Biuro Podróży

1(2pkt). Stwórz abstrakcyjną klasę SrodekLokomocji.

a) dodaj w niej pola iloscMiejsc (int) oraz cenaBiletu(double) z mod. protected

b) dodaj konstruktor domyślny (pusty)

c) dodaj metodę wirtualną ObliczCene() typu void, bez parametru, która za pole cenaBiletu podstawia wartość 50,

d) dodaj zwykłą metodę zwracającą cenę biletu (można użyć właściwości choć nie trzeba)

2(2pkt). Stwórz klasę Autobus dziedziczącą z klasy SrodekLokomocji. W klasie potomnej wykonaj następujące czynności:

a) dodaj konstruktor parametryczny z parametrem typu int, parametr ma być podstawiony w pole iloscMiejsc, w konstruktorze należy wywołać metodę ObliczCene() – w wersji z klasy bazowej

b) przesłoń metodę ToString() tak, aby zwracała informacje o Autobusie np.

Autobus: ilość miejsc: 50, cena biletu: 50.

3(2pkt). Stwórz klasę Pociag dziedziczącą z klasy SrodekLokomocji. W klasie potomnej wykonaj następujące czynności:

a) dodaj pole dlugoscTrasy typu int – mod. protected

b) przesłoń metodę ObliczCene() następująco: jeśli długość trasy jest większa niż 100 za cenę biletu należy podstawić wartość dlugoscTrasy * 1,43; w przeciwnym wypadku należy wywołać zachowanie z klasy bazowej – czyli za cenę biletu podstawić wartość 50.

c) dodaj konstruktor parametryczny z dwoma parametrami (oba typu int), parametry należy podstawić odpowiednio do pól iloscMiejsc i dlugoscTrasy, w konstruktorze należy również wywołać metodę ObliczCene() – w wersji przesłoniętej

d) przesłoń metodę ToString() tak, aby zwracała informacje o pociągu:

Pociąg: ilość miejsc: 50, długość trasy: 200, cena biletu: 286.

4(2pkt). Stwórz klasę Express dziedziczącą z klasy Pociag. W klasie Express wykonaj następujące czynności:

a) przesłoń metodę ObliczCene() tak, aby podstawiała za cenę biletu zawsze 200.

b) przesłoń metodę ToString() tak, aby zwracała informacje o expressie:

Express: ilość miejsc: 50, długość trasy: 200, cena biletu: 200.

5(4pkt). W klasie Program i metodzie Main stwórz program testujący o następujących właściwościach:

a) stwórz kolejkę na obiekty typu SrodekLokomocji (ma być to kolejka generyczna)

b) dodaj na kolejkę po 3 obiekty klas utworzonych w punktach 2-4 (kolejność dowolna), można dorobić brakujące konstruktory, obiekty mają mieć tyle parametrów, aby można dla każdego z nich obliczyć cenę

c) oblicz i wyświetl na konsoli łączny koszt podróży (trzeba zsumować ceny wszystkich elementów na kolejce, ważne – po skończonej operacji nie należy usuwać niczego z kolejki)

d) wypisz na konsoli wszystkie elementy z kolejki kolejno usuwając je z kolekcji

6(2pkt). Stwórz klasę Podroz.

a) dodaj w tej klasie prywatne pole miasto typu string,

b) dodaj w tej klasie prywatne pole ileDni typu int

c) dodaj w klasie konstruktor parametrycznymi z  parametrami typu string i int, parametry trzeba ustawić odpowiednio  jako pole miasto, ile Dni.

d) przesłoń metodę ToString() tak aby zwracała informacje np.:

Miasto: Warszawa, liczba dni: 3.

7(2pkt) Podepnij do klasy Podroz interfejs IComparable<T> i dodaj w niej implementację metody CompareTo następująco:

- obiekty mają być najpierw sortowanie po nazwie alfabetycznie

- jeśli nazwy są identyczne – po liczbie dni: od największej do najmniejszej

8(2pkt) W klasie Program i instrukcji Main (oddzielone linijką komentarza od instrukcji z punktu 5) dodaj program testujący następująco:

a) stwórz listę z obiektami typu Podroz

b) dodaj na nią co najmniej 5 obiektów typu Podroz za pomocą konstruktora parametrycznego

c) posortuj listę używając funkcji Sort(), po tym wypisz elementy na konsoli w odwrotnej kolejności niż na liście po sortowaniu

9 (2pkt) Stwórz mechanizm głębokiej kopii dla klasy Podroz. Następnie skopiuj trzeci element na liście z obiektami typu Podroz i dodaj go na listę.

Max – 20 pkt.

bardzo dobry (5,0) - 91-100%;

dobry plus (4,5) - 86-90%;

dobry (4,0) - 75-85%;

dostateczny plus (3,5) - 70-74%;

dostateczny (3,0) - 50-69%;

niedostateczny (2,0) - 0-49%.