Programowanie Obiektowe – C#

Zadania\_9

1. Zdefiniować klasę bazową Wielokąt i dziedziczące z niej klasy pochodne Prostokąt, Kwadrat, Trójkąt.

# Wielokąt

# Prostokąt

# Kwadrat

# Trójkąt

Do klas pochodnych wprowadzić odpowiednie składowe reprezentujące boki tych figur. W klasie bazowej zdefiniować abstrakcyjne funkcje wirtualne ObliczObwód oraz WyświetlDane i przeciążyć je w klasach pochodnych. Konstruktory klas pochodnych mają zapraszać do wprowadzania długości boków i wczytywać odpowiednie wartości. W programie głównym zadeklarować listę obiektów klasy Wielokąt i zrealizować obsługę następujących opcji:

pro - utworzenie nowego obiektu klasy Prostokąt i wpisanie jego referencji  
 do kolejnej pozycji tablicy,

kwa - utworzenie nowego obiektu klasy Kwadrat i wpisanie jego referencji   
 do kolejnej pozycji tablicy,

tro - utworzenie nowego obiektu klasy Trójkąt i wpisanie jego referencji   
 do kolejnej pozycji tablicy,

wys – wyświetlenie danych opisujących wszystkie figury,

obw - obliczenie i wyświetlenie sumy obwodów wszystkich figur zawartych w tablicy,

kon - koniec programu.

Zastosować kolekcję Dictionary przechowującą pary <nazwa\_opcji, delegacja>.

1. Opracować program, który zawiera hierarchię 2 klas reprezentujących dowolne obiekty świata rzeczywistego pozostające w relacji generalizacja – specjalizacja. Dla klas tych zdefiniować odpowiednie składowe, konstruktory i funkcję wirtualną Pokaz, która wyświetla wartości składowych na ekranie monitora. Funkcja wirtualna z klasy pochodnej powinna wywoływać funkcję wirtualną z klasy bazowej. Program główny zawiera tablicę obiektów klasy bazowej i udostępnia opcje umożliwiające wprowadzanie nowych obiektów każdej z tych klas i wyświetlanie opisów wszystkich obiektów wskazywanych przez poszczególne pozycje tablicy.
2. Opracować program, który zawiera definicję klasy Komputer o publicznych składowych Zegar [GHz] i Pamięć [GB]. Klasa Komputer ma zawierać konstruktor dwuargumentowy. W klasie tej zrealizować interfejs IComparable, czyli zdefiniować funkcję CompareTo służącą porównywaniu dwu obiektów klasy Komputer. Porównanie to ma przebiegać następująco:
   * jeżeli wartości składowej Zegar porównywanych obiektów różnią się o więcej niż 0,2 GHz to lepszy jest obiekt o większej wartości Zegara,
   * w przeciwnym przypadku porównywane są wartości składowych Pamięć obu obiektów.

W programie głównym utworzyć kolekcję List<Komputer>, wpisać do niej kilka przykładowych obiektów klasy Komputer, posortować te obiekty i wyprowadzić listę posortowanych obiektów.

UWAGA.

Jeżeli dla X lepszego od Y : X . CompareTo(Y) = 1, to po sortowaniu otrzymamy   
 kolejność od najgorszego do najlepszego

1. Opracować program symulujący loterię, w którą gra 3 graczy. Klasa Loteria udostępnia odpowiednią delegację i 3 zdarzenia L1, L2, L3. Każdy obiekt klasy Gracz rejestruje losowo swoją funkcję Zakład w jednym z tych zdarzeń. Następnie wykonywana jest funkcja Losowanie z klasy Loteria, która wybiera losowo jedno z wymienionych zdarzeń i zgłasza je. Gracze wygrywający otrzymują po1 punkcie – dodania punktu dokonuje funkcja Zakład. Program kończy się po uzyskaniu przez dowolnego gracza 100 punktów (wyświetlić wynik końcowy wszystkich graczy).
2. Zdefiniować klasę Student (Imię, Nazwisko, Album, RokUro, Semestr, Średnia) i klasę Wykładowca (Imię, Nazwisko, Semestr - przyjąć, że wykładowca prowadzi zajęcia na tylko jednym semestrze) oraz klasę Listy zawierającą kolekcję List<Student> oraz kolekcję List<Wykładowca>. Utworzyć obiekt klasy Listy i wczytać do niego dane z pliku SW.xml za pomocą serializacji XML. Następnie zdefiniować wyrażenia LINQ, które wyznaczają następujące dane (i wyznaczone dane wyświetlić):
3. pełne dane wszystkich studentów uporządkowane rosnąco wg. semestrów,
4. imiona i nazwiska studentów I stopnia studiów (sem. I do VII) uporządkowane malejąco wg. numerów albumów,
5. odpowiedź na pytanie, czy w zbiorze studentów (mężczyzn) są starsi niż 26 lat,
6. pełne dane studentek z VII semestru,
7. przeciętną średnią studentów II stopnia studiów (sem. VIII - X),
8. nazwiska i imiona studentów, z którymi prowadzi zajęcia wskazany wykładowca,
9. imię i nazwisko studenta o najwyższej średniej i studenta o najniższej średniej,
10. imiona, nazwiska i nr albumu studentów podgrupowane wg. numerów semestrów.

Ponadto zapisać wyrażenia LINQ, które:

1. zmieniają wszystkim studentom numery semestrów na o 1 większe (poza studentami X semestru - dla nich wpisać nr semestru -1),
2. zmniejszają studentom o podanym imieniu numer semestru o 1,

oraz wyświetlić zmodyfikowaną kolekcję danych studentów.

1. Opracować program kodujący/dekodujący plik tekstowy (zapytać o nazwę pliku) wczytywany do tablicy znaków. Kodowanie polega na zamianie kodu litery na kod litery następnej (z => a, Z => A, pozostałe znaki bez zmian), dekodowanie odwrotnie. Wykonanie każdej z tych operacji może nastąpić sekwencyjnie (w pojedynczym wątku) lub współbieżnie (za pomocą 4 współbieżnych wątków przetwarzających po 1/4 tablicy). Po zakończeniu operacji należy wyświetlić czas jej trwania i zapisać zakodowany/zdekodowany tekst do pliku wyjściowego (zapytać o nazwę pliku).
2. Opracować program gry Bieg, w której dwa pionki (o różnych kolorach) posuwają się z lewej do prawej strony okna konsoli (każdy w swoim wierszu). W każdym ruchu pionek przesuwa się o losową liczbę pozycji z przedziału (-8, 12). Zakończenie gry następuje, gdy jeden z pionków osiąga cel (kolumnę 60) - komunikat sygnalizujący zakończenie należy wyświetlić poniżej pionków stosując litery takiego samego koloru jak kolor pionka - zwycięzcy.

W programie należy zdefiniować klasę Pionek, która posiada konstruktor trójargumentowy (kolor, numer wiersza, obiekt synchronizujący) oraz funkcję Skok, która oblicza nową pozycję i wyświetla pionek synchronizując dostęp do konsoli (lock). Dla każdego pionka należy utworzyć odrębny wątek. Kolejne ruchy pionków powinny odbywać się co 500 ms.

1. Opracować program umożliwiający rysowanie w oknie konsoli za pomocą klawiszy strzałek. Po uruchomieniu programu kursor zostaje ustawiony na środku konsoli, a następnie użycie dowolnego z czterech klawiszy strzałek powoduje przesunięcie kursora w odpowiednim kierunku i zapełnienie pola znakowego:

* strzałka w górę : kolorem czerwonym,
* strzałka w dół : kolorem niebieskim,
* strzałka w lewo : kolorem zielonym,
* strzałka w prawo : kolorem żółtym.

Naciśnięcie klawisza Esc kończy działanie programu. Należy zapobiec przesunięciu kursora poza okno konsoli.

1. Opracować program, który realizuje operację ekstrakcji koloru zielonego z plików graficznych ( jeżeli dla danego piksela G > 1.5 \* R i G > 1.5 \*B, to należy go zastąpić pikselem 0,255,0 ; przeciwnie zastąpić pikselem 255, 255, 255). Przetwarzaniu mają podlegać wszystkie pliki zawarte w katalogu .\Pictures – pliki przetworzone należy zapisać w katalogu .\Green. Program wykonuje to zadanie dwa razy:  
   - sekwencyjnie, korzystając tylko z wątku głównego,  
   - współbieżnie powołując tyle wątków i jest plików do przetworzenia (wykorzystać   
    Parallel.ForEach).  
   W każdej z tych opcji dokonywany jest pomiar czasu przetwarzania – obliczony czas wyświetlany jest na monitorze. Ponadto należy wyświetlić liczbę działających współbieżnie procesorów dostępnych w danym komputerze.
2. Opracować program, który na podstawie tablicy T1, zawierającej 1M liczb typu double wygenerowanych losowo, wyznacza tablicę T2, której elementy przyjmują wartości:

T2[i] = sin2(T1[i] - 12,5) + cos2(T1[i] + 15,7)

Obliczenie wartości elementów tablicy T2 wykonać trzy razy:

a) sekwencyjnie za pomocą 1 wątku,

b) współbieżnie za pomocą 2 wątków przetwarzających połówki tablicy T1  
 (wykorzystać Parallel.For),

c) współbieżnie za pomocą 1M wątków przetwarzających tablicę T1

(wykorzystać Parallel.For).

Dla każdego przypadku zmierzyć czas wykonania obliczeń i wyprowadzić te czasy   
 wyrażone w milisekundach.

1. Opracować program, który dla każdego z plików zawartych w katalogu Dane wyznacza liczbę wyrazów zawierających przynajmniej jedną literę 'e', stosując odpowiednie wyrażenie LINQ. Po ustaleniu tej liczby jest ona wyprowadzana na monitor razem z nazwą pliku oraz identyfikatorem wątku. Obliczenie to wykonać dwukrotnie:

a) sekwencyjnie za pomocą 1 wątku,

b) współbieżnie za pomocą instrukcji Parllel.ForEach.

Dla każdego przypadku zmierzyć czas wykonania obliczeń i wyprowadzić te czasy   
 wyrażone w milisekundach.

1. Opracować program, który przetwarza współbieżnie tablicę CC zawierającą 1K liczb losowych, całkowitych, dodatnich, mniejszych od 1000 za pomocą 4 wątków, z których każdy modyfikuje kolejną ćwiartkę tej tablicy.

* ćwiartka 0 : wartości zamienić na przeciwne,
* ćwiartka 1 : do każdej wartości dodać 10000,
* ćwiartka 2 : od każdej wartości odjąć 10000,
* ćwiartka 3 : zastąpić wszystkie wartości liczbą 0.

Następnie wyprowadzić elementy tablicy CC po 16 w jednym wierszu oddzielając   
 ćwiartki wierszem pustym.

1. Wszystkie słowa z pliku testowego E wczytać do kolekcji List<string> (każde słowo ma w niej wystąpić tylko raz, zamienić duże litery na małe) a następnie, stosując wyrażenia LINQ, wyznaczyć:
2. wszystkie słowa zaczynające się na literę **s**, posortowane alfabetycznie – wynik wpisać pliku s-słowa.txt,
3. wszystkie słowa o liczbie liter równej 5 posortowane alfabetycznie – wynik wpisać do pliku 5-słowa.txt,
4. słowa o złożone z 5 liter i zaczynające się na **s** – wpisać do pliku s5-słowa.txt (operacja na zbiorach danych),
5. minimalną, średnią i maksymalną liczbę liter w słowach zaczynających się na **s** - wyświetlić,
6. słowa o najmniejszej liczbie liter i słowa o największej liczbie liter spośród słów zaczynających się na **s** - wyświetlić.