Zápočtová úloha z předmětu KIV/ZSWI

OBJEKTOVÝ NÁVRH APLIKACE

pro program na vytváření lentikulárních obrázků

22. 3. 2015

Tým: Lentilky

Členové:

Lukáš Hruda	hrudalu@students.zcu.cz
Štěpán Baratta	BarattaStepan@gmail.com
Jan Albl	alblj@students.zcu.cz
Tomáš Matějka	matejka@students.zcu.cz

1. Úvod

Tento dokument popisuje návrh aplikace pro vytváření lentikulárních obrázků.
Nejdříve bude popsán kontext systému a jeho architektura, poté bude následnovat celkový návrh systému a podrobný diagram tříd s jejich popisem.

1.1 Účel systému

Program umožní načíst několik obrázků v běžně používaných formátech a provést proces jejich proložení podle zadaných parametrů a finální převzorkování pro tiskovou velikost. Umožní uložení výstupního obrázku do libovolného z běžně používaných formátů do libovolného adresáře pod libovolným názvem. Formát, umístění a název výstupního souboru budou nastavitelné.
Umožní zadání všech parametrů potřebných k proložení obrázků – šírka a výška výstupního obrázku v cm, mm a palcích, DPI (reps. DPCM) pro výpočet velikosti v pixelech, počet lentikulí na palec (resp. centimetr) použité fólie, interpolační algoritmus použitý při změně velikosti pro obě fáze prokládání (1. převedení všech obrázkůna velikost podle počtu lentikulí, 2. převedení výsledného obrázku na tiskovou velikost). Umožní přidání pasovacích značek volitelné šířky a barvy s volitelným odsazením od obrázku. V rámci uživatelského rozhraní umožní snadnou orientaci v načtených obrázcích a umožní určit pořadí, ve kterém budou vstupní obrázky zpracovány.

1.2 Slovníček definic, pojmů a zkratek

Lentikulární obrázek – Obrázek, který společně s optickou deskou tvoří iluzní představu o měnicím se nebo přesouvajícím se obrázku v závislosti na úhlu pohledu.

Běžně používané formáty – Za běžné používané formáty obrázků předpokládáme obrázky s koncovkou **png**, **tif**, **jpg**, **gif** a **bmp**

DPI - Dots per inch je údaj určující, kolik obrazových bodů (pixelů) se vejde do délky jednoho palce (2,54cm).

Lentikule – jeden sloupeček optické desky, pod kterým je vždy jeden sloupeček ze včech vstupních obrázků.

Interpolační algoritmus – metoda sloužící k nalezení přibližné barevné hodnoty neznámého pixelu z okolních známých pixelů, například při zvětšování nebo zmenšování obrázku.

Pasovací značka - Pomocné čáry, které pomáhají při usazení optické desky na lentikulární obrázek.

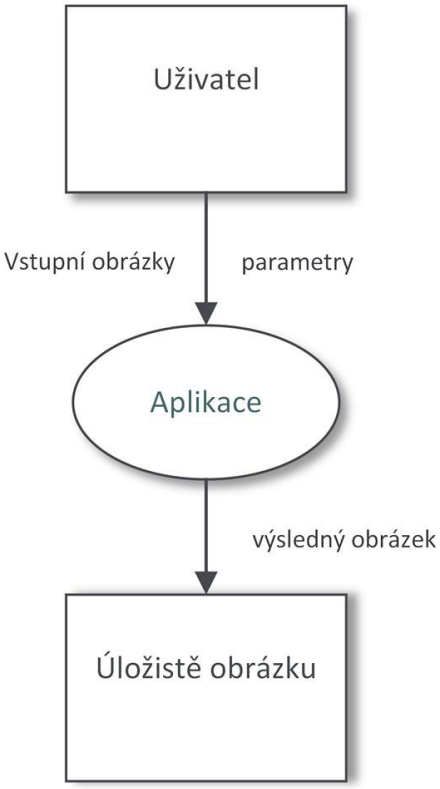
Bitmapová reprezentace obrázku - obrázek se rozdělí na jednotlivá políčka. Každé políčko má nějakou barvu a jas. V bitmapovém souboru jsou potom uloženy informace o každém z těchto políček.

DLL knihovna – implementace konceptu sdílených knihoven společnosti Microsoft pro operační systém Microsoft Windows

2. Kontext a architektura systému

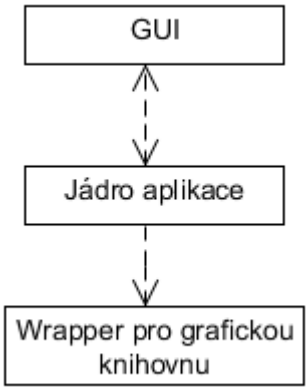
2.1 Kontext systému

Jde o nově vyvíjený program jehož výstupní obrazová data budou určena pro tisková zařízení s rozlišením 300 až 6000 DPI. Vstup i výstup obrazových dat je předpokládán v RGB.



2.2 Architektura systému, přehled podsystémů

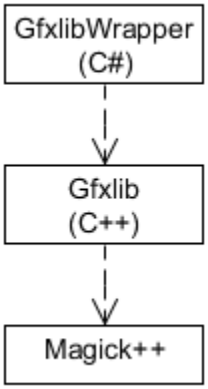
Celý systém bude rozdělen do tří vrstev, viz následující diagram.



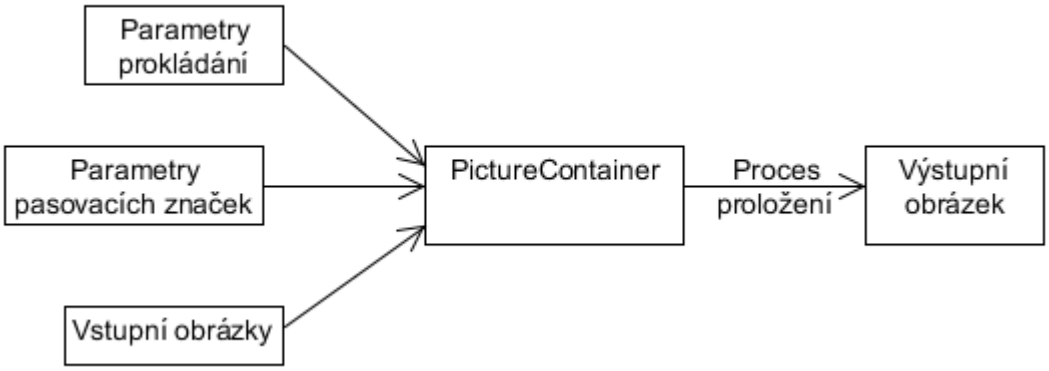
Jádro aplikace používá wrapper pro grafickou k načítání obrázků a převedení do bitmapové reprezentace a pro potřebné transformace, především změnu velikosti.

Mezi GUI (grafickým uživatelským rozhraním) a jádrem aplikace je obousměrná komunikace, protože některé informace zpracované uživatelským rozhraním je potřeba vyhodnotit jádrem aplikace a předat GUI výsledek k zobrazení.

Wrapper pro grafickou knihovnu:



- Magick++ je C++ wrapper pro grafickou knihovnu ImageMagick, která bude použita pro práci s obrázky – načítání, ukládání, přístup k pixelům, transformace.
- Gfxlib je DLL knihovna, která slouží jako interpret funkcí knihovny Magick++ pro C#. Jsou v ní definovány funkce pro všechny operace, pro které budeme knihovnu Magick++ používat.
- GfxlibWrapper je objektový wrapper pro Gfxlib. Zabaluje funkčnost knihovny do několika tříd v C#

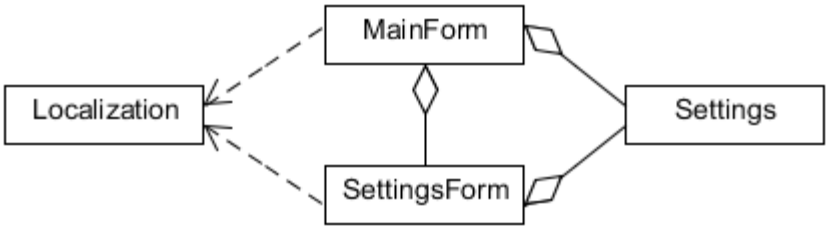


(podrobněji viz diagram tříd).

Jádro aplikace:

- PictureContainer je třída, která v konstruktoru přijme vstupní data, tedy parametry prokládání, parametry pasovacích značek a vstupní obrázky (více viz diagram tříd), a na jejich základě vytvoří obrázek výstupní.

GUI:



- Main form je třída hlavního formulář aplikace. Ten obsahuje komponenty pro nastavení paremetrů prokládání, nastavení parametrů pasovacích značek a systém pro výběr vstupních obrázků.
- SettingsForm je třída formluláře, který je použit pro zobrazení a upravení nastavení aplikace (jazyk, délkové jednotky, jednotky rozlišení). Otevírá se z hlavního formuláře.
- Localization je statická třída, která se stará o nastavení jazyka aplikace.
- Settings je třída, která obsahuje informace o aktuálním nastavení aplikace, které lze zobrazit a upravit v SettingFormu. MainForm obsahuje jednu instanci této třídy jako atribut a při vytvoření instnace SettingFormu mu tuto instanci předává v konstruktoru, stejná instance je pak tedy atributem i třídy SettingForm.

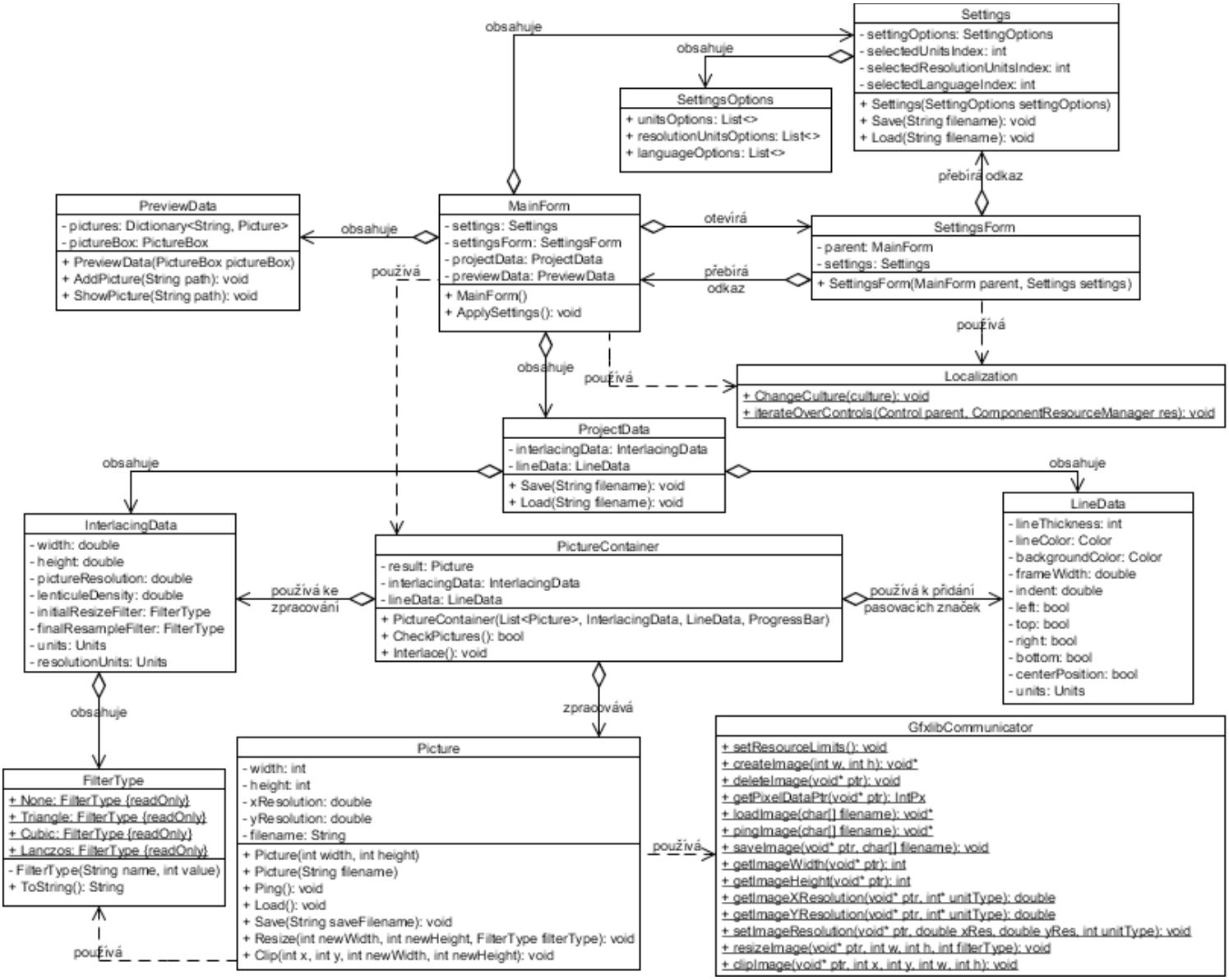
2.3 Zvolená technologie, programovací jazyk ad., důvody

Pro implementaci vrstev **GUI** a **Jádro aplikace** jsme zvolili programovací jazyk C# s použitím .NET frameworku. Důvodem je poměrně snadná tvorba uživatelského rozhraní a také doporučení zadavatelem.
Implementace vrstvy **Wrapper pro grafickou knihovnu** bude částečně v C# a částečně v C++, jak je vidět v diagramu výše. Část v C# je potřebná z důvodu snadného použití ve zbytku kódu. V C++ bude napsána pouze část komunikující přímo s knihovnou Magick++. C++ jakožto programovací jazyk pro tuto část je určen knihovnou samotnou.

3. Typy informací zpracovávané systémem

Vstupními i výstupními soubory jsou především obrázky v běžně používaných formátech – JPEG, bmp, png, tiff.
Dále bude aplikace umět ukládat projekty do souborů a znich projekty načítat. Navíc bude do konfiguračního souboru ukládat aktuální nastavení aplikace. Formát těchto souborů bude blíže specifikován při implementaci, pravděpodobně půjde o formát ve stylu co řádek, to hodnota, kde tyto hodnoty pak budou aplikací čteny ve předem specifikovaném pořadí a přiřazovány konkrétním atributům.

4. Návrh systému



4.1 Přehled tříd

4.1.1 Třída GfxlibCommunicator

Statická třída, která obstarává komunikaci s grafickou knihovnou. Obsahuje pouze naimportované funkce z Gfxlib.dll.

4.1.1.1 Konstruktory

Nejsou.

4.1.1.2 Metody

public static extern void setResourceLimits()

- nastaví limity pro cache knihovny Magick++

public static extern void* createImage(int w, int h)
- vytvoří obrázek v podobě třídy Magick::Image v C++ a vrátí pointer na tento objekt
- w – šířka v pixelech
- h – výška v pixelech
- return – pointer na nově vytvořenou instanci

public static extern void deleteImage(void* ptr)
- dealokuje instanci třídy Magick::Image
- ptr – pointer na instanci pro dealokaci

public static extern IntPx* getPixelDataPtr(void* ptr)
- získá pole pixelů z instance třídy Magick::Image
- ptr – pointer na instanci třídy Magick::Image
- return – pointer na začátek pole pixelů

public static extern void* loadImage(char[] filename)
- načte obrázek ze do instance třídy Magick::Image
- filename – název souboru
- return – instance třídy Magick::Image, do které byl obrázek načten

public static extern void* pingImage(char[] filename)
- načte obrázek bez jeho dekomprese, aby z něj bylo možné zjistit některé informace
- filename – název souboru
- return – instance třídy Magick::Image, do které byl obrázek načten

public static extern void saveImage(void* ptr, char[] filename)
- uloží obrázek do souboru
- ptr – pointer na instanci třídy Magick::Image, která obsahuje data obrázku
- filename – název souboru

public static extern int getImageWidth(void* ptr)
- vrátí šířku obrázku v pixelech
- ptr – pointer na instanci třídy Magick::Image, která obsahuje data obrázku
- return – šířka obrázku v pixelech

public static extern int getImageHeight(void* ptr)
- vrátí výšku obrázku v pixelech
- ptr – pointer na instanci třídy Magick::Image, která obsahuje data obrázku
- return – výška obrázku v pixelech

public static extern double getImageXResolution(void* ptr, int* unitType)
- vrátí obrazové rozlišení obrázku na ose X
- ptr – pointer na instanci třídy Magick::Image, která obsahuje data obrázku
- unitType – pointer na proměnnou, do které se uloží informace o jednotkách
- return – rozlišení obrázku na ose X

public static extern double getImageYResolution(void* ptr, int* unitType)
- vrátí obrazové rozlišení obrázku na ose Y
- ptr – pointer na instanci třídy Magick::Image, která obsahuje data obrázku
- unitType – pointer na proměnnou, do které se uloží informace o jednotkách
- return – rozlišení obrázku na ose Y

public static extern void setImageResolution(void* ptr, double xRes, double yRes, int unitType)
- nastaví obrázku jeho obrazové rozlišení
- ptr – pointer na instanci třídy Magick::Image, které má být rozlišení nastaveno
- xRes – rozlišení na ose X
- yRes – rozlišení na ose Y

- unitType – informace o jednotkách, ve kterých má být rozlišení nastaveno

public static extern void resizeImage(void* ptr, int w, int h, int filterType)
- provede změnu velikosti obrázku
- ptr – pointer na instanci třídy Magick::Image, na které má být změna velikosti provedena
- w – nová šířka v pixelech
- h – nová výška v pixelech
- filterType – informace o filtru (interpolačním algoritmu), který má být použit

public static extern void clipImage(void* ptr, int x, int y, int w, int h)
- ořízne obrázek
- ptr - pointer na instanci třídy Magick::Image, na které má být provedeno oříznutí
- x – x souřadnice počátku vyříznuté oblasti
- y – y souřadnice počátku vyříznuté oblasti
- w – šířka v pixelech vyříznuté oblasti
- h – výška v pixelech vyříznuté oblasti

4.1.2 Třída Picture

Třída reprezentující obrázek v bitmapové podobě.

4.1.2.1 Konstruktory

public Picture(int width, int height)
- vytvoří obrázek o zadaných rozměrech v pixelech
- width – šířka v pixelech
- height – výška v pixelech

public Picture(String filename)
- vytvoří instanci a nastaví jí jméno souboru pro pozdější načtení obrázku
- filename – jméno souboru

4.1.2.2 Metody

public void Ping()
- načte informace o obrázku, ale nezíská informace o pixelech

public void Load()
- načte obrázek

public void Save(String saveFilename)
- uloží obrázek do souboru
- filename – jméno výstupního souboru

public void Resize(int newWidth, int newHeight, FilterType filterType)
- změní velikost obrázku
- newWidth – nová šířka v pixelech
- newHeight – nová výška v pixelech
- filterType – použitý filter (interpolační angoritmus)

public void Clip(int x, int y, int newWidth, int newHeight)
- ořízne obrázek
- x – x souřadnice počátku vyříznuté oblasti
- y – y souřádnice počátku vyříznuté oblasti
- newWidth – šířka vyříznuté oblasti v pixelech

- newHeight – výška vyříznuté oblasti v pixelech

4.1.3 Třída FilterType

Slouží k reprezentaci typu filtru (interpolačního algoritmu) pro změnu velikosti obrázku.

4.1.3.1 Konstruktory

private FilterType(String name, int value)
- nastaví jméno filtru a hodnotu, která odpovídá danému filtru v knihovně Magick++

4.1.3.2 Metody

public override string ToString()
- vrací jméno filtru

4.1.3.3 Veřejné atributy

- každý atribut odpovídá jednomu typu filtru
public static readonly FilterType None = new FilterType("Nearest neighbour", 1) - nejbližší soused
public static readonly FilterType Triangle = new FilterType("Triangle", 3) – trojúhelníkový filtr (bilineární interpolace)
public static readonly FilterType Cubic = new FilterType("Cubic", 10) – kubický filtr (bikubická interpolace)
public static readonly FilterType Lanczos = new FilterType("Lanczos", 22) – Lanczos filtr

4.1.4 Třída InterlacingData

Uchovává všechny parametry potřebné k provedení procesu proložení

4.1.4.1 Konstruktory

Žádné. Možná budou později doplněny, ale pro funkčnost aplikace nejsou potřeba.

4.1.4.2 Metody

Pouze gettery a settery.

4.1.5 Třída LineData

Uchovává všechny parametry potřebné k přidání pasovacích značek do proloženého obrázku.

4.1.5.1 Konstruktory

Žádné. Možná budou později doplněny, ale pro funkčnost aplikace nejsou potřeba.

4.1.5.2 Metody

Pouze gettery a settery.

4.1.6 Třída ProjectData

Slouží pro načítání a ukládání projektů, tedy parametrů pro proces prokládání a přidání pasovacích značek.
Zároveň slouží jako zapouzdrovací třída pro třídy InterlacingData a LineData.

4.1.6.1 Konstruktory

Žádné. Možná budou později doplněny, ale pro funkčnost aplikace nejsou potřeba.

4.1.6.2 Metody

public void Save(String filename)
- uloží projekt do souboru
- filename – název souboru

public void Load(String filename)
- načte projekt ze souboru
- filename – název souboru

4.1.7 Třída PictureContainer

Slouží ke zrpacování vstupních obrázku na základě parametrů prokládání a parametrů pasovacích značek.

4.1.7.1 Konstruktory

public PictureContainer(List<Picture> pictures, InterlacingData interlacingData, LineData lineData, ProgressBar progressBar = null)
- inicializuje objekt potřebnými informacemi
- pictures – seznam obrázků reprezentovaných objekty třídy Picture
- interlacingData – parametry potřebné k provedení procesu proložení
- lineData – parametry potřebné k přidání pasovacích značek
- progessBar – progress bar použitý k zobrazení postupu prokládání

4.1.7.2 Metody

public bool CheckPictures()
- projde vstupní obrázky a zjistí, zda jsou všechny stejně velké, pokud ano vrátí true, pokud ne vrátí false a při procesu prokládání se pak obrázky oříznou na šířku nejužšího obrázku a výšku nejnižšího obrázku

public void Interlace()
- provede zpracování vstupních obrázků a vytvoří obrázek výstupní, ten je uložen do atributu **result**

4.1.8 Třída SettingOptions

Obsahuje informace o všech možnostech pro nastavení. Funguje spíše jako struktura.

4.1.8.1 Konstruktory

Žádné. Možná budou později doplněny, ale pro funkčnost aplikace nejsou potřeba.

4.1.8.2 Metody

Žádné

4.1.8.3 Veřejné atributy

public List<> unitsOptions;
- možnosti nastavení délkových jednotek

public List<> resolutionUnitsOptions;
- možnosti nastavení jednotek pro obrazové rozlišení a hustotu čoček

public List<> languageOptions;
- možnosti nastavení jazyka

4.1.9 Třída Settings

Obsahuje informace o aktuálním nastavení aplikace.

4.1.9.1 Konstruktory

public Settings(SettingOptions settingOptions)
- inicializuje objekt instancí třídy SettingOptions, na terou bude nastavení vázáno

4.1.9.2 Metody

public void Save(String filename)
- uloží nastavení do souboru
- filename – název souboru

public void Load(String filename)
- načte nastavení ze souboru
- filename – název souboru

4.1.10 Třída Localization

Statická třída, která slouží k nastavení jazyka aplikace

4.1.10.1 Konstruktory

Nejsou

4.1.10.2 Metody

public static void changeCulture(culture)
- změní nastavení kultury programu
- culture – kultura, na kterou bude program nastaven

public static void iterateOverControls(Control parent, ComponentResourceManager res)
- projde všechny komponenty formuláře a nastaví jim aktuákně používaný jazyk

- parent – kořenová komponenta
- res – instance třídy **ComponentResourceManager** použitá pro nastavení jazyka

4.1.11 Třída PreviewData

Slouží k uchovávání obrázků pro náhledy a jejich zobrazování.

4.1.11.1 Konstruktory

PreviewData(PictureBox pictureBox)
- nastaví picture box, do kterého bude instance zobrazovat náhledy

4.1.11.2 Metody

public void AddPicture(String path)
- přidá do instance obrázek k náhledu
- path – název souboru

public void ShowPicture(String path)
- zobrazí obrázek ve svém picture boxu
- path – název souboru pod kterým je obrázek v instanci uchováván

4.1.12 Třída MainForm

Třída hlavního formuláře aplikace.

4.1.12.1 Konstruktory

MainForm()
- inicializuje formlulář a jeho komponenty, nastaví jazyk, nastaví hodnoty komponent a výchozí

4.1.12.2 Metody

public void ApplySettings()
- aplikuje akutální nastavení obsažené v atributu settings

4.1.13 Třída SettingsForm

Třída formuláře, který slouží k zobrazení a upravení nastavení aplikace.

4.1.13.1 Konstruktory

SettingsForm(MainForm parent, Settings settings)
- inicializuje formlulář a jeho komponenty, nastaví zayk, nastaví hodnoty komponent podle instance settings třídy **Settings**
- parent – hlavní formulář aplikace, na který tento formulář přebírá odkaz
- settings – instance třídy **Settings**, která obsahuje aktuální nastavení aplikace, je předána odkazem ze třídy **MainForm**

4.1.13.2 Metody

Žádné.