Rząd 1. Wypożyczalnia samochodów.

Pamiętaj: podstawą do rozpoczęcia sprawdzania jest poprawna kompilacja programu i przesłanie go na Githuba.

0. Stwórz projekt aplikacja konsolowa, zapisz go w folderze na Pulpicie, nazwij go Poprawa+swój numer indeksu np. poprawa130000.

1. Stwórz interfejs IKoszt i dodaj w nim metodę ObliczKoszt z parametrem typu int i zwracającą tym double.

2. Stwórz abstrakcyjną klasę Pojazd. W nowoutworzonej klasie wykonaj czynności:

a) dodaj pola marka (string), model (string), cenaZaDzien (double), dobierz modyfikator dostępu tak, aby pamiętać o hermetyzacji i pola były dostępne dla klas potomnych

b) stwórz konstruktor z parametrami (string, string, double), pobrane parametry ustaw jako odpowiednie pola w klasie

c) do klasy podepnij interfejs IKoszt i dodaj deklarację abstrakcyjnej metody ObliczKoszt

3. Stwórz klasę Osobowy dziedziczącą z klasy Pojazd. W nowej klasie wykonaj czynności:

a) dodaj prywatne pola klimatyzacja i kombi (oba tylko bool)

b) dodaj konstruktor z parametrami (string, string, double, bool, bool), pobrane parametry należy ustawić jako odpowiednie pola

c) przesłoń metodę ToString() tak, aby zwracała informacje o sam. osobowym np.

Osobowy, marka: Fiat, model: Panda, cena za dzień: 300, klimatyzacja: tak, kombi: nie.

d) przesłoń metodę ObliczKoszt następująco:

- jeśli samochód nie ma klimatyzacji i nie jest kombi, należy zwrócić liczbę wg wzoru (numer indeksu podzielony na 100+cenaZaDzien)

- jeśli samochód jest kombi należy pomnożyć cenaZaDzień przez pobrany parametr,

- w przeciwnym wypadku należy zwrócić 500.

4. Stwórz klasę Autobus dziedziczącą z klasy Pojazd. W nowej klasie wykonaj czynności:

a) dodaj prywatne pole liczbaMiejsc (int)

b) dodaj konstruktor z parametrami (string, string, double, int), pobrane parametry należy ustawić jako odpowiednie pola

c) przesłoń metodę ToString() tak, aby zwracała informacje o autobusie np.

Autobus, marka: MAN, model: Lion's Regio, cena za dzień: 400, ilość miejsc: 55.

d) przesłoń metodę ObliczKoszt następująco:

- jeśli autobus ma więcej niż 50 miejsc, to należy pomnożyć parametr przez 100

- w przeciwnym wypadku, należy pomnożyć pole cenaZaDzień przez pobrany parametr

5. W klasie Program i metodzie Main wykonaj czynności:

a) stwórz listę pojazdy na obiekty typu Pojazd

b) za pomocą konstruktora parametrycznego dodaj na listę pojazdy 10 różnych obiektów

c) wypisz na konsoli elementy, których indeks na liście jest nieparzysty

d) dla elementów których indeks jest parzysty wywołaj metodę ObliczKoszt wstawiając za parametr liczbę 7, otrzymane wartości wypisz na konsoli

6. Stwórz klasę Wypozyczalnia. W nowej klasie wykonaj czynności:

a) dodaj prywatne pola ileWolnych oraz ileAut (oba typu int)

b) dodaj konstruktor z parametrami (int, int), który parametry podstawi jako odpowiednie pola w klasie

c) podepnij interfejs IKoszt i zaimplementuj metodę ObliczKoszt następująco:

- jeśli ilość wolnych aut jest mniejsza lub równa połowie wszystkich aut (pole ileAut) to pobrany parametr należy pomnożyć raz 34;

- w przeciwnym wypadku należy zwrócić 734.

7. Do kodu z punktu 5c wykonaj następujące czynności:

a) stwórz 3 obiekty typu Wypozyczalnia za pomocą konstruktora parametrycznego

b) dla obiektów z punktu 7a wywołaj metodę ObliczKoszt, zwrócone wartości wyświetl na konsoli

8. Do klasy Pojazd podepnij interfejs IComparable<Pojazd> i zaimplementuj metodę do porównywania tak, aby odbywało się to po model odwrotnie alfabetycznie (Z->A).

9. Posortuj listę pojazdy stworzoną w punkcie 5. Posortowane elementy wypisz na konsoli.

10. Elementy z listy pojazdy skopiuj (poprzez dowolny mechanizm kopiowania) na generyczny stos. Następnie wypisz je na konsoli.

Punktacja

Polecenia 1-5 – łącznie 6 punktów

Polecenia 6-7 – łącznie 2 punkty

Polecenia 8-9 – łącznie 1 punkt

Polecenie 10 – 1 punkt

Aby otrzymać ocenę dostateczną na koniec, trzeba uzyskać min. 6 punktów.