Programowanie w C#

Paweł Biesiada

Zadanie 1.3

- Napisz program, który znajduje najmniejszą liczbę Fibonacciego większą od:
  - 1000
  - **Parametru z args**
  - **1,1,2,3,5,8,13,21,34,55**

Zadanie 1.4

- Napisz program, który sprawdza czy zadana liczba jest liczbą pierwszą
  - Wydziel nową metodę, która zwraca wartość bool
  - Argumentem wejściowym jest rozpatrywana liczba

Zadanie 1.5

- Sprawdź ile dni minęło od dnia Twoich urodzin do teraz.
  - Wprowadź datę z konsoli (Console.ReadLine())
  - Wypisz datę urodzin (sam dzień) w formacie polskim
    - CultureInfo
  - Wynik wypisz do konsoli
  - **Policz czas również w godzinach.**

Zadanie 1.6

- Napisz program, który na podstawie następującego tekstu „*Lorem ipsum dolor sit amet, consectetur adipiscing elit.*”:
  - Zlicza liczbę wyrazów (Split)
  - Wypisuje fragment tekstu do przecinka (bez przecinka) (IndexOf, Substring)
    - wypisuje cały tekst za przecinkiem (bez przecinka)
  - Wypisuje tylko trzeci wyraz (Split)
  - Łączy w jeden string co drugi wyraz (Split, Concat)
  - Zlicza wystąpienie litery ‘e’ (foreach or Split, or IndexOf and Substring)

Zadanie 1.7

- Porównaj prędkość działania:
  - Konkatenacji stringów
  - StringBuilder'a

Zadanie 2.3

- Napisz cztery klasy o nazwach Employee, Secretary, Director oraz Programmer.
  - klasa Employee jest klasą bazową zawierającą właściwość Name oraz metodę CalculateSalary, która wypisuje „Obliczam wypłatę dla [nazwaOsoby]”
  - klasy pochodne nadpisują CalculateSalary() dopisując nową linię z wysokością pensji
  - ponadto niech Director zawiera metodę GetBonus() (Niech wypisze na konsolę „Wypłacam bonus.”)
  - Napisz metodę, która tworzy po 2 obiekty Secretary oraz Programmer i 1 obiekt Director. Zapisz je w dowolnej kolekcji
  - Napisz metodę, która jako parametr przyjmuje tą kolekcję i wykonuje dla wszystkich obiektów metodę CalculateSalary(), a dla Director również GetBonus();

Zadanie 2.5

- Przeciąż metodę porównującą obiekty (Equals()) tak, aby:
  - zwracała true, jeżeli porównywane obiekty typu Employee mają takie samo Id
  - W innych przypadkach powinna zwracać false
  - Pamiętaj o przeciążeniu GetHashCode()!

Zadanie 3.1a

- Wykorzystaj klasę Users, do następujących operacji wykorzystujących LINQ (pobierz kolekcję za pomocą CreateCollection.GetUsers():
  - Sprawdź czy kolekcja posiada jakikolwiek element (*Any*)
  - Policz wszystkie nie nullowe elementy (*Where, Count*)
  - Wypisz wszystkich użytkowników, których imię zaczyna się na literę “M” (Where)
  - wyciągnij 5 pierwszych użytkowników posortowanych po imieniu (*Take, OrderBy*)
    - wypisz drugą piątkę (*Take*, *Skip*)
  - Wypisz wszystkie imiona bez powtórzeń (Select + Distinct)
  - Wypisz wszystkie imiona, które się powtarzają (GroupBy)

Zadanie 3.1b

- Wykorzystaj klasę Users, do następujących operacji wykorzystujących LINQ (pobierz kolekcję za pomocą CreateCollection.GetUsers():
  - Wypisz tylko obiekty typu SuperUser (OfType)
  - Wypisz wszystkich użytkowników, którzy są aktywnymi administratorami (OfType, Where)
  - Pobierz tylko jednego użytkownika, który jest administratorem. (OfType, Where, FirstOrDefault)
  - Utwórz obiekt Dictionary<string, int> gdzie kluczem jest imię użytkownika, a wartością liczba użytkowników o takim samym imieniu (GroupBy, ToDictionary)

Zadanie 4.5

- Napisz program, który (SqlConnect):
  - Łączy się z bazą danych
  - pobiera listę grup oraz użytkowników
  - Nieaktywnych użytkowników zapisuje do pliku
  - Nieaktywnych użytkowników usuwa z bazy danych za
  - DELETE FROM Users WHERE Id =2
  - pomocą procedury składowanej (z parametrem Id)
  - Pobierz dane z wielu tabel (rozszerz podpunkt b) oraz wypisz je na konsolę

Zadanie 4.6

- Napisz metodę, która za pomocą EF wykonuje następującą akcję:
  - Dodaje nową grupę do bazy
  - Usuwa użytkowników nieaktywnych
  - Wypisz na konsole użytkowników i grupy do nich przypisane