PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Univerzální grafické uživatelské rozhraní s editorem formulářů pro řádkové utility

2010

Michal Němec

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně.

30. července 2010 Michal Němec

Anotace

Cílem této bakalářské práce bylo vytvořit aplikaci pro spouštění předloh pro tzv. řádkové utility.

Děkuji rodičům za podporu při studiu.

Obsah

Seznam obrázků

1. ÚVOD

Aplikace Univerzální grafické uživatelské rozhraní s editorem formulářů pro řádkové utility (dále jen aplikace) vznikla jako závěrečná bakalářská práce studia Aplikované informatiky na Univerzitě Palackého v Olomouci. Toto téma práce bylo zvoleno z témat prací vypsaných fakultou a je určeno pro dva studenty.

1.1. Rozdělení práce

Tato bakalářská práce se zabývá řešením první části zadání a to je `Spouštěč řádkových utilit v grafickém rozhraní`, dále jen `Spouštěč`. Druhou část tématu, kterou je `Editor šablon řádkových utilit`, dále jen `Editor`, řeší ve své práci [1]. Obě části práce jsou zpracovány jako dvě samostatné aplikace se dvěma dynamicky linkovanými knihovnami (aplikace + jedna knihovna), ve kterých jsou sdíleny společné části obou aplikací. Toto rozdělení jasně vymezuje práci každého studenta. Je jasné že na společných částech projektu, které jsou součástí dynamicky linkovaných knihoven, museli oba studenti prokázat nutnou míru spolupráce v podobě společných konzultací. Výsledný kód je pak už výhradně prací každého řešitele.

1.2. Cíle práce

Cílem této práce je zpřístupnit méně zdatným uživatelům počítačů programy ovládané pouze zadáváním parametrů na příkazové řádce, tzv. řádkové utility v grafickém rozhraní Windows.

Úkolem této bakalářské práce je navrhnout univerzální systém, který by umožnil definovat uživatelská rozhraní pro nejběžnější řádkové utility bez nutnosti cokoliv programovat. Každý program bude mít svou šablonu, u které půjde jednoduchým způsobem definovat, jaké parametry lze u daného programu na příkazové řádce zadávat. Vytvořený systém umožní pohodlně vytvářet tyto šablony, a potom je použít pro spouštění programů pracujících na příkazové řádce.

1.3. Existující řešení

V minulosti toto zadání bakalářské práce na Přírodovědecké fakultě Univerzity Palackého v Olomouci bylo řešeno již několikrát, vždy ale formou jednoduché aplikace která umožňuje načíst a pracovat jen s jednou šablonou v daný čas a postrádají jakoukoliv správu šablon. Je možné, že řešení mimo fakultu existuje mnohem více, avšak nebyla zjištěna.

2. ODBORNÁ ČÁST

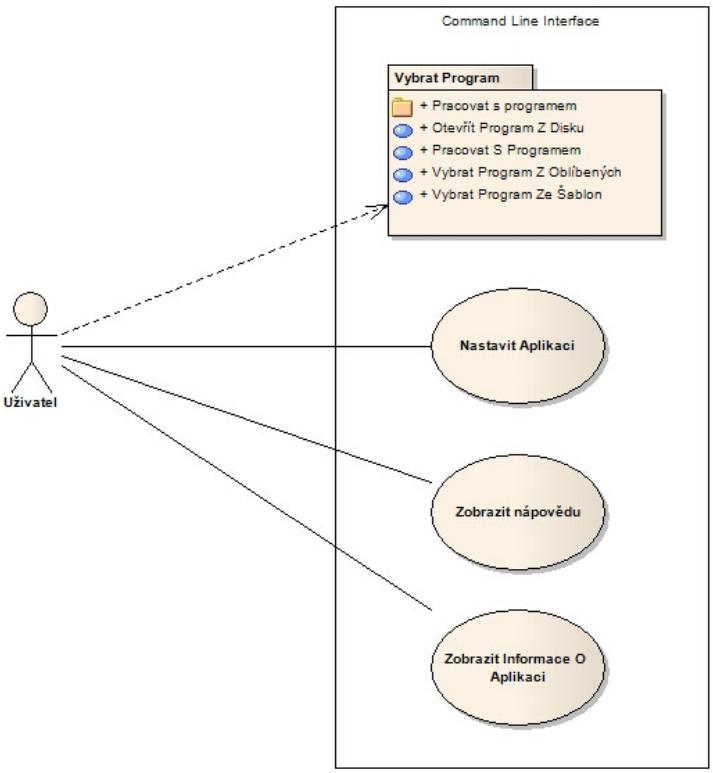
2.1. Aktuální stav řešené problematiky

Uživatel doposud využívá příkazový řádek operačního systému, do kterého nejprve zapíše název řádkové utility. Protože nezná všechny její parametry, musí ji nejprve spustit bez parametrů (nebo z parametrem nápovědy), tak aby se zobrazila nápověda k utilitě. Poté musí znovu zapsat název utility do příkazového řádku již s potřebnými parametry. Pokud chce spustit novou instanci, musí znovu zapsat utilitu i s parametry do příkazového řádku.

2.2. Analýza požadavků

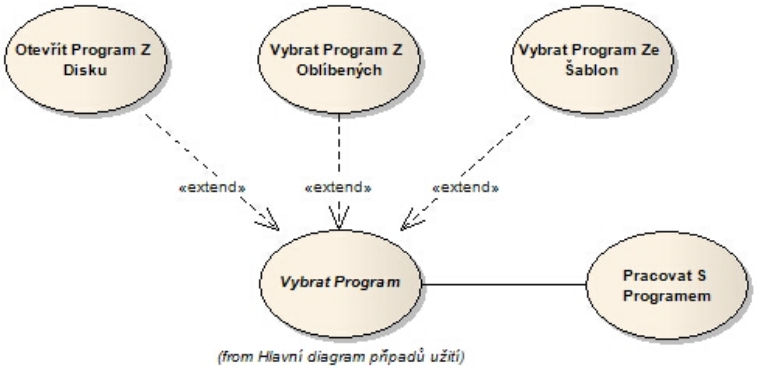
Případy užití

Sběrem a analýzou požadavků vznikl model případů užití aplikace. Pro přehlednost je rozdělen na několik částí. Obrázek č. 1. ukazuje základní kostru aplikace.



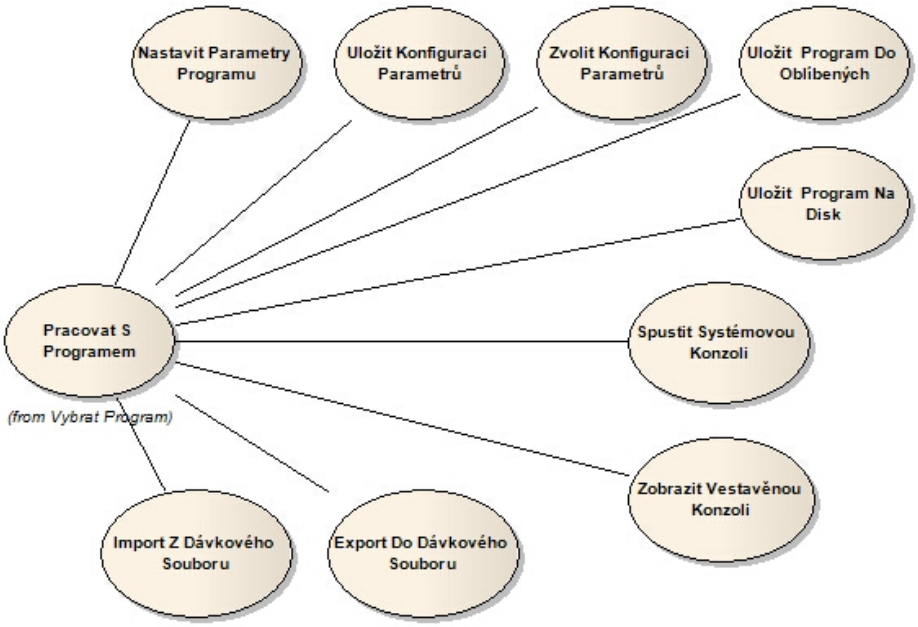
Obrázek 1. Hlavní model aplikace

Obrázek č. 2. ukazuje možnosti výběru programu, respektive šablony programu, dále jen *programu*.

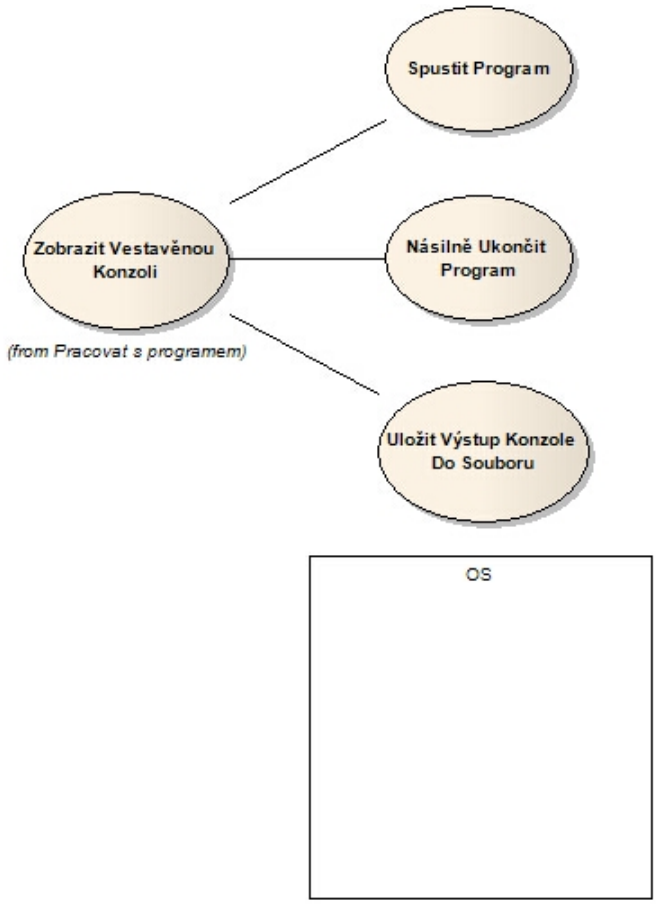


Obrázek 2. Model části aplikace - výběr programu

Obrázek č. 3. a 4. ukazuje možnosti práce s programem a se systémovou a vestavěnou konzolou.



Obrázek 3. Model části aplikace - práce s programem



Obrázek 4. Model části aplikace - práce s vestavěnou konzolou

Další požadavky

Jako jazyk aplikace je zadavatelem požadován výhradně **anglický jazyk** a na další jazykové mutace si neklade žádné nároky. Z tohoto důvodu bude celé uživatelské rozhraní v anglickém jazyce.

2.3. Struktura aplikace

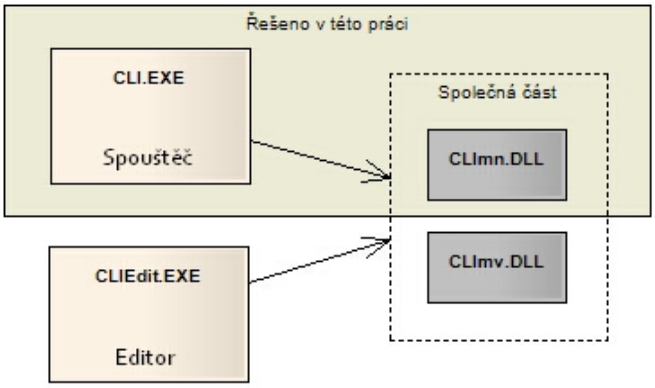
Celý projekt je rozdělen do čtyř modulů (obr. 5.) a to výhradně proto, že zadání bakalářské práce je pro dva studenty.

Moduly řešené v této části práce:

- `Spouštěč` - *CLI.EXE*
- knihovna společných funkcí - *CLImn.DLL*

Moduly řešené v druhé části práce kolegou[1]:

- `Editor` - *CLIedit.EXE*
- knihovna společných funkcí - *CLImv.DLL*



Obrázek 5. Moduly aplikace

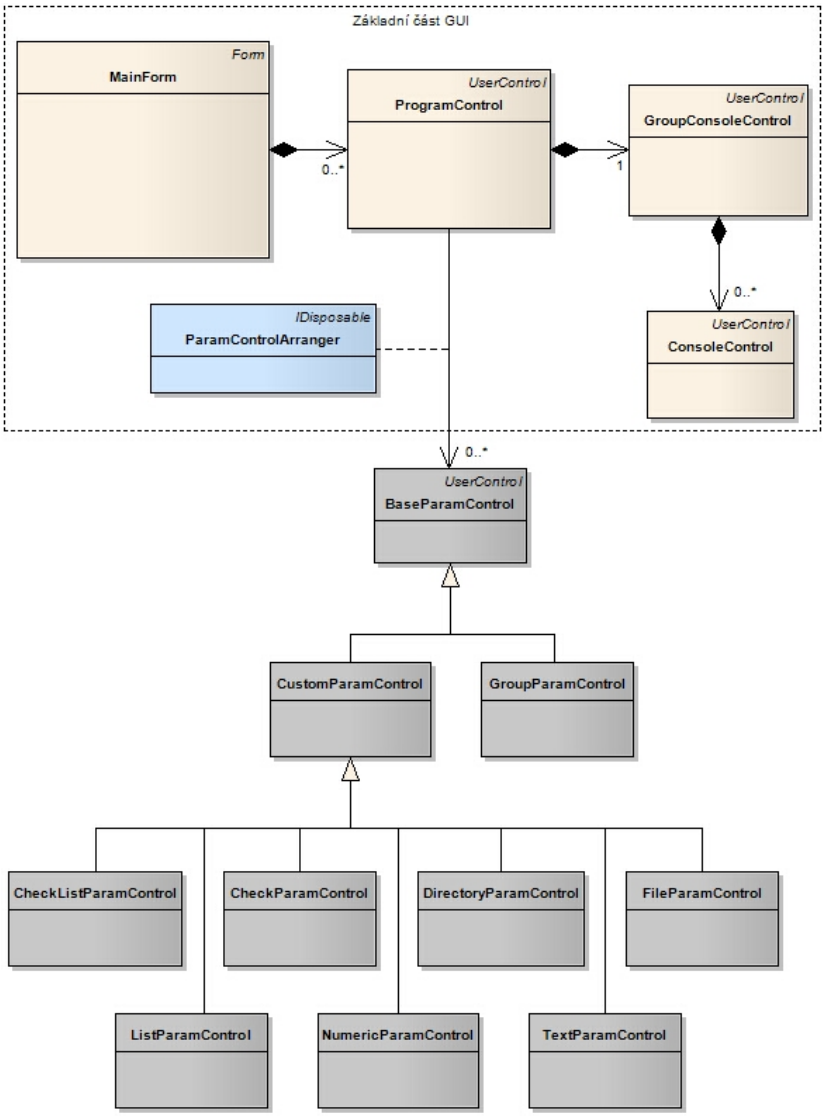
Jak již bylo napsáno, rozdělení na dva spustitelné soubory a dvě dynamicky linkované knihovny, je pouze z důvodu jednoznačného rozdělení práce a zdánlivé vzájemné nezávislosti obou řešitelů. Následující text se bude zabývat popisem pouze prvních dvou modulů a na další dva se jen odkazovat.

2.4. Stavba aplikace

Aplikace je sestavena z jednotlivých samostatných tříd a uživatelských ovládacích prvků, které nesou v aplikaci různou funkčnost a zodpovědnost a jsou vzájemně spojeny rozhraním. Aplikace je navržena jako modulární pro lepší přehlednost a správu zdrojového kódu. Třídy jednotlivých modulů tak řeší vždy pouze problémy, které spadají do jednoho logického celku.

2.4.1. GUI aplikace a jeho třídy

Základní část grafického uživatelského rozhraní aplikace je tvořena čtyřmi částmi (obr. 6.). První částí je vlastní formulář aplikace, který jako první umožňuje interakci s uživatelem. Na základě požadavku uživatele vlastní formulář aplikace zajistí další obsluhu (vytvoření, zviditelnění nebo zánik) navazujících uživatelských prvků.



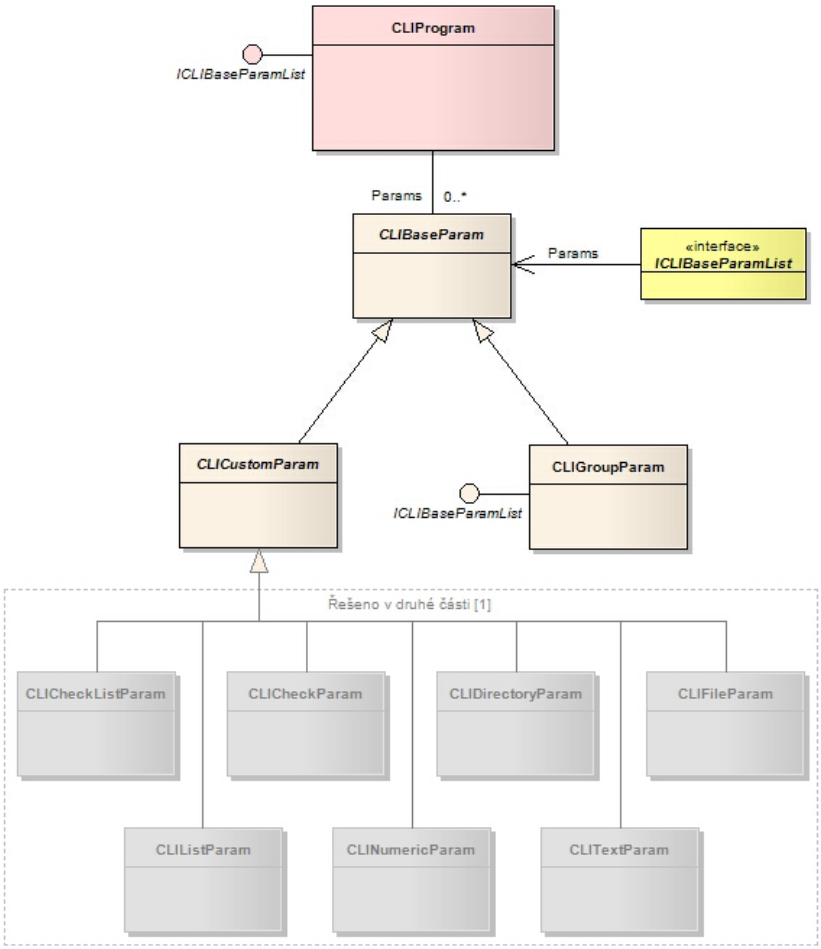
Obrázek 6. GUI aplikace a jeho třídy

Popis tříd GUI aplikace

- `MainForm` hlavní formulář aplikace - slouží jako první nárazník pro uživatele.
- `ProgramControl` karta programu - vytvoří se pro uživatelem vybraný *Program* a zobrazí všechny dostupné informace daného *Programu*. Uživatel tak může pomocí této karty nastavovat jednotlivé prvky (*Parametry*) *Programu*.
- `ParamControlArranger` aranžér grafických ovládacích prvků (*Parametrů*) *Programu* - je neviditelná součást, která má za úkol správné rozmístění grafických ovládacích prvků odvozených z `BaseParamControl`.
- `GroupConsoleControl` skupinová komponenta pro `ConsoleControl` - drží správu nad jednotlivými vestavěnými konzolemi. Aplikace tak umožňuje mnohonásobně spustit daný *Program* a mít přehled nad jednotlivými instancemi spuštěného programu.
- `ConsoleControl` karta vestavěné konzole - poskytuje uživateli kontrolu nad spuštěným procesem, zachytává výstup procesu a zobrazuje vybrané informace o procesu.
- `BaseParamControl` základní předek všech ovládacích prvků *Parametrů* - nese společné vlastnosti všech ovládacích prvků jednotlivých *Parametrů*.
- `GroupParamControl` ovládací prvek - slučuje další *Parametry* do logické skupiny.
- `CustomParamControl` rozšířený předek všech ovládacích prvků *Parametru* (mimo `GroupParamControl`) - přidává další společné vlastnosti pro ovládací prvky jednotlivých *Parametrů* .
- `CheckListParamControl` ovládací prvek parametru - zaškrkávací seznam
- `CheckParamControl` ovládací prvek parametru - pouze zaškrávací prvek typu má/nemá být vybrán
- `DirectoryParamControl` ovládací prvek parametru - zadání adresáře (nebo výběr adresáře z disku)
- `FileParamControl` ovládací prvek parametru - zadání souboru (nebo výběr souboru z disku)
- `ListParamControl` ovládací prvek parametru - výběr položky ze seznam
- `NumericParamControl` ovládací prvek parametru - zadání čísla
- `TextParamControl` ovládací prvek parametru - zadání textu

2.4.2. Serializace objektů do XML

Na základě požadavku načítat a ukládat šablony *Programů* ze souborů a do souborů ve formátu XML, vzniklo serializovatelné jádro (obr. 7.).



Obrázek 7. Serializace objektů do XML

Popis serializace

Serializace má za úkol převést objekt do nějakého perzistentního stavu, v tomto případě na soubor ve formátu XML uložený na disku. Pro serializaci je použita vestavěná třída `XmlSerializer` z prostředí .NET Frameworku která má za úkol serializovat třídu, respektive instanci třídy `CLIProgram`. S touto třídou se také zároveň serializují asociované třídy, které za tímto účelem jsou naprogramovány jako serializovatelné. Výsledek serializace je pak směrován za pomocí třídy `TextWriter` z prostředí .NET Frameworku do souboru na disku.

Řízení serializace

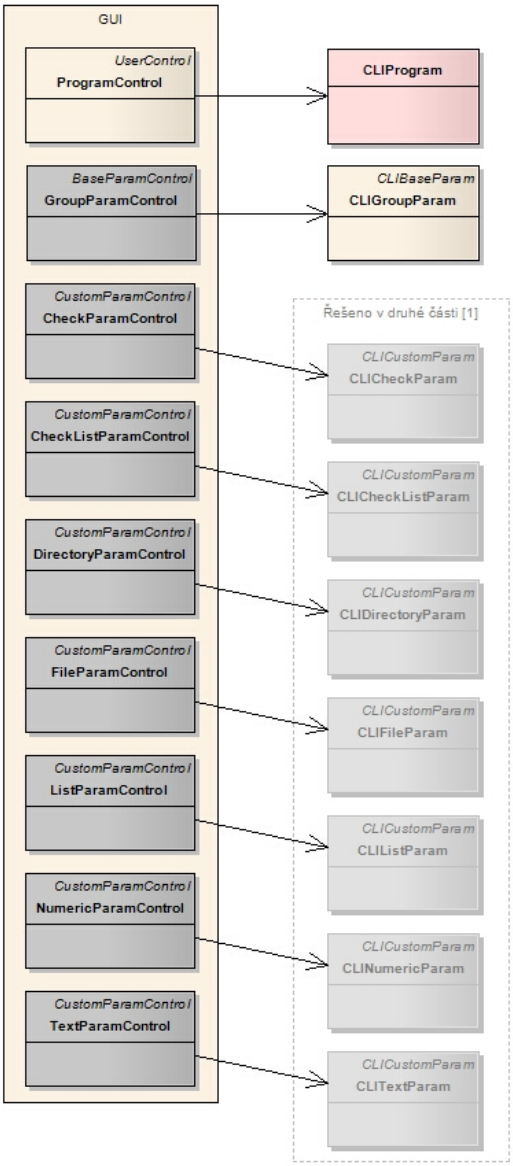
Při použití tohoto způsobu serializace se standartně serializují všechny veřejné vlastnosti daného objektu do xml elementu. Protože není účelné všechny tyto veřejné vlastnosti objektu takto serializovat, tak je na řízení serializace použit systém atributů z prostředí .NET Frameworku, které dodatečně říkají serializátoru, co má s danou vlastností objektu udělat. Pro názornost je uveden příklad.

```
[XmlRoot("Program")]
public class CLIProgram : Params.ICLIBaseParamList
{
    [XmlIgnore]
    [Browsable(false)]
    public string SourceFileName { get { return sourceFileName; } }

    [XmlIgnore]
    [Browsable(false)]
    public Params.ICLIBaseParamList Owner { get { return owner; } }

    [XmlAttribute("name")]
    public string Name { get; set; }

    [XmlAttribute("category")]
    public string Category { get; set; }

    [XmlAttribute("exeFile")]
    public string ExeFile
    {
     .
     .
     .
```

Tato aplikace tak využívá jeden z více možných způsobů serializace pomocí prostředí .NET Frameworku.

2.4.3. Propojení GUI s objekty XML serializace

Pro zobrazení serializovaného objektu uživateli slouží kolekce grafických prvků, ke kterým jsou za pomocí třídy `ParamControlArranger` jednosměrně asociovány objekty z kolekce třídy `CLIProgram` (obr. 8.). Grafické prvky se pak v interakci s uživatelem automaticky starají o zobrazení a nastavení vlastností serializovaného objektu.



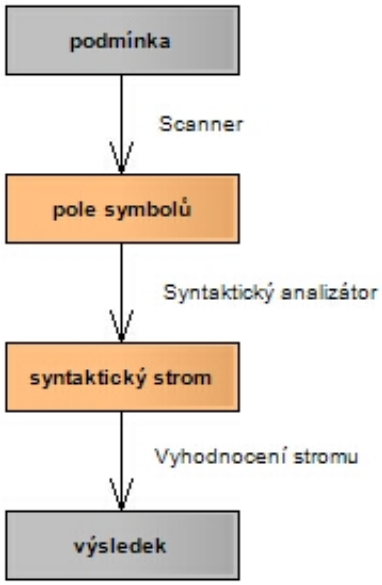
Obrázek 8. Propojení GUI s objekty XML serializace

2.4.4. Podmínkový analyzátor

Jednou z vlastností třídy `CLIBaseParam` je vlastnost EnabledCondition, do které je možno zadat za jaké podmínky bude ovládací prvek přístupný uživateli. Za účelem vyhodnocení podmínky existuje v aplikaci *Podmínkový analyzátor*. Vstupem analyzátoru je tedy podmínka, jejímž vyhodnocením je výsledek typu *true, false*.

Stručný popis funkce

Funkci *Podmínkového analyzátoru* znázorňuje obrázek 9..



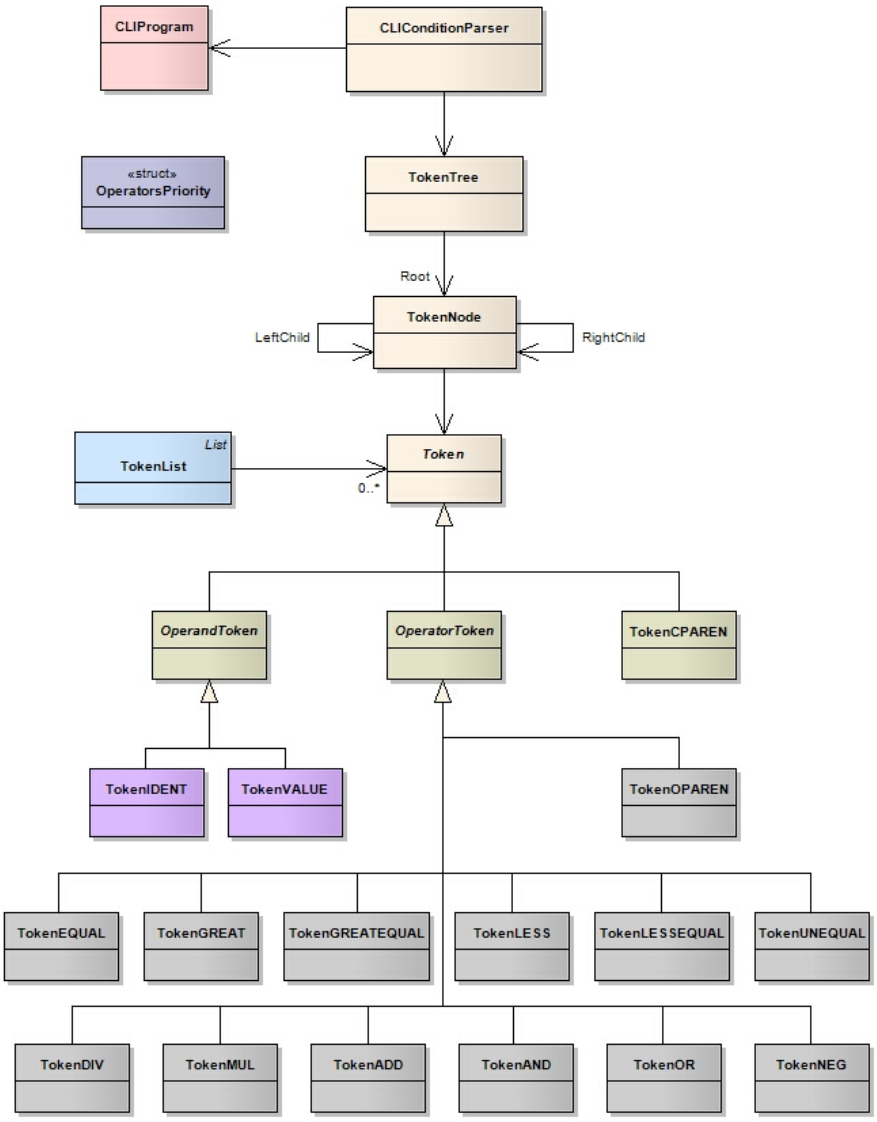
Obrázek 9. Schema podmínkového analyzátor

Scaner má za úkol ze vstupní podmínky vytvořit pole symbolů. Jako první podmínku rozdělí na části (číslo, text, operátor, ...) a poté vytvoří skenováním pole symbolů reprezentující části vstupní podmínky. Jednotlivé symboly v poli jsou přitom seřazeny tak, jak byly zapsány ve vstupní podmínce.

Syntaktický analyzátor naskenované pole symbolů převede na postfixovou notaci viz. [6] a poté z něho sestaví syntaktický strom. Při převodu se detekují syntaktické chyby, které vedou k vyvolání vyjímky. Ta je pak vhodným způsobem v aplikaci odchycena. Takto vytvořený syntaktický strom je následně vyhodnocen.

Syntaktický strom je struktura reprezentující kód, ve které jsou hodnoty listy a operace uzly stromu. Struktura stromu určuje, v jakém pořadí mají být jednotlivé operace vykonány.

Na obrázku 10. jsou vidět třídy podmínkového analyzátoru. Hlavní třídou je třída `CLIConditionParser`, která ostatní třídy analyzátoru využívá. Pro práci s analyzátorem slouží jediná statická metoda bool Evaluate(CLIProgram program, string condition), která zajistí zpracování a vyhodnocení výrazu a výsledek vrátí jako svoji návratovou hodnotu. Odkaz na *Program* v parametru program slouží analyzátoru k nalezení identifikátorů, které mohou být uvedené ve výrazu vstupní podmínky. Identifikátor může mít tvar jmenoParametru.Vlastnost.



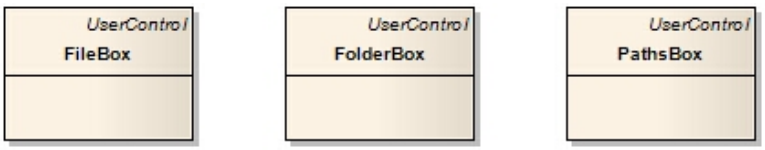
Obrázek 10. Třídy podmínkového analyzátoru

Pro tvorbu *Podmínkového analyzátoru* byly čerpány informace ze zdrojů [3], [4] a [6].

2.4.5. Pomocné uživatelské komponenty

Pro potřeby aplikace byly vytvořeny další nezávislé uživatelské komponenty (obr. 11.).

- Komponenta **FileBox** - slouží pro zadání souboru.
- Komponenta **FolderBox** - slouží pro zadání složky.
- Komponenta **PathsBox** - slouží pro zadání více diskových cest vzájemně oddělených středníkem.



Obrázek 11. Pomocné uživatelské komponenty

Tyto komponenty jsou použity v některých ovládacích prvcích *Parametrů* a v okně pro nastavení aplikace.

2.5. Použité technologie a jejich výběr

Pro vývoj aplikace bylo zvoleno vývojové prostředí Microsoft Visual Studio 2010 s prostředím .NET Framework 4 a knihovnou Windows Forms. Implementačním jazykem byl zvolen jazyk C#. Výběr této technologie byl jedním z požadavků zadání. Osobně jsem byl s tímto požadavkem spokojen, protože tato technologie se řadí k nejmodernějším a nejvyspělejším technologiím vývoje aplikací této doby. Skýtá velkou podporu při tvorbě aplikací a hlavně v této práci poskytuje potřebné nástroje pro serializaci objektů.

3. UŽIVATELSKÁ ČÁST

3.1. Úvod

3.1.1. O aplikaci

Aplikace slouží pro spouštění řádkových utilit v grafickém režimu. Řádkovou utilitu, která se běžně spouští v příkazovém řádku, zde zastupuje šablona řádkové utility. Šablona řádkové utility je XML soubor, se kterým tato aplikace umí pracovat a na základě jeho obsahu uživateli vygeneruje na obrazovku kolekci parametrů v podobě různých zaškrkávacích, výběrových a editačních polí. Aplikace umožňuje i správu takto vytvořených šablon v podobě *Oblíbených položek*, do kterých si uživatel může svoji nakonfigurovanou šablonu uložit a poté kdykoliv znovu vyvolat. Aplikace umožňuje dvojím způsobem spuštění utility. První je možnost spustit utilitu ve vestavěné konzoli. Druhá možnost je spustit utilitu ve standartní konzoli operačního systému. Vestavěná konzole umožňuje uživateli spustit více instancí dané utility zároveň a mít přehled o jejich průběhu. Je také možné vytvořit několik konfigurací dané utility a s těch pak vybírat. Jednotlivé konfigurace je také možné exportovat do dávkového souboru, nebo importovat z dávkového souboru. Dále v tomto textu budeme záložce se šablonou řádkové utility říkat *Program* a jejím jednotlivým položkám *Parametry* programu.

3.1.2. Požadavky na systém

- `Podporované operační systémy:` Windows 7; Windows Server 2003 Service Pack 2; Windows Server 2008; Windows Server 2008 R2; Windows Vista Service Pack 1; Windows XP Service Pack 3
- `Běhové prostředí:` Microsoft .NET Framework 4
- `Procesor:` Pentium 1 GHz nebo vyšší
- `RAM:` 512MB nebo více
- `Pevný disk:` 2MB volného místa nebo více, při současné instalaci balíčku Microsoft .NET Framework 4 až 850MB volného místa

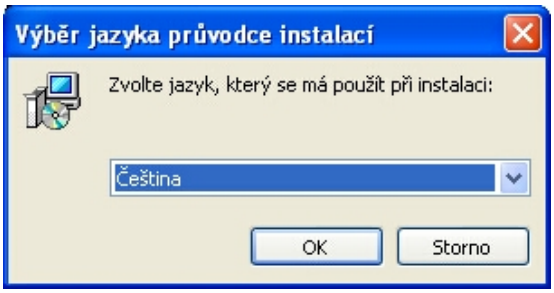
3.2. Instalace aplikace

3.2.1. Před instalací aplikace

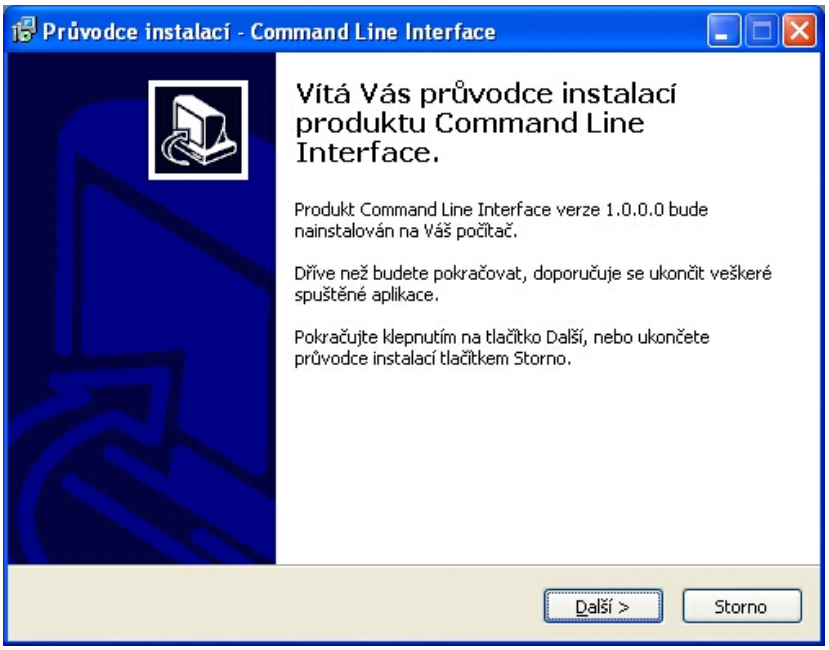
Před začátkem instalace je nutné ověřit, zda je v systému nainstalován Microsoft .Net Framework 4. V případě nepřítomnosti Microsoft .Net Framework 4. instalátor upozorní na tuto skutečnost a instalace bude přerušena.

3.2.2. Instalace aplikace

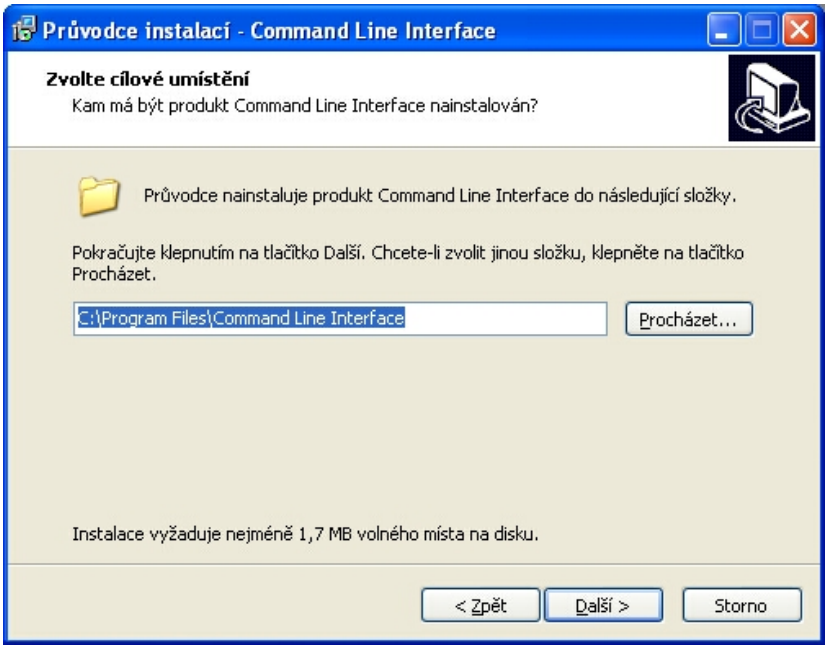
1. Spusťte z instalačního CD instalační program `CLIsetup.exe`.
2. Zvolte jazyk instalace a klikněte myší na tlačítko `OK`. Instalátor spustí instalační program.



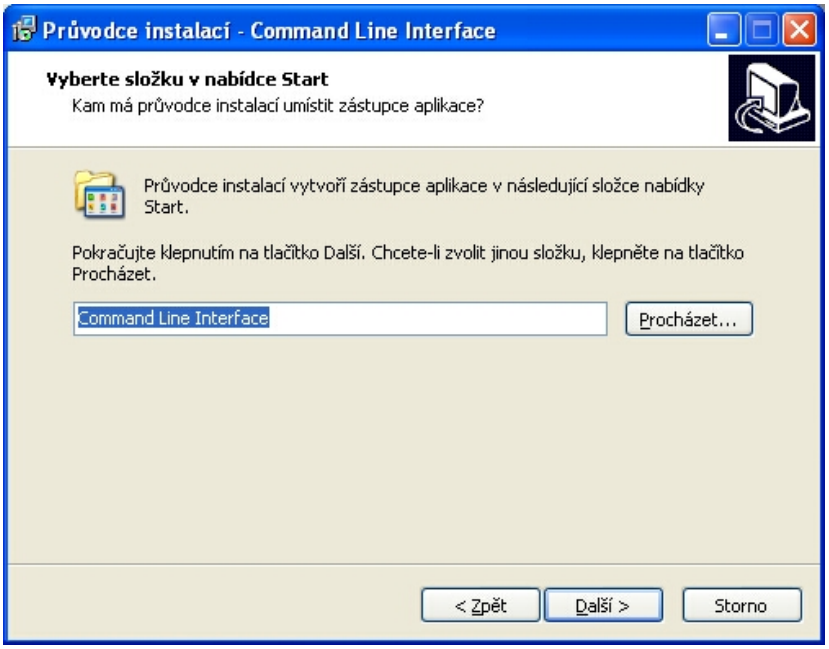
3. Postupujte podle pokynů na obrazovce a klikněte myší na tlačítko `Další`.



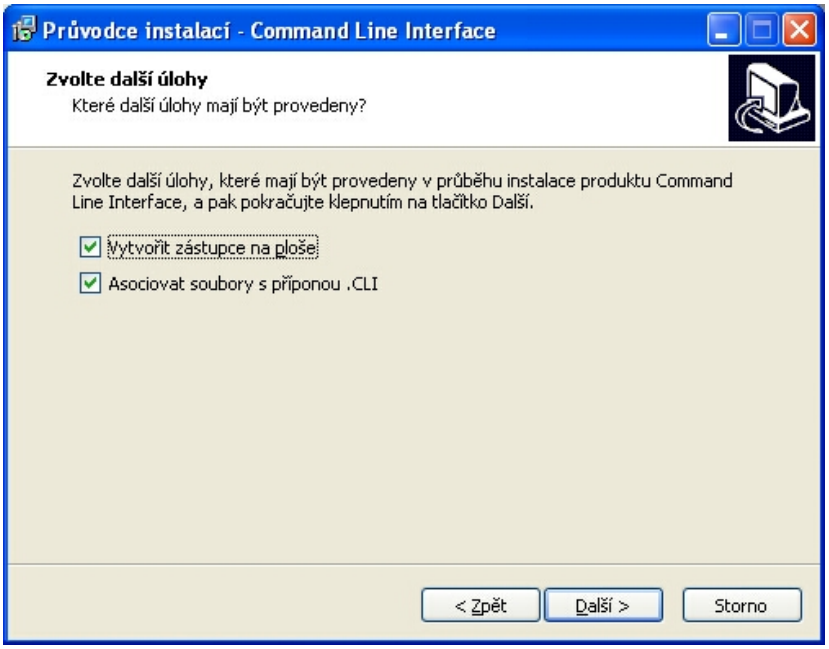
4. Postupujte podle pokynů na obrazovce a klikněte myší na tlačítko `Další`.



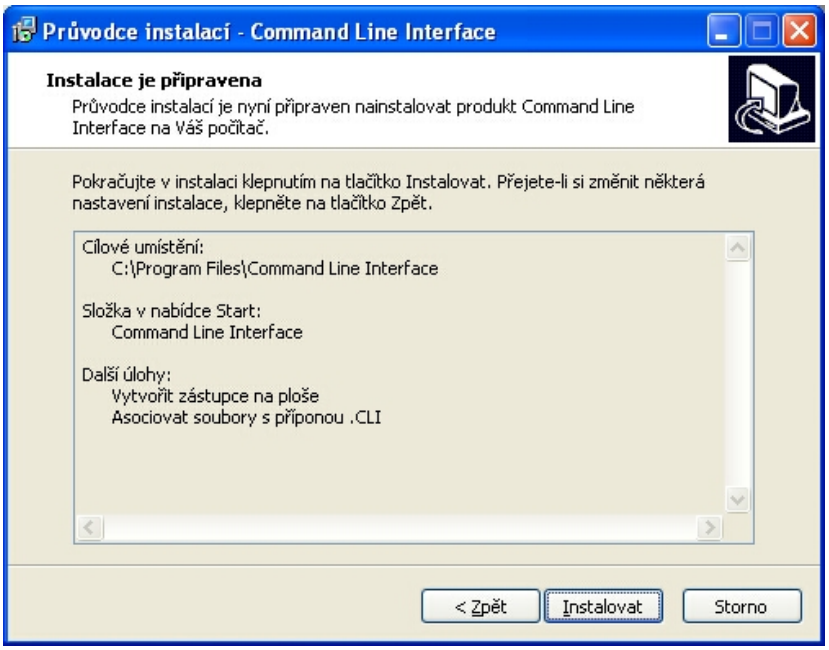
5. Postupujte podle pokynů na obrazovce a klikněte myší na tlačítko `Další`.



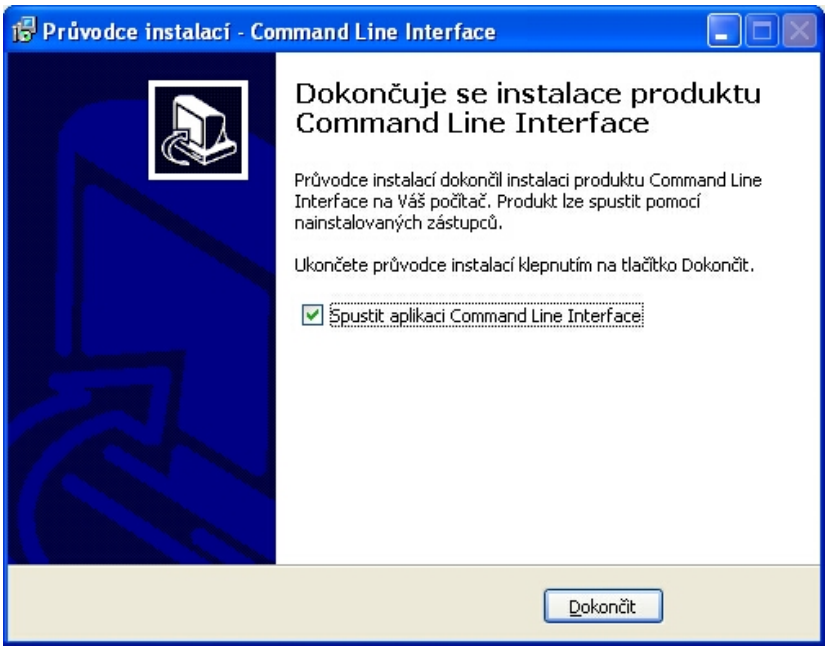
6. Postupujte podle pokynů na obrazovce a klikněte myší na tlačítko `Další`.



7. Postupujte podle pokynů na obrazovce a klikněte myší na tlačítko `Instalovat`.



8. Postupujte podle pokynů na obrazovce a klikněte myší na tlačítko `Dokončit`.



3.2.3. Odinstalování aplikace

1. Zvolte možnost `Start > Programy > Command Line Interface > Odinstalovat aplikaci Command Line Interface`.
2. Postupujte podle pokynů na obrazovce, dokud nebudou soubory aplikace odebrány.

nebo

1. Zvolte možnost `Start > Nastavení > Ovládací panely`.
2. Klikněte na ikonu `Přidat nebo odebrat programy`.
3. V seznamu nainstalovaných programů klikněte na položku `Command Line Interface`.
4. Klikněte myší na tlačítko `Odebrat`.
5. Postupujte podle pokynů na obrazovce, dokud nebudou soubory programu odebrány.

3.3. Obsluha aplikace

3.3.1. Spuštění aplikace

Aplikaci `Command Line Interface` je možné spustit několika způsoby:

- z nabídky start,
- z plochy (pokud tato možnost byla zvolena při instalaci),
- poklepání na soubor šablony *Programu* s příponou `.CLI` (pokud tato možnost byla zvolena při instalaci),
- z příkazového řádku viz. kapitola 3.3.2..

3.3.2. Spuštění aplikace z příkazového řádku

Aplikaci lze spustit z příkazového řádku pomocí parametrů zadaných za jméno spustitelného souboru aplikace. Parametry pro spuštění lze také zadat do zástupce aplikace a spouštět tak pohodlným způsobem různé konfigurace aplikace různě nakonfigurovanými zástupci aplikace.

Formát příkazu je:

```
CLI [-fav | -nofav] [-opn <program> [-opn <program>]...]

<program> = šablona [příkazový_řádek]
```

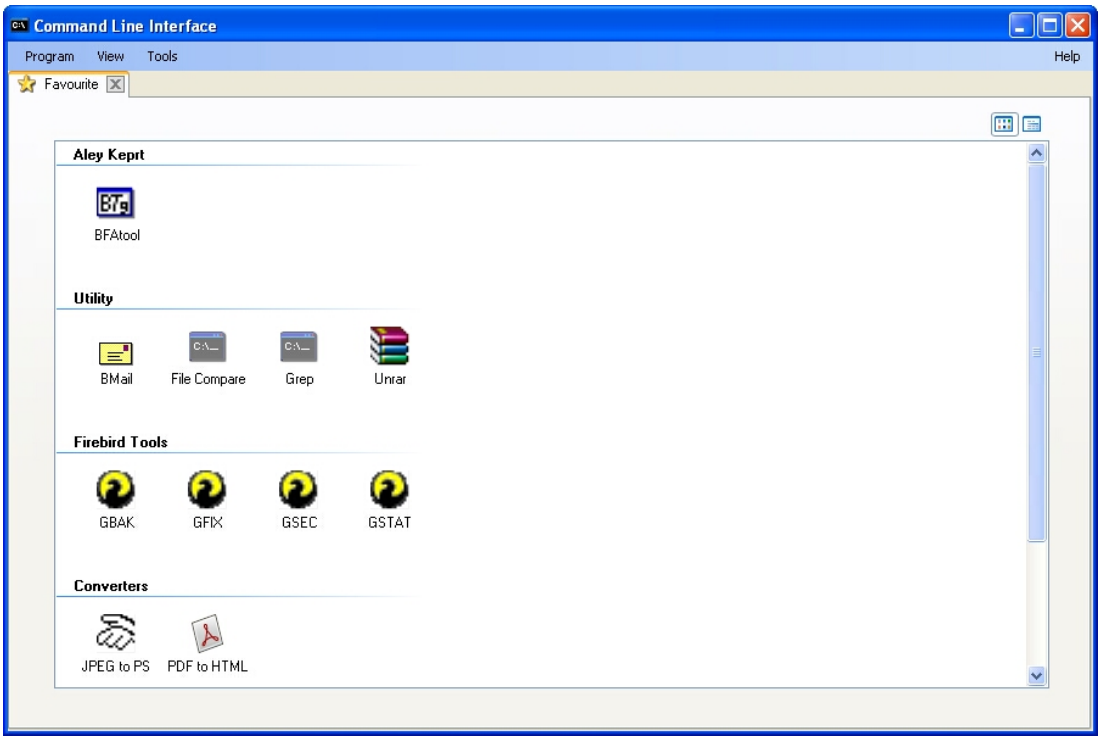
Přepínač	Popis
-fav	Zobrazí záložku s oblíbenými položkami
-nofav	Nezobrazí záložku s oblíbenými položkami
-opn	Vytvoří záložku dle `šablona` a nastaví pole s parametry na hodnoty dle `příkazový_řádek`
šablona	Jméno souboru šablony. Pokud není uvedena přípona, tak je jméno doplněno příponou `.CLI`. Pokud není uvedena kompletní cesta k souboru šablony, tak je nejprve vyhledána ve složce oblíbených položek a poté ve složce šablon viz. nastavení aplikace, kapitola 3.3.11.
příkazový_řádek	Řetězec ve formátu příkazového řádku pro program dané šablony

Příklad:

```
CLI -nofav -opn Ping ping.exe -a www.centrum.cz
```

3.3.3. Hlavní okno aplikace

Po prvním spuštění aplikace se zobrazí hlavní okno (obr. 12.) ve výchozím nastavení s jedinou záložkou *Oblíbených položek* (*Favourite*), na které jsou již z výroby předchystány nějaké položky.



Obrázek 12. Hlavní okno aplikace

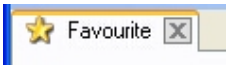
- Jednotlivé oblíbené položky na záložce lze přejmenovat nebo smazat vyvoláním kontextové nabídky u dané položky (obr. 13.).

Obrázek 13. Smazání oblíbené položky

- Změnu velikosti ikon na záložce oblíbených položek lze měnit dvojicí tlačítek v pravém horním rohu (obr. 14.).

Obrázek 14. Změna velikosti ikon

- Jakoukoliv záložku je možné zavřít stiskem tlačítka v pravé části ouška záložky (obr. 15.). Záložku s oblíbenými položkami je možno znovu vyvolat příkazem z hlavního menu (obr. 16.).



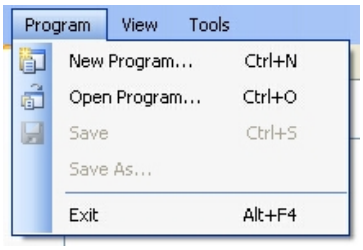
Obrázek 15. Tlačítko pro uzavření záložky

Obrázek 16. Hlavní menu - položka *View*

Po spuštění aplikace je možné vybrat *Program* několika způsoby:

- ze záložky *Oblíbených položek*
- vyvoláním dialogu pro přidání nového *Programu* (obr. 18.) z hlavního menu `Program > New Program...`
- vyvoláním systémového dialogu pro výběr souboru z hlavního menu `Program > Open Program...` a otevřít tak *Program* z libovolného umístění na disku

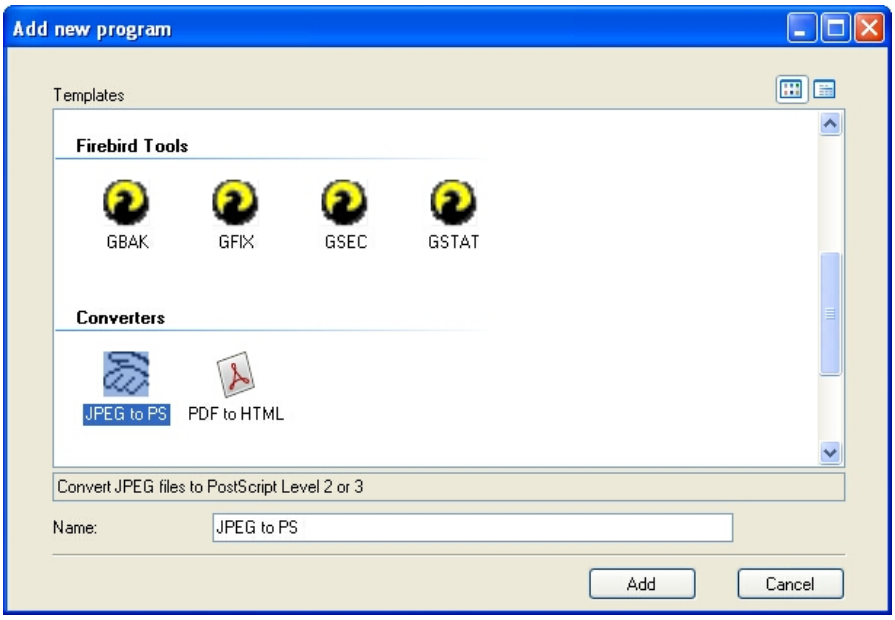
Možnosti výběru z nabídky jsou vidět na obrázku 17.. Po úspěšném výběru bude přidána nová záložka s *Programem* (obr. 19.).



Obrázek 17. Hlavní menu - položka *Program*

3.3.4. Dialog přidání nového programu

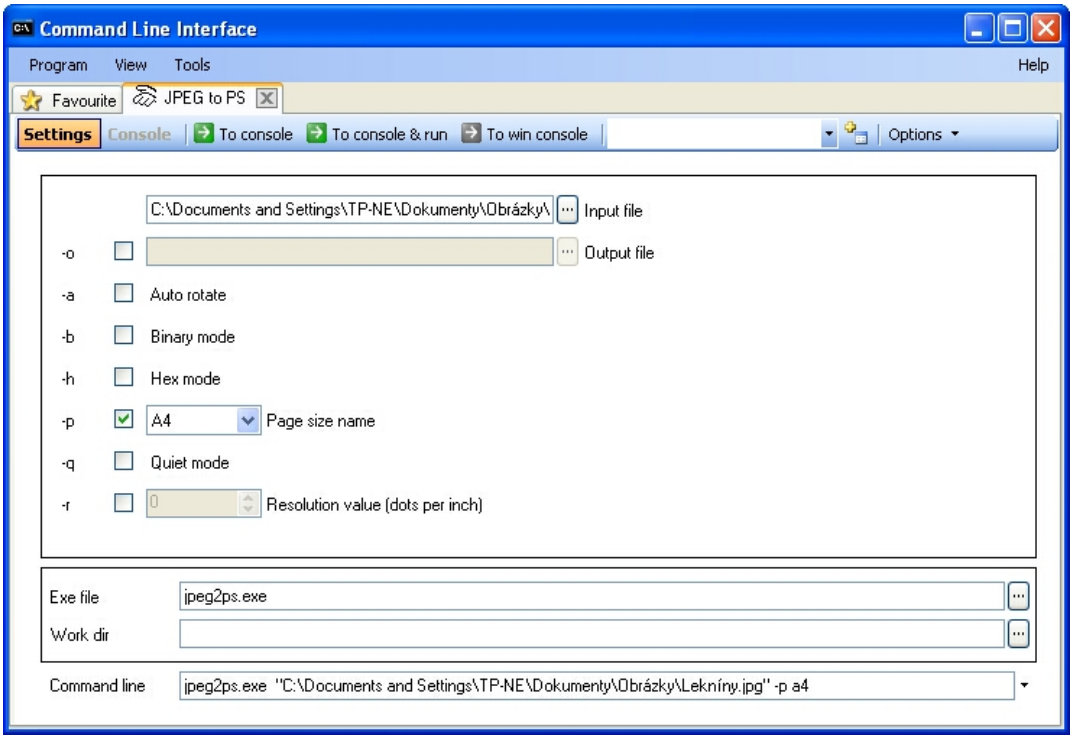
V dialogu přidání nového programu je možné vybrat jednu z předdefinovaných šablon programů a pojmenovat ji v poli *Name*. Pro úspěšné přidání je potřeba potvrdit akci stiskem tlačítka *Add*, nebo akci stornovat tlačítkem *Cancel*.



Obrázek 18. Dialog přidání nového programu

3.3.5. Karta *Programu*

Na záložce *Programu* (obr. 19.) jsou systémem vygenerovány všechny ovládací prvky *Parametrů Programu* tak, jak jsou definovány v šabloně *Programu*. Jednotlivé *Parametry* je možno dle potřeb nastavit a výsledný příkazový řádek je pak zobrazován ve spodní části karty *Programu*.

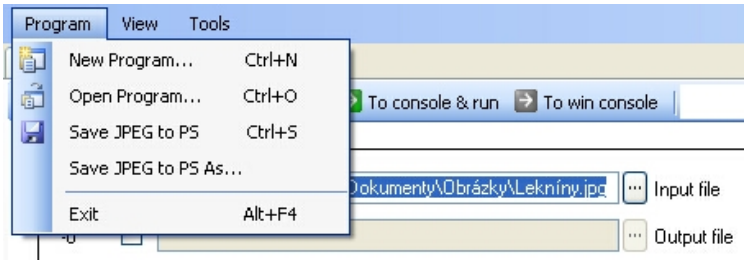


Obrázek 19. Karta *Programu*

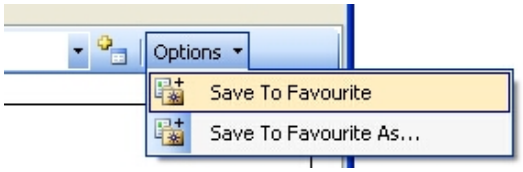
3.3.6. Uložení *Programu*

Uložení nastavení jednotlivých *Parametrů Programu* je možné provést volbou z hlavního menu `Program > Save` (obr. 20.). Uložení do jiného umístění je možné provést volbou z hlavního menu `Program > Save As...`.

Uložení do *Oblíbených položek* je možno provést volbou na kartě *Programu* `Option > Save To Favourite`, nebo volbou `Option > Save To Favourite As` (obr. 21.)



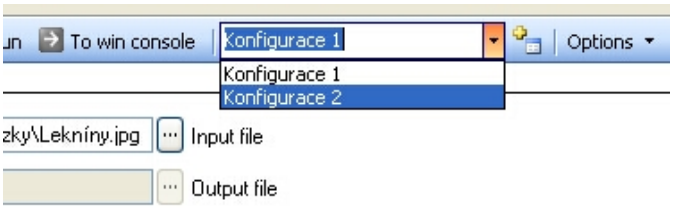
Obrázek 20. Uložení *Programu*



Obrázek 21. Uložení *Programu* do *Oblíbených položek*

3.3.7. Uložení konfigurace *Programu*

Na kartě *Programu* je možné uložit více *konfigurací Parametrů* a z těch pak pohodlným způsobem vybírat (obr. 22.). Pro uložení nové konfigurace je potřeba vyplnit nové jméno konfigurace do výběrového boxu konfigurací a následně provést uložení nové konfigurace stiskem tlačítka  napravo od výběrového boxu.



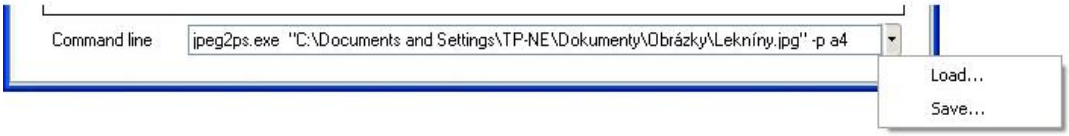
Obrázek 22. Uložení konfigurace *Programu*

Aby nově pořízené *konfigurace* byly k dispozici i po znovu spuštění aplikace, je potřeba na závěr provést i uložení celého *Programu* volbou z hlavního menu `Program > Save`.

3.3.8. Příkazový řádek na kartě *Programu*

Příkazový řádek je možné uložit (`Save...`) do dávkového souboru, nebo načíst (`Load...`) z dávkového souboru pomocí menu, které je umístěno na pravé straně příkazového řádku (obr. 23.). Příkazový řádek je možno editovat i přímo. Pokud systém nedokáže příkazový řádek úspěšně rozložit, bude v levé části editačního pole zobrazena ikona signalizující chybu s plovoucí nápovědou popisující tuto chybu (obr. 24.), v opačném případě jsou automaticky změněny hodnoty v ovládacích prvcích *Parametrů* dle zadaní v příkazovém řádku. Chyba bude zobrazena i v případě chybného načtení příkazového řádku z dávkového souboru.

Z dávkového souboru je možné tímto způsobem načíst pouze první řádek souboru. Při ukládání příkazového řádku do existujícího souboru je uživatel dotazován zda má být soubor přepsán, nebo zda má být příkazový řádek uložen na konec souboru.

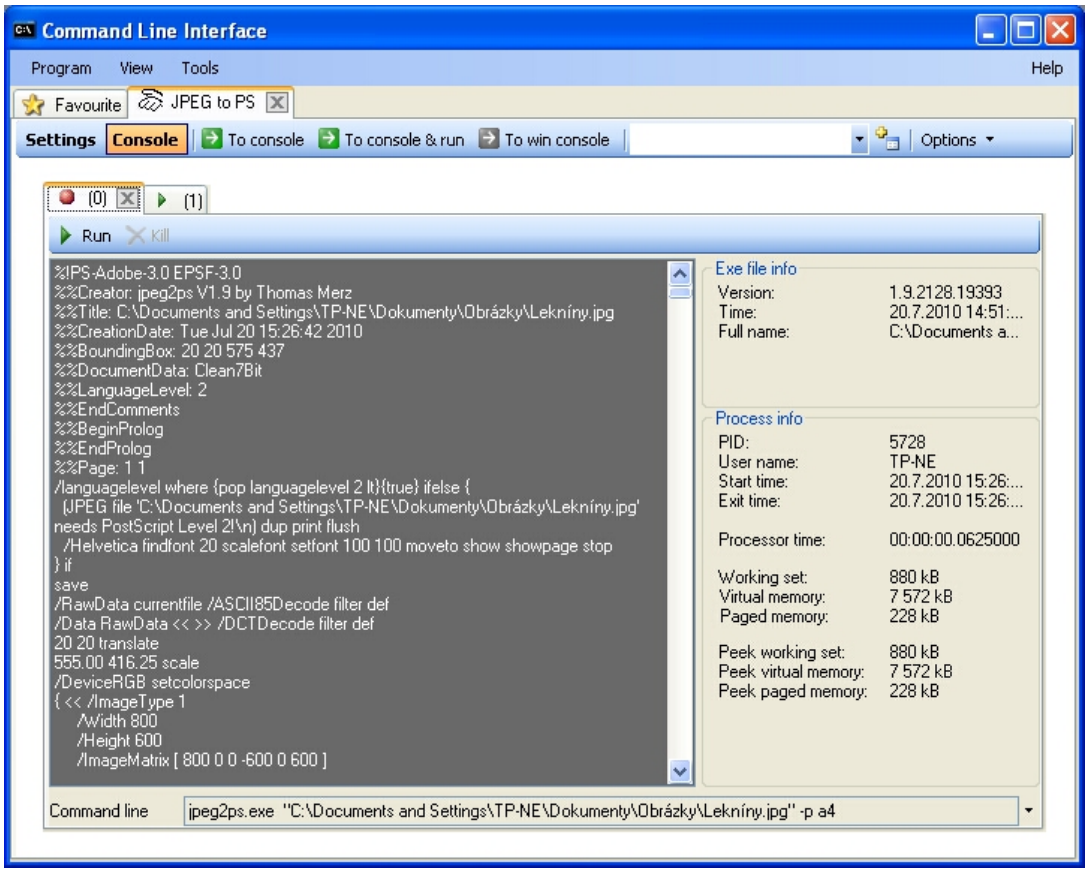


Obrázek 23. Menu příkazového řádku

Obrázek 24. Chyba na příkazovém řádku

3.3.9. Vestavěná konzole

Vestavěná konzole aplikace (obr. 25.) umožňuje uživateli pohodlnou správu nad běžícími procesy spuštěnými dle dané konfigurace na kartě *Programu*. Novou kartu konzole lze vytvořit příkazem **To console** na nástrojové liště *Programu* a příkazem **To console & run** lze konzoli i zároveň spustit.



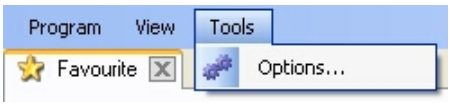
Obrázek 25. Vestavěná konzole *programu*

3.3.10. Systémová konzole

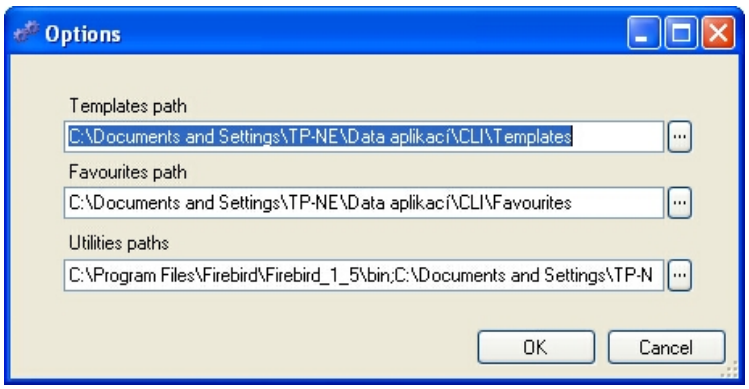
Proces v systémové konzoli operačního systému Windows lze spustit příkazem **To win console** na nástrojové liště *Programu*. Tato možnost je určena uživatelům, kteří vyžadují možnosti systémové konzole jako jsou například speciální funkce `Ctrl+C`, `Ctrl+Break` nebo barevný výstup konzole.

3.3.11. Nastavení aplikace

Okno *Options* s nastavením aplikace (obr. 27.) lze vyvolat volbou v hlavním menu `Tools > Options` (obr. 26.). V tomto okně *Options* je možno nastavit cesty k různým složkám, které aplikace využívá pro svůj chod.



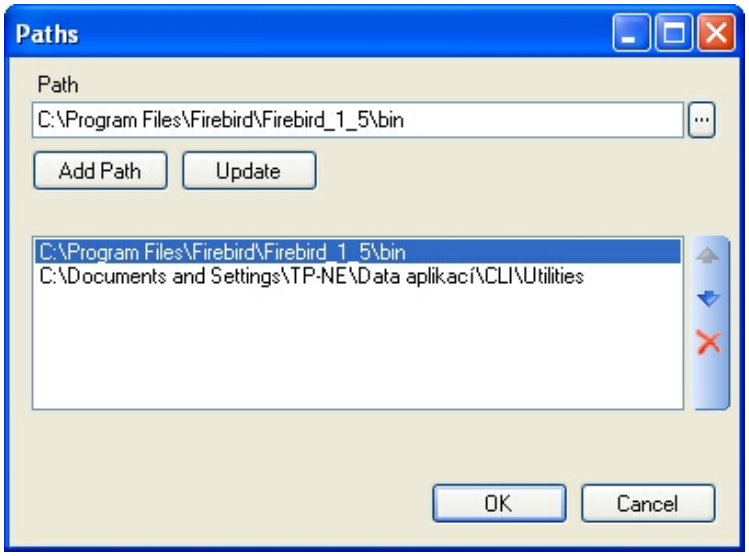
Obrázek 26. Hlavní menu - položka *Tools*



Obrázek 27. Nastavení aplikace

- **Templates path** - cesta ke složce se šablonami *Programů*
- **Favourites path** - cesta ke složce s oblíbenými položkami *Programů*
- **Utilities paths** - cesty ke složkám se spustitelnými soubory utilit. Jednotlivé cesty musí být odděleny středníkem.

Do položky **Utilities paths** je možno pomocí dialogu *Paths* (obr. 28.) nastavit všechny cesty ke složkám, v nichž se mají dané utility vyhledávat. Jednotlivé cesty se v daném pořadí při spuštění aplikace nastaví do vyhledávacích cest prostředí aplikace. Z tohoto důvodu je potřeba pro akceptaci změny v **Utilities paths** aplikaci znovu spustit.



Obrázek 28. Nastavení cest ke složkám

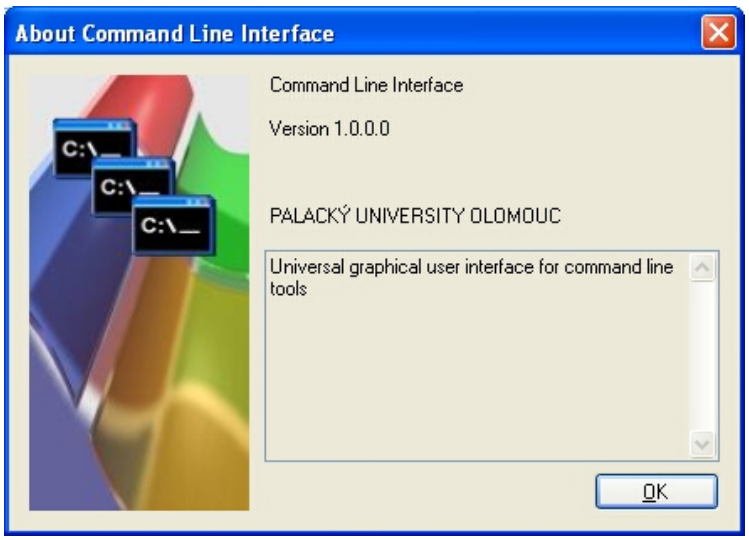
3.3.12. Vyvolání nápovědy k programu

Nápovědu k aplikaci lze vyvolat volbou z hlavního menu `Help > View Help`, nebo horkou klávesou `Ctrl+F1`.

3.3.13. Zobrazení okna *O Aplikaci*

Okno *O aplikaci* lze vyvolat volbou v hlavním menu `Help > About Command Line Interface` (obr. 29.). Poté se zobrazí okno *About Command Line Interface* (obr. 30.) s informacemi o aplikaci a její verzi.

Obrázek 29. Hlavní menu - položka *Help*



Obrázek 30. Okno *O Aplikaci*

4. POJMY A ZKRATKY

Program Objekt popisující konzolovou aplikaci (program pracující na příkazovém řádku), obsahuje hlavičku s informacemi o programu a seznam parametrů programu.

Parametr Objekt popisující jeden parametr *Programu*, objekt vychází z jedné bázové třídy, z níž jsou odvozeny různé modifikace. Každý typ parametru má pak svou vizuální podobu, která je dostupná na kartě *Programu*.

Karta programu Objekt patřící jednomu *Programu*. Na kartě jsou umístěny všechny informační a ovládací prvky daného *Programu*. Tato karta je vizuální prvek v aplikaci.

Karta konzole Objekt patřící jednomu *Programu*. Na kartě jsou umístěny všechny informační a ovládací prvky konzole *Programu*. Tato karta je vizuální prvek v aplikaci.

Oblíbené (položky) Seznam *Programů*, které byly uloženy jako oblíbená položka. Uživatel může z tohoto seznamu znovu *Program* vyvolat. Seznam je zobrazen na zvláštní kartě označené jako *Favourite*. Tato karta je dostupná z hlavního okna aplikace.

Podmínkový analyzátor Třída, jenž umí vyhodnotit podmínku, která může být složena z logických či matematických výrazů. Výsledkem je vždy logická hodnota (true, false). Slouží pro práci s *Parametry*, které se navzájem dle podmínky mohou ovlivňovat (povolit / zakázat).

Dávkový soubor Dávkový soubor je neformátovaný textový soubor, který obsahuje jeden nebo více příkazů a jeho název má příponu .bat nebo .cmd

GUI Grafické uživatelské rozhraní (z anglického Graphical User Interface)

XML Rozšiřitelný značkovací jazyk (z anglického Extensible Markup Language), viz. [5]

Serializace Uložení instance do nějakého perzistentního úložiště (relační databáze, soubor . . .).

5. ZÁVĚR

Přestože zadání aplikace se jeví jako poměrně jednoduché, bylo na jeho zpracování vyvinuto nemalé úsilí. Cílem bylo použít nejnovějších technologií firmy Microsoft a to zejména v serializaci objektů do XML. Zdrojové kódy, které mohou působit nenáročně (což bylo i cílem) však skrývají mnoho práce v podobě studování prostředí .NET Framework a psaní různých testovacích projektů, které měly za cíl ověřit funkce různých částí aplikace. Výsledkem je tak modulární jednoduše spravovatelná aplikace. Věnované úsilí této práci považuji za užitečné, protože mi pomohla prohloubit znalosti práce s MS Visual Studiem, jazykem C# a s prostředím .NET Framework, zejména co se týká serializace a použití různých komponent pro uživatelské rozhraní.

Reference

[1] Michal Václavský řešitel druhé části této práce

[2] Webový portál http://msdn.microsoft.com

[3] Aleš Šturala : DLR - Scanner, Webový portál http://www.vyvojar.cz, Vydáno 9.9.2008

[4] Aleš Šturala : DLR - Parser, Webový portál http://www.vyvojar.cz, Vydáno 24.9.2008

[5] Wikipedie : Extensible Markup Language
http://cs.wikipedia.org/wiki/extensible_markup_language

[6] Wikipedie : Postfixová notace
http://cs.wikipedia.org/wiki/postfixov%c3%a1_notace

A. Dodané šablony programů

Pro otestování aplikace bylo nutné vytvořit reálné šablony programů, které jsou součástí instalace. Následující seznam popisuje šablony programů, které byly vytvořeny jako součást této práce. Ostatní šablony programů, které nejsou v tomto seznamu a jsou součástí instalace byly vytvořeny [1].

GBAK Utilita pro zálohování a obnovení databází FireBird

GFIX Utilita pro údržbu databáze FireBird

GSEC Utilita pro správa bezpečnostní databáze FireBird

GSTAT Utilita pro výpis statistiky databáze FireBird

JPG to PS Utilita pro konverzi obrázků z formátu JPG do formátu EPS (spustitelný soubor je součástí instalace této aplikace)

Ping v4 Utilita pro zjištění odezvy v počítačové síti (spustitelný soubor je součástí operačního systému)

Pro zdárné spuštění utility je zapotřebí mít v počítači nainstalovaný patřičný spustitelný soubor dané utility. Například pro používání utilit databázového systému FireBird, je zapotřebí mít nainstalován tento databázový systém a mít správně nastavené cesty k jeho utilitám, které jsou součástí instalace tohoto databázového systému. Každý uživatel může používat jinou verzi tohoto databázového systému a tak nelze tyto utility paušálně šířit v instalaci této aplikace.

B. Struktura přiloženého CD

Přiložené CD obsahuje tyto adresáře (složky):

bin aplikaci připravenou ve spustitelné podobě

src zdrojové kódy aplikace

doc všechny dokumenty, které byly použity pro tvorbu této aplikace

install instalační balíček programu (podrobnosti k instalaci jsou uvedeny v uživatelské části této dokumentace)