Pociąg

1. Stwórz klasę Lokomotywa.

a) Dodaj w niej prywatne pola model (typ string) i masę (typ int).

b) Dodaj w niej konstruktor parametryczny ustawiający jednocześnie model i masę.

c) dodaj w niej metodę zwracają masę lokomotywy (typ zwracany int), metoda ma być bez parametru.

d) dodaj metodę Informacje() zwracającą string z informacjami o lokomotywie (jej model i masę). np.

Lokomotywa: model Pafawag 102E, masa: 80 000.

2. Stwórz abstrakcyjną klasę Wagon.

a) dodaj w niej pola model (typ string) i masę (typ int) z modyfikatorem protected.

b) dodaj deklarację abstrakcyjnej metody Informacje() zwracającą typ string;

c) dodaj deklarację i implementację zwykłej metody zwracającej masę wagonu (typ zwracany int).

3. Stwórz klasę Osobowy dziedziczącą z klasy Wagon.

a) dodaj w niej prywatne pole rodzaj (na przechowywanie informacji o rodzaj wagon np. przedziałowy, bezprzedziałowy, restauracyjny, kuszetka), typ string

b) dodaj konstruktor parametryczny ustawiający jednocześnie model, masę i rodzaj wagonu

c) dodaj implementację metody Informacje() tak, aby w jednej linii zostały zwrócone informacje o wagonie osobowy takie jak model, masa, rodzaj (oddzielone średnikiem lub przecinkiem). Np.
Wagon osobowy: model 112A, waga: 33500, rodzaj: przedziałowy, 1 klasa.

4. Stwórz klasę Towarowy dziedziczącą z klasy Wagon.

a) dodaj w niej prywatne pole ladunek (na przechowywanie informacji o ładunku przewożonym w wagonie, np. węgiel, paliwo, zboże), typ string

b) dodaj konstruktor parametryczny ustawiający jednocześnie model, masę i ładunek wagonu

c) dodaj implementację metody Informacje() tak, aby w jednej linii zostały zwrócone informacje o wagonie osobowy takie jak model, masa, ładunek (oddzielone średnikiem lub przecinkiem). Np.
Wagon towarowy: model 401Z, waga: 19500, ładunek: węgiel, ruda żelaza.

5. Stwórz w projekcie dwa interfejsy:

a) IPoprawnyPociag w którym będzie deklaracja metody MozeJechac() bez parametru zwracającej typ bool,

b) IUzupelnijSklad w którym będzie deklaracja metod typu void z następującymi parametrami (kolejność może być inna, ale później należy zachować konsekwencję):
DodajOsobowy(int masa, string model, string rodzaj)
DodajTowarowy(int masa, string model, string ladunek)
DodajLokomotywe(int masa, string model)

6. Dodaj w projekcie klasę Pociag i podepnij do niej stworzone interfejsy w poleceniu nr 5.

a) stwórz w klasie prywatne pola: wagony typu List<Wagon> oraz lokomotywy z typie List<Lokomotywa>

b) zainicjuj listy w dowolny sposób

c) dodaj implementację metod z interfejsów

- metoda MozeJechac() zwraca true jeśli masa wszystkich lokomotyw jest większa lub równa masie wszystkich wagonów oraz w składzie pociągu jest co najmniej jedna lokomotywa; w przeciwnym wypadku zwraca false;

- metody DodajOsobowy i DodajTowarowy dodają odpowiedni wagon na pole wagony w tej klasie

- metoda DodajLokomotywe dodaje lokomotywę na pole lokomotywy w tej klasie.

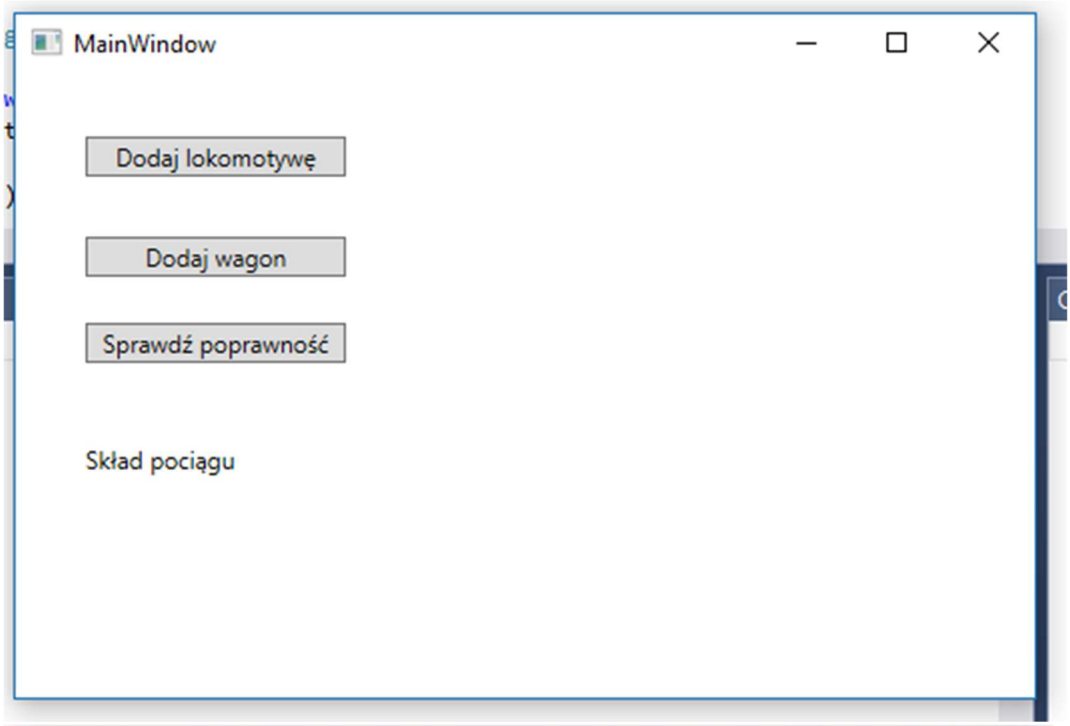
d) dodaj w tej klasie metodę Informacje(), która zwraca string z informacjami o składzie pociągu, przy czym informacje o kolejnych lokomotywach/wagonach mają być w kolejnych wierszach. Np.

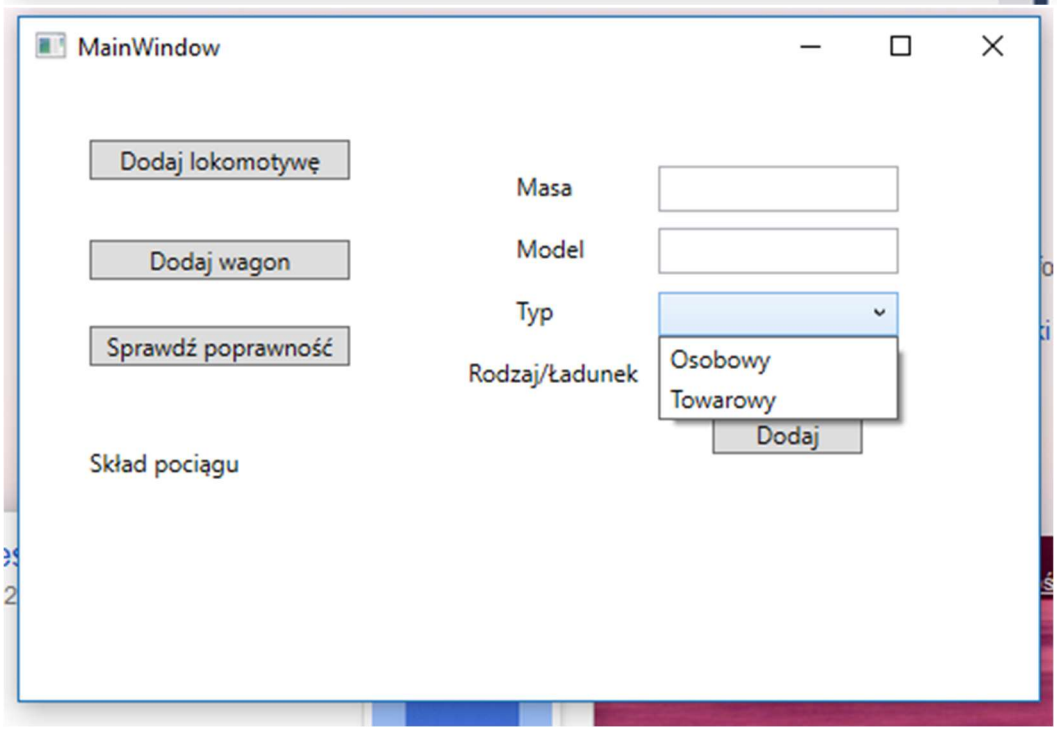
Skład pociągu:

Lokomotywa: model Pafawag 102E, masa: 80 000.
Wagon towarowy: model 401Z, waga: 19500, ładunek: węgiel, ruda żelaza.
Wagon osobowy: model 112A, waga: 33500, rodzaj: przedziałowy, 1 klasa.

Nie ma potrzeby sortowania elementów(!).

7. Zaimplementuj aplikację WPF do przetestowania kodu. Przykładowe okna:





Uwaga: aplikacja powinna poprawnie obsłużyć możliwe wyjątki przy konwersji string na liczbę oraz to, że masa powinna być liczbą dodatnią.

Punktacja:

Podpunkt 1 – 1pkt

Podpunkt 2 – 1pkt

Podpunkt 3 – 1pkt

Podpunkt 4 – 1pkt

Podpunkt 5 – 1pkt

Podpunkt 6 – 2pkt

Podpunkt 7 – 2pkt

Poprawna obsługa wyjątków – 1pkt