Univerzita Pardubice

Dopravní fakulta Jana Pernera

Aplikace pro výpočet jízdních dob vlaku

Libor Bajer

Bakalářská práce

2013

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: 2012/2013

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Libor Bajer**
Osobní číslo:	**D07003**
Studijní program:	**B3709 Dopravní technologie a spoje**
Studijní obor:	**Aplikovaná informatika v dopravě**
Název tématu:	**Aplikace pro výpočet jízdních dob vlaku**
Zadávající katedra:	**Katedra informatiky v dopravě**

Zásady pro vypracování:

Cílem práce je navrhnout a implementovat počítačovou aplikaci pro výpočet jízdních dob vlaku na dané trati/traťovém úseku v rozsahu odpovídajícím a pomocí aparátu procvičovaném v rámci předmětu Technologie a řízení železniční dopravy.
V textové části se předpokládá rešerše metod výpočtu jízdních dob počítaných bez pomoci výpočetní techniky i implementovaných v rámci existujících softwarových prostředků pro konstrukci GVD a simulačních nástrojů určených pro modelování a simulaci železniční dopravy.
V praktické části se předpokládá návrh a implementace

- systematického popisu infrastruktury ve smyslu vstupních údajů programu;
- datové struktury programu;
- výpočetního jádra.

Součástí textové části práce bude též verbální a grafický popis návrhu a uživatelská příručka pro přípravu dat a ovládání aplikace.
Vývojové prostředí Delphi (jazyk Object Pascal).

Rozsah grafických prací:

Rozsah pracovní zprávy: **minimálně 30 stran**

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

Seznam odborné literatury:

[1] **Vonka, J., Molková, T., Široký, J. Technologie a řízení dopravy II. - GVD. Pardubice : Univerzita Pardubice, 2000. ISBN 80-7194-286-3.**
[2] **ČD D23 : Služební předpis pro stanovení provozních intervalů a následných mezidobí.**

Vedoucí bakalářské práce: **Ing. Viktor Patras, Ph.D.**
OLTIS Group a. s., Hálkova 171/2, 772 00 Olomouc

Datum zadání bakalářské práce: **5. prosince 2012**
Termín odevzdání bakalářské práce: **31. května 2013**

prof. Ing. Bohumil Culek, CSc.
děkan

L.S.

doc. Ing. Josef Volek, CSc.
vedoucí katedry

V Pardubicích dne 5. prosince 2012

Prohlášení autora

Prohlašuji:

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

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 29.5.2013

Bajer

Libor Bajer

Souhrn

Předmětem bakalářské práce je vytvoření aplikace, která bude schopná vypočítat jízdní dobu vlaku na uživatelem zadané trati metodou používanou v předmětu Technologie a řízení železniční dopravy.

Před vlastním návrhem a vytvořením aplikace se práce zabývá popisem této metody a také existujícím programům, které výpočet jízdních dob umožňují.

Klíčová slova

Výpočet jízdních dob, železniční doprava, objektově orientované programování

Title

Application for calculation traveling times of train

Summary

The topic of this thesis is to create an application that will be able to calculate the travel time of train on the desired route by method used in subject Techniques and Control in railway transport.

Before the design and creation of application, this thesis describes this method of calculation and also existing programs, which allow calculation of travel time.

Keywords

Traveling times calculation, railway transport, object-oriented programming

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Viktoru Patrasovi za cenné rady, čas a připomínky, které mi umožnily napsat tuto práci.

Obsah

Seznam tabulek....8
Seznam obrázků....9
Úvod....10
1 Možnosti výpočtu pomocí SW....11
1.1 Viriato....11
1.2 Opentrack....12
1.3 iPlan....12
2 Metodika výpočtu....14
2.1 Pojmy z technologie železniční dopravy....14
2.2 Výpočet....15
2.2.1 Základní vzorce....15
2.2.2 Postup výpočtu....17
2.3 Modelové příklady....20
2.3.1 Osobní vlak....20
2.3.2 Manipulační vlak....23
3 Obecný postup pro řešení problému....26
3.1 Pojmy z objektově orientovaného programování....26
3.1.1 Postuláty OOP....26
3.1.2 Skládání a obsažení....27
3.1.3 Dědičnost a polymorfismus....27
3.1.4 Třída....28
3.1.5 Třívrstvá architektura aplikace....28
3.2 Hledání tříd....29
3.2.1 Analytický model....29
3.2.2 Návrhový model....32
3.3 Datové struktury....33
3.3.1 Pole....34
3.3.2 Seznam....35
3.3.3 Strom....36
4 Analýza a návrh aplikace....37
4.1 Analýza problému....37
4.2 Specifikace návrhových tříd a vazeb....41
4.2.1 Specifikace atributů a změny v uspořádání tříd....41
4.2.2 Použití skládání nebo obsažení....42
4.2.3 Použití kolekcí....43
4.2.4 Specifikace metod....43
4.2.5 Třída TVypocet....44
5 Realizace aplikace....46
5.1 Seznámení s programem....46
5.1.1 Editace tratě....47
5.2 Výpočet....48
6 Porovnání výstupu....49
Závěr....51
Použitá literatura....52

Seznam tabulek

Tabulka 1 Typy vozidlového odporu 15
Tabulka 2 Dílčí úseky - osobní vlak 22
Tabulka 3 Jízdní doby - osobní vlak 23
Tabulka 4 Dílčí úseky - manipulační vlak 24
Tabulka 5 Jízdní doby - manipulační vlak 25
Tabulka 6 Symboly násobností 31
Tabulka 7 Porovnání výsledků - osobní vlak 49
Tabulka 8 Porovnání výsledků - manipulační vlak 49

Seznam obrázků

Obrázek 1 Rozklad sil na sklonu 16
Obrázek 2 Diagram rychlostí - osobní vlak 21
Obrázek 3 Diagram rychlostí - manipulační vlak 24
Obrázek 4 Třívrstvá architektura aplikace 29
Obrázek 5 Pole 35
Obrázek 6 Lineární spojový seznam 35
Obrázek 7 K-cestný strom 36
Obrázek 8 Analytické třídy 39
Obrázek 9 Analytické třídy II 40
Obrázek 10 Hlavní okno aplikace 46
Obrázek 11 Okno editace tratě 47
Obrázek 12 Editace stanice 47
Obrázek 13 Editace traťového úseku 47
Obrázek 14 Vykreslení výsledku 48

Úvod

Jízdní doby vlaku jsou jedním ze základních počitatelných podkladů pro tvorbu jízdního řádu. V dnešní době postupující liberalizace železničního prostředí je přesnost výpočtu těchto podkladů (jízdní doby, provozní intervaly) více než důležitá s ohledem pro maximálně efektivní využití kapacity tratě. Využití výpočetní techniky, byť i pouhým zautomatizováním ručního výpočtu, napomáhá zjednodušit práci konstruktérům, ale i zpřesnit samotné výsledky při použití přesnějších, ale náročnějších, metod výpočtu.

Tato práce se zabývá výpočtem jízdních dob a vychází z postupů výpočtu procvičovaných v předmětu Technologie a řízení železniční dopravy, tj. pomocí vztahů z kinematiky hmotného bodu. Hlavním cílem je vytvoření aplikace, která tyto jízdní doby spočítá na zadané trati a se zadanými parametry vlaku. Tato aplikace by tedy měla uživateli umožnit tuto trať sestavit a editovat, ale i uložit a načíst pro pozdější použití.

Správnost výstupu aplikace bude ověřena porovnáním výsledků modelových příkladů počítaných ručním výpočtem a naprogramovanou aplikací.

1 Možnosti výpočtu pomocí SW

V dnešní době se v prostředí české železnice na centrální úrovni používá k sestavně jízdních řádů komplexní systém KANGO (komplexní aplikace návrhu grafikonu online), tento projekt byl nasazen na konci roku 2009 a plně nahrazuje původní systém SENA JŘ-VT. Původním záměrem projektu SENA JŘ-VT byla konstrukce listu nákresného jízdního řádu a základních pomůcek na něj navazujících, ale přerostl do všech oblastí konstrukce grafikonu včetně plánů vlakotvorby a oběhů a pomůcek jak služebních, tak i pro informování cestujících. Systém KANGO jednotlivé části původního systému nahrazuje nově vyvinutými, s výjimkou součásti pro konstrukci jízdního řádu a tvorbu většiny tiskových výstupů, která převzala původní knihovny.[2][3]

Tento systém je provozován pouze na vnitřní síti SŽDC, s.o. a není jinak komerčně využíván. Tato práce bude ovšem více zaměřena na veřejně dostupný, byť i komerční, software, proto v následujících podkapitolách zmíním dva programy používané při výuce na Dopravní fakultě Jana Pernera a jeden program, který je používán na ČVUT v Praze. Tyto programy ovšem nejsou jediné, které v oblasti výpočtu jízdních dob existují.

1.1 Viriato

Viriato je produktem švýcarské společnosti SMA. Poprvé byl uveden v roce 1996 a od té doby byl rozšířen o různé moduly, např. zobrazení podle zeměpisné polohy. Na podzim roku 2012 bylo Viriato kompletně přepsáno do jazyka C#.NET. Aplikace pro svůj běh vyžaduje databázový server, pro samostatné instalace nebo malé pracovní skupiny postačuje MS Access, pro velké pracovní skupiny využívá ORACLE. Systém poskytuje také volitelná rozhraní pro přístup do databází švýcarských, německých, portugalských, belgických, finských i jiných správců železniční infrastruktury.

Umožňuje hrubý odhad jízdních dob pro tvorbu předběžných oběhů vozidel, které umožní plánovačům určit optimální jízdní řády i díky možnosti porovnání alternativ. Rozhraní umožňuje zobrazit nákresný jízdní řád, analýzy jízdních dob, vyhledávání konfliktů, diagramy sítě, diagramy obsazení staničních kolejí, ale také jízdní řády určené pro cestující. Viriato také umožňuje jednoduše vytvářet taktové jízdní řády. Upravovat časy odjezdů vlaku, řešit konflikty nebo obsazení staničních kolejí je možné jednoduše řešit přetažením myší. Viriato umožňuje také vytvářet výlukové jízdní řády.[4]

1.2 Opentrack

Jedná se o projekt Švýcarského spolkového technologického institutu, byl zadán v polovině devadesátých let 20. století. Opentrack představuje simulační nástroj, který je schopný nejen výpočtů jízdních dob a konstrukce jízdního řádu, ale i následné analýzy využití stanic, kapacity tratí i analýzy, zda-li by bylo vhodné investovat do úpravy infrastruktury nebo vozidel.

Základem pro simulaci je železniční síť popsaná dvoubodovými grafy, kterou může uživatel graficky editovat. Tuto síť, ale i další podklady, jako jsou trakční charakteristiky nebo již vytvořené jízdní řády, je možné i načíst z různých formátů, nejjednodušeji ovšem pomocí RailML, což představuje jednotný formát výměny železničních dat, který je postavený na klasickém XML.

Další možností tohoto programu je možnost provést simulaci se zadanými nebo náhodně generovanými zpožděními vlaků a následně vyhodnocení, jak tyto zpoždění ovlivní vytvořený jízdní řád.

Výstupní data program umožňuje zobrazit v grafické podobě nebo exportovat v řadě formátů od prostého textu po MS Excel, samozřejmostí je formát RailML. Opentrack také poskytuje rozhraní pro připojení externích aplikací, které s ním komunikují prostřednictvím Opentrack-API, které je realizováno posíláním SOAP zpráv po HTTP protokolu.[5]

1.3 iPlan

iPlan je součástí balíku FBS[1] německého Institutu pro plánování regionální a dálkové dopravy. Tento systém je vytvářen od roku 1997. Institut spolupracuje i s ČVUT, proto poskytuje i zastoupení pro Českou republiku.

iPlan obsahuje několik podprogramů. Jedním z nich je FPL, který slouží k návrhu nákresného jízdního řádu, a to jak k výpočtu jízdní doby vlaku, tak i k jeho následnému vykreslení na listu. Pro výpočet jízdní doby v požadované trase uživatel nejdříve musí zadat číslo vlaku, přirážky k jízdní době, určit, kde vlak bude zastavovat, čas odjezdu a hmotnost vlaku a hnacího vozidla. FPL potom sám nabídne nejbližší volné trasy, příp. navrhne křižování na jednokolejných tratích. Úpravy časové polohy jsou možné posunutím myší. Je možné i kopírování vlaku v taktu.

[1] Fahrplanbearbeitungssystem – systém zpracování jízdního řádu

Dalším je podprogram BFO. Tento podprogram je určen pro vytvoření přehledu obsazení staničních kolejí.

Posledním podprogramem je NETZ, který slouží pro spojování jednotlivých souborů z programu FPL do sítě. NETZ obsahuje moduly pro vytváření provozních podmínek (oběhy a sešitové jízdní řády), ale i knižních jízdních řádů pro cestující.[6]

2 Metodika výpočtu

Před samotným rozborem výpočtu je potřeba pro správné pochopení a řešení problému si ujasnit některé pojmy z technologie železniční dopravy, proto v následujících odstavcích bude vysvětlen význam těch, které jsou pro řešený problém významné.

2.1 Pojmy z technologie železniční dopravy

Základním pojmem je vlak, rozumí se jím sestavená a svěšená skupina drážních vozidel (nebo samostatné drážní vozidlo), která je schopná samostatného pohybu. Z pohledu jízdního řádu je vlakem také spoj v něm uvedený. Ovšem pro tento řešený problém představuje vlak skupinu vozidel podle první definice.

Dalšími pojmy jsou rychlosti. Konstrukční rychlost se vztahuje k vozidlu a představuje maximální rychlost, pro jakou je dané vozidlo projektováno a vyrobeno. Dále stanovená rychlost, je rychlost předepsaná vlaku s ohledem na konstrukční rychlost každého z vozidel vlaku, tzn. stanovená rychlost je minimem ze všech těchto hodnot, ale může být stanovena i na hodnotu nižší. Traťová rychlost se vztahuje k určitému úseku traťové koleje a určuje, jakou nejvyšší rychlostí tento úsek smí být projížděn. Tato rychlost se vyznačuje v daném místě u traťové koleje návěstmi – rychlostníky, příp. návěstmi pro pomalou jízdu. Okamžitá rychlost představuje rychlost, jakou se vlak v určitém okamžiku pohybuje.

Dopravna je místo na dráze, které slouží k řízení sledu jízdy vlaků. Základní členění dopraven lze provést na dopravny s kolejovým rozvětvením a bez kolejového rozvětvení. Do první skupiny patří stanice a odbočky, do druhé skupiny náleží hradla, hlásky, ale i oddílová návěstidla automatického bloku nebo automatická hradla, která jsou ovšem důležitá s ohledem na následná mezidobí, čili ne pro tento řešený problém. Mezi dopravny se neřadí zastávky a nákladiště. Zastávkou je označené místo na dráze, které slouží pro nástup a výstup cestujících.[7]

Důležitými pojmy jsou hlavně časy příjezdu a odjezdu, resp. průjezdu. Časem příjezdu se rozumí okamžik, kdy vlak přijede do dopravny nebo zastávky a čelem zastaví na určeném místě. Časem odjezdu se rozumí okamžik, kdy se vlak dá do pohybu při opuštění dopravny nebo zastávky. Pokud vlak v dané dopravně nezastavuje, určuje se místo času příjezdu a odjezdu pouze čas průjezdu. Časem průjezdu se rozumí okamžik, kdy čelo vlaku míjí příslušné hlavní návěstidlo.

Zátěžovou tabulkou se rozumí tabulka technického normativu hmotnosti. Vytváří se pro každou řadu hnacího vozidla a pro každý typ vozidlového odporu. Tabulka potom udává, jakou rychlost je hnací vozidlo schopné vyvinout s vlakem zadané hmotnosti na určité třídě sklonu. Použití těchto typů odporu je uvedeno v tabulce 1. Třída sklonu, zapisovaná římskou číslicí, představuje celočíselnou část výsledku následujícího vzorce

$$TS = \frac{sklon + 2}{2} \qquad (1)$$

Tabulka 1 Typy vozidlového odporu

Typ jízdního odporu	Použití
R	Vlaky sestavené z podvozkových osobních vozů normální stavby (včetně podvozkových vozů na přepravu aut) a osobních vozů lehké stavby o délce větší než 20 metrů
S	Vlaky sestavené z dvounápravových osobních vozů normální stavby nebo z nákladních vozů při průměrné hmotnosti připadající na jedno vozové dvoukolí 10 až 15 tun
T	Nákladní vlaky při průměrné hmotnosti připadající na jedno vozové dvoukolí větší než 15 tun
U	Nákladní vlaky při průměrné hmotnosti připadající na jedno vozové dvoukolí menší než 10 tun
M	Vlaky sestavené z vozů lehké stavby o délce do 20 metrů

2.2 Výpočet

Pro výpočet jízdních dob v semestrální práci z předmětu Technologie a řízení železniční dopravy byly použity základní fyzikální vzorce pro pohyb hmotného bodu. Tyto použiji i zde.

2.2.1 Základní vzorce

Rozjezd (zrychlování) vlaku popisuje dvojice vzorců:

$$s = v_0 t + \frac{1}{2} a_r t^2 \qquad (2)$$

$$v = v_0 + a_r t \qquad (3)$$

Zastavení (brzdění) vlaku popisuje dvojice, která se oproti předchozím liší pouze znaménkem:

$$s = v_0 t - \frac{1}{2} a_b t^2 \qquad (4)$$

$$v = v_0 - a_b t \qquad (5)$$

Vzorec pro jízdu stálou rychlostí získáme dosazením nulové hodnoty za zrychlení:

$$s = vt \tag{6}$$

Do těchto vzorců ovšem zasahuje také sklon tratě:

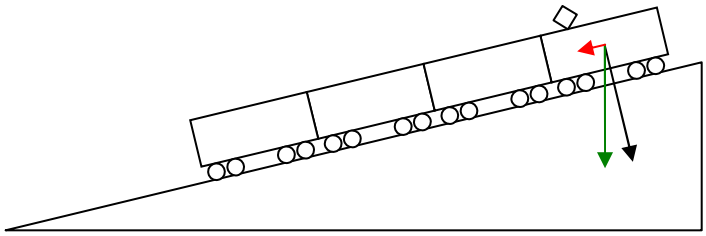
$$a_o = \frac{x}{100} \tag{7}$$

potom pro zrychlení a zpomalení platí následující vztahy, kde a_{rz} a a_{bz} představují původní zadané hodnoty:

$$a_r = a_{rz} - a_o \tag{8}$$

$$a_b = a_{bz} + a_o \tag{9}$$

toto zjednodušení lze provést, budeme-li vycházet z následujícího nákresu:



Obrázek 1 Rozklad sil na sklonu

zelená šipka představuje tíhovou sílu, červená šipka představuje odporovou sílu působící proti pohybu. Při jízdě v opačném směru (v klesání) tato síla působí ve směru pohybu a proto je urychlující. Vztah mezi těmito silami zachycuje následující vzorec:

$$F_o = F_G \sin\alpha \tag{10}$$

vydělením celé rovnice hmotností získáme následující vzorec:

$$a_o = g \sin\alpha \tag{11}$$

Stejný poměr platí i mezi délkou tratě a převýšením

$$h = l \sin\alpha \tag{12}$$

proto je možné postavit tyto dva vzorce do rovnosti a vyjádřit požadované a_o

$$\sin\alpha = \frac{h}{l} = \frac{a_o}{g} \tag{13}$$

$$a_o = g\frac{x}{1000} \tag{14}$$

v případě, že délka tratě (l) je 1000 metrů, představuje poté převýšení (h) samotnou hodnotu sklonu (x). Po dosazení hodnoty 10 za g, získáme původní vzorec (7).

2.2.2 Postup výpočtu

Správné pochopení pojmů z technologie železniční dopravy je potřeba hlavně v tomto kroku, protože je nezbytné pro určení správných hranic úseků nebo použití správné hodnoty rychlosti.

Úplně na začátku je potřeba zjistit, jaké jsou zadané parametry vlaku, jestli se jedná o osobní vlak nebo nákladní. To ovlivní hlavně hodnoty zrychlení a zpomalení a také četnost zastavování. Pokud není vlak zadaný délkou a normativem hmotnosti, musí se tyto vlastnosti dopočítat. Délka vlaku se vypočte jako součet délek přes nárazníky všech vozidel ve vlaku, tj. včetně hnacích vozidel. Hmotností vlaku se rozumí hmotnost tažených vozidel, ta je součtem hmotností ložených vozů v případě nákladního vlaku, nebo hmotnost plně obsazeného vozu v případě vlaku osobního. Plně obsazený vůz představuje vůz, kde jsou obsazena všechna místa k sedění. Na jedno místo se potom počítá s váhou 0,08 tuny (80 kg). Do hmotnosti vlaku se tedy nezapočítávají činná hnací vozidla, proto když je vlak veden pouze samotným motorovým vozem, je tato hmotnost nula tun a ze zátěžové tabulky se použije sloupec, který je většinou označen „Vůz sólo“.

Dále se musí určit, po kterých kolejích bude vlak projíždět, to určuje místa zastavení ve stanicích a zastávkách a také lokální omezení rychlosti hlavním návěstidlem při jízdě do odbočky.

Poté se trasa rozděluje na menší úseky podle míst zastavení, podle míst změn rychlostí včetně lokálních omezení rychlosti hlavním návěstidlem při jízdě do odbočky a podle míst změn sklonů. Při rozdělování podle rychlostí je potřeba pamatovat, že vlak může začít zrychlovat až, když je celou délkou mimo úsek s nižší rychlostí, proto se místo zrychlování posouvá o délku vlaku ve směru jízdy za rychlostník, příp. poslední výhybku v případě lokálního omezení rychlosti hlavním návěstidlem při jízdě odbočkou. Naopak při vjezdu do úseku s nižší rychlostí musí mít vlak tuto rychlost již v okamžiku, kdy čelo vlaku míjí daný rychlostník nebo hlavní návěstidlo omezující rychlost při jízdě do odbočky. Při rozdělování úseku podle je potřeba si uvědomit, jak začíná působit odporová síla proti pohybu. Nezačne působit najednou, ale postupně narůstá, jak vozy vjíždějí na sklon. Za předpokladu rovnoměrně rozložené hmotnosti vlaku narůstá tato síla lineárně, proto je možné použít začátek vlivu sklonu od půlky délky vlaku včetně hnacího vozidla, zejména u dlouhých vlaků. U krátkých vlaků tento přístup můžeme použít také, protože při délce současných používaných vozidel by tento posun představoval maximálně deset metrů, což je z pohledu

následujícího výpočtu zanedbatelný rozměr. (Při rychlosti 90 km.h^{-1} urazí vlak vzdálenost 10 metrů za 0,4 sekundy, při rychlosti 40 km.h^{-1} pak za 0,9 sekundy.)

Při dělení podle změn sklonů je potřeba také vždy zkontrolovat v zátěžové tabulce, zdali je pro hnací vozidlo s vlakem o zadané hmotnosti na tomto sklonu možné dosáhnout požadované rychlosti. Pokud této rychlosti není schopno dosáhnout, začne vlak na tomto úseku zpomalovat pouze se zpomalením, které odpovídá vlivu sklonu – vzorec (7).

Před samotnými výpočty je potřeba provést převod jednotek! Doporučuje se všechny hodnoty převést na jednotky obsahující metry – *m*, *m.s*$^{-1}$ a *m.s*$^{-2}$ – na místo jednotek s kilometry. Lze ovšem použít i délkové jednotky v metrech a rychlosti v *km.h*$^{-1}$, v tom případě je ale nutné nezapomenout u rychlosti na převodní koeficient.

Následně se pro každý úsek zjistí způsob výpočtu, mohou nastat následující situace:

1. vlak celý úsek projíždí konstantní rychlostí,
2. vlak na začátku úseku zrychluje a poté jede konstantní rychlostí,
3. vlak na začátku úseku zpomaluje a poté jede konstantní rychlostí,
4. vlak jede konstantní rychlostí a na konci úseku brzdí,
5. vlak na začátku úseku zrychluje, na konci úseku brzdí a mezi tím jede konstantní rychlostí,
6. vlak v úseku nestihne zrychlit na požadovanou rychlost,
7. vlak v úseku nestihne z dané rychlosti zbrzdit,
8. vlak v úseku zrychluje i brzdí, ale nestihne mezi tím dosáhnout maximální rychlosti.

Výpočet pro první situaci spočívá pouze v použití vzorce (6).

Pro situaci 2 se nejprve musí zjistit, jak dlouho bude vlak zrychlovat – podle vzorce (3). Tento čas se dosadí do vzorce (2) a tím je určena dráha, na jaké toto zrychlení proběhne. Pokud je tato dráha větší než délka úseku, musí se tento úsek počítat podle situace 6. Jinak se pro zbývající dráhu úseku použije vzorec (6).

Situace 3 nastane v okamžiku, kdy hnací vozidlo není schopno s vlakem dosáhnout požadované rychlosti, proto vlak bude zpomalovat vlivem působení tíhové síly a to zpomalením odpovídajícím sklon. V tomto případě se tedy použijí vzorce (4) a (5), ovšem s rozdílem, že místo a_b bude použito a_o podle vzorce (7). Doba jízdy po zbývající dráze úseku se vypočítá podle vzorce (6).

Výpočet u situace 4 bude probíhat podobně jako v případě 2, pouze s tím rozdílem, že se použijí vzorce pro brzdění – vzorce (4) a (5), kterými se zjistí dráha brzdění a na zbývající dráhu se opět použije vzorec (6). Pokud dráha brzdění bude delší než délka úseku, musí se použít výpočet podle situace 7.

V situaci 5 se nejdříve zjistí čas a dráha zrychlování – vzorce (2) a (3) – a poté čas a dráha brzdění – vzorce (4) a (5). Na zbývající dráhu se opět použije vzorec (6). Pokud je součet dráhy zrychlování a brzdění větší než délka úseku, musí se úsek spočítat podle situace 8.

Výpočet pro situaci 6 spočívá v tom, že se nejdříve určí, jaké rychlosti v tomto úseku může vlak dosáhnout. Vyjádřením času ze vzorce (3), jeho dosazením do vzorce (2) a následnou úpravou vyjde vztah (15), resp. vztah (16) pro dosazení rychlosti v km.h^{-1}.

$$v = \sqrt{2a_r s + v_0^2} \qquad (15)$$

$$v = \sqrt{2 \cdot 3{,}6^2 a_r s + v_0^2} \qquad (16)$$

Tato zjištěná rychlost se může tomuto úseku nastavit jako maximální a současně se následujícímu úseku nastaví jako počáteční rychlost tato vypočtená.

Výpočet pro situaci 7 bude probíhat podobně jako v předchozím případě. Nejdříve se zjistí, z jaké rychlosti dokáže vlak na délce úseku zbrzdit na požadovanou rychlost. Proto se ze vzorce (5) vyjádří čas, dosadí se do vzorce (4) a po úpravách vyjde vztah

$$v_0 = \sqrt{2a_b s + v^2} \qquad (17)$$

$$v_0 = \sqrt{2 \cdot 3{,}6^2 a_b s + v^2} \qquad (18)$$

resp. varianta pro možnost dosazení rychlosti v km.h^{-1}.

Rychlost zjištěná podle vzorce (17) nebo (18) se určí v daném úseku za maximální a dále se musí přepočítat předchozí úsek, ale s upravenou koncovou rychlostí, kterou bude tato zjištěná.

Situace 8 je na výpočet asi nejnáročnější. Je možné ji řešit tak, že se zkusí odhadnout hodnota rychlosti, které vlak dosáhne než bude muset začít brzdit. Ovšem tento postup je časově náročný, než je nalezna správná hodnota, proto je výhodnější – a hlavně i výpočetně přesnější – vyřešit tuto situaci soustavu rovnic, v tomto případě pěti.

$$
\begin{aligned}
s_c &= s_r + s_c \\
v_{\max} &= v_p + a_r t_r \\
v_k &= v_{\max} - a_b t_b \\
s_r &= v_p t_r + \tfrac{1}{2} a_r t_r^2 \\
s_b &= v_{\max} t_b - \tfrac{1}{2} a_b t_b^2
\end{aligned}
\qquad (19)
$$

Postupnými úpravami této soustavy se vyjádří doba rozjezdu t_r, ovšem toto vyjádření vychází ve tvaru klasické kvadratické rovnice

$$Ax^2 + Bx + C = 0 \qquad (20)$$

kde pro členy A, B a C vychází následující hodnoty

$$
\begin{aligned}
A &= a_r \left(a_b + a_r\right) \\
B &= 2v_p \left(a_b + a_r\right) \\
C &= 2vp\left(v_p - v_k\right) - \left(v_p - v_k\right)^2 - 2a_b s_c
\end{aligned}
\qquad (21)
$$

z těchto hodnot musí diskriminant být nezáporné číslo, jinak rovnice nemá řešení a stejně tak i tento počítaný úsek. Doba rozjezdu se tedy vypočte

$$t_r = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \qquad (22)$$

a tato hodnota se dosadí do vzorce pro dobu brzdění, který byl získán během řešení soustavy rovnic

$$t_b = \frac{v_p + a_r t_r - v_k}{a_b} \qquad (23)$$

Je vidět, že tento způsob výpočtu je sice složitější, ale na druhou stranu se nemusí řešit mezivýsledky a získají se přímo požadované výsledné hodnoty doby rozjezdu a brzdění. To také zpřesní samotný výpočet.

2.3 Modelové příklady

Pro modelové příklady použiji trať zadanou v semestrální práci na předmět Technologie řízení železniční dopravy. Z této semestrální práce použiji i osobní vlak ve směru od začátku ke konci a manipulační vlak v polovině jeho trasy od konce k začátku.

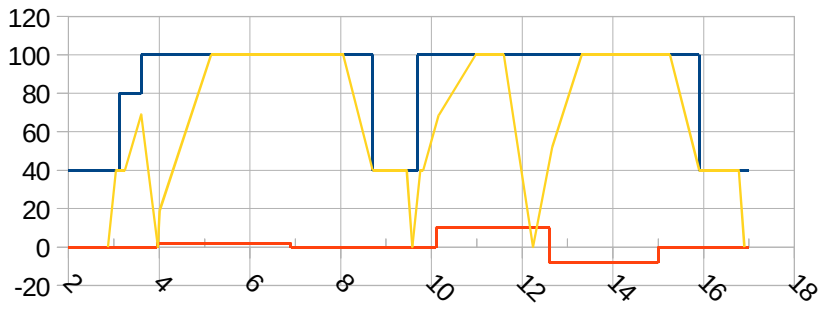
2.3.1 Osobní vlak

Osobní vlak je zadán složením soupravy: 130 + BDs + 3xBp. Stanovená rychlost vlaku je 100 $km.h^{-1}$, zrychlení 0,35 $m.s^{-2}$ a zpomalení 0,5 $m.s^{-2}$.

Nejdříve se musí spočítat délka vlaku: 20 (obecná délka lokomotivy) + 24,5 (délka vozu BDs) + 3x24,5 (délka vozu Bp) = 118 metrů. A také hmotnost vlaku: 39+40*0,08 (hmotnost vozu BDs se čtyřiceti obsazenými místy) + 3x(34+88*0,08) = 165,32 tuny. Ze zátěžové tabulky proto bude vybrán sloupec s nejbližší vyšší hodnotou, tedy 200.

Osobní vlak jede ze stanice A do stanice C. Ze stanice A odjíždí z druhé koleje, ve stanici B zastavuje na druhé koleji a ve stanici C zastaví na druhé koleji.

Následující obrázek zobrazuje diagram rychlostí, ve kterém osa x představuje kilometrickou polohu na trati. Modře je vyznačena maximální povolená rychlost (traťová rychlost s omezeními při jízdě do odbočky), červená linka představuje sklon ve směru jízdy a žlutě je vyznačena okamžitá rychlost (pro zjednodušení vykreslena jako přímka namísto paraboly).



Obrázek 2 Diagram rychlostí - osobní vlak

Osobní vlak se rozjíždí z určeného místa ve stanici A – 10 metrů před koncem nástupiště u druhé staniční koleje (km 2,871). Zrychluje na rychlost 40 km.h^{-1}, neboť tuto rychlost umožňuje návěst na hlavním návěstidle, touto rychlostí pokračuje až do okamžiku opuštění poslední výhybky posledním vozem – čelo vlaku bude v km 3,241. Od této chvíle může vlak zrychlovat na traťovou rychlost 80 km.h^{-1}. Ovšem následuje zastávka Z, kde vlak zastavuje čelem 10 m před koncem nástupiště – km 3,969. Mezitím se změnil sklon, ovšem jeho vliv začne být významný až, když na něm bude vlak polovinou své délky, tj. od km 4,009. Proto ze zastávky Z vlak zrychluje 40 m bez vlivu sklonu, po těchto 40 metrech zrychluje na stanovenou rychlost 100 km.h^{-1} se zrychlením sníženým o vliv sklonu. Vliv sklonu přestane být významný po opuštění sklonu polovinou soupravy – tj. v km 6,959. Následuje úsek, na jehož konci u vjezdového návěstidla do stanice B (km 8,707) již vlak musí jet rychlostí pouze 40 km.h^{-1}. Touto rychlostí vlak pokračuje přes zhlaví až na druhé staniční koleji zastaví u nástupiště. Protože ale u této koleje v tomto směru je návěstidlo umístěno

ještě před koncem nástupiště, nebude vlak stavět deset metrů před koncem, ale zastaví deset metrů před návěstidlem (km 9,578). Z tohoto místa se následně vlak rozjíždí na rychlost 40 km.h^{-1} až do okamžiku, kdy posledním vozem opustí poslední výhybku – km 9,754. Od tohoto místa vlak může zrychlovat na stanovenou rychlost, ale mezitím trať mění sklon, který začne působit od km 10,159, čili vlak pokračuje ve zrychlování sníženém o vliv tohoto sklonu. Po dosažení stanovené rychlosti se vlak blíží k zastávce s hradlem V, kde bude zastavovat deset metrů před koncem nástupiště – km 12,242. Protože je vlak stále na sklonu 10‰, bude vlak brzdit o vliv sklonu rychleji. Z tohoto místa se rozjíždí s vlivem sklonu sníženým zrychlením až do místa – km 12,659 – kde přestane působit vliv tohoto sklonu a začne působit jiný sklon, klesání. Vlak pokračuje ve zrychlování, tentokrát navíc zrychleným vlivem klesání tratě. Vlak dosáhne stanovené rychlosti, kterou pokračuje až do místa, kde jeho vliv odezní – km 15,059. Vlak pokračuje stanovenou rychlostí kolem návěstidla předvěstícího rychlost 40 km.h^{-1} od následujícího hlavního návěstidla v km 15,905. Od tohoto návěstidla vlak pokračuje touto rychlostí přes zhlaví až k místu zastavení deset metrů před koncem nástupiště u druhé staniční koleje – km 16,900.

Tabulka 2 Dílčí úseky - osobní vlak

	od	do	délka	počáteční r.	maximální r.	koncová r.	sklon
1	2,871	3,241	370	0	40	40	0
2	3,241	3,718	477	40	80	80	0
3	3,718	3,969	251	80	100	0	0
4	3,969	4,009	40	0	100	100	0
5	4,009	6,959	2950	100	100	100	2
6	6,959	8,707	1748	100	100	40	0
7	8,707	9,578	871	40	40	0	0
8	9,578	9,816	238	0	40	40	0
9	9,816	10,159	343	40	100	100	0
10	10,159	12,242	2083	100	100	0	10
11	12,242	12,659	417	0	100	100	10
12	12,659	15,059	2400	100	100	100	-8
13	15,059	15,905	846	100	100	40	0
14	15,905	16,900	995	40	40	0	0

Následuje samotný výpočet časů. Podle situace 1 se budou počítat úseky, kde jsou hodnoty všech rychlostí stejné, tomu odpovídají úseky 5 a 12. Podle situace 2 budou počítány úseky, ve kterých je maximální rychlost shodná s koncovou rychlostí, ale počáteční je menší, tedy úseky 1, 2, 4, 8, 9 a 11. Situace 4 odpovídá úsekům, kde se počáteční rychlost rovná maximální rychlosti, ale koncová rychlost je nižší, tedy úseky 6, 7, 10, 13 a 14. Situaci 5 odpovídají úseky, kde se hodnoty počáteční a koncové rychlosti liší od maximální rychlosti, zde se jedná o úsek 3. Situace 3 v tomto případě vůbec nenastala.

Během výpočtu ale zjistíme následující skutečnosti

- na výpočet podle situace 6 povedou úseky 2, 4, 9 a 11.
- na výpočet podle situace 7 povede úsek 3

a z toho vyplývají následující úpravy

- úseku 10 se změní počáteční rychlost a přepočítá se podle situace 5
- úsekům 5 a 12 se změní počáteční rychlost a přepočítají se podle situace 2
- úseky 2 a 3 se mohou sloučit do jednoho, který se přepočítá podle situace 8, tuto úpravu bylo možné odhadnout již z diagramu rychlosti.

Výsledné hodnoty jsou uvedeny v tabulce Jízdní doby - osobní vlak.

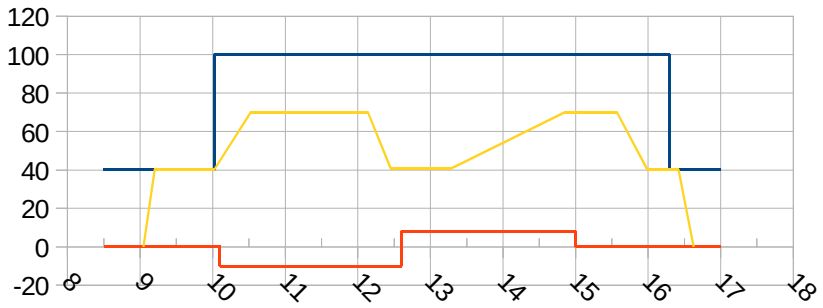
Tabulka 3 Jízdní doby - osobní vlak

	od	do	čas A-Z	čas Z-B	čas B-V	čas V-C
1	2,871	3,241	49,18			
2	3,241	3,969	61,98			
3						
4	3,969	4,009		15,08		
5	4,009	6,959		133,81		
6	6,959	8,707		72,92		
7	8,707	9,578		89,50		
8	9,578	9,816			37,30	
9	9,816	10,159			22,73	
10	10,159	12,242			103,60	
11	12,242	12,659				57,78
12	12,659	15,059				93,83
13	15,059	15,905				40,45
14	15,905	16,900				100,66
jízdní doby			111,16	311,31	163,63	292,72
jízdní doby – minuty			1	5	2	4
jízdní doby – sekundy			51,16	11,31	43,63	52,72
JD po zaokrouhlení			2	5,5	3	5

2.3.2 Manipulační vlak

Manipulační vlak je zadán délkou a normativem hmotnosti: 742 S500/300m. Stanovená rychlost vlaku je 70 $km.h^{-1}$, zrychlení 0,3 $m.s^{-2}$ a zpomalení 0,4 $m.s^{-2}$.

Manipulační vlak jede ze stanice C do stanice B. Ze stanice C odjíždí ze sedmé koleje, ve stanici B zastavuje na čtvrté koleji.



Obrázek 3 Diagram rychlostí - manipulační vlak

Vlak se rozjíždí z určeného místa ve stanici C na sedmé staniční koleji – km 16,625 – a zrychluje na rychlost 40 km.h^{-1} povolenou návěstí na hlavním návěstidle. Po opuštění poslední výhybky posledním vozem (čelo vlaku v km 16,143) může vlak dále zrychlovat až na stanovenou rychlost 70 km.h^{-1}, kterou pokračuje až k začátku stoupání, resp. k místu, kde vliv stoupání začne být významný – km 14,850. Protože lokomotiva není s tímto vlakem schopná na tomto stoupání udržet požadovanou rychlost, začne vlivem sklonu zpomalovat až na rychlost 41 km.h^{-1} zjištěnou ze zátěžové tabulky. Po dosažení této rychlosti vlak pokračuje až k místu, kde vliv tohoto sklonu odezní a začíná působit následující klesání (km 12,450), na kterém vlak zrychluje se zrychlením zvětšeným o vliv sklonu. Mezitím v km 12,104 míjí oddílové návěstidlo hradla V. Pokračuje ve zrychlování na stanovenou rychlost. Ještě před koncem klesání, ale musí začít brzdit, neboť se vlak blíží k vjezdovému návěstidlu (km 10,020) stanice B, které dovoluje jízdu rychlostí 40 km.h^{-1}. Toto návěstidlo bude míjet ještě v době, kdy působí vliv sklonu. Od tohoto místa pokračuje touto rychlostí až do místa, kde musí začít brzdit ke konečnému zastavení deset metrů před návěstidlem u čtvrté koleje – km 9,045km.

Tabulka 4 Dílčí úseky - manipulační vlak

	od	do	délka	počáteční r.	maximální r.	koncová r.	sklon
1	16,625	16,143	482	0	40	40	0
2	16,143	14,850	1293	40	70	70	0
3	14,850	12,450	2400	70	70	41	8
4	12,450	12,104	346	41	70	70	-10
5	12,104	10,020	2084	70	70	40	-10
6	10,020	9,045	975	40	40	0	0

Následuje samotný výpočet časů. Podle situace 2 budou počítány úseky, ve kterých je maximální rychlost shodná s koncovou rychlostí, ale počáteční je menší, tedy úseky 1, 2 a 4. Situace 4 odpovídá úsekům, kde se počáteční rychlost rovná maximální rychlosti, ale koncová

rychlost je nižší, tedy úseky 5 a 6. Situaci 3 odpovídají úseky, kde vlak není schopen dosáhnout požadované rychlosti, zde se jedná o úsek 3.

Během výpočtu nejsou zjištěny žádné mimořádné skutečnosti. Výsledné hodnoty jsou uvedeny v tabulce 5

Tabulka 5 Jízdní doby - manipulační vlak

	od	do	čas C-V	čas V-B
1	16,625	16,143	61,9	
2	16,143	14,850	72,45	
3	14,850	12,450	175,12	
4	12,450	12,104	21,97	
5	12,104	10,020		113,13
6	10,020	9,045		101,64
jízdní doby			331,44	214,77
jízdní doby – minuty			5	3
jízdní doby – sekundy			31,44	34,77
JD po zaokrouhlení			6	4

3 Obecný postup pro řešení problému

V současné době je při návrhu desktopových aplikací standardem využití objektově orientovaného programování, proto zde nejdříve bude uvedeno několik základních vlastností této programovací techniky a poté činnosti prováděné ve dvou hlavních fázích vývoje aplikace, kterými jsou analýza problému a návrh aplikace.

3.1 Pojmy z objektově orientovaného programování

Objekt je v realitě synonymem pro předmět. Obecně pro libovolnou část reality, kterou lze označit podstatným jménem. Takový objekt většinou má nějaké vlastnosti a může vykonávat nějaké činnosti. Podobně se na objekt nahlíží z hlediska programování, jako na prvek modelované reality, kde ovšem dochází k zavádění abstrakce, kdy se zanedbají některé vlastnosti a činnosti, které nejsou z hlediska řešeného problému důležité. Tyto vlastnosti jsou potom nazývány atributy a činnosti metodami objektu. Z hlediska programování je tedy objekt strukturou dat, která umožňuje současně definovat jak atributy, tak i metody.

3.1.1 Postuláty OOP

Postulát je obecný předpoklad, o jehož pravdivosti se nerozhoduje[2], případně není možno rozhodnout, přitom je přijímán jako pravdivý. Důsledkem používání těchto postulátů jsou některé odvozené vlastnosti objektově orientovaného programování a to zapouzdření, dědičnost a polymorfismus. Někdy tyto odvozené vlastnosti bývají mylně označovány za vlastnosti základní.

Tyto postuláty jsou:

1. Objekt má vnitřní paměť.

Tato vnitřní paměť objektu umožňuje pamatovat si potřebné atributy zachycující jeho stav. Je čistě záležitostí objektu, jak si tyto atributy bude uchovávat, protože paměť je zvenku objektu nedostupná.

2. Objekt má metody.

Metody představují jedinou možnost práce s vnitřní pamětí. Metodami se rozumí funkce a procedury. Stejně jako vnitřní paměť i metody jsou zvenku objektu nedostupné a nezáleží

[2] Tvrzení určité vědecké teorie přijaté bez důkazů a tvořící její východisko (http://slovnik-cizich-slov.abz.cz/web.php/slovo/postulat)

tedy na jejich implementaci. Dojde-li ke změně implementace metody, nezmění se tím funkčnost celého programu.

3. Objekt má schopnost přijmout a zpracovat zprávu.

Toto se děje prostřednictvím protokolu zpráv, což je mechanismus, který při přijetí zprávy, nalezne odpovídající metodu, kterou zavolá. Po vykonání metody se zprávě vrátí výstupní parametry metody.

To dokazuje, proč tyto ty tři postuláty způsobují, že na objekt se nahlíží jako na „černou skříňku“.

4. Objekt může obsahovat další objekty.

V objektu mohou být vnořené další objekty. Tyto objekty také splňují tyto čtyři postuláty, proto jediná možná komunikace s nejvíce vnořeným objektem je postupné posílání zpráv zvnějšku přes protokoly zpráv vnořených objektů a zpět.

3.1.2 Skládání a obsažení

Tyto obě vlastnosti vychází ze čtvrtého postulátu. Hlavním rozdílem mezi nimi je těsnost vazby dvou takto spojených objektů. Skládání, též agregace, představuje volnou vazbu umožňující samostatnou existenci součásti, tj. vnořeného objektu, jež je využíván nadřazeným objektem. Proti tomu obsažení, též kompozice, je vazba těsnější, která neumožňuje samostatnou existenci vnořeného objektu. Tento vnořený objekt navíc přísluší pouze jednomu nadřazenému objektu, na rozdíl od skládání, kde jeden vnořený objekt může být využíván více nadřazenými objekty. Příkladem agregace může být počítač s jeho periferiemi, které mohou být využívány i jiným počítačem. Jako příklad kompozice lze uvést strom s jeho listy – je zřejmé, že listy není možné „připojit“ na jiný strom.

3.1.3 Dědičnost a polymorfismus

Dědičnost představuje vztah, kdy odvozený objekt konkretizuje nadřazený objekt, např. strom a dub – strom je nadřazený objekt, dub je konkrétní druh stromu. Odvozený objekt potom od nadřazeného přebírá veškeré chování (tj. vnitřní paměť, metody, protokol zpráv a vnořené objekty) od nadřazeného objektu. Toto chování může upravit svými specifiky – přidat další atributy, metody, vnořené objekty nebo rozšířit protokol zpráv, překrýt existující metody, ale nikdy nemůže odebírat existující. Z uvedeného vyplývá, že odvozený objekt může kdekoli zastoupit prvek nadřazený.

Hierarchii dědičnosti je možné vytvářet buď postupným zobecňováním, nebo postupným specifikováním. V prvním případě se u objektů hledají jejich společné vlastnosti a chování a toto se následně přiřadí nadřazenému objektu. Při tomto postupu se ovšem musí dbát i na skutečnost, aby i takto vytvořený objekt existoval v realitě! Příkladem tohoto postupu může být zobecnění objektů pes, kočka a vrána do objektu zvíře. Opačným případem je např. specifikování plošných geometrických tvarů, kde z objektu tvar lze specifikovat kruh, trojúhelník, čtyřúhelník, šestiúhelník atd.

Polymorfismus představuje možnost zajistit rozdílné chování objektů při zavolání stejné zprávy přes protokol zpráv. Jak bylo zmíněno výše, odvozený objekt přebírá od nadřazeného kompletní chování, které může změnit. Překrytím metod nadřazeného objektu je docíleno této požadované změny chování. Polymorfismus je důsledkem dědičnosti a nesmí být podnětem pro hledání dědičnosti!

Oba z výše uvedených příkladů lze použít pro vysvětlení i zde. Zvíře vydává zvuk, každé z uvedených jiný. Tvar se může nakreslit, každý jinak.

3.1.4 Třída

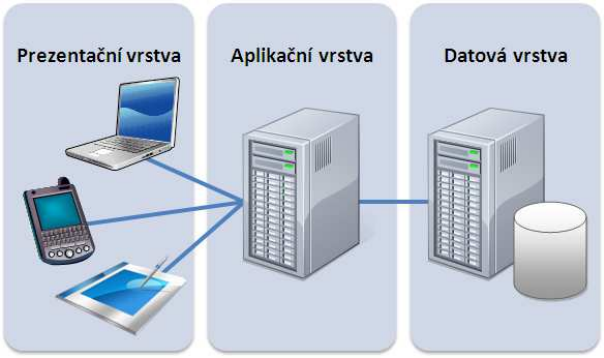
Jak bylo zmíněno výše, slovem objekt byl označen jeden předmět, jedna část reality. Ovšem i v reálném světě je spousta objektů, které jsou stejného druhu, ale mají jiné vlastnosti – teď není myšlen vztah zvíře vs. pes, kočka, ale vztah černý pes vs. bílý pes. O takové skupině objektů je možné prohlásit, že mají stejnou „šablonu“. Z pohledu objektově orientovaného programování je touto šablonou objektů právě třída. Třída typově určuje, jaké vlastnosti je možné u objektů pozorovat. Jednotlivé objekty, které se v tomto případě nazývají instance třídy, se potom mohou lišit hodnotami těchto vlastností. Třída tedy představuje předpis pro tvorbu objektů.

3.1.5 Třívrstvá architektura aplikace

Třívrstvá architektura představuje způsob vývoje aplikací nebo informačních systémů, kde jsou odděleny základní odpovědnosti za zobrazení, výpočty a práci se vstupními daty. Tyto vrstvy jsou na sobě nezávislé a změna kterékoli vrstvy neovlivní chování celé aplikace, tzn. při zachování protokolu zpráv mezi objekty v jednotlivých vrstvách je možné měnit způsob implementace jednotlivých metod.

Jednotlivé vrstvy se označují prezentační, aplikační a datová. Nejvýznamnější je vrstva aplikační, která provádí samotné výpočty a požadované funkcionality. Tato vrstva ovládá datovou vrstvu, která slouží hlavně k načítání a ukládání dat z databáze nebo jiných zdrojů.

Nejvyšší vrstvou je prezentační (též označovaná jako uživatelská) vrstva, přes kterou je uživatelem ovládána aplikační vrstva, tzn. umožňuje uživateli zadávat požadavky, vstupní hodnoty a zobrazit navrácené výsledky.[10]



Obrázek 4 Třívrstvá architektura aplikace

3.2 Hledání tříd

Nalezení vhodných tříd je pro návrh a následnou implementaci systému základní potřeba. Vhodně navržené třídy a vazby mezi nimi umožní zjednodušení a zpřehlednění kódu, neopakování jeho částí a v neposlední řadě také zjednodušení práce při případných pozdějších úpravách.

Během procesu hledání tříd se vyskytnou dva druhy tříd – analytické a návrhové. Hlavním rozdílem mezi nimi je míra jejich konkrétnosti. Zatímco analytická třída je málo konkrétní, návrhová třída je konkrétní natolik, že je možné ji implementovat. Z toho vyplývá, že návrhová třída se vyvine z analytické třídy. Dalším rozdílem mezi těmito dvěma druhy je skutečnost, kterou daný typ zachycuje – analytická třída zachycuje požadované chování systému (co se má stát), návrhová třída již toto chování přesně specifikuje (jak se to má stát).

3.2.1 Analytický model

Analytická třída obsahuje hlavní atributy a služby třídou poskytované, často pouze ve formě nástinu a většinou i bez specifikace datových typů, parametrů a přístupových práv. Hlavním požadavkem na analytickou třídu je zachycení konkrétního pojmu z reality, ovšem zjednodušený pouze na oblast, která souvisí s problémem řešeným v daném systému. V některých případech mohou být u automobilu vyžadovány vlastnosti jako je barva,

registrační značka, výrobní kód a další, v jiném případě je potřebná pouze registrační značka a ostatní jsou pro daný problém zbytečné.

Dobrá analytická třída se vyznačuje v první řadě svým názvem, který musí jednoznačně určovat její smysl v porovnání s prvkem reality, který má tato třída modelovat. Dalším znakem je, že obsahuje malou skupinu odpovědností, tj. služeb, které třída bude poskytovat ostatním. Vhodný počet odpovědností je mezi třemi a pěti, velké množství tříd, které mají mnoho odpovědností nebo naopak poskytují jen jednu službu, způsobuje zhoršení přehlednosti modelu. V takovém případě je vhodné se pokusit několik malých tříd sloučit do jedné nebo příliš velké třídy rozdělit na menší. Dále by dobrá analytická třída měla poskytovat pouze ty služby, které logicky vyplývají a dají se očekávat z jejího názvu. Tato skutečnost se nazývá soudržnost. Třídy spolu spolupracují, ovšem, stejně jako objekty v realitě, je spolupráce se spoustou dalších spíše na škodu, protože způsobuje nepřehlednost.

Pro hledání analytických tříd existují osvědčené techniky, které vedou k vhodným závěrům. Jednou z těchto technik je analýza podstatných jmen a sloves. Tato metoda je jedním z nejjednodušších způsobů hledání tříd, ovšem i zde musí být řešitel opatrný. Nejprve musí dostatečně rozumět a chápat řešený problém, je ale také potřeba dát si pozor na synonyma (různá slova označující stejnou věc) a na homonyma (slova stejně znějící, ale označující různé věci), která mohou vést k nevhodným třídám.

Po shromáždění maxima dostupných důležitých informací se zvýrazní podstatná jména, spojení podstatných jmen, slovesa a slovesné fráze. Podstatná jména a jejich spojení poté bude představovat možné budoucí třídy nebo jejich atributy, zatímco slovesa a fráze s nimi budou vymezovat odpovědnosti nebo operace.

Nebo může být použita metoda štítků CRC[3]. Dělí se na dvě fáze, kde první fází je získávání informací. To může probíhat tak, že je kolektiv požádán o pojmenování objektů, které znají z oblasti řešeného problému. Tyto se tak stávají kandidáty na třídu. Dále se stejným způsobem zjišťují odpovědnosti, které se zapíší k odpovídajícím kandidátům na třídu. Nakonec se zkouší najít, které třídy (kandidáti na třídu) by mohly společně spolupracovat. V další fázi probíhá analýza získaných informací. Pokud název některého z kandidátů vystihuje podstatný pojem z řešeného problému, stane se třídou. Pokud se jeví jako zřejmá součást jiného kandidáta, může se s přihlédnutím k rozsahu odpovědností a množiny

[3] Z anglického Class, Responsibilities and Collaborators, česky Třída, Odpovědnosti a Spolupracovníci

spolupracovníků stát buď třídou, nebo atributem. Pokud se název jeví jako málo významný, stane se atributem třídy, ke které má nejblíže.

Dalším zdrojem pro hledání analytických tříd mohou být fyzické objekty z reálného světa, tištěné formuláře související s řešeným problémem atd.

V objektově orientovaném programu spolu musí objekty nějakým způsobem spolupracovat, proto se musí hledat vazby, kterými tato spolupráce bude probíhat. K nalezení vazeb pomůže objektový diagram. Tento diagram zobrazuje jednotlivé instance a spojení mezi nimi tak, jak se v určitém čase mohou za běhu programu objevit. Existuje-li spojení mezi instancemi, musí potom nutně existovat i vazba mezi třídami. Tyto vazby mohou být specifikované názvem, průchodností, násobností nebo názvem rolí. Název vazby je většinou sloveso nebo slovesná vazba, průchodnost vazby (označení šipkou) značí, který objekt ovládá jiný objekt. Zprávy mezi objekty je potom možné posílat pouze ve směru této šipky. Násobnost vazby vyjadřuje omezení počtu objektů, které se dané vazby v určitém čase mohou účastnit a je vyjádřena seznamem čárkami oddělených intervalů. V tabulce Symboly násobností je uvedeno několik příkladů.

Mezi vazbami a atributy tříd je souvislost. Vazba totiž téměř vždy vyjadřuje, že třída bude mít jako atribut objektový odkaz na jinou třídu. Pokud je násobnost vazby 1, může být tento atribut vyjádřen přímo objektovým odkazem. Pokud je násobnost větší, je potřeba, aby tento atribut byl později v návrhové třídě vyjádřen kolekcí.

Tabulka 6 Symboly násobností

Násobnost	**Význam**
0..1	nula nebo jedna
1	právě jedna
* nebo 0..*	nula nebo více
1..*	jedna nebo více
1..4	jedna až čtyři
1..3,7,11..*	jedna až tři nebo přesně sedm nebo jedenáct nebo více

Nejtěsnější vazbou mezi dvěma třídami je dědičnost, proto je nutné s touto vazbou zacházet velmi opatrně a použít pouze tehdy, je-li pro její použití logický důvod. Pokud je mezi dvěma objekty vztah zjednodušení-upřesnění i ve skutečném světě, je možné dědičnost

použít, ovšem je třeba věnovat pozornost hloubce hierarchie dědění, která by neměla být neúměrně hluboká. I tři úrovně již mohou být příliš.

3.2.2 Návrhový model

Jsou-li nalezeny vhodné analytické třídy a vazby mezi nimi, je možné přejít k hledání návrhových tříd. Návrhové třídy se nacházejí zpřesňováním analytických tříd, kdy může dojít i k rozpadnutí analytické třídy na několik návrhových tříd nebo rozhraní. Další návrhové třídy lze nalézt v samotném vývojovém prostředí (knihovny, komponenty nebo kolekce).

Pro přechod od analytické k návrhové třídě je nejprve potřeba přesně specifikovat typy atributů. Dále následuje rozložení odpovědností analytické třídy na konkrétní metody, které budou mít definované typy návratových hodnot i všechny parametry včetně jejich typů. Nastaví se také přístupová práva pro jednotlivé atributy a metody. Po této specifikaci atributů a metod je vhodné zkontrolovat, zda v návrhu nevznikla neobvykle velká třída, a případně ji rozdělit na několik menších.

I návrhová třída má několik znaků, při jejichž splnění je možné ji označit za dobrou. Návrhová třída musí být úplná a dostatečná. Třída je úplná, když ostatním uživatelům (tj. třídám, které využívají jejích služeb) poskytuje vše, co je možné od ní podle jejího názvu očekávat. Dostatečná je třída tehdy, jsou-li její metody zaměřeny pouze na její účel. Dále by dobrá návrhová třída měla být jednoduchá, tzn. měla by nabízet jeden způsob dosažení požadovaného cíle. Návrhová třída by měla být vysoce soudržná. Pod tímto označením je možné si představit, že atributy a metody této třídy jsou navrženy tak, aby prováděly malou skupinu zodpovědností třídy. To činí třídu i snadněji srozumitelnou. S tím souvisí i vazby na ostatní třídy, které je potřeba využít k plnění vlastních odpovědností. Tyto vazby by měly být minimalizovány na pouze nezbytně nutné.

Dalším úkolem je upřesnění vazeb mezi třídami, které vznikly během analýzy. Tyto vazby budou nejčastěji převedeny do podoby skládání nebo obsažení. Rozdíl mezi skládáním a obsažením je v míře volnosti vazby mezi třídami a byl popsán v kapitole 3.1.2 Skládání a obsažení. Toto upřesnění probíhá v několika krocích. Nejdříve se musí doplnit násobnosti, pokud nejsou doplněny. Dalším krokem je určení, která z tříd představuje celek, tj. která třída využívá služeb druhé třídy – součásti. Ve směru od celku k části přidáme šipku – směr průchodnosti. Podle hodnoty na straně celku určíme, jedná-li se o obsažení, nebo skládání. Pokud tato hodnota je rovna jedné (relace 1:1 nebo 1:N), je možné použít obsažení, jinak (M:1) je nutné použít skládání. Navíc pokud toto spojení je v relaci 1:1, nabízí se otázka, zda-

li součást nezačlenit přímo do celku. Proti tomuto řešení může být některé z výše zmíněných pravidel pro dobrou návrhovou třídu, např. její velikost nebo soudržnost, a proto výsledkem bude použití obsažení, neboť se jedná o těsnou vazbu.

V dalším kroku se rozhoduje, jak budou implementovány násobnosti. Pokud je na straně součásti násobnost 1, implementuje se toto v třídě celku jednoduchým objektovým odkazem. V případě násobnosti větší je potřeba zvolit vhodnou kolekci, která tuto násobnost zrealizuje. O kolekcích bude řeč dále v kapitole 3.3 Datové struktury. Programovací jazyky často nabízejí pouze základní kolekci, kterou je pole, to ovšem není vždy vhodné a proto je nutné implementovat vlastní kolekce ve dalších třídách. Použití pole se v návrhové třídě vyznačí hranatými závorkami za názvem typu, objektového odkazu. V opačném případě se jako datový typ uvede objektový odkaz na třídu, ve které je implementována požadovaná kolekce, a použije se obsažení, protože logicky mezi třídou a kolekcí uchovávající součásti vzniká relace 1:1 – kolekce není sdílená mezi třídami a třída celku používá pouze jednu kolekci pro uchování těchto součástí, vazba mezi kolekcí a součástmi odpovídá původní vazbě mezi třídou celku a třídou součásti.

Speciálním případem je relace M:N, kterou běžně programovací jazyky přímo implementovat nedokáží. V takovém případě je nutné použít asociační třídu, což je třída, která současně představuje i spojení. Umožňuje zachytit souvislost mezi dvěma instancemi různých tříd. Stejně jako obecné třídy i asociační třídy mohou mít své atributy a metody. Ovšem omezení spočívá v jejím druhém významu – spojení – mezi jedním prvkem první množiny (M) a jedním prvkem druhé množiny (N) může v jednu chvíli existovat nejvýše jedno takové spojení. Instance asociační třídy je potom jednoznačně určena právě kombinací objektů na obou koncích. Příkladem pro využití asociační třídy může být vztah Firma – Zaměstnanec, kdy zaměstnanec může pracovat pro více firem současně.

Problém potom ale může nastat v případě, že je potřeba mít takových spojení v jeden okamžik více. I ten je ovšem možné vyřešit, a to rozkladem relace M:N na dvě 1:M a N:1 s obyčejnou třídou uprostřed. Tento postup je používaný např. v relačních databázích.[1]

3.3 Datové struktury

Datové struktury představují způsob uložení dat v paměti. Pro různé řešené problémy jsou vhodné různé struktury, záleží také na skutečnosti, je-li pro řešení potřeba úspora paměti nebo časová úspora při zpracování. Paměťovou náročnost dané struktury není možné vylepšit, pouze použít v programu jinou strukturu.

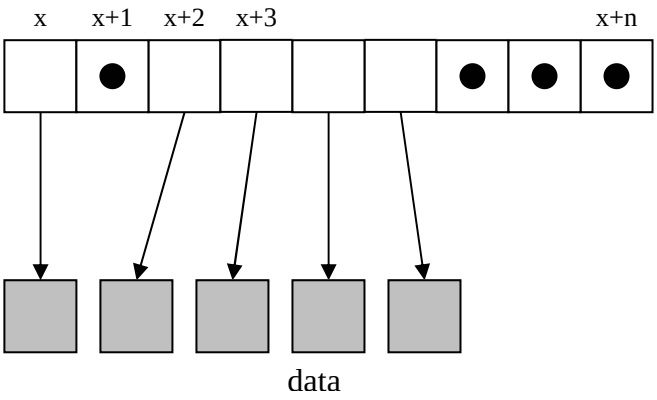
Datová struktura také obsahuje sadu operací pro práci s ní samotnou a s prvky, které obsahuje. Mezi tyto operace patří konstruktor a destruktor (první typ) a dále selektory a modifikátory. Selektory slouží ke zjištění hodnoty určitého prvku, modifikátory slouží k úpravám prvků.

Datové struktury se dělí na základní a odvozené. Mezi základní patří proměnná, pole, struktura a objekt. Proměnná představuje místo v paměti, jehož velikost odpovídá deklarovanému typu, kde je uložená hodnota této proměnné. Pole je poté sekvencí těchto proměnných stejného typu předem určeného rozsahu. Zatímco pole je sekvencí homogenní, struktura je sekvencí heterogenních proměnných, tj. proměnných různých typů. Tyto základní struktury za běhu programu nemění svůj rozsah.

Odvozené datové struktury (také abstraktní datové struktury – ADS) jsou implementovány objekty, proto mohou za běhu programu měnit svůj rozsah. ADS představují způsob uložení dat a operací s nimi bez ohledu na konkrétně zvolenou implementaci. Určují tím také složitost jednotlivých metod selektorů a modifikátorů. Tato složitost se také liší, pokud jsou data ve struktuře setříděná nebo nikoliv.[8]

3.3.1 Pole

Základní kolekcí, kterou přirozeně poskytuje naprostá většina programovacích jazyků, je pole. Tato kolekce uchovává konečný počet prvků stejného typu. K jednotlivým prvkům se přistupuje pomocí indexu označujícího pořadí tohoto prvku v rámci pole. Z toho vyplývá, že operace s konkrétním prvkem pole jsou velmi rychlé. Horší je hledání určité položky v poli. Pokud je dané pole setříděné, lze využít vyhledávání půlením intervalu, které má logaritmickou časovou složitost. Ovšem pokud pole není setříděné, časová náročnost se stává až lineární, protože v nejhorším případě je potřeba projít celé pole položku po položce. Oproti výhodám v rychlosti stojí nevýhoda v paměťové náročnosti pole, které alokuje prostor v paměti pro všechny prvky – jedná se o statickou kolekci. V případě potřeby většího pole je nutné alokovat nové větší pole a následně do něj zkopírovat nebo přesunout prvky ze starého pole.

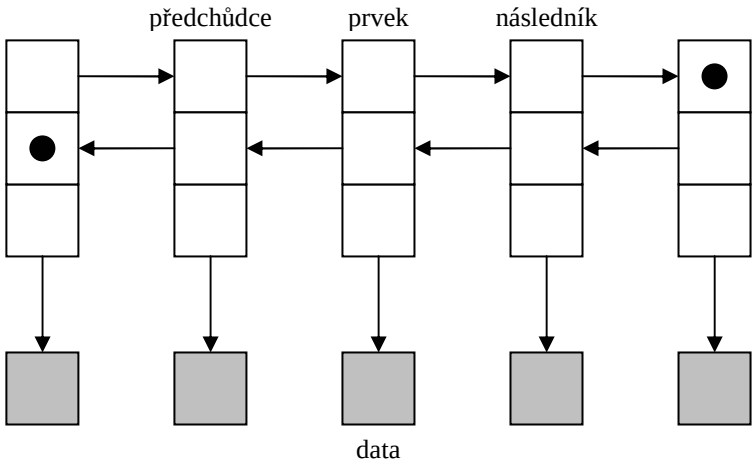


Obrázek 5 Pole

Použití pole je vhodné v případech, které se podobají jeho povaze, tj. čísly indexované prvky stejného typu existující po celou dobu, např. sedadla v kině, háčky v šatně ap.

3.3.2 Seznam

Spojový seznam je oproti statickému poli datová struktura dynamická, která alokuje pro data prostor v paměti podle potřeby. Časová složitost operací nad daty seznamu je lineární. Seznam může být lineární (acyklický) nebo cyklický, jednosměrný nebo obousměrný. Lineární seznam nemá oproti cyklickému svázaný konec se začátkem. V jednosměrném seznamu každý prvek odkazuje pouze na následující, v obousměrném prvky odkazují i na prvek předcházející.

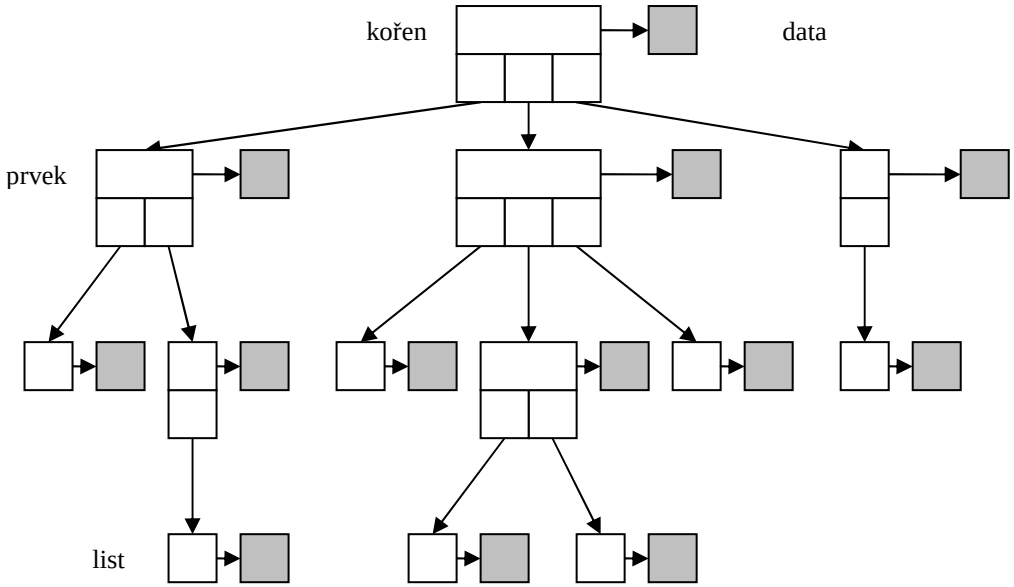


Obrázek 6 Lineární spojový seznam

Spojový seznam je vhodné použít v ostatních případech, na které je lineární struktura vhodná. Již podle názvu jimi mohou být různé neohraničené seznamy nebo číselníky.

3.3.3 Strom

Strom představuje nelineární datovou strukturu, konkrétně hieratickou. Každý prvek stromu obsahuje buď odkaz na svého předka, nebo obsahuje kolekci svých následníků. První typ představuje unární strom, další mohou být binární strom (prvek má maximálně dva následovníky) nebo např. k-cestný strom, kdy každý prvek může obsahovat až *k* odkazů na následníky. Vstupními body do této struktury jsou listy nebo kořen. Listy představují prvky, které nemají žádné další následníky.



Obrázek 7 K-cestný strom

Z uvedené charakteristiky vyplývá příklad využití k-cestného stromu např. k zachycení organizační struktury. Binární stromy je vhodné použít při vyhledávání, díky čemuž je vyhledávací algoritmus schopen dosáhnout logaritmické složitosti.

4 Analýza a návrh aplikace

Před samotnou realizací jsem si formuloval požadavky na program. Účelem aplikace je spočítání jízdní doby jednoho vlaku.

Prvním požadavkem bylo sestavení tratě, její následné ukládání a také načítání. Zde jsem zvolil grafickou reprezentaci s možností manipulovat s jednotlivými objekty pomocí myši. Pro editace jednotlivých objektů na trati jsem se rozhodl vyvolat vždy odpovídající editační okno, kde se již ale většina hodnot zadává textově. Pro ukládání jsem zvolil CSV[4] soubor. Strukturu tohoto souboru jsem ale mohl určit až po definování návrhových tříd.

Dalším požadavkem bylo zadání parametrů vlaku. Tyto parametry jsem se rozhodl zadávat přímo v hlavním okně programu, protože se ve většině jedná o jednorozměrné hodnoty, které je možné zadávat relativně rychle. Výjimkou je tabulka normativu, kterou jsem se rozhodl načítat z CSV souboru. Dále jsem do hlavního okna umístil i zadání dalších parametrů výpočtu, tj. vstupní a výstupní podmínky, pobyty ve stanicích a zastávkách.

Nakonec jsem se rozhodoval v jaké formě prezentovat výsledky. Zvolil jsem grafický výstup vykreslením segmentu nákresného jízdního řádu.

Dále jsem při realizaci aplikace postupoval podle postupů popsaných v kapitole 3.2 Hledání tříd.

4.1 Analýza problému

Již od začátku práce mi bylo jasné, že zachycení tratě bude nejnáročnější na celé aplikaci. Proto jsem si popsal tento problém a na tento text jsem použil metodu analýzy podstatných jmen a sloves. Podstatná jména, resp. vazby s nimi, jsou zvýrazněna tučně, slovesa a jejich vazby jsou zvýrazněny kurzívou a podtržením.

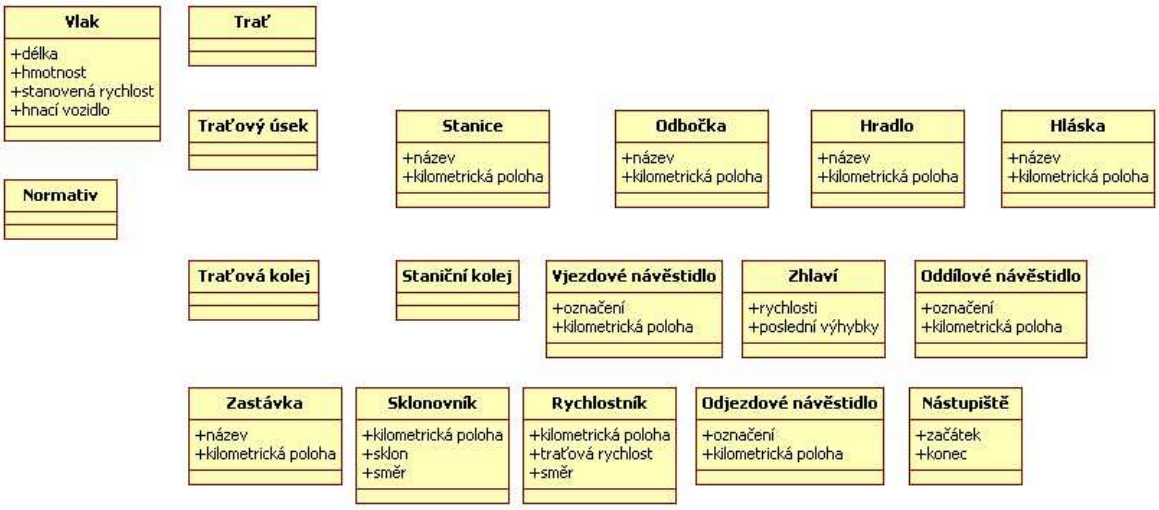
Vlak jede po trati. **Trať** *se skládá* z traťových úseků, stanic a odboček, hlásek a hradel a zastávek. Vlastnosti tratě *udávají* rychlostníky a sklonovníky. **Stanice** *je určena* **názvem** a **kilometrickou polohou** dopravní kanceláře, *je ohraničena* **vjezdovými návěstidly**. Stanice *sestává* ze staničních kolejí. **Staniční kolej** *je ohraničena* **návěstidly** (odjezdové nebo cestové) případně námezníky. U staniční koleje *může být* nástupiště, nástupiště *může příslušet* více kolejím. **Nástupiště** *je určeno* **kilometrickou polohou** začátku a konce. Staniční kolej *je* v rámci stanice *určená* **číslem**. Stanice *má* liché a sudé **zhlaví**, které *určuje* **rychlosti** při jízdě

[4] Coma Separated Values – čárkou oddělené hodnoty

z traťové koleje na staniční kolej a **poslední výhybky** v cestě. **Odbočka** *je určena* **názvem** a **kilometrickou polohou** dopravní kanceláře, *je ohraničena* **vjezdovými návěstidly**. Odbočka *má* pouze jedno **zhlaví**, které *určuje* **rychlosti** při jízdě mezi jednotlivými traťovými kolejemi a **poslední výhybky** v cestě. **Hradlo** i **hláska** *jsou určeny* **názvem** a **kilometrickou polohou** dopravní kanceláře, *sestávají* z **oddílového návěstidla** pro lichý směr a **oddílového návěstidla** pro sudý směr pro každou traťovou kolej. Hradlo i hláska *může* současně *plnit* funkci zastávky, tzn. má nástupiště. Hradlo i hláska *nemají* žádné zhlaví. **Traťový úsek** *je ohraničený* **dopravnami**. Traťový úsek *se skládá* z jedné až dvou traťových kolejí, označených čísly 1 nebo 2. Traťový úsek *může obsahovat* zastávky. Vlastnosti traťové koleje *jsou dané* na ní umístěnými rychlostníky a sklonovníky. **Traťová kolej** *je ohraničena* odpovídajícími vjezdovými návěstidly dopravny nebo oddílovými návěstidly hlásky nebo hradla. U traťové koleje *může být* nástupiště příslušící zastávce v traťovém úseku. **Zastávka** *je určena* **názvem** a **kilometrickou polohou**, *skládá se* z nástupišť při traťové koleji. **Návěstidlo** *je určeno* **označením**, **kilometrickou polohou** a **směrem**. **Sklonovník** *je určen* **kilometrickou polohou** a **směrem** a *nese* informaci o **sklonu** koleje (až k dalšímu sklonovníku v daném směru). **Rychlostník** *je určen* **kilometrickou polohou** a **směrem** a *nese* informaci o **traťové rychlosti** koleje (až k dalšímu rychlostníku v daném směru). Vlak *je určen* **délkou**, **hmotností**, **stanovenou rychlostí** a **hnacím vozidlem**. K vlaku *se váže* **normativ** – podle druhu hnacího vozidla a jejich počtu.

Z tohoto textu vyplynulo, že podstatná jména a jejich spojení následující po slovesné vazbě „je určeno“ budou představovat atributy uvedené třídy. Tyto třídy jsou uvedeny na obrázku Analytické třídy.

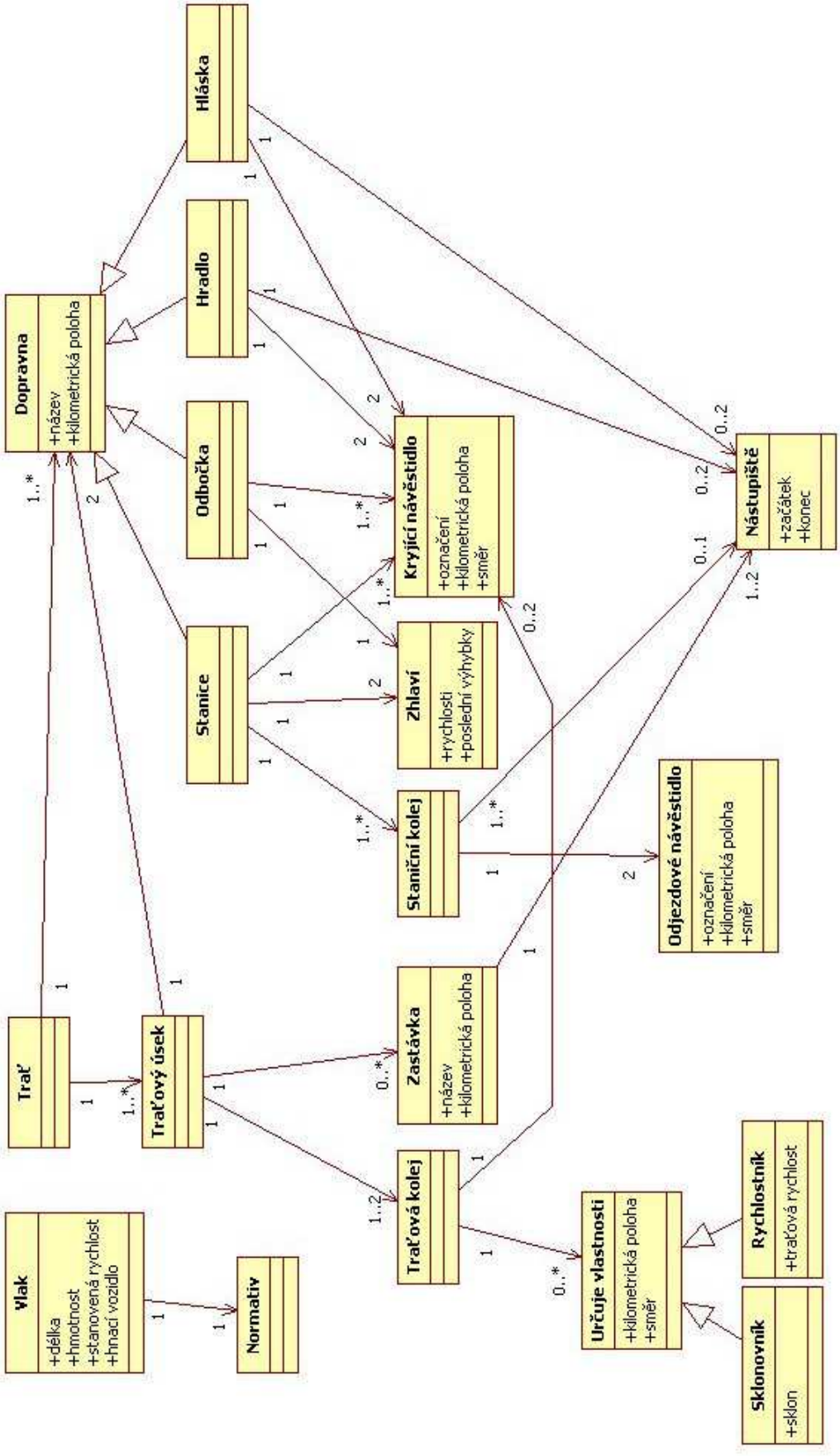
Také jsem vymezil odpovědnosti. Stanice je odpovědná za spravování staničních kolejí, nástupišť, rychlostí na zhlavích a posledních výhybek. Konkrétně to znamená přidávat, odebírat a zpřístupňovat staniční koleje, přidávat a odebírat nástupiště a také je spojovat nebo oddělovat od staničních kolejí. Zhlaví musí v závislosti na vstupujících traťových kolejích nebo staničních kolejích určit velikost tabulek rychlostí a posledních výhybek a následně upravovat tyto hodnoty. Traťová kolej zodpovídá za přidávání a odebírání sklonovníků a rychlostníků. Traťový úsek přidává, odebírá a zpřístupňuje traťové koleje a zastávky. Odpovědnostmi tratě je správa dopraven a jejich propojování traťovými úseky včetně rušení traťových úseků.



Obrázek 8 Analytické třídy

Pokračoval jsem nakreslením objektového diagramu. Z tohoto diagramu a také z textu jsem určil vazby a násobnosti.

Také s přihlédnutím k dělení dopraven – kapitola 2.1 Pojmy z technologie železniční dopravy – jsem zavedl společného předka *Dopravna* pro třídy *Stanice*, *Odbočka*, *Hradlo* a *Hláska*. Stejně tak jsem i podle věty „*Vlastnosti tratě udávají rychlostníky a sklonovníky.*“ zavedl společného předka *Určuje vlastnosti* pro třídy *Sklonovník* a *Rychlostník*. Do těchto předků jsem přesunul společné atributy jejich následníků. Také jsem se v tomto kroku rozhodl sloučit třídy *Vjezdové návěstidlo* a *Oddílové návěstidlo* do jedné třídy *Kryjící návěstidlo*, protože plní stejnou funkci – ukončují traťový úsek.



Obrázek 9 Analytické třídy II

4.2 Specifikace návrhových tříd a vazeb

V této podkapitole popíši zásadní provedené změny při přechodu z analytického modelu na návrhový model a také provedu zdůvodnění použití skládání nebo obsažení a použití datových struktur kolekcí.

4.2.1 Specifikace atributů a změny v uspořádání tříd

V prvním kroku jsem specifikoval atributy návrhových tříd. Atributy *název* a *označení* jsem definoval jako textový datový typ *string* – kvůli očekávanému výskytu písmen. Atributy *začátek* a *konec* ve třídě *Nástupiště* a *kilometrická poloha* v ostatních třídách, kde se vyskytuje, jsem definoval desetinným datovým typem *float*. Dále atribut *směr* – ve třídách *Určuje vlastnosti*, *Odjezdové návěstidlo* a *Kryjící návěstidlo* – může nabývat pouze dvou hodnot, proto jsem se rozhodl definovat jej jako typ *boolean*, kde hodnota pravda bude představovat směr jízdy od začátku tratě ke konci. Atributy *sklon* ve třídě *Sklonovník* a *traťová rychlost* ve třídě *Rychlostník* představují celočíselný datový ty *integer*. Dále ve třídě *Zhlaví* oba atributy musí představovat tabulku hodnot, proto jsem v tomto případě zvolil dvourozměrné pole. V případě atributu *rychlosti* se jedná o pole celočíselných hodnot, u atributu *poslední výhybky* se jedná o pole desetinných čísel. Dále ve třídě vlak jsem se rozhodl v tento okamžik vyřadit atribut *hnací vozidlo* a nahradit jej atributem *normativ*, protože tato skutečnost bude dostatečně určena samotným normativem. Tento atribut bude objektovým odkazem na instanci třídy *Normativ*. Do třídy Normativ jsem vložil atribut tabulka, který je definován dvourozměrným polem celočíselných hodnot a představuje vlastní zátěžovou tabulku.

V tomto okamžiku jsem také přejmenoval všechny třídy a atributy podle konvencí pro pojmenování v Pascalu, proto od této chvíle se třídy nazývají *TNazevTridy* a atributy *aNazevAtributu*.

Z důvodu rozdílné struktury a zacházení s tabulkami ve třídě *TZhlavi* jsem její atributy *aRozvetveni* a *aPosledniVyhybka* přesunul přímo do třídy *TOdbocka* a do třídy *TStanice* dvakrát, pro jedno zhlaví s koncovkou *L*, pro druhé s koncovkou *S*, kde *L* představuje zhlaví blíže k začátku tratě. Původní třídy *TZhlavi* jsem zrušil. Pro snadnější manipulaci nejen s výše uvedenými poli (atributy *aRozvetveni* a *aPosledniVyhybka*) jsem do tříd konkrétních dopraven přidal atributy *aTKolejeL* a *aTKolejeS*, které jsou definované jako jednorozměrné pole objektových odkazů na instance třídy *TTratovaKolej*. Ovšem z důvodu rozdílného zacházení

s těmito atributy, jsem do hierarchie dědičnosti dopraven vložil jednu úroveň, a to konkrétně rozdělení na dopravny s rozvětvením a bez něj – třídy *TSRozvetvenim* a *TBezRozvetveni*.

Třída *TStanice* bude také obsahovat atributy *aNastupiste* (definované stejně jako v předchozím případě) a *aKoleje*. Tento atribut je definovaný jako pole odkazů na instance třídy *TStanicniKolej*.

Třídě *TStanicniKolej* jsem přiřadil dvojici atributů *aNavestidloL* a *aNavestidloS*, pro uchování odkazů na instanci třídy *TUkoncujiciKolej* (Odjezdové návěstidlo) pro každý směr odděleně, proto je z této třídy možné vypustit atribut *aSmer*. Tuto úpravu jsem následně provedl i ve třídě *TTratovaKolej* pro vjezdová návěstidla a i pro samotné kolekce návěstí určujících vlastnosti. Proto jsem atribut *aSmer* vypustil i z tříd *TKryjiciDopravnu* a *TUrcujiciVlastnostiTrati*. Dále jsem také třídám *TUkoncujiciKolej*, *TKryjiciDopravnu* a *TUrcujiciVlastnostiTrati* určil společného předka v podobě třídy *TNavest*, protože ve všech těchto případech se jedná o návěsti, ale každý typ má jiný význam.

Místo třídy *Vlak* jsem použil třídu *TSituace*, která plní funkci přenosu parametrů výpočtu z prezentační vrstvy do aplikační vrstvy, konkrétně třídě *TVypocet*. Tato třída obsahuje funkce potřebné k samotnému výpočtu.

Dále jsem přidal třídy datové vrstvy, které slouží k ukládání a načítání dat, a třídy prezentační vrstvy (formuláře), které slouží k ovládání aplikace a zadávání dat uživatelem.

4.2.2 Použití skládání nebo obsažení

V předchozí podkapitole (4.2.1) jsem zmínil začlenění atributů z analytické třídy *Zhlaví* do tříd *TStanice* a *TOdbocka*. K tomuto kroku mě kromě uvedených důvodů (zjednodušení manipulace a přístupu k hodnotám a mírně odlišná struktura) vedla i násobnost této relace 1:1, která toto umožnila. Zároveň tato změna není ani proti předpokládaným odpovědnostem.

Těsnou vazbu v relaci 1:1 jsem také pozoroval např. mezi staniční kolejí a návěstidly, která ji ohraničují. Pokud by v realitě byla staniční kolej zrušena (snesena), pozbývají daná návěstidla smysl (a nejspíše by byla odstraněna společně s kolejí). Proto jsem zde použil obsažení, které se projevilo skutečností, že ohraničující návěstidla jsou požadována již jako parametr konstruktoru a existují celou dobu existence staniční koleje, lze pouze měnit jejich polohu a označení.

Volnou vazbu se staniční kolejí jsem oproti předchozímu případu zpozoroval u nástupiště. Je to dáno tím, že jedno nástupiště může příslušet více kolejím, čili vzniká relace

M:1. Příkladem volnosti této vazby z reality může být opět rušení staniční koleje – kolej se zruší, ale nástupiště je případně stále využíváno u jiné koleje.

Těsná vazba, ovšem v relaci 1:N, se také vyskytuje ve vztahu tratě s traťovými úseky a dopravnami. Zruší-li se trať (ve smyslu je snesena), jsou zrušeny i traťové úseky a dopravny. Stejný případ představují i návěsti určující vlastnosti trati ve vztahu k traťové koleji, kdy tyto návěsti mají smysl pouze za existence příslušné traťové koleje.

4.2.3 Použití kolekcí

O použití kolekcí jde hlavně v případě relací 1:N. Pokud se někde objeví relace M:1, je potřeba někde ve struktuře programu vytvořit jakýsi číselník, který bude udržovat jednotlivé součásti. Tím ovšem vzniká vztah číselník k součásti s relací 1:N.

V tomto programu jsem použil pouze dvoje kolekce – pole a lineární seznamy.

Pole jsem použil v případech, kdy jsou součásti organizovány podle daného klíče – čísla. Takovým případem jsou staniční koleje a nástupiště v rámci stanice nebo traťové koleje v rámci traťového úseku. Výhodou tohoto řešení je rychlý přístup k požadovanému objektu. Nevýhodou je například paměťová náročnost v případě nesymetricky rozvinutých stanic, kdy je v paměti vymezeno místo i pro tyto neobsazené pozice.

V ostatních případech jsem použil lineární seznamy.

Ve třídě *TTrat* jsem pro uchování dopraven a traťových úseků použil lineární seznamy nesetříděné, protože v tomto případě je hlavním účelem uchování jednotlivých objektů bez ohledu na pořadí.

Naopak potřeba využít setříděný lineární seznam vyvstala při uchování návěstí určující vlastnosti tratě ve třídě *TTratovaKolej* nebo zastávek ve třídě *TTratovyUsek*, kde je i s ohledem na následující výpočet toto setřídění podle kilometrické polohy žádoucí.

4.2.4 Specifikace metod

Specifikace metod spočívá v rozložení odpovědností na jednotlivé operace a následném konkretizování parametrů a návratových hodnot.

Na příklad správa staničních kolejí ve třídě *Stanice* představuje přidání nové koleje, odebrání existující koleje a ověření, zda požadovaná kolej existuje. Z tohoto vyplývají tři metody: *PridatKolej*, *OdebratKolej* a *ExistujeKolej*. První dvě metody jsem ponechal bez návratových hodnot, poslední metoda vrací hodnotu typu *boolean*. Protože jsem tyto dvě

metody nemají návratové hodnoty, je v případě chyby, např. již existující kolej se zadaným číslem, vyvolána výjimka. Všechny metody mají parametr celočíselného typu, který určuje číslo koleje, se kterou se má daná operace provést. Metoda *PridatKolej* má navíc druhý parametr *paKolej*, kterým je předána samotná kolej pro vložení do pole staničních kolejí.

Ukázkou polymorfismu může být případ, kdy si každá dopravna organizuje propojení se sousedními úseky, proto zde vznikly metody *PridatZausteni* a *OdebratZausteni*. Opět se jedná o procedury bez návratových hodnot, parametry jsou přidávaný, nebo odebíraný úsek a směr, ze kterého tento úsek ústí do dopravny. Ovšem dopravny bez rozvětvení tuto operaci provedou jinak než dopravny s rozvětvením, kde také stanice toto zaústění organizuje odlišně od odbočky. Proto tyto metody jsou na vrcholu hierarchie dědění deklarované jako abstraktní a implementovány jsou až ve třídách *TBezRozvetveni*, *TOdbocka* a *TStanice*. Ovšem tuto vlastnost jsem zde nemohl plně využít z důvodu omezení vývojovým prostředím, konkrétně cyklickou deklarací používaných souborů. Proto jsem musel tyto metody umístit až do druhé úrovně hierarchie, kde ve třídě *TBezRozvetveni* jsou přímo implementovány a ve třídě *TSRozvetvenim* zůstávají abstraktní.

Dále se během implementace datové vrstvy objevila potřeba přetížení metody *PropojitDopravny* ve třídě *TTrat*. Jedna verze vytváří nový traťový úsek a následně jej zařadí do seznamu traťových úseků, proto jejími parametry jsou parametry konstruktoru třídy *TTratovyUsek*, tedy první dopravna, druhá dopravna, počet kolejí a pole návěstidel kryjících tyto dopravny. Druhá verze je používaná při načítání tratě ze souboru, kdy je traťový úsek vytvořen při načítání a je potřeba tento úsek pouze zařadit do seznamu, proto druhá verze má parametr pouze jeden, kterým je traťový úsek k zařazení.

V předchozím odstavci byl zmíněn konstruktor třídy *TTratovyUsek*. U určování parametrů konstruktoru, jsem také vycházel ze situace, představuje-li tato třída nějaký celek, který obsahuje součásti a také jak těsná vazba v tom případě je. Pokud se jedná o zmíněný traťový úsek, jsou požadovány i hodnoty parametrů pro vytvoření traťových kolejí tohoto úseku ihned v rámci jeho vytváření.

4.2.5 Třída TVypocet

V případě této třídy byla potřeba vytvořit metody, které budou provádět jednotlivé dílčí výpočty. Jedná se o výpočty dráhy a časů podle vzorců 2 až 6, vyjádření třídy sklonu, zjištění maximální rychlosti v tabulce normativu, určení míst zastavení, resp. místa, ke kterému se počítá průjezd vlaku atd. Dále tato třída také obsahuje metody pro rozdělení úseků na menší a

také funkci pro zaokrouhlení výsledného času na půlminuty. Všechny tyto metody jsou soukromé, zvenku nepřístupné.

Třída *TVypocet* má pouze jednu veřejně přístupnou metodu (*Spocitat*), kterou se vyvolá samotný proces výpočtu. Požadovanými vstupními parametry jsou instance třídy *TSituace* se zadanými hodnotami pro výpočet a odkaz na pole jízdního řádu, které tato metoda naplní hodnotami.

Metoda *Spocitat* provede výpočet v následujících krocích. Celá trasa, určená místem odjezdu z výchozí stanice a místem zastavení v koncové stanici, je rozdělena nejdříve podle míst v dopravnách, kde vlak zastavuje, nebo projíždí, poté i podle zastavení v zastávkách na trati, kde vlak zastavuje. Poté se tyto úseky dělí podle míst, kde se mění traťová rychlost, tj. podle polohy rychlostníku příp. posunutí o délku vlaku v případě, že traťová rychlost je vyšší než v předchozím úseku. Následují úpravy úseků podle lokálních omezení rychlosti při jízdě do odbočky, v tomto kroku může dojít i ke slučování několika úseků a jejich následnému rozdělení nebo jen k úpravě hodnot rychlosti. Dále se úseky rozdělí podle míst, kde se mění působení sklonu, tj. poloha sklonovníku posunutá o polovinu délky vlaku.

První z kontrol je ověření dosažitelnosti stanovené rychlosti se zadaným vlakem na sklonu v jednotlivých úsecích. Dále se od posledního k prvnímu projdou všechny úseky a ověří se, jestli v těchto úsecích stihne vlak zbrzdit, pokud tomu tak není, jsou upraveny hodnoty rychlostí v daném i předcházejícím úseku. Poslední kontrolou je ověření dráhy rozjezdu pro každý úsek od začátku ke konci. V tomto kroku může dojít také k úpravám hodnot rychlostí, nebo i ke sloučení dvou úseků v případě, kdy délka jednoho úseku nebyla dostatečná pro požadované zbrzdění a délka předcházejícího úseku nebyla dostatečná pro zrychlení na požadovanou rychlost.

Takto rozdělené úseky se spočítají funkcí pro výpočet času na jednom úseku, která vrací zaokrouhlenou celočíselnou hodnotu – čas v sekundách. Toto řešení jsem zvolil, protože mi přijde dostatečně přesné i s ohledem na následné zaokrouhlování na násobky třiceti.

Hodnoty z jednotlivých úseků jsou sečteny ve skupinách odpovídajících úseku mezi dvěma místy, která jsou významná pro grafikon, tj. mezi odjezdy nebo průjezdy stanicemi, odbočkami, hradly nebo hláskami nebo zastavením v zastávce, a následně zaokrouhleny na půl minuty a zapsány na odpovídající místo v poli, které bylo předáno jako druhý parametr a představuje jízdní řád.

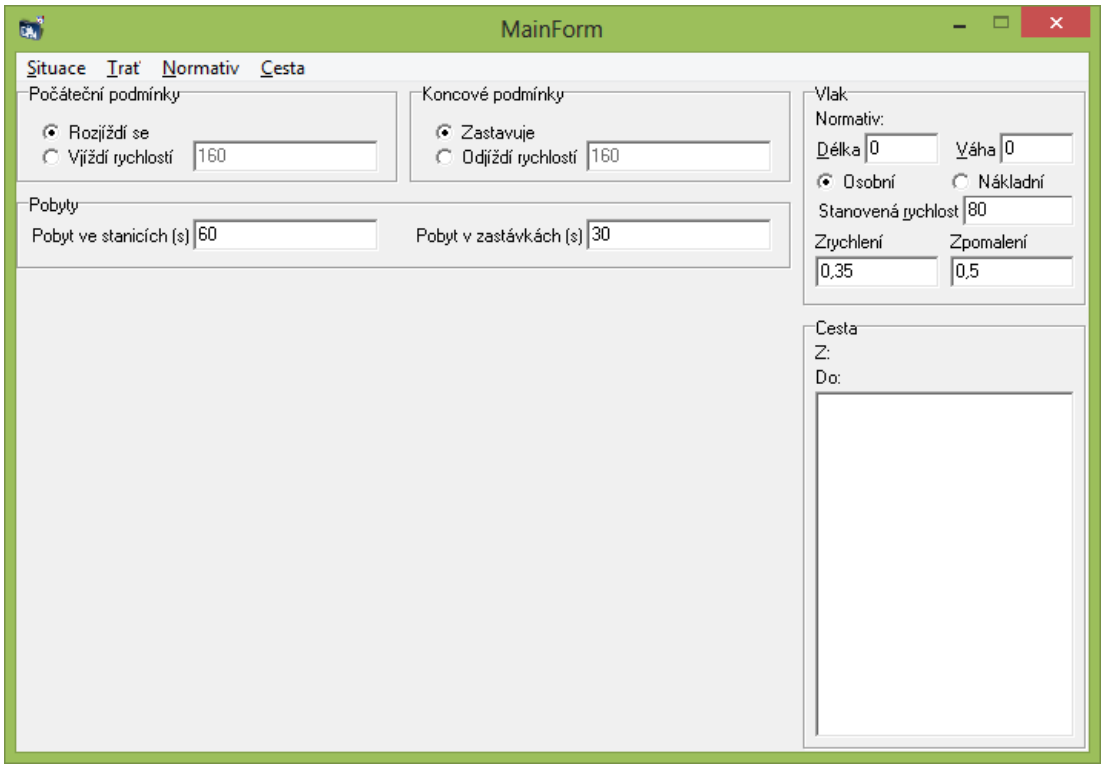
5 Realizace aplikace

Aplikaci jsem vytvářel v programovacím jazyce Object Pascal ve vývojovém prostředí Delphi 7 společnosti Borland. Program, který je zkompilován v tomto prostředí je možné spouštět v prostředí OS Windows, ovšem musí v systémové složce nebo ve složce se spouštěcím souborem aplikace existovat soubor ***qtintf70.dll***, který představuje knihovnu uživatelského rozhraní. K aplikaci je přiložen ve složce *data*.

K programu je také přiložen soubor s tratí použitou v modelových příkladech.

5.1 Seznámení s programem

Po spuštění aplikace se zobrazí hlavní okno, kde se v horní části okna nachází menu, dále se na hlavním okně nachází možnosti nastavení výpočtu sdružené do skupin. Poslední část okna je určena pro vykreslení výsledku.

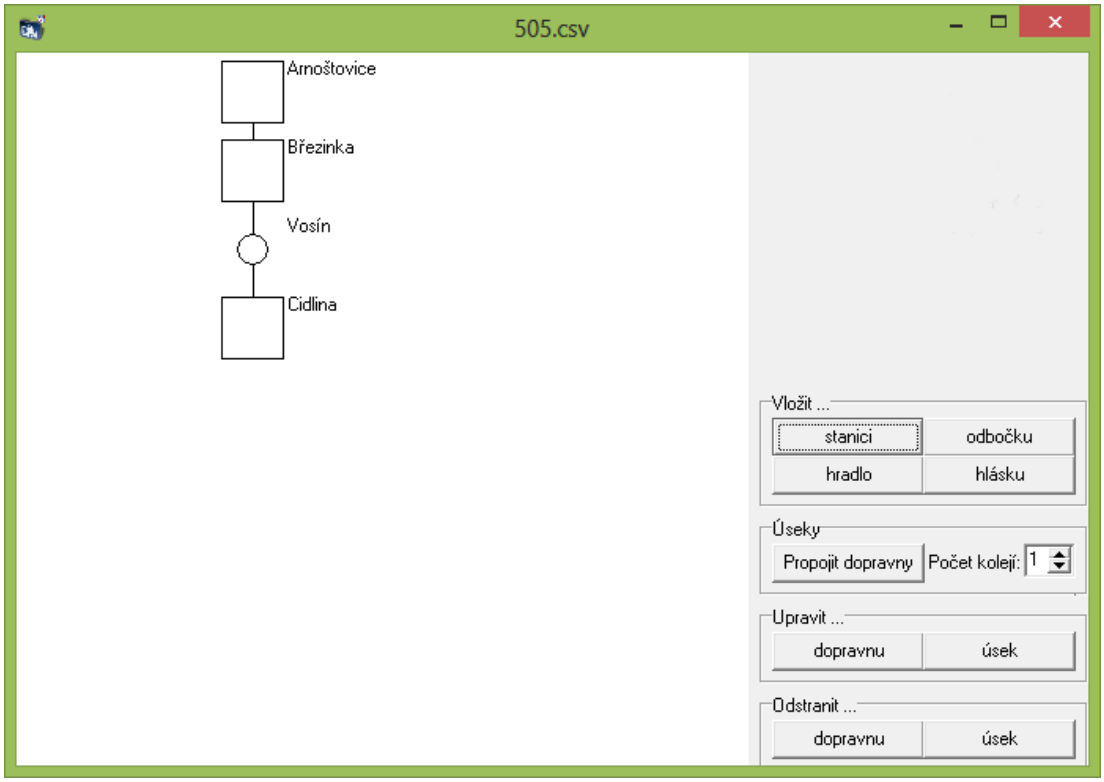


Obrázek 10 Hlavní okno aplikace

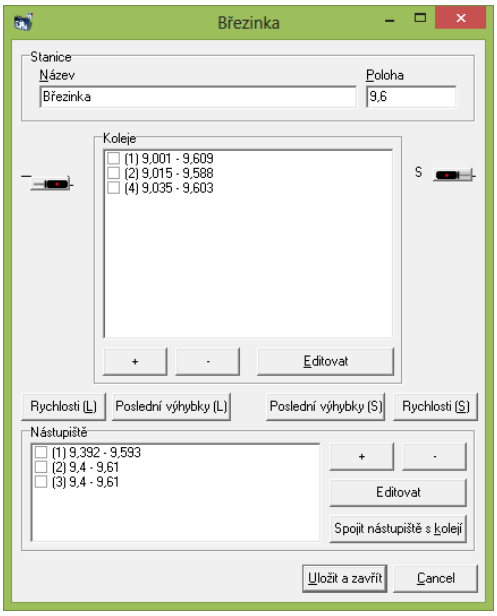
Nabídka obsahuje možnosti pro práci s tratí, jako je vytvoření nové tratě, načtení uložené, uložení aktuálně upravované i pod jiným jménem a možnost pro vyvolání okna pro editace tratě. Další položkou menu je Normativ, která jej umožňuje pouze načíst nebo zobrazit. Poslední položkou je Cesta, jejíž položka umožní vybrat cestu vlaku zvolením počáteční a koncové stanice.

5.1.1 Editace tratě

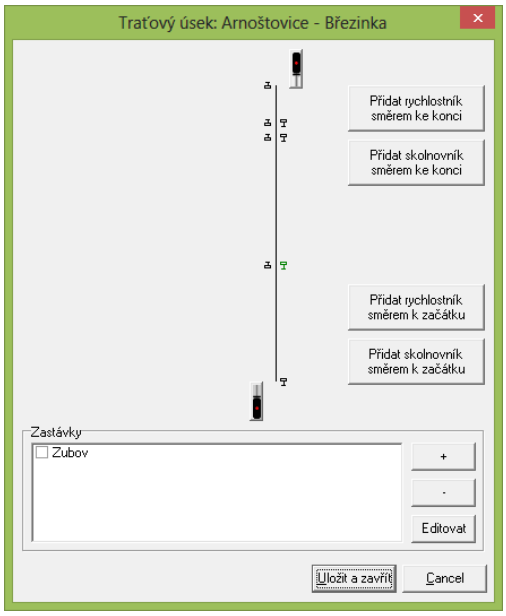
Tato možnost vyvolá okno pro editaci tratě. Pokud v programu není žádná trať načtena, je uživatel dotázán na vytvoření nové tratě. Toto okno umožňuje úpravy tratě zvolením požadované možnosti v pravé části. V levé části je prostor, kde se vykresluje schéma tratě.



Obrázek 11 Okno editace tratě



Obrázek 12 Editace stanice



Obrázek 13 Editace traťového úseku

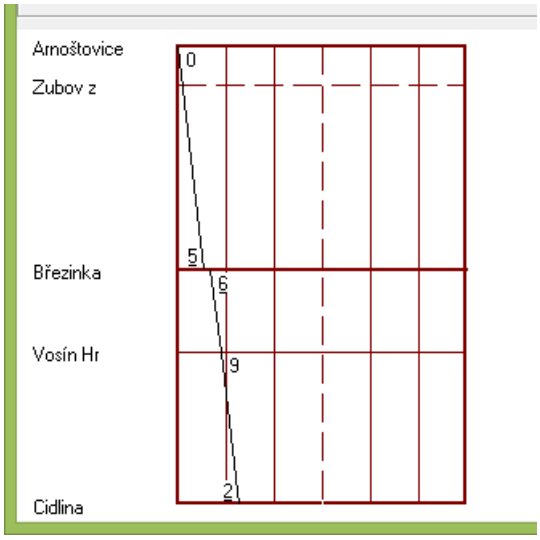
Obrázky 12 a 13 představují okna, kterými se zadává nebo upravuje stanice a traťový úsek.

Ovládání programu je detailně popsáno v uživatelské příručce na CD, které je přiloženo k této práci.

5.2 Výpočet

Samotný výpočet lze spustit až po načtení nebo vytvoření tratě, vybrání cesty na ní a po načtení normativu. Spustí se výběrem možnosti *Počítat* v možnosti *Situace* a probíhá ve dvou krocích, nejdříve je uživatel postupně dotazován na jednotlivé koleje ve stanicích a na traťových úsecích, po kterých má vlak projíždět, a následně probíhá samotný výpočet.

Po provedení výpočtu proběhne podle navrácených dat vykreslení výsledku v levé dolní části hlavního okna.



Obrázek 14 Vykreslení výsledku

6 Porovnání výstupu

V tabulkách 7 a 8 jsou uvedeny rozdíly ve výsledcích mezi ručním výpočtem a výpočtem pomocí aplikace.

Tabulka 7 Porovnání výsledků - osobní vlak

			ruční výpočet				aplikace			
	od	do	A-Z	Z-B	B-V	V-C	A-Z	Z-B	B-V	V-C
1	2,871	3,241	49,18				49			
2	3,241	3,969	61,98				62			
3	3,969	4,009		15,08				15		
4	4,009	6,959		133,81				134		
5	6,959	8,707		72,92				73		
6	8,707	9,578		89,5				90		
7	9,578	9,816			37,3				37	
8	9,816	10,159			22,73				23	
9	10,159	12,242			103,6				104	
10	12,242	12,659				57,78				58
11	12,659	15,059				93,83				94
12	15,059	15,905				40,45				40
13	15,905	16,900				100,66				101
jízdní doby			111,16	311,31	163,63	292,72	111	312	164	293
jízdní doby – minuty			1	5	2	4	1	5	2	4
jízdní doby – sekun			51,16	11,31	43,63	52,72	51	12	44	53
JD po zaokrouhlení			2	5,5	3	5	2	5,5	3	5

Tabulka 8 Porovnání výsledků - manipulační vlak

			ruční výpočet		aplikace	
	od	do	čas C-V	čas V-B	čas C-V	čas V-B
1	16,625	16,143	61,9		62	
2	16,143	14,850	72,45		72	
3	14,850	12,450	175,12		175	
4	12,450	12,104	21,97		22	
5	12,104	10,020		113,13		113
6	10,020	9,045		101,64		
6a	10,020	9,950				6
6b	9,950	9,045				95
jízdní doby			331,44	214,77	331	214
jízdní doby – minuty			5	3	5	3
jízdní doby – sekundy			31,44	34,77	31	34
JD po zaokrouhlení			6	4	6	4

U osobního vlaku (tabulka 7) tyto hodnoty odpovídají po zaokrouhlení na celá čísla hodnotám spočítaným programem.

U manipulačního vlaku (tabulka 8) se tyto hodnoty také shodují, až na případ úseku 6, resp. 6a a 6b, které se mezi sebou po zaokrouhlení liší o jednu sekundu. To je způsobené

rozdělením úseku 6 v ručním výpočtu na dva úseky 6a a 6b ve výpočtu v aplikaci a následným zaokrouhlením. Při ručním výpočtu úseků 6a a 6b vycházejí hodnoty 6,3 a 95,34. Tyto hodnoty se samostatně správně zaokrouhlí, jak jsou uvedeny v tabulce 8, ovšem, pokud se nejprve sečtou – v modelové případě bylo popsáno, že vlak míjí vjezdové návěstidlo požadovanou rychlostí, čili v dalším úseku bylo možné tento vliv sklonu (70 metrů z 975) zanedbat, protože na této části vlak ještě nebrzdil k zastavení – vychází hodnota 101,64, která se již ale zaokrouhluje nahoru a vzniká tedy rozdíl jedné sekundy. Ve výsledném zaokrouhlení se ale tento rozdíl ztratí ve většině případů. Tato okolnost by mohla ovlivnit výsledky, pouze v případech, kdy celkový součet časů mezi dvěma místy, ke kterým se počítají jízdní doby, pohybuje okolo celé minuty nebo půlminuty.

Proto lze prohlásit, že výsledky aplikace jsou reálné.

Závěr

V práci jsou zmíněny možnosti výpočtu jízdních dob použitím některých na trhu dostupných aplikací. Dále je také detailně popsán postup ručního výpočtu i s potřebnými vzorci a zjednodušeními. Také jsou popsány základní pojmy z technologie železniční dopravy související s problematikou výpočtů jízdních dob a základní pojmy z objektově orientovaného programování.

V práci byl popsán obecný postup analýzy a návrhu aplikace, podle kterého byla vytvořena vlastní aplikace. Aplikace splnila požadavky, tj. umožňuje uživateli sestavit trať, zadat parametry vlaku, ze kterých následně vypočítá jízdní doby, a může být použita na příklad vyučujícím předmětu Technologie a řízení železniční dopravy k přípravě příkladů na cvičení nebo ke kontrole hodnot v semestrálních pracích.

Jak bylo dokázáno porovnáním výsledků modelových příkladů, poskytuje aplikace na výstupu správné výsledky.

Použitá literatura

[1] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. Vyd. 1. Překlad Bogdan Kiszka. Brno: Computer Press, 2007, 567 s. ISBN 978-80-251-1503-9.

[2] ŠOTEK, Karel, Hynek BACHRATÝ, Karel GREINER, Miroslav GÁBOR, Petr VESELÝ a Viliam TAVAČ. Tvorba jízdního řádu pomocí výpočetní techniky na Českých drahách. [online]. [cit. 2013-05-29]. Dostupné z: http://spz.logout.cz/zabezpec/sena/sena.html

[3] GREINER, Karel. Distribuovaná aplikace editoru vlaků. Perner's contacts [online]. 2009, roč. 4, č. 2 [cit. 2013-05-29]. Dostupné z: http://pernerscontacts.upce.cz/14_2009/greiner.pdf

[4] Viriato - Timetable planning at your fingertips. *SMA und Partner AG* [online]. [cit. 2013-05-29]. Dostupné z: http://www.sma-partner.ch/index.php?option=com_content&view=article&id=368&Itemid=210&lang=en

[5] Simulace železnice. *OpenTrack Railway Technology* [online]. [cit. 2013-05-29]. Dostupné z: http://www.opentrack.cz/opentrack_cz.html

[6] iPlan - tak se dělá jízdní řád. *iRFP - Institut pro plánování regionální a dálkové dopravy* [online]. [cit. 2013-05-29]. Dostupné z: http://www.irfp.de/cesky/fbs/iplan.html

[7] Vyhláška 173 Ministerstva dopravy ze dne 22. června 1995, kterou se vydává dopravní řád drah, ve znění vyhlášky č. 242/1996 Sb. a vyhlášky č. 174/2000 Sb. In: *Sbírka zákonů České republiky*. 1995.

[8] BAYER, Tomáš. Datové struktury a datové typy: Základní datové typy. Odvozené datové typy. Základní datové struktury. Odvozené datové struktury. [online]. [cit. 2013-05-29]. Dostupné z: http://web.natur.cuni.cz/~bayertom/Prog1/programovani3.pdf

[9] Základní datové struktury. In: [online]. [cit. 2013-05-29]. Dostupné z: http://flashi.wz.cz/materialy/szz/13-Z%C3%A1kladn%C3%AD%20datov%C3%A9%20struktury%20%28pole,%20z%C3%A1sobn%C3%ADk,%20bin%C3%A1rn%C3%AD%20strom%20atd.%29,%20datov%C3%A9%20struktury%20vhodn%C3%A9%20pro%20fyzickou%20implementaci%20rela%C4%8Dn%C3%ADch%20dat%20v%20S%C5%98BD%20%28ha%C5%A1ovac%C3%AD%20tabulky,%20B-strom%29..pdf

[10] Třívrstvá architektura (Three-tier architecture). *Management mania* [online]. [cit. 2013-05-29]. Dostupné z: http://managementmania.com/cs/trivrstva-architektura-three-tier-architecture

PŘÍLOHY

Obsah přiloženého CD

- Tato práce ve formátu PDF
- Zdrojové kódy aplikace
- Spustitelná verze aplikace
- Detailní uživatelská příručka k aplikaci