Projekt zaliczeniowy

Programowanie obiektowe
Rok akademicki 2025/2026

Autorzy:
Patryk Tatarczyk
Szymon Szumański
Karol Szydłowski

System Obsługi Przychodni

Celem projektu było stworzenie aplikacji w C#, która służy do zarządzania pacjentami i lekarzami w przychodni. Nasz program pozwala na ewidencje osób, przechowywanie danych medycznych, filtrowanie, sortowanie oraz trwały zapis danych.

Podział ról

Patryk Tatarczyk:

- Stworzenie hierarchii klas
- Stworzenie algorytmu walidacji PESEL oraz walidacji numeru PWZ
- Zastosowanie interfejsów
- Zabezpieczenie danych przed błędnym wprowadzeniem

Szymon Szumański:

- Zaprojektowanie głównego okna aplikacji WPF
- Implementacja mechanizmu Data Binding
- Stworzenie dynamicznego interfejsu
- Oprogramowanie panelu statystyk

Karol Szydłowski:

- Napisanie klasy KlinikaManager
- Stworzenie i oprogramowanie osobnego okna dialogowego do zarządzania historią chorób
- Implementacja obsługi własnych wyjątków
- Przygotowanie testów jednostkowych

Klasy użyte w projekcie:

1) Osoba
   a) Rola w projekcie:
      i) Jest fundamentem dla całego systemu. Reprezentuje ogólny model człowieka w systemie.
   b) Uzasadnienie stworzenia:
      i) Zastosowanie dziedziczenia pozwala uniknąć duplikowania kodu w klasach dziedziczących (Pacjent i Lekarz)
   c) Modyfikatory dostępu:
      i) Pola prywatne (imie, nazwisko, pesel) zapewniają hermetyzację danych.
      ii) Właściwości publiczne do których dostęp jest możliwy dzięki właściwością. W set'cie użyto logikę walidację, dzięki czemu niemożliwe jest przypisanie błędnego numeru PESEL.

2) Pacjent
   a) Rola w projekcie:
      i) Reprezentuje pacjenta przychodni, jest rozszerzeniem klasy „Osoba" o informacje medyczne.
   b) Uzasadnienie stworzenia:
      i) Pacjent różni się od lekarza, tym że posiada on historię chorób.
   c) Modyfikatory dostępu:
      i) Lista „historiaChorob" jest zarządzana przez właściwość, co pozwala kontrolować sposób jej pobierania i ustawiania.
      ii) Implementacja metody „Clone()" jest publiczna, aby menadżer lub GUI mogły tworzyć kopie obiektu bez naruszania oryginału.

3) Lekarz:
   a) Rola w projekcie:
      i) Reprezentuje pracownika medycznego poprzez rozszerzenie klasy „Osoba" o uprawnienia i identyfikatory zawodowe.
   b) Uzasadnienie stworzenia:
      i) Lekarz różni się od pacjenta, tym że posiada pola: „specjalizacja" i „numerPWZ".
   c) Modyfikatory dostępu:
      i) Pola wymienione powyżej są prywatne zapewniając hermetyzację danych.
      ii) W publicznym set'cie dla „NumerPWZ" zastosowano walidację za pomocą funkcji Regex.

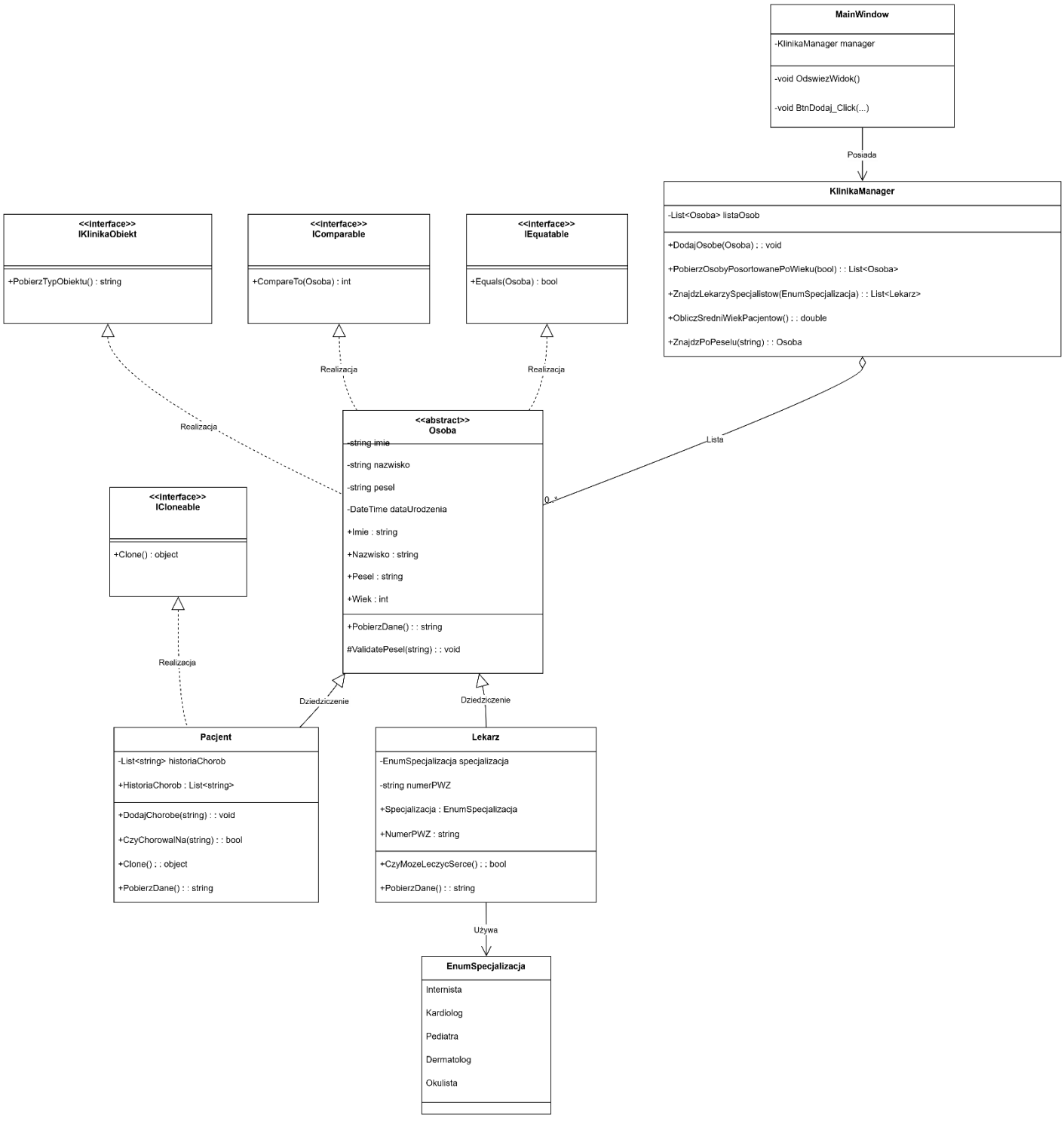
4) KlinikaManager:
   a) Rola w projekcie:
      i) Jest ”mózgiem” aplikacji, który łączy dane z GUI.
   b) Uzasadnienie stworzenia:
      i) Dzięki niej kod okna jest czysty, a lista osób jest zabezpieczona przed przypadkowym usunięciem.
   c) Modyfikatory dostępu:
      i) Lista jest prywatna, aby nie można było z niej usuwać elementów ani dodawać bez wlidacji.
      ii) Metody publiczne („DodajOsobe”, „ZnajdzPoPeselu”) stanowią interfejs API dla reszty aplikacji. Tylko przez nie można modyfikować stan bazy.

5) WiekComparer:
   a) Rola w projekcie:
      i) Klasa pomocnicza służąca do definiowania niestandardowego sposobu sortowania.
   b) Uzasadnienie stworzenia:
      i) Klasa „Osoba” implementuje domyślne sortowanie po nazwisku. Aby umożliwić alternatywne sortowanie po wieku stworzono osobny komparator.

6) BlednaDataException, NiepoprawnyPeselException
   a) Rola w projekcie:
      i) Sygnalizacja błędów specyficznych dla domeny problemu.
   b) Uzasadnienie stworzenia:
      i) Własne wyjątki pozwalają w GUI poezyjnie reagować na konkretne błędy walidacji

7) IKlinikaObiekt
   a) Rola w projekcie:
      i) Kontrakt, który musza spełnić obiekty w systemie.
   b) Uzasadnienie stworzenia:
      i) Zapewnia, że każda klasa implementująca ten interfejs będzie posiadała metodą identyfikującą jej typ.

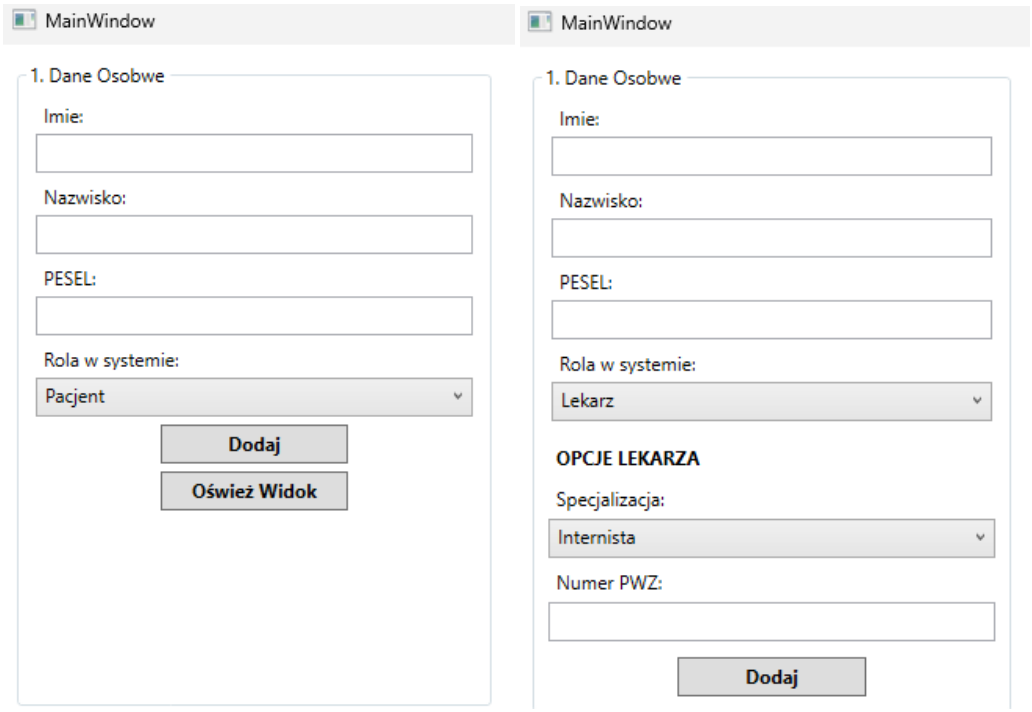
Diagram klas


MainWindow
-KlinikaManager manager
-void OdswiezWidok()
-void BtnDodaj_Click(...)
Posiada
KlinikaManager
-List<Osoba> listaOsob
+DodajOsobe(Osoba) : : void
+PobierzOsobyPosortowanePoWieku(bool) : : List<Osoba>
+ZnajdzLekarzySpecjalistow(EnumSpecjalizacja) : : List<Lekarz>
+ObliczSredniWiekPacjentow() : : double
+ZnajdzPoPeselu(string) : : Osoba
<<interface>>
IKlinikaObiekt
+PobierzTypObiektu() : string
<<interface>>
IComparable
+CompareTo(Osoba) : int
<<interface>>
IEquatable
+Equals(Osoba) : bool
Realizacja
Realizacja
Realizacja
Lista
0..*
<<abstract>>
Osoba
-string imie
-string nazwisko
-string pesel
-DateTime dataUrodzenia
+Imie : string
+Nazwisko : string
+Pesel : string
+Wiek : int
+PobierzDane() : : string
#ValidatePesel(string) : : void
<<interface>>
ICloneable
+Clone() : object
Realizacja
Dziedziczenie
Dziedziczenie
Pacjent
-List<string> historiaChorob
+HistoriaChorob : List<string>
+DodajChorobe(string) : : void
+CzyChorowalNa(string) : : bool
+Clone() : : object
+PobierzDane() : : string
Lekarz
-EnumSpecjalizacja specjalizacja
-string numerPWZ
+Specjalizacja : EnumSpecjalizacja
+NumerPWZ : string
+CzyMozeLeczycSerce() : : bool
+PobierzDane() : : string
Używa
EnumSpecjalizacja
Internista
Kardiolog
Pediatra
Dermatolog
Okulista

Opis funkcjonalności

Projekt realizuje szereg funkcji umożliwiający kompleksowe zarzadzanie mała placówką medyczną.

1. Rejestracja osób z dynamicznym formularzem
   Użytkownik może dodać do systemu nową osobę, wybierając jej rolę. Formularz dostsowuje widoczne pola do wybranej roli – w przypadku lekarza pojawiają się dodatkowe pola na numer PWZ i specjalizację.
   Odpowiedzialne klasy:
   - MainWindow – obsługuje zdarzenie wyboru roli i steruje widocznością kontrolek. Pobiera dane z pól tekstowych.
   - KlinikaManager – przymuje nowy obiekt i po udanych sprawdzeniu unikalności dodaje go do listy.
   - Lekarz / Pacjent – konstruktory tych klas tworzą odpowiednie obiekty.



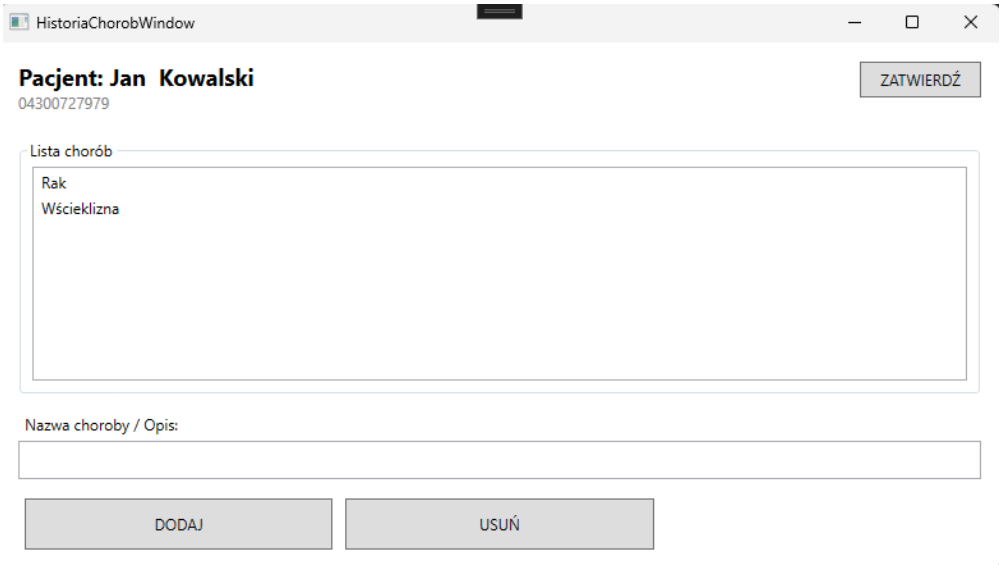
2. Walidacja poprawności danych
   System blokuje możliwość wprowadzenia błędnych danych. Weryfikowana jest suma kontrolna numeru PESEL, poprawność daty urodzenia oraz format numeru PWZ.
   Odpowiedzialne klasy:
   - Osoba – zawiera algorytm sprawdzania sumy kontrolnej PESEL.
   - Lekarz – weryfikuje numer PWZ za pomocą Regex
   - BlednaDataException / NiepoprawnyPeselException – klasy wyjątków sygnalizujące błędy użytkownikowu

3. Przeglądanie i sortowanie listy osób
   Dane wyświetlane są w tabeli za pomocą DataGrid. Użytkownik ma możliwość posortowania listy według wieku.
   Odpowiedzialne klasy:
   - MainWindow – wyświetla dane przy użyciu mechanizmu Data Binding
   - KlinikaManager – udstępnia metodę PobierzOsobyPosortowanePoWieku, która zwraca uporządkowaną listę.
   - WiekComparer – klasa pomocnicza implementująca interfejs IComparer, definiująca logikę porównywania dat urodzenia.

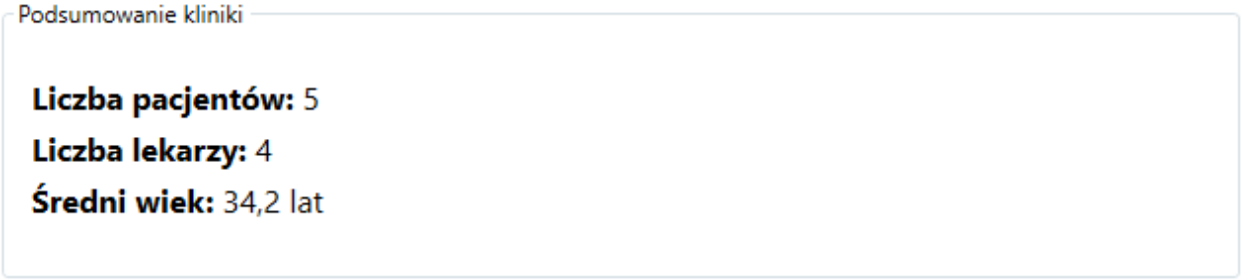
Typ	Imie	Nazwisko	PESEL	Wiek	Data Urodzenia	Specjalizacja
Pacjent	Maciej	Solejuk	08300531444	17	10/5/2008 12:00:00 AM	
Pacjent	Jan	Kowalski	04300727979	21	10/7/2004 12:00:00 AM	
Pacjent	Jarosław	Niewidomy	96110335682	29	11/3/1996 12:00:00 AM	
Lekarz	Mariusz	Kolano	78122965671	47	12/29/1978 12:00:00 AM	Pediatra
Pacjent	Adrian	Nowak	78122468839	47	12/24/1978 12:00:00 AM	
Pacjent	Stefan	Siarzewski	68091849296	57	9/18/1968 12:00:00 AM	
Lekarz	Ferdynand	Kiepski	66071672298	59	7/16/1966 12:00:00 AM	Internista
Lekarz	Grzegorz	Brzęczyszczykiewicz	64060618285	61	6/6/1964 12:00:00 AM	Kardiolog
Lekarz	Katrzyna	Nowak	52051563412	73	5/15/1952 12:00:00 AM	Dermatolog

4. Filtrowanie lekarzy według specjalizacji
   Pozwala na szybkie znalezienie lekarzy o określonej specjalizacji.
   Odpowiedzialne klasy:
   - KlinikaManager – metoda ZnajdzLekarzySpecjalistow przeszukuje listę i zwraca tylko obiekty typu Lekarz pasujące do kryterium
   - EnumSpecjalizacja – definiuje dostępne typy specjalizacji.

5. Zarządzanie historią chorób pacjenta
   Dla każdego pacjenta dostępny jest osobny moduł, w którym można dodawać przebyte choroby lub je usuwać.
   Odpowiedzialne klasy:
   - HistoriaChorobWindow – osobne okno WPF obsługujące interakcję z użytkownikiem
   - Pacjent – przechowuje liste chorób i udostępnia metody do jej modyfikacji.



6. Statystyki kliniki
   Aplikacja w czasie rzeczywistym oblicza i wyświetla kluczowe wskaźniki.
   Odpowiedzialne klasy:
   - KlinikaManager – metody obliczeniowe
   - MainWindow – odświeża te wartości na widoku po każdej operacji na danych.



7. Trwałość danych
   System umożliwia zapisanie stanu aplikacji do pliku XML oraz jego późniejsze wczytanie. Dzięki temu dane nie są tracone po zamknięciu programu.
   Odpowiedzialne klasy:
   - XmlSerializer – klasa systemowa użyta do serializacji listy obiektów
   - Osoba – klasa bazowa oznaczona atrybutami „[XmlInclude]”, co pozwala na zapis obiektów.