Zajęcia 3. Klasy pochodne `Informatyk`, `Lekarz`, `Nauczyciel`

3.1. Wstęp

Zaprojektowana do tej pory kolekcja typu **`List<Pracownik>`** służy do dokonywania różnych operacji na liście pracowników. Przechowywane obiekty typu `Pracownik` są jednak elementami ogólnymi (w pewnym sensie abstrakcyjnymi) z niewyszczególnionymi cechami typowymi dla konkretnych pracowników. Opisana w instrukcji rozbudowa projektu umożliwi zdefiniowanie bardziej precyzyjnych obiektów. Będą one reprezentowane przez typy: **`Informatyk`**, **`Lekarz`** oraz **`Nauczyciel`**, pochodne po klasie **`Pracownik`**, które będzie można zarówno dodawać, usuwać, sortować, wyszukiwać na liście jak i dokonywać przy ich użyciu operacji wejścia-wyjścia. Aby udostępnić taką funkcjonalność aplikacji, konieczne będzie poszerzenie projektu o wyżej wymienione klasy. Muszą one zostać umieszczone w projekcie **AplikacjaPracownicy**.

3.2. Dodanie do projektu klasy `Informatyk`

Klasa o nazwie **`Informatyk`** ma być klasą pochodną po klasie **`Pracownik`**. Dla tej klasy należy zdefiniować następujące składowe:

- Prywatne pola:
  - **`string adresEmail;`**
  - **`string stronaInternetowa;`**
- Publiczne właściwości:
  - **`AdresEmail`** zwracającą i ustawiającą wartość pola **`adresEmail`**.
  - **`StronaInternetowa`** zwracającą i ustawiającą wartość pola **`stronaInternetowa`**.
  - **`Zawod`** zwracającą składową **`Informatyk`** wyliczenia **`Zawody`**. Właściwość ma nadpisywać tę samą właściwość z klasy podstawowej (słowo kluczowe **`override`**).
- Publiczne metody:
  - bezparametrowy konstruktor inicjalizujący pola składowe zerami. W konstruktorze, na liście inicjalizacyjnej należy wywołać bezargumentowy konstruktor klasy bazowej.
  - konstruktor inicjalizujący wszystkie pola składowe na podstawie argumentów. Konstruktor ma posiadać te same argumenty co konstruktor klasy **`Pracownik`** i dodatkowe parametry potrzebne do zainicjalizowania pól składowych obiektu typu **`Informatyk`**).
  - konstruktor kopiujący inicjalizujący wszystkie pola składowe na podstawie argumentu wzorcowego typu **`Informatyk`**.
    Uwaga: W każdym konstruktorze, na liście inicjalizacyjnej, należy wywołać odpowiedni konstruktor klasy bazowej w celu ustawienia niedostępnych pól klasy bazowej.
  - **`override Pracownik Clone()`**, zwracającą instancję nowostworzonego obiektu typu **`Informatyk`** na wzór danego (wywołanie konstruktora kopiującego).
  - **`override string SzczegolyZawodu()`** zwracającą łańcuch w formie: adres email _s_ strona internetowa, gdzie _s_ jest znakiem separatora **`'\t'`**.
  - **`override string ToString()`** zwracającą łańcuch opisujący dane pracownika w formie: imię nazwisko dzień miesiąc rok ulica numer domu miasto adres email strona internetowa.
  - **`override string FormatWyjsciowy()`** zwracającą łańcuch opisujący dane pracownika w formie:
    Zawód: Informatyk
    Imię i nazwisko: imię nazwisko
    Data urodzenia: dzień miesiąc rok
    Adres zamieszkania: ulica numer domu miasto
    Dane dodatkowe: adres email strona internetowa
    Definicja tej metody wymaga odpowiedniego sformatowania informacji oraz, ze względu na odniesienie się do danych z klasy bazowej, wywołania metody **`FormatWyjsciowy`** z klasy **`Pracownik`**. Implementacja zaprezentowana jest na Listingu 3.1.

```
public override string FotrmatWyjsciowy()
{
   return String.Format("Zawód: {0}\n{1}Dane dodatkowe: {2}, {3}",
          this.Zawod, base.FotrmatWyjsciowy(), adresEmail, stronaInternetowa);
}
```

Listing 3.1: Definicja metody formatującej dane informatyka do wymaganej postaci.

- **override void OdczytConsole()**, której zadaniem jest wczytanie z klawiatury wszystkich danych dla informatyka. Obiekt typu **Informatyk** przechowuje również dane zawarte w typie **Pracownik** (odnoszące się do imienia, nazwiska, daty urodzenia i adresu zamieszkania) w związku z tym w definicji metody konieczne będzie wywołanie metody **OdczytConsole** z klasy bazowej. Metoda **OdczytConsole** zdefiniowana dla klasy **Informatyk** przesłania tę samą metodę w klasie **Pracownik** co umożliwia zastosowanie polimorfizmu.
- **override void ZapisConsole()**, której zadaniem jest wypisanie na ekran wszystkich danych informatyka. W przypadku definicji tej metody ma miejsce taka sama sytuacja jak podczas odczytu danych z konsoli: w trakcie wypisania na ekran konieczne jest uwzględnienie informacji o danych podstawowych informatyka (wywołanie metody **OdczytConsole** z klasy **Pracownik.**)

3.3. Dodanie do projektu klasy Lekarz

Klasa o nazwie **Lekarz** ma być klasą pochodną po klasie **Pracownik**. Dla tej klasy należy zdefiniować następujące składowe:

- Prywatne pola:
  - **string specjalizacja;**
  - **string tytul;**
- Publiczne właściwości:
  - **Specjalizacja** zwracającą i ustawiającą wartość pola **specjalizacja**.
  - **Tytul** zwracającą i ustawiającą wartość pola **tytul**.
  - **Zawod** zwracającą składową **Lekarz** wyliczenia **Zawody**. Właściwość ma nadpisywać tę samą właściwość z klasy podstawowej (słowo kluczowe **override**).
- Publiczne metody:
  - bezparametrowy konstruktor inicjalizujący pola składowe zerami. W konstruktorze, na liście inicjalizacyjnej należy wywołać bezargumentowy konstruktor klasy bazowej.
  - konstruktor inicjalizujący wszystkie pola składowe na podstawie argumentów. Konstruktor ma posiadać te same argumenty co konstruktor klasy **Pracownik** i dodatkowe parametry potrzebne do zainicjalizowania pól składowych obiektu typu **Lekarz**).
  - konstruktor kopiujący inicjalizujący wszystkie pola składowe na podstawie argumentu wzorcowego typu **Lekarz**.
    Uwaga: W każdym konstruktorze, na liście inicjalizacyjnej, należy wywołać odpowiedni konstruktor klasy bazowej w celu ustawienia niedostępnych pól klasy bazowej.
  - **override Pracownik Clone()**, zwracającą instancję nowostworzonego obiektu typu **Lekarz** na wzór danego (wywołanie konstruktora kopiującego).
  - **override string SzczegolyZawodu()** zwracającą łańcuch w formie: specjalizacja _s_ tytuł, gdzie _s_ jest znakiem separatora **'\t'**.
  - **override string ToString()** zwracającą łańcuch opisujący dane pracownika w formie: imię nazwisko dzień miesiąc rok ulica numer-domu miasto specjalizacja tytuł.
  - **override string FormatWyjsciowy()** zwracającą łańcuch opisujący dane pracownika w formie:
    Zawód: Lekarz
    Imię i nazwisko: imię nazwisko
    Data urodzenia: dzień miesiąc rok

Adres zamieszkania: ulica numer domu miasto
Dane dodatkowe: specjalizacja tytuł

- **`override void OdczytConsole()`**, której zadaniem jest wczytanie z klawiatury wszystkich danych dla lekarza. Uwaga: W przypadku pobrania z klawiatury informacji o lekarzu, należy uwzględnić jego dane podstawowe (podpunkt 3.3).
- **`override void ZapisConsole()`**, której zadaniem jest wypisanie na ekran wszystkich danych lekarza. Uwaga: W przypadku wyświetlenia informacji o lekarzu, należy uwzględnić jego dane podstawowe (podpunkt 3.3).

3.4. Dodanie do projektu klasy `Nauczyciel`

Klasa o nazwie **`Nauczyciel`** ma być klasą pochodną po klasie **`Pracownik`**. Dla tej klasy należy zdefiniować następujące składowe:

- Prywatne pola:
  - **`string przedmiot;`**
  - **`string tytul;`**
- Publiczne właściwości:
  - **`Przedmiot`** zwracającą i ustawiającą wartość pola **`przedmiot`**.
  - **`Tytul`** zwracającą i ustawiającą wartość pola **`tytul`**.
  - **`Zawod`** zwracającą składową **`Nauczyciel`** wyliczenia **`Zawody`**. Właściwość ma nadpisywać tę samą właściwość z klasy podstawowej (słowo kluczowe **`override`**).
- Publiczne metody:
  - bezparametrowy konstruktor inicjalizujący pola składowe zerami. W konstruktorze, na liście inicjalizacyjnej należy wywołać bezargumentowy konstruktor klasy bazowej.
  - konstruktor inicjalizujący wszystkie pola składowe na podstawie argumentów. Konstruktor ma posiadać te same argumenty co konstruktor klasy **`Pracownik`** i dodatkowe parametry potrzebne do zainicjalizowania pól składowych obiektu typu **`Nauczyciel`**).
  - konstruktor kopiujący inicjalizujący wszystkie pola składowe na podstawie argumentu wzorcowego typu **`Nauczyciel`**.
    Uwaga: W każdym konstruktorze, na liście inicjalizacyjnej, należy wywołać odpowiedni konstruktor klasy bazowej w celu ustawienia niedostępnych pól klasy bazowej.
  - **`override Pracownik Clone()`**, zwracającą instancję nowostworzonego obiektu typu **`Nauczyciel`** na wzór danego (wywołanie konstruktora kopiującego).
  - **`override string SzczegolyZawodu()`** zwracającą łańcuch w formie: przedmiot _s_ tytuł, gdzie _s_ jest znakiem separatora **`'\t'`**.
  - **`override string ToString()`** zwracającą łańcuch opisujący dane pracownika w formie: imię nazwisko dzień miesiąc rok ulica numer-domu miasto przedmiot tytuł.
  - **`override string FormatWyjsciowy()`** zwracającą łańcuch opisujący dane pracownika w formie:
    Zawód: Nauczyciel
    Imię i nazwisko: imię nazwisko
    Data urodzenia: dzień miesiąc rok
    Adres zamieszkania: ulica numer domu miasto
    Dane dodatkowe: przedmiot tytuł
  - **`override void OdczytConsole()`**, której zadaniem jest wczytanie z klawiatury wszystkich danych dla nauczyciela. Uwaga: W przypadku pobrania z klawiatury informacji o nauczycielu, należy uwzględnić jego dane podstawowe (podpunkt 3.3, 3.4).
  - **`override void ZapisConsole()`**, której zadaniem jest wypisanie na ekran wszystkich danych nauczyciela. Uwaga: W przypadku wyświetlenia informacji o nauczycielu, należy uwzględnić jego dane podstawowe (podpunkt 3.3, 3.4).

3.5. Metoda `Main`

W metodzie głównej **`Main`** klasy **`Program`** należy zdefiniować obiekt klas **`Informatyk`**, **`Lekarz`** i **`Nauczyciel`** i oraz wywołać wybrane metody na rzecz tych obiektów.