápisy atributů, lze zapsat i jiným způsobem. Prvky klíč a hodnota, lze v zápise otočit, na načítání to vliv mít nebude. Není tedy rozdíl v zápisu *křižovatka – U Práce* a *U Práce – křižovatka*.

Výjimka pro zápis se týká hlavně atributů zpracovávající směry pruhu. Testovací soubor obsahuje zápis `směrPruhu = RP`, tuto informaci lze zapsat i takto `směrPruhu = vpravo = rovně`, výsledek načtení bude stejný. Program si zkratku RP po načtení sám sestaví. Hodnoty jednotlivých směrů se dají také zapsat různě. Můžeme zapsat dvojici *směr = hodnota*, nebo *projelo = směr = hodnota*.

```
směrPruhu = vpravo = rovně
vpravo = 12     < ekvivalentní zápis >     projelo = vpravo = 12
rovně = 10      < ekvivalentní zápis >     projelo = rovně = 12
```

Obr. 46:Ekvivalentní zápis pruhu

Směr vpravo, vlevo může být zapsán jako R a P nebo v angličtině. Všechny tyto typy zápisu program bezpečně rozezná. Složený směr z LPR nemusí mít při zápisu jasné pořadí písmen. Lze zapsat `směr pruhů = RP` nebo `směr pruhů = PR`.

Nemusíme se bát, že pokud prohodíme některé z písmen, data by se mu nenačetla. Program umí rozpoznat všechny kombinace těchto znaků. Pokud bychom zaznamenali vozidla pruhu RP, jak stojí v řadě, ale už nevíme, které vozidlo kam odbočilo, můžeme zadat celkový počet. Lze zapsat `směrPruhu = vpravo = rovně = 16` nebo `směrPruhu = vpravo = rovně` a na další řádek `projelo = 16`.

A.8 Struktura XML

Soubor typu `xml` se skládá z elementů, které jsou do sebe postupně zapouzdřené. Zde tedy data zapisujeme do elementů a připisujeme k nim atributy. Atributy budou data,

která popisují vlastnosti elementu. Pokud bych chtěl například popisovat dopravní vozidlo a jeho vlastnosti, jméno elementu bude vozidlo a jeho vlastnosti (typ vozidla, barva, počet kol, atd.) budou atributy elementu.

```
<rameno jmenoUlice=”Americká”>
     <pruhVjezd smer=”PR”>
          <pocetPrujezdu projelo=”5” smer=”P” />
          <pocetPrujezdu projelo=”10” smer=”R” />
     </pruhVjezd>
</rameno>
```

Obr. 47:Ukázka zapouzdření elementů

Důležité je, aby podobně jako u `txt` byla v souboru informace o křižovatce. Nemusí zde být jméno, postačí element pojmenovaný například *křižovatka*. Dále v soubor musí obsahovat údaj o čase, kdy bylo měření provedeno. Příklad struktury zápisu všech elementů a atributů je vidět v testovacím souboru *mereni – U Prace.xml*. Nadřazený element má v sobě zapouzdřené elementy, které ten nadřazený popisují. Například rameno se skládá z pruhů a pruhy mají nějaké vlastnosti.

```
<krizovatka>
  <nazev>U náměstí míru</nazev>
  <pocetRamen>4</pocetRamen>
  <zacatekMereni>2013-03-10 11:54</zacatekMereni>
  <konecMereni>2013-03-10 12:09</konecMereni>
  <rameno>
    <jmenoUlice>Klatovská - jih</jmenoUlice>
    <pruhVjezd oznaceni="11" sipky="RP">
      <pocetPrujezdu projelo=”5” smer=”P” />
      <pocetPrujezdu projelo=”10” smer=”R” />
    </pruhVjezd>
  </rameno>
<krizovatka>
```

Obr. 48:Přepis atributu na element

Při vytváření souboru musíme hlavně dodržovat základní pravidla pro tvorbu `xml`. Elementy se nesmí křížit, pouze zanořovat. Každý element musí být řádně ukončen, řádkový element na konci lomítkem a párový koncovým elementem. Hodnoty atributů jsou vždy v uvozovkách. Stejně jako u `txt` i zde lze některá data zapsat jinak, než je tomu v testovacím souboru. Název křižovatky nemusí být zapsán jako atribut elementu,

ale lze jej osamostatnit do párového elementu. To samé lze udělat i s počtem ramen. Podobným způsobem lze zaznamenat čas měření a jméno ulice. Pro *pocetPrujezdu* také platí vice možností zápisu. Jak je vidět v ukázce, počet projetých vozidel je zaznamenán v atributu *projelo*.

```
<pocetPrujezdu smer="P">5</pocetPrujezdu>
<pocetPrujezdu smer="R">10</pocetPrujezdu>
```

Obr. 49:Ekvivalentní zápis pro počet průjezdů pruhem

Pokud budete vytvářet soubory podle testovacího souboru, je zde několik pravidel, kterými se musíte řídit. Atributy v elementu mohou být v jiném pořadí, než se v testovacím souboru. Nesmí být ovšem zapsány u jiného elementu. Vnořené elementy mohou mít různé pořadí. Nesmí být na stejné nebo vyšší úrovni zapouzdření, jako nadřazený element. Na začátku každého souboru uvedeno kódování. Program bezpečně načte pouze kódování UTF-8. Soubor musí obsahovat elementy o křižovatce a čase. Ostatní druhy elementů se v souboru mohou vyskytovat, ale nejsou podmínkou pro správné načtení.

Pro oba typy souboru platí stejný tvar zápis času. Čas musí mít přesný tvar, jinak nebude programem načten. Záznam musí obsahovat rok, měsíc, den hodinu a minutu měření. Formát, který program akceptuje je následovný:

RRRR-MM-DD HH:MM -> rok-měsíc-den hodiny-minuty -> 2014-06-24 01:21

Program dokáže akceptovat zápis, kdy bude datum a čas prohozen. Jiné alternativy zápisu neexistují.

A.9 Ant

Ve složce Program se nachází soubor *build.xml*. Tento soubor slouží k překladu programu a obsahuje ještě několik doplňujících funkcí. Soubor dokáže program odstranit, znovu přeložit a obnovit původní nastavení souborů s ekvivalenty. Soubory s ekvivalenty obsahují neznámá slova, která si program zapamatoval po určení významu uživatelem.

Kdybychom chtěli program smazat, zadáme do příkazového řádku příkaz: `ant clean`. Ve složce tak zůstane pouze *build.xml* a složka *konfiguracniSoubory*. Abychom mohli program znovu používat, musíme ho znovu přeložit. To provedeme příkazem:

`ant`. Program se znovu přeloží, vytvoří potřebné složky, spustitelný jar soubor a vygeneruje dokumentaci. Dostaneme se do stavu, ve kterém jsme teď. Obnovení původního nastavení souborů s ekvivalenty provedeme příkazem: `ant restart`.

B Testovací soubory

B.1 Soubor TXT

Testovací soubor je uložen na CD ve složce Testovaci_soubory pod jménem *mereni – U Práce.txt*.

```
křižovatka = U Práce
početRamen = 4
začátekMěření = 2014-06-20 12:50
konecMěření = 2014-06-20 13:00

ramenoKřižovatky = Klatovská třída - jih
typPruhu = vjezd
směrPruhu = RP
rovně = 43
vpravo = 21
typPruhu = vjezd
směrPruhu = RL
rovně = 29
vlevo = 7
typPruhu = výjezd
početVýjezdů = 2
projelo = 103

typSemaforu = kulatý
indukčníSmyčka = ano
barva = zelená
trváníBarvy = 15
barva = žlutá
trváníBarvy = 2
barva = červená
trváníBarvy = 46

typSemaforu = tramvajový

typSemaforu = chodecký
indukčníSmyčka = ano
barva = zelená
```

```
trváníBarvy = 10
barva = červená
trváníBarvy = 70

přechod = vlevo = 12
přechod = vpravo = 7

zastávka = U Práce - Bory
čekáNaZastávce = 5
zastávka = U Práce - Košutka
čekáNaZastávce = 15

ramenoKřižovatky = Americká
typPruhu = vjezd
směrPruhu = RP
rovně = 7
vpravo = 34
typPruhu = vjezd
směrPruhu = L
vlevo = 15
typPruhu = výjezd
početVýjezdů = 1
projelo = 83

typSemaforu = šipkový - automobilový
indukčníSmyčka = ano
barva = zelená
trváníBarvy = 8
barva = žlutá
trváníBarvy = 1
barva = červená
trváníBarvy = 22

typSemaforu = chodecký
indukčníSmyčka = ano
barva = zelená
trváníBarvy = 7
barva = červená
trváníBarvy = 30
```

```
přechod = vlevo = 12
přechod = vpravo = 8

typVozidla = tramvaj
směrLinky = Košutka
čísloLinky = 4
zastavkaMHD = U Práce
nastoupilo = 6
vystoupilo = 4

typVozidla = tramvaj
směrLinky = Bory
čísloLinky = 4
zastavkaMHD = U Práce
nastoupilo = 2
vystoupilo = 10
```

Obr. 50:Obsah textového souboru

B.2 Soubor XML

Testovací soubor je uložen na CD ve složce Testovaci_soubory pod jménem mereni – U *Práce.xml*.

```
<?xml version="1.0" encoding="UTF-8"?>
<krizovatka nazev="U Práce" pocetRamen="4">
     <cas zacatekMereni="2014-06-20 12:50"
konecMereni="2014-06-20 13:00" />
     <rameno jmenoUlice="Klatovská třída - jih">
          <pruhVjezd sipky="RP">
               <pocetPrujezdu smer="R" projelo="43" />
               <pocetPrujezdu smer="P" projelo="21" />
          </pruhVjezd>
          <pruhVjezd sipky="RL">
               <pocetPrujezdu smer="R" projelo="29" />
               <pocetPrujezdu smer="L" projelo="7" />
          </pruhVjezd>
          <pruhVyjezd pocet="2">
               <pocetPrujezdu projelo="103" />
          </pruhVyjezd>
          <semafor typ="kulatý" indukcniSmycka="ano">
               <interval barva="zelená" trvaniBarvy="15" />
```

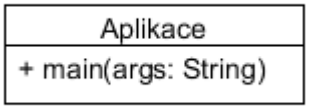
```
                <interval barva="žlutá" trvaniBarvy="2" />
                <interva barva="červená" trvaniBarvy="46" />
          </semafor>
          <semafor typ="tramvajový" />
          <semafor typ="chodecký" indukcniSmycka="ano">
                <interval barva="zelená" trvaniBarvy="10" />
                <interval brava="červená" trvaniBarvy="70"
/>
          </semafor>
          <prechod cekaVlevo="12" cekaVpravo="7" />
          <zastavka popis="U Práce - Bory" ceka="5" />
          <zastavka popis="U Práce - Košutka" ceka="15" />
     </rameno>
     <rameno jmenoUlice="Americká">
          <pruhVjezd sipky="RP">
                <pocetPrujezdu smer="R" projelo="7" />
                <pocetPrujezdu smer="P" projelo="34" />
          </pruhVjezd>
          <pruhVjezd sipky="L">
                <pocetPrujezdu smer="L" projelo="15" />
          </pruhVjezd>
          <pruhVyjezd pocet="1">
                <pocetPrujezdu projelo="83" />
          </pruhVyjezd>
          <semafor typ="šipkový - automobilový"
indukcniSmycka="ano">
                <interval barva="zelená" trvaniBarvy="8" />
                <interval barva="žlutá" trvaniBarvy="1" />
                <interva barva="červená" trvaniBarvy="22" />
          </semafor>
          <semafor typ="chodecký" indukcniSmycka="ano">
                <interval barva="zelená" trvaniBarvy="7"
indukcniSmycka="ano" />
                <interval brava="červená" trvaniBarvy="30"
/>
          </semafor>
          <prechod cekaVlevo="12" cekaVpravo="8" />
     </rameno>
     <mhd typMHD="tramvaj" cisloLinky="4"
smerLinky="Košutka">
```

```
          <zastavka popis="U Práce" nastoupilo="6"
vystoupilo="4" />
     </mhd>
     <mhd typMHD="tramvaj" cisloLinky="4" smerLinky="Bory">
          <zastavka popis="U Práce" nastoupilo="2"
vystoupilo="10" />
     </mhd>
</krizovatka>
```

Obr. 51:Obsah XML souboru

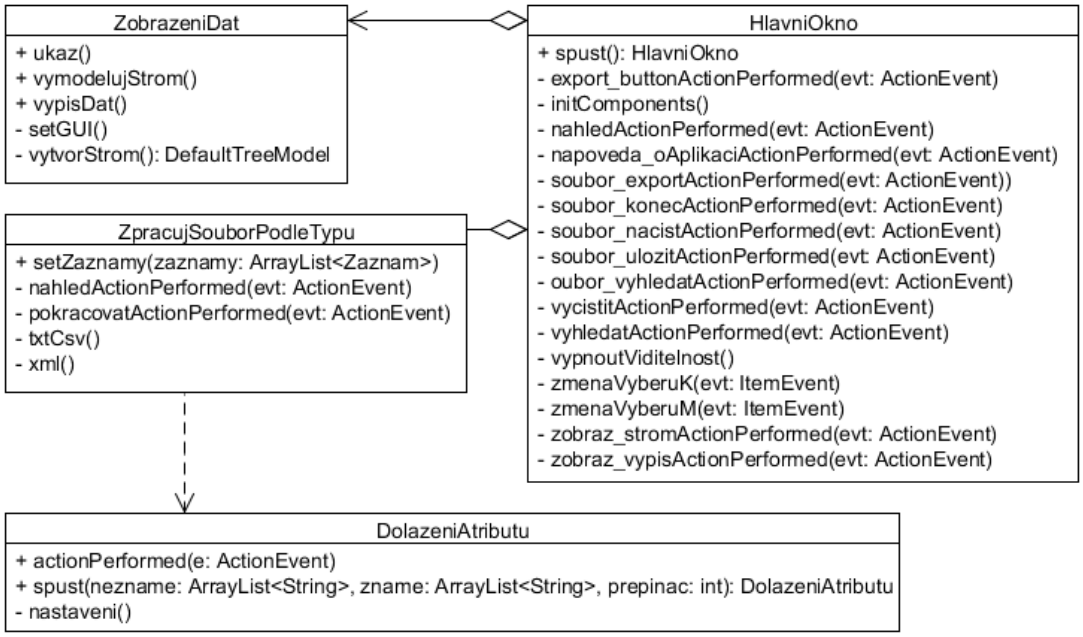
C Diagramy tříd

C.1 Diagram tříd balíku aplikace



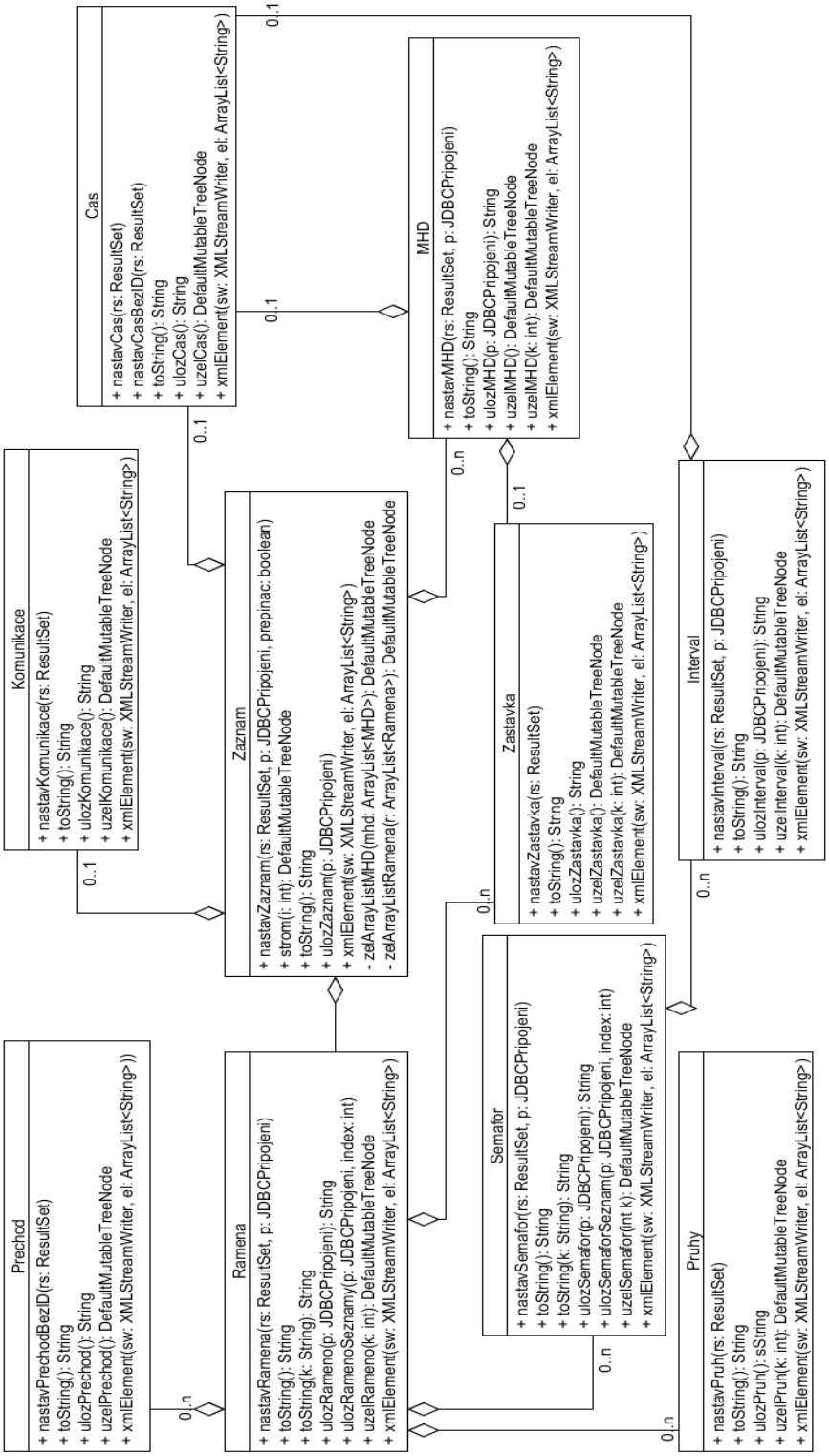
Obr. 52:Diagram tříd balíku aplikace

C.2 Diagram tříd balíku aplikace.grafika



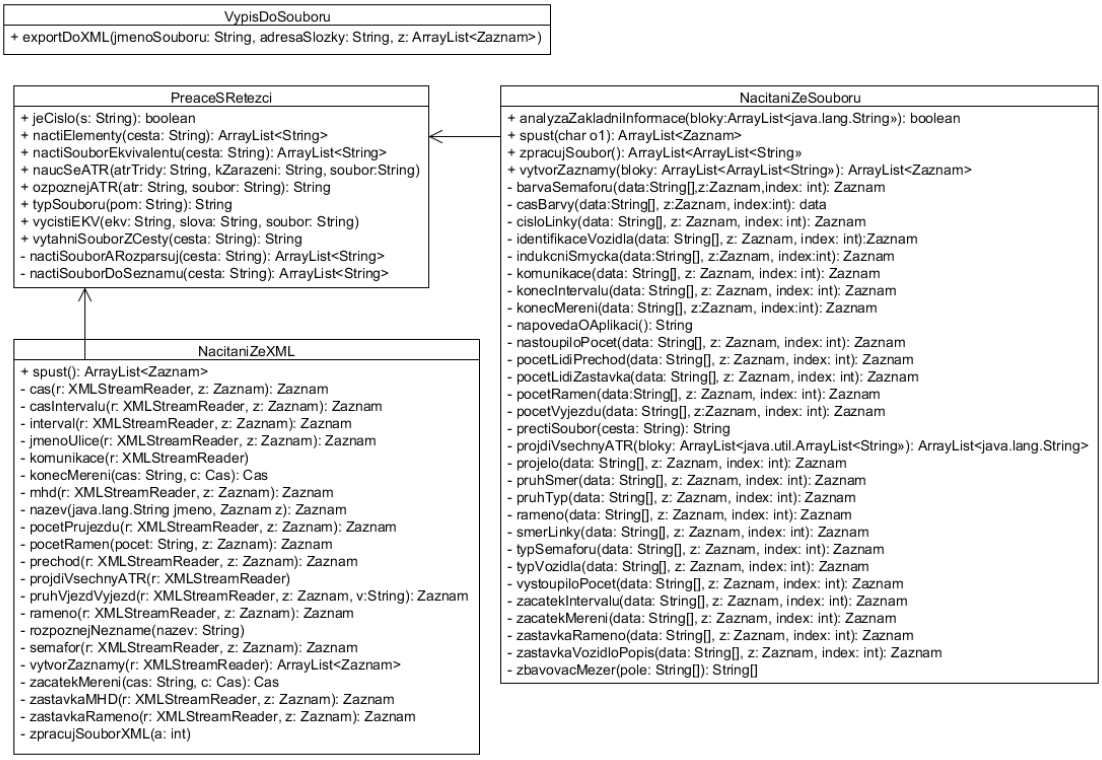
Obr. 53:Diagram tříd balíku aplikace.grafika

C.3 Diagram tříd balíku aplikace.tridyDB



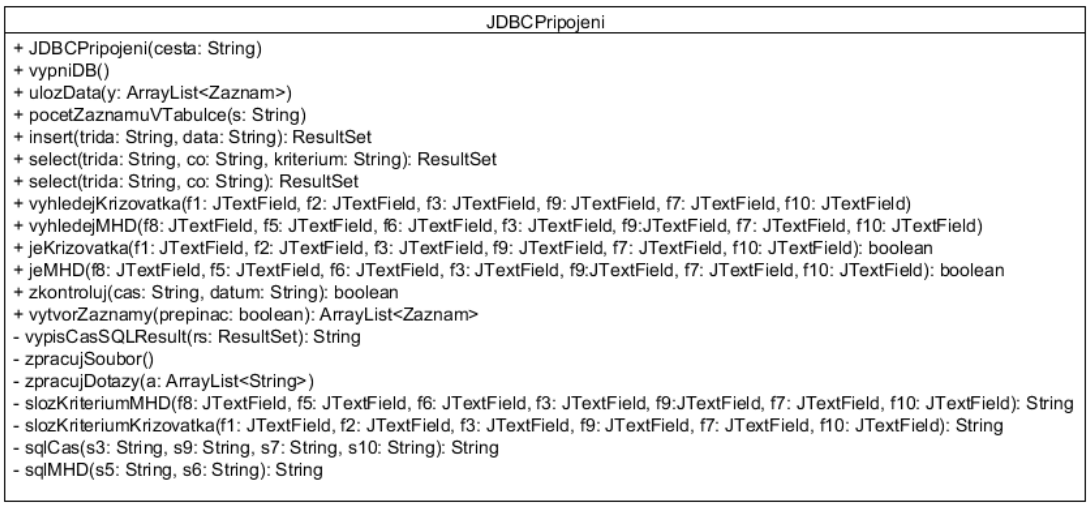
Obr. 54:Diagram tříd balíku aplikace.tridyDB

C.4 Diagram tříd balíku aplikace.data



Obr. 55:Diagram tříd balíku aplikace.data

C.5 Diagram tříd balíku aplikace.pripojeni



Obr. 56:Diagram tříd balíku aplikace.pripojeni