Programowanie Komputerów 4					
Rok akademicki	Termin	Rodzaj studiów	Kierunek	Prowadzący	Grupa
2019/2020	Zajęcia zdalne w trakcie pandemii	**SSI**	**INF**	**KP**	warunek

Temat:

Skala szarości w C#

Autor:
Dominik Nowocień

Przedstawienie i omówienie realizowanego problemu

Celem projektu było stworzenie prostego edytora zapewniającego zmianę kolorów wybranego obrazu na jego odpowiednik w kolorach czarno-białych. Ze względu na dużą popularność algorytmów wyciągających średnią z kolorów (czerwonego, zielonego i niebieskiego) zdecydowałem się nie używać wyżej wymienionego sposobu tylko wyszukać inne funkcje realizujące zadany problem.

Uzasadnienie wyboru projektu:

Łatwa możliwość późniejszej rozbudowy w celu poszerzania wiedzy i uczenia się nowych technik programowania obiektowego jak i pracy z zewnętrznymi bibliotekami.

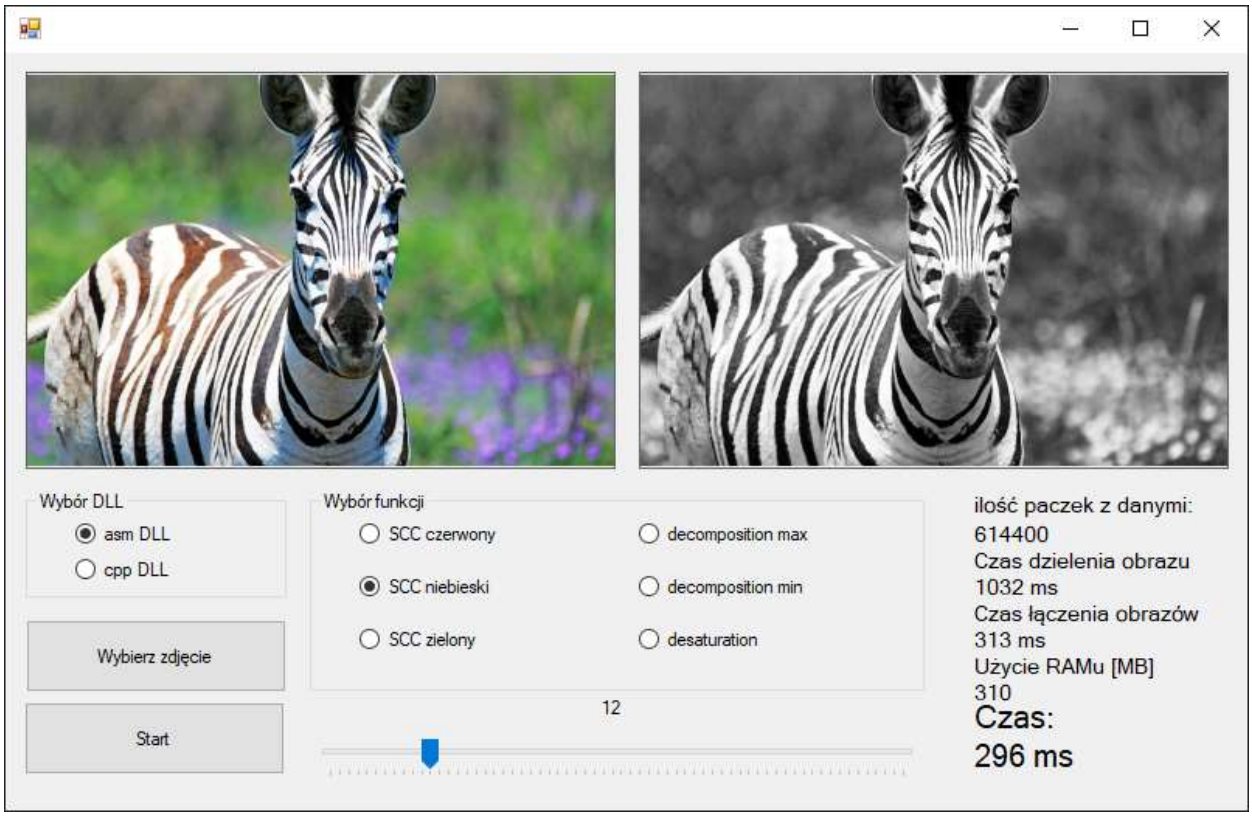
Zdefiniowanie wybranego fragmentu kodu realizowanego z wykorzystaniem bibliotek zewnętrznych (ASM/CPP)

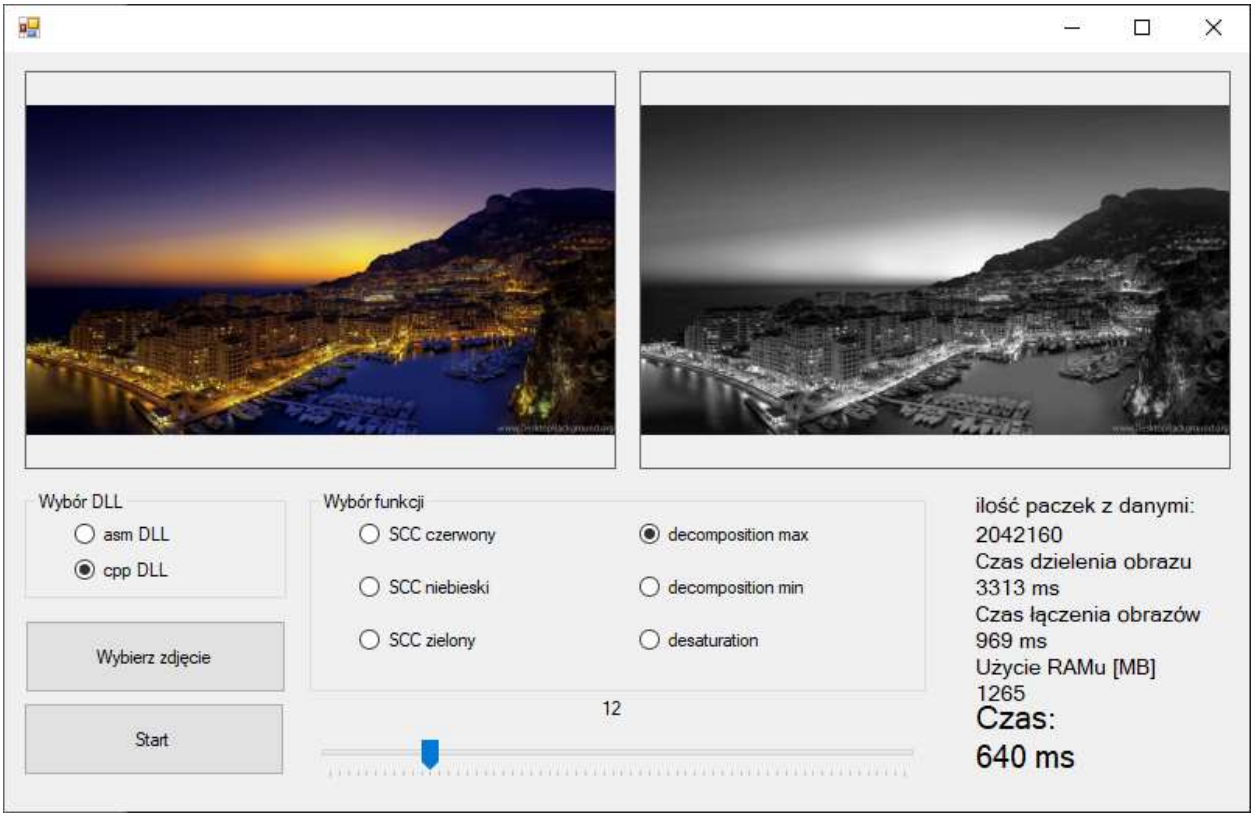
```
public void ChangeColorToGrayScale(object data)
{
    int i = ThreadService.GetI();

    while (i < Pixels.Length)
    {
        ProcessingMethod(Pixels.GetRed(i), Pixels.GetGreen(i), Pixels.GetBlue(i));
        i = ThreadService.GetI();
    }
}
```

Wybrany fragment kodu jest wykonywany równolegle przez zdefiniowaną przez użytkownika ilość wątków. Polega na iterowaniu po kolejnych nieprzetworzonych paczkach obrazu aż do wyczerpania ich ilości. Całość jest wydzielonym w osobnej klasie fragmentem realizowanym w języku c#.

Realizowanie tej części sprawozdania pozwoliło na usunięcie nadmiarowości kodu w postaci pętli for na sztywno ustawionej na jedną iterację. Wnioski: sumienna praca nad dokumentacją pozwala na lepsze zrozumienie własnego kodu i znalezienie błędów.

Pokazanie działania programu





Użycie poszczególnych składowych języka.

Threads

```
ThreadStart ts = ProcessingFunction;
ts += ThreadDone;

threads = new Thread[ThreadsCount];
for (int i = 0; i < ThreadsCount; i++)
{
    threads[i] = new Thread(ts);
    threads[i].IsBackground = true;
    threads[i].Start();
}
```

Nastąpiła próba implementacji prostego menadżera wątków. Dalsza część kodu obejmuje dodatkowo próbę ich synchronizacji poprzez użycie zmiennej Locker.

Templates

```
public class PixelPackageHelper<T>
{
    private const int Size = 16;

    Odwołania: 2
    public PixelPackageHelper(int len)
    {
        Length = len;
        Red = new T[len][];
        Green = new T[len][];
        Blue = new T[len][];

        for (int i = 0; i < len; i++)
        {
            Red[i] = new T[Size];
            Green[i] = new T[Size];
            Blue[i] = new T[Size];
        }
    }
```

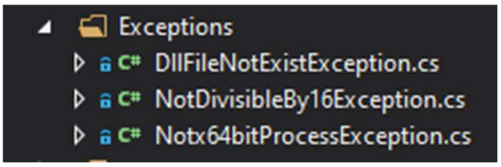
Jako klasa helper do przechowywania danych o kolorach pixela w zgrabnych paczkach.

Regex

```
private string regexPattern = String.Empty;

1 odwołanie
public string SearchString(string separator, IEnumerable<string> lines)
{
    Regex regex = new Regex(regexPattern);

    var regexMatch = lines.Where<string>(x => regex.IsMatch(x));

    return String.Join(separator, regexMatch);
}
```

Mechanizm służący do przeszukiwania logów. „Ubrany” w przystępny dla użytkownika WinForm.

Exceptions



Zaimplementowałem własne 3 wyjątki oraz w programie występują przykłady użycia zarówno tych z bibliotek systemowych jak i własnych.

Przykłady:

```
catch (System.IO.FileNotFoundException exception)
{
    loger.Error(exception.Message);
    MessageBox.Show(String.Format("Nie znaleziono pliku:\n\r{0}", exception.Message));
}
catch (NotDivisibleBy16Exception exception)
{
    loger.Error(exception.Message);
    MessageBox.Show(String.Format("Jestem leniwym programista i zamiast obrobic zdjęcie
}
catch (Exception exception)
{
    loger.Error(exception.Message);
```

```
if (imageToProcess.Height % 16 != 0)
{
    throw new NotDivisibleBy16Exception();
}
```

Inteligentne wskaźniki

Ze względu na rozbieżności w językach Cpp i C# ich użycie jest zbędne dzięki GC

Źródło: https://en.wikipedia.org/wiki/Smart_pointer

STL

```
var regexMatch = lines.Where<string>(x => regex.IsMatch(x));
```

```
RadioButton checkedRadioButton = groupBox.Controls.Cast<RadioButton>()
                                                  .Where(x => x.Checked)
                                                  .FirstOrDefault();

Dll = dllTypes.Where(x => checkedRadioButton.Text.ToUpper().Contains(x.Key))
              .First()
              .Value;
```

Jako takich list/kolejek nie zastosowałem lecz za każdym razem występują one niejawnie i są wykorzystywane na obiektach typu IEnumerable. Znajdują się w ns Linq.

RTTI

```
if (exception is ArgumentException)
{
    MessageBox.Show(exception.GetType().ToString());
}
```

Ich zastosowanie w języku C# sprawiło mi duże problemy ponieważ nie jestem pewny czy zastosowane przeze mnie struktury należą do RTTI

Opis bibliotek zewnętrznych

```
LIBRARY C_DLL
EXPORTS SingleColorChannel_Red_CPP
EXPORTS SingleColorChannel_Green_CPP
EXPORTS SingleColorChannel_Blue_CPP
EXPORTS Decomposition_max_CPP
EXPORTS Decomposition_min_CPP
EXPORTS Desaturation_CPP
```

```
LIBRARY ASM_DLL_2017
EXPORTS SingleColorChannel_Red_ASM
EXPORTS SingleColorChannel_Green_ASM
EXPORTS SingleColorChannel_Blue_ASM
EXPORTS Decomposition_max_ASM
EXPORTS Decomposition_min_ASM
EXPORTS Desaturation_ASM
```

Poszczególne funkcje są odpowiadają sobie i są swoimi analogami. Ponad to kod wynikowy napisany w CPP jest przekształcany podczas asemblacji do kodu praktycznie identycznego jak w bibliotece ASM.

Każda z funkcji przyjmuje jako parametr trzy wskaźniki na 16 bajtowe paczki z danymi o kolorach danego pixelu. Zwracanym typem zawsze jest void, ponieważ pracujemy bezpośrednio na danych.

Single Color Channel przyporządkowuje dany kolor(widniejący w nazwie funkcji) do pozostałych kolorów.

Dekompozycja max/min wyciąga odpowiednio największą/najmniejszą wartość spośród trzech składowych koloru RGB i rozpropagowuje na inne składowe koloru

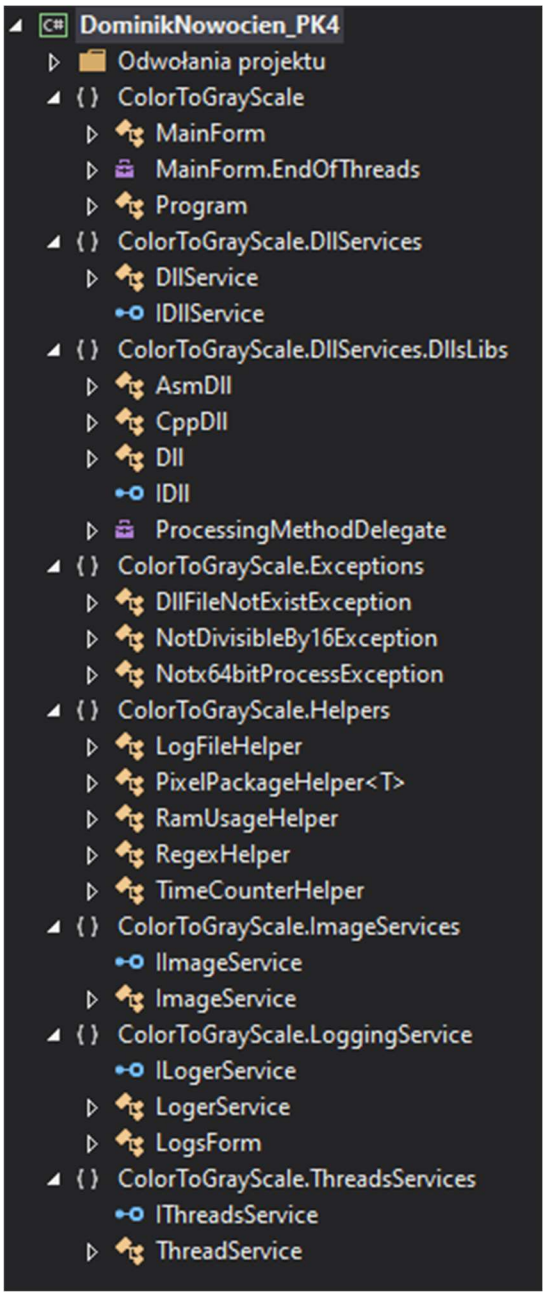
Desaturacja wyciąga wartość średnią z największej i najmniejszej wartości spośród RGB i przypisuje do wszystkich trzech kolorów.

```
void C_DLL::SingleColorChannel_Blue_CPP(unsigned char* r, unsigned char* g, unsigned char* b)
{
    __m128i* r_sse = (__m128i*)r;
    __m128i* g_sse = (__m128i*)g;
    __m128i* b_sse = (__m128i*)b;

    _mm_storeu_si128(g_sse, *b_sse);
    _mm_storeu_si128(r_sse, *b_sse);
}
```

```
SingleColorChannel_Blue_ASM PROC
movdqu xmm1, [rcx]
movdqu xmm2, [rdx]
movdqu xmm3, [r8]
movdqu [rcx], xmm3
movdqu [rdx], xmm3
movdqu [r8], xmm3
ret
SingleColorChannel_Blue_ASM endp
```

Przykładowy kod w CPP i jego odpowiednik w ASM

Specyfikacja wewnętrzna



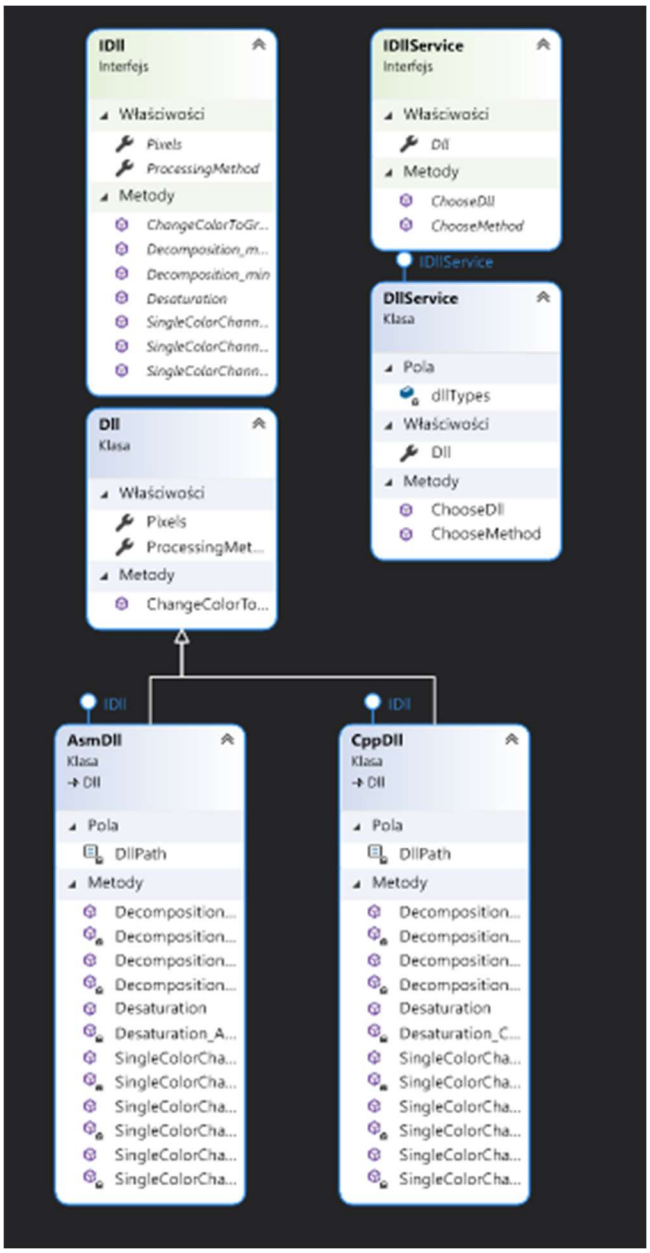
Cały projekt jest podzielony na 7 namespacesów tj.:

Exceptions, Helpers, DllService, DllService.DllLibs, ImageService, LoggingService, ThreadsService

Dla 5 ostatnich zostały wystawione odpowiednie interfejsy.

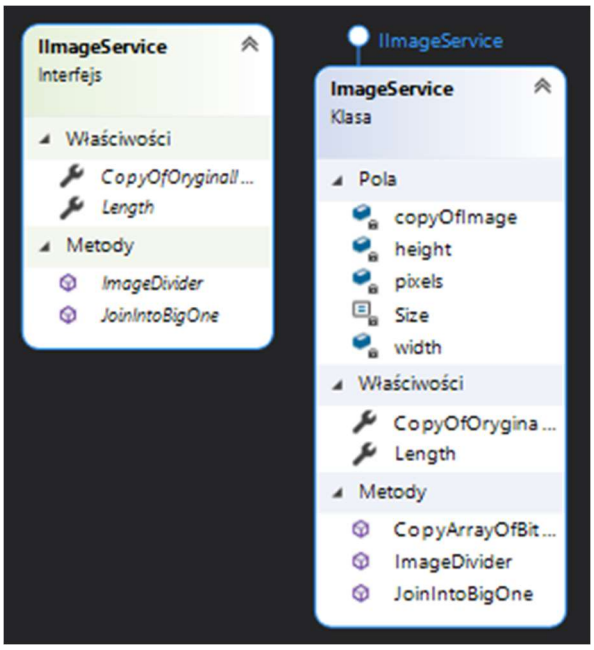
DLLService i DllLibs

Obsługa bibliotek dll



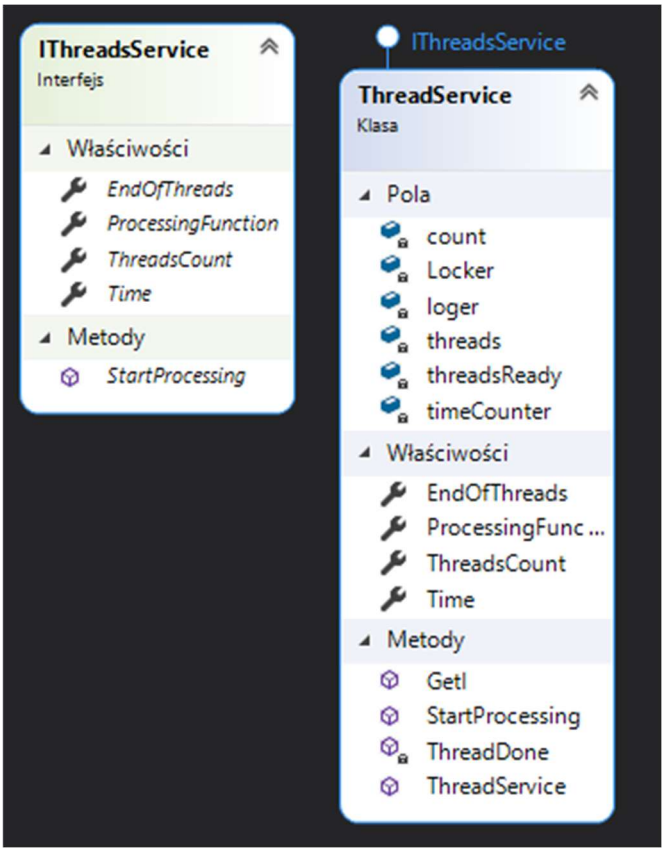
Pozwala na obsługę odpowiednich bibliotek i pozwala wybrać odpowiadające metody z zewnętrznych dll.

ImageService



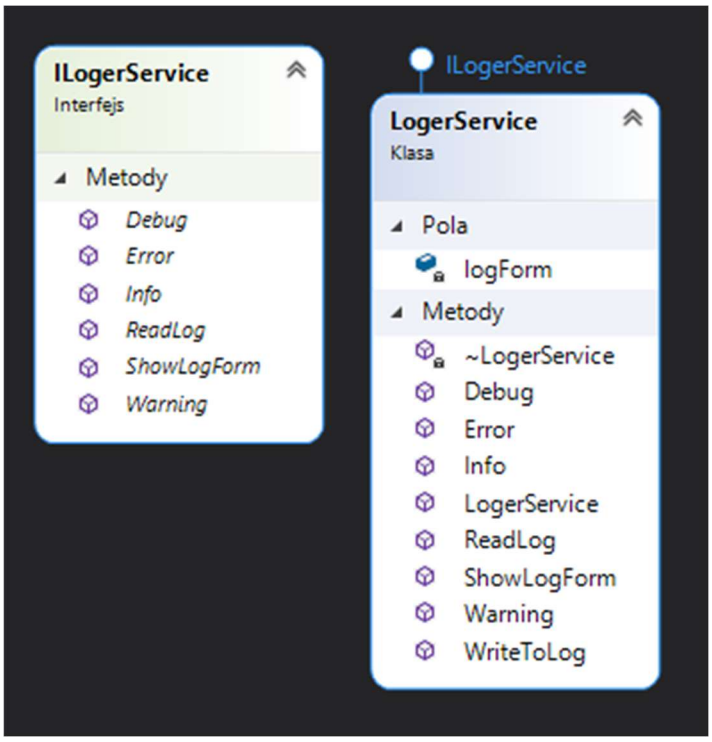
Przygotowuje obraz do zmiany koloru oraz składa do po zakończonym procesie.

ThreadsService



Dostarcza mechanizm wątków i przetwarzania obrazu w sposób zrównoleglony.

LogerService

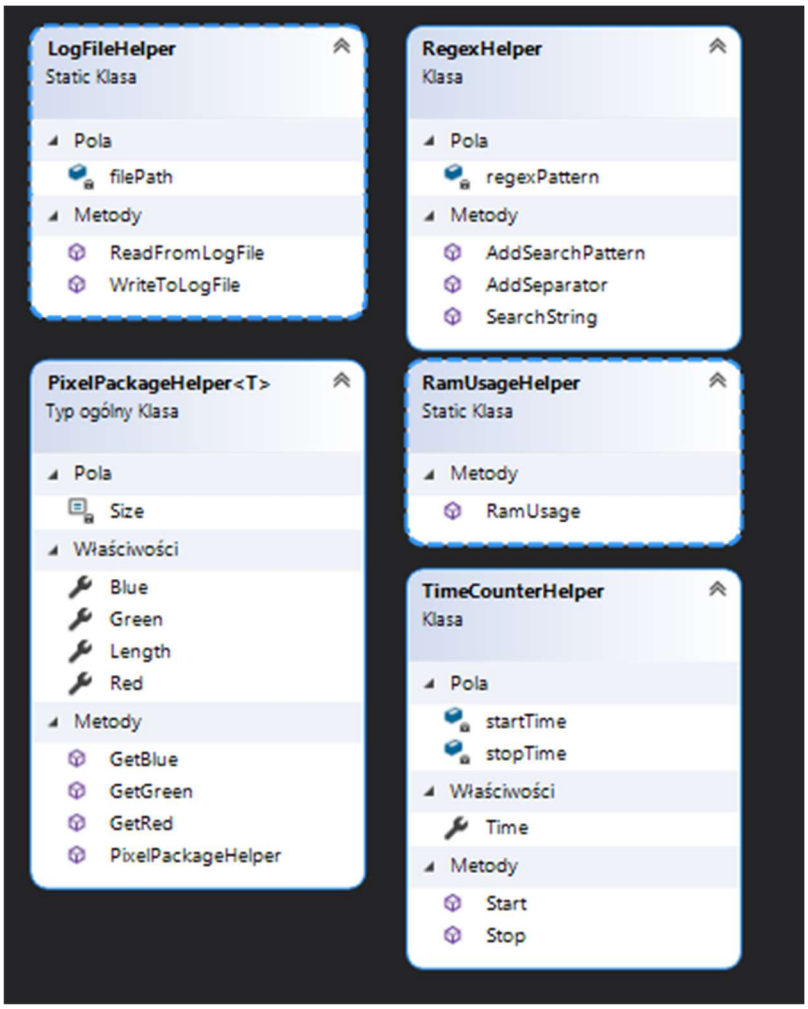


Pozwala logować do pliku, oraz odczytywać w sposób przyjemny dla użytkownika, dostarcza WindowsForm z oknem ułatwiającym wizualizację.

Exceptions

Odpowiada za dostarczenie własnych wyjątków.

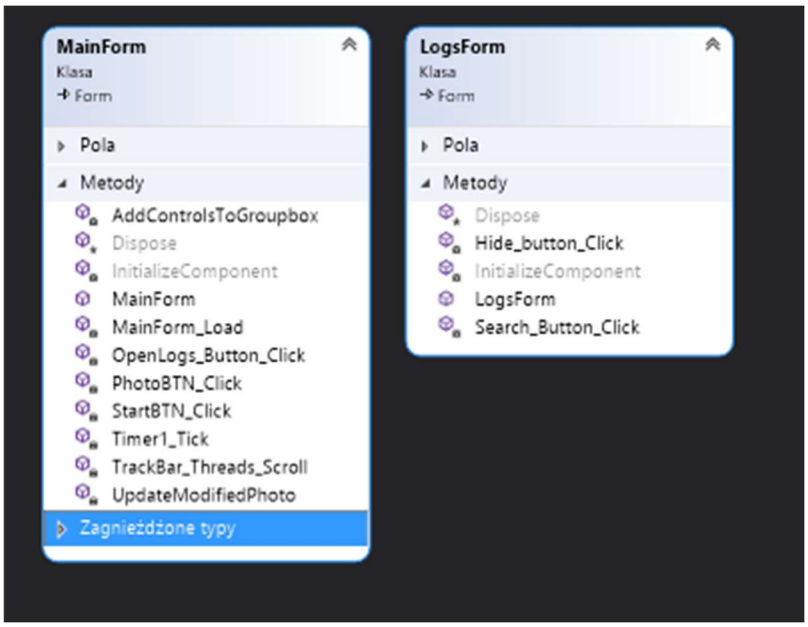
Helpers



Dostarcza klasy pomocnicze które odpowiednio:

- **LogFile** pomaga odczytywać i zapisywać do pliku logu wpisy.
- **PixelPackageHelper** przechowuje informacje o paczce pixeli i ich kolorach. Ponad to mamy zastosowany template.
- **Regex** pomaga w wysługiwaniach z użyciem regex
- **RamUsage** dostarcza informacji i aktualnym zużyciu pamięci RAM.
- **TimeCounter** pomaga liczyć czas trwania metod/funkcji.

WindowForm

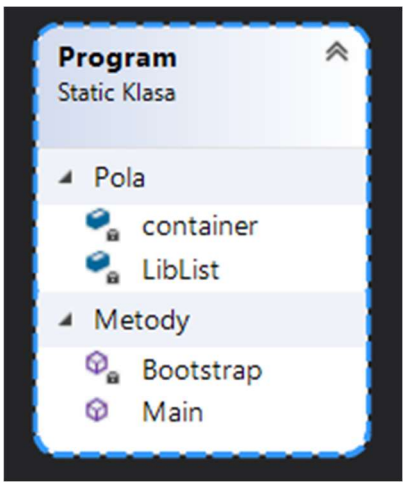


Program zawiera 2 Formsy:

MainfForm jest wywoływany z funkcji main i jest głównym oknem programu.

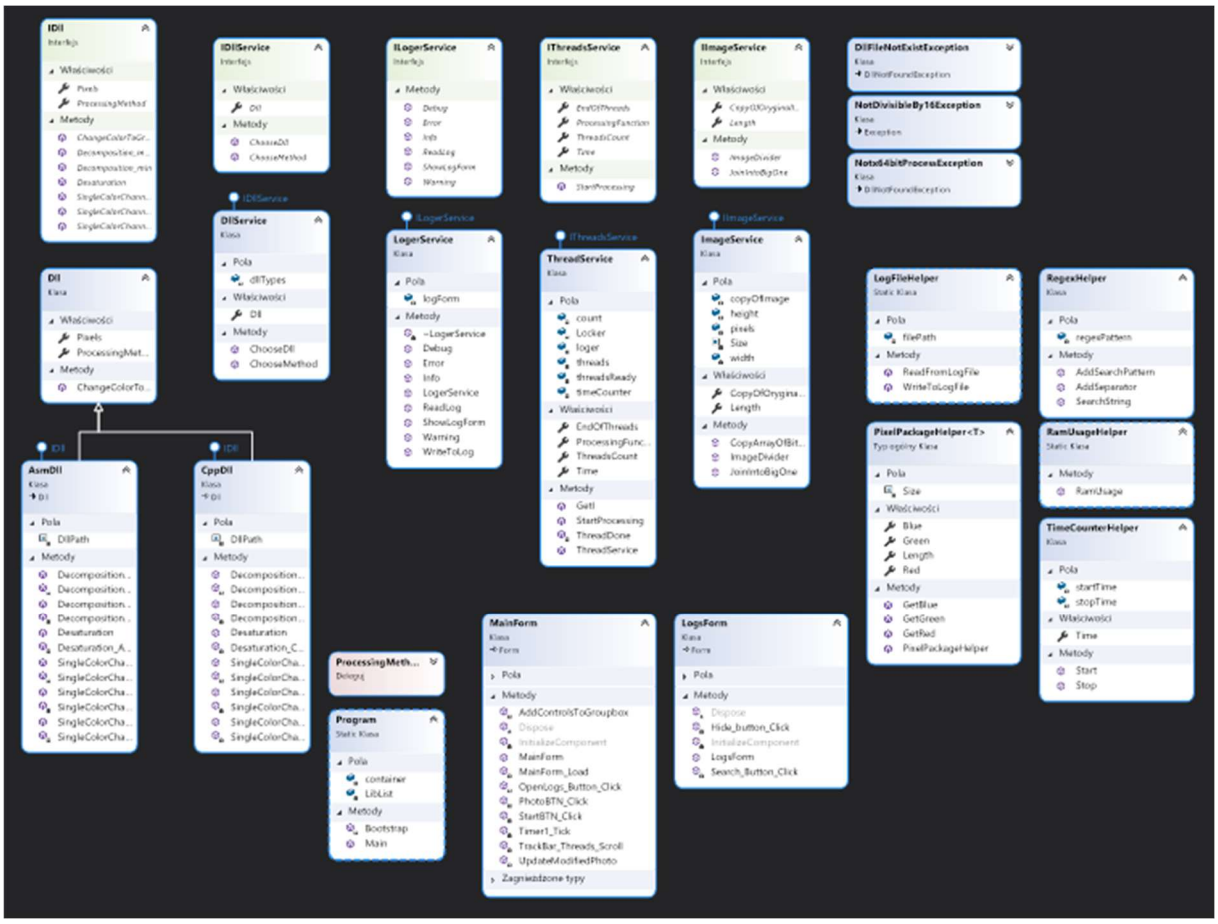
LogsForm jest dodatkowym oknem pozwalającym na uporządkowanie logów z działania programu.

Start programu



Rozruch następuje klasycznie z metody main. Jednak przed inicjalizacją rejestrowane są usługi w celu późniejszego wykorzystania wstrzykiwania zależności

Diagram klas



Wnioski

Efekt końcowy projektu spełnia jego założenia – możliwa jest zamiana obrazu kolorowego na czarno-biały z wykorzystaniem bibliotek napisanych w języku asembler i c++. Wydajność programu jest ograniczona ze względu na jednowątkowe przygotowywanie i sklejanie otrzymanego programu. Funkcję te zostaną w przyszłości rozwinięte w ramach indywidualnej pracy z w środowisku .NET. Największą częścią pod względem czasowym okazała się konfiguracja bibliotek dll i ich podpięcie. Problemem była konfiguracja środowiska Visual Studio. Całość projektu upewniła mnie w założeniu, że praca z GITem jest dobrą praktyką i pomaga w zarządzaniu kodem, tworzeniu backupu oraz rozwijaniu projektów niepewnych kierunkach bez ponoszenia konsekwencji utraty wartościowego kodu. Zdecydowałem się również na próbę stworzenia menagera wątków zamiast realizacji kodu w oparciu gotowe rozwiązania. Zdecydowanie utrudniło to moją pracę jednak osiągnięty efekt dydaktyczny był warty wysiłku.

Wnioski po rozszerzeniu o projekt na PK4

Rozwijanie projektu w ramach przedmiotu programowanie komputerów 4 pozwoliło mi zweryfikować swoje założenie co do użycia wstrzykiwania zależności. Jest to całkiem ciekawa metoda pracy jednak w większości przypadków w tym projekcie jest całkowicie nieuzasadniona jednak ze względu na chęć „zabawy kodem” zdecydowałem się nie wycofywać z początkowej koncepcji.

Dużej modyfikacji uległa logika głównego okna aplikacji przez próbę trzymania SRP.

Utworzone zostały dodatkowe klasy *Helper, a cały kod został podzielony pod względem funkcjonalności. Do projektu została dołożona funkcjonalność logów ze względu na chęć zrozumienia zagadnienia wyrażeń regularnych. Praca z gitem była przydatna ze względu na częste błędne ścieżki rozwoju kodu.

Wnioski3

Warto pisać na bieżąco dokumentację techniczną ponieważ nadrabianie po czasie jest zdecydowanie utrudnionym elementem tworzenia oprogramowania. Warto prowadzić na bieżąco sprawozdania z pracy lub/i stosować automatyczną dokumentację. Która znacząco poprawi zrozumienie kodu po pewnym czasie od napisania.