Programowanie Obiektowe INEW00002L

Zarządzanie inteligentnym budynkiem

Wykonał Olaf Krawczyk

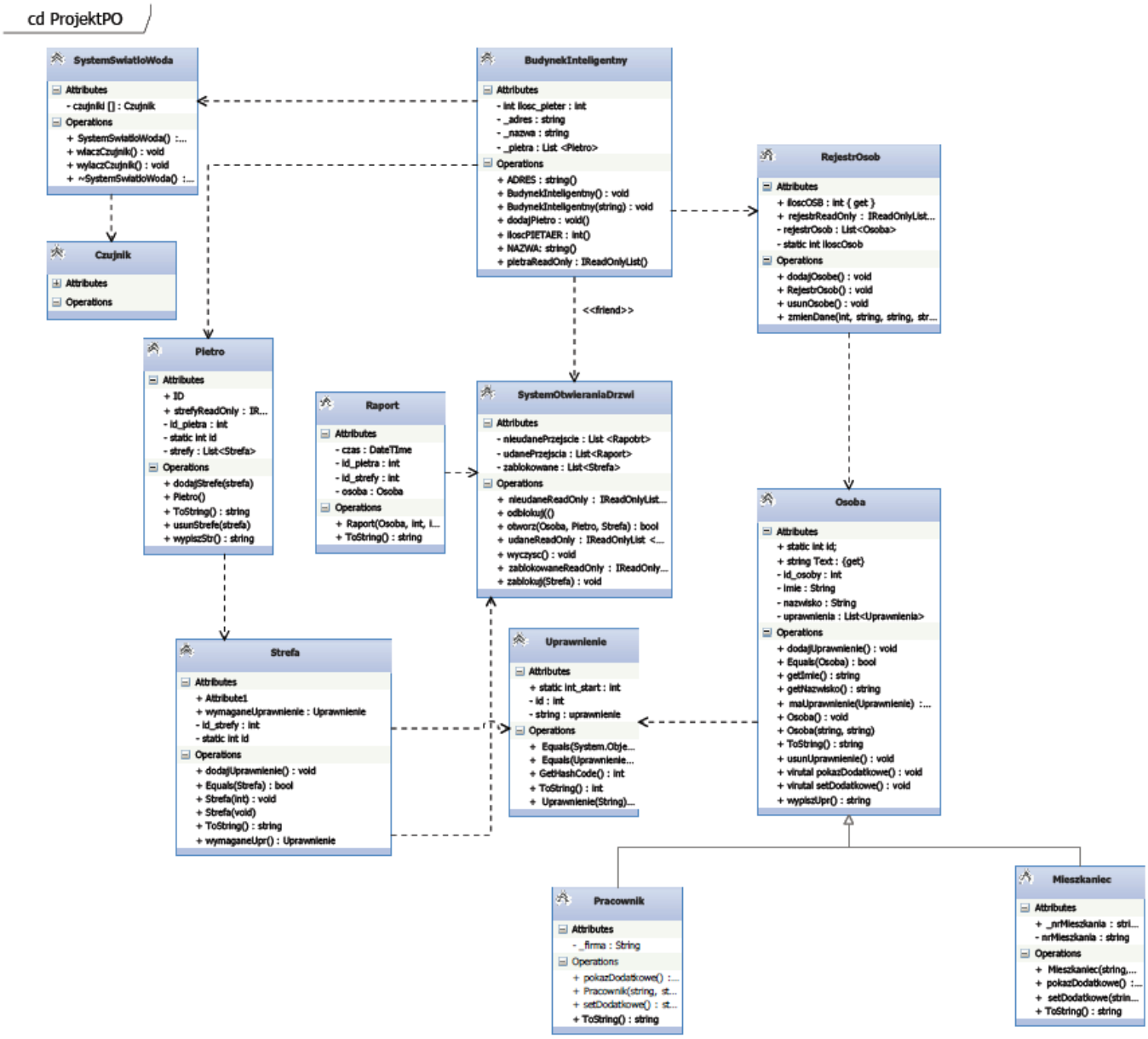
Sprawdzający Mgr inż. Karol Puchała

Wrocław, 03.06.*2015*

I. Opis projektu

Program Zarządzanie budynkiem inteligentnym ma za zadanie wspomagać obsługę nowoczesnych budynków. Jego głównym zadaniem jest umożliwienie administracji kontrolowania dostępu do poszczególnych stref. Program przechowuje informacje o piętrach budynku oraz strefach na jakie zostały podzielone poszczególne piętra. Każda ze stref przechowuje informacje o uprawnieniach jakie należy posiadać, aby wejść do danej strefy. W aplikacji prowadzony jest również rejestr osób, który jest pewnego rodzaju bazą danych. Dane o osobach jakie może przechowywać to imię, nazwisko, uprawnienia posiadane przez daną osobę oraz w zależności od osoby nazwę firmy w jakiej pracuje lub nr mieszkania. Za przyznawanie dostępu do poszczególnych stref odpowiedzialny jest moduł Otwórz. Na podstawie wprowadzonych danych decyduje czy dana osoba może przekroczyć daną strefę. Wszystkie próby dostania się do poszczególnych stref są zapisywane. Na podstawie tych danych jest możliwe wygenerowanie prostego raportu.

II. *Diagram klas UML*



Rysunek 1 - Pełna wersja w osobnym pliku

III. *Fragmenty plików .cs z atrybutami oraz nazwami metod*

BudynekInteligentny.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BudynekInt
{
    public class BudynekInteligetny
    {
        //  Podstawowe dane o budynku - nazwa, adres
        private string _adres;
        private string _nazwa;
        public string ADRES {get { return _adres; } set { _adres = value; }}
        public string NAZWA { get { return _nazwa; } set { _nazwa = value; }
}
        private int ilosc_pieter;
        public int iloscPIETER { get { return ilosc_pieter; } set {
this.ilosc_pieter = value; } }

        // Lista generyczna przechowująca obiekty typu piętro oraz getter
        public List<Pietro> _pietra = new List<Pietro>();
        public IReadOnlyList<Pietro> pietraReadOnly { get { return
_pietra.AsReadOnly(); } } // getter pięter

        public BudynekInteligetny()
        public BudynekInteligetny(string iNazwa)
        public void dodajPietro
}
```

Czujnik.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BudynekInt
{
    // Klasa w fazie przygotowań
    class Czujnik
    {
    }
}
```

Mieszkaniec.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BudynekInt
{
    class Mieszkaniec : Osoba
```

```
    {
        // Klasa mieszkaniec, rozszerza klasę osoba o dodatkowe informacje -
nr mieszkania
        private string _nrMieszkania { get; set; }
        public string nrMieszkania {
            set { this._nrMieszkania = value; }
            get { return this._nrMieszkania; }
        }

        public Mieszkaniec(string iImie, string iNazwisko, string
iNrMieszkania) : base(iImie, iNazwisko)
        public override string pokazDodatkowe()
        public override void setDodatkowe(string add)
        public override string ToString()
    }
}
```

Osoba.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BudynekInt
{
    public class Osoba
    {
        public static int id = 0;

        private int id_osoby;
        public int getID { get { return id_osoby; } }
        private string imie;
        private string nazwisko;

        public string Text { get { return imie + " " + nazwisko; } }

        private List<Uprawnienie> uprawnienia = new List<Uprawnienie>();
        public IReadOnlyList<Uprawnienie> uprawnieniaReadOnly { get { return
uprawnienia.AsReadOnly(); } }

        public Osoba()
        public Osoba(string iImie, string iNazwisko)

        // Wirtualna metoda wyświetlająca imformacje o firmie lub nr
mieszkania
        // w zależności od klasy
        public virtual string pokazDodatkowe()

        // Wirtualna metoda ustalająca dodatkowy parametr firma/nr
mieszkania
        public virtual void setDodatkowe(string add)
        // gettery i settery klasy osoba
        public string getImie()
        public string getNazwisko
        public void setImie(string iImie)
        public void setNazwisko(string iNazwisko)

        // Metody do obsługi uprawnień osoby
        public void dodajUprawnienie(Uprawnienie iUpr)
```

```
        public void usunUprawnienie(Uprawnienie iUpr)
        public bool maUprawnienie(Uprawnienie iUpr)
        public string wypiszUpr(
        public bool Equals(Osoba iOsb)
        public override string ToString()
    }
}
```

Pietro.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BudynekInt
{
    public class Pietro
    {
        private static int id = 0;
        private int id_pietra;
        public int ID { get { return id_pietra; } }
        private List<Strefa> strefy = new List<Strefa>();
        public IReadOnlyList<Strefa> strefyReadOnly { get { return
strefy.AsReadOnly(); } }
        public Pietro()
        public void dodajStrefe(Strefa iStref)
        public void usunStrefe(Strefa iStref)
        public string wypiszStr
        public override string ToString()
    }
}
```

Pracownik.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BudynekInt
{
    public class Pracownik : Osoba
    {
        private string firma { get; set; }

        public Pracownik(string iImie, string iNazwisko, string iFirma)
            : base(iImie, iNazwisko)
        public override string pokazDodatkowe()
        public override void setDodatkowe(string add)
        public override string ToString()
    }
}
```

RejestrOsob.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace BudynekInt
{
    public class RejestrOsob
    {
        static private int iloscOsob = 0;
        public int iloscOSB { get { return iloscOsob; } }

        private List<Osoba> rejestrOsob = new List<Osoba>();
        public IReadOnlyList<Osoba> rejestrReadOnly { get { return
rejestrOsob.AsReadOnly(); } }

        public void dodajOsobe(Osoba iOsb)
        public void usunOsobe(Osoba iOsb)
        public void zmienDane(int i, string imie, string nazwisko, string
add = "")
    }
}
```

Strefa.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BudynekInt
{
    public class Strefa
    {
        private static int id = 0;
        private int id_strefy;
        public int statID { get { return id; } set { id = value; }}
        public int ID { get { return id_strefy; } } // Getter id

        Uprawnienie wymaganeUprawnienie;

        public Strefa(int startId)
        public Strefa()

        //zmiana uprawnien
        public void dodajUprawnienie(Uprawnienie iUpr)

        //zwraca wymagane uprawnienia
        public Uprawnienie wymaganeUpr()
        // porównanie stref ze względu na id oraz wymagane uprawnienia
        public bool Equals(Strefa iStr)
        public override string ToString()
    }
}
```

SystemOtwieraniaDrzwi.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace BudynekInt
{
    public class SystemOtwieraniaDrzwi
    {
        private List<Raport> udanePrzejscia = new List<Raport>();
        private List<Raport> nieudanePrzejscia = new List<Raport>();
        private List<Strefa> zablokowane = new List<Strefa>();

        //gettery i settery do list;
        public IReadOnlyList<Raport> udaneReadOnly { get { return
udanePrzejscia.AsReadOnly(); } }
        public IReadOnlyList<Raport> nieudaneReadOnly { get { return
nieudanePrzejscia.AsReadOnly(); } }
        public IReadOnlyList<Strefa> zablokowaneReadOnly { get { return
zablokowane.AsReadOnly(); } }

        public SystemOtwieraniaDrzwi()

        public bool otworz(Osoba iOsb, Pietro iPiet, Strefa iStref)
        public void zablokuj(Strefa iStref)
        public void odblokuj(Strefa iStref)
        public void wyczyść()
    }
}
```

SystemSwiatloWoda.h

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BudynekInt
{
    public class SystemSwisatloWoda
    {
        // Klasa w przygotowaniu
        List<Czujnik> czujniki = new List<Czujnik>();

    }
}
```

Raport.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BudynekInt
{
    public class Raport
    {
        // Klasa raport wykorzystywana przez klasę SystemOtwieraniaDrzwi do
przechowywania informacji
        // o osobach i strefach jakie chciały przekroczyć
        private Osoba osoba;
        int id_pietra, id_strefy;
        private DateTime czas;
```

```
        public Raport(Osoba iOsb, int iId_pietra, int iID_strefy)
        public override string ToString()
    }
}
```

Uprawnienie.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BudynekInt
{
    public class Uprawnienie : IEquatable<Uprawnienie>
    {
        public static int id_start = 0;
        private int id;
        private string uprawnienie;

        public Uprawnienie(String iUpr)
        {
            uprawnienie = iUpr;
            id = id_start;
            id_start++;
        }

        public override bool Equals(System.Object iObject)
        public override int GetHashCode()
        public bool Equals(Uprawnienie iUpr)
        public override string ToString()
    }
}
```

IV. Podsumowanie

Projekt Zarządzanie Inteligentnym Budynkiem, pomimo drugiej wersji w dalszym ciągu pozwala na dalszy rozwój. Podobnie jak w poprzedniej wersji rozbudowy wymaga System ŚwiatłoWoda, oraz klasa Czujnik pozwalająca na kontrolowanie zużycia mediów w budynku oraz informowanie o ewentualnych awariach, pożarach i innych wypadkach losowych. W obecnej wersji udało mi się poprawić raporty generowane przez program. Zostały one wzbogacone o informację o dacie w jakiej nastąpiła próba przejścia przez daną strefę. Formatowanie wpisów w raporcie ułatwia jego analizę. Największą zmianą jest stworzenie programu opartego na Windows Forms. Sposób przechowywania danych przez program nie zmienił się znacząco. Poprawiona została również obsługa błędów, oraz sposób nadawania uprawnień. Program w obecnej wersji można wzbogacić o przechowywanie danych w plikach. Dzięki serializacji obiektów dostępnej w C# nie stanowiłoby to problemu.