dr. inż. Janusz Rafałko

Kontrola lotów

Przygotuj następującą hierarchię wyjątków:

1. WyjatekTransportu dziedziczący z Exception
2. WyjatekBrakuPaliwa i WyjatekBiznesowy dziedziczą z WyjatekTransportu

Wszystkie utworzone wyjątki posiadają konstruktor z parametrem typu String, w którym przekazują krótki opis powstałego błędu.

Przygotuj interfejs Pojazd, który posiada deklarację następujących metod:

- getIloscPaliwa() wyrzucający wyjątek WyjatekBrakuPaliwa,
- zuzyciePaliwa(), gdzie parametrem będzie ilość traconego paliwa.

Przygotuj następujący zestaw klas:

1. Klasa Samolot implementująca interfejs Pojazd.
2. Klasa SamolotPasazerski dziedzicząca po klasie Samolot.
3. Klasa KomputerKontrolny

Dobierz modyfikatory klas w taki sposób, aby: nie można było utworzyć żadnych obiektów klasy Samolot i nie można było dziedziczyć od SamolotPasazerski.

Pola i metody następujących klas:

1. Samolot:
    - pola: iloscPaliwa, iloscZalogi, miejsceOdlotu, czasPrzylotu (tabela dwuelementowa składająca się z godziny oraz minut),
    - metody:
        - odpowiednie gettery oraz toString(),
        - metoda getIloscPaliwa() wyrzuca wyjątek z informacją: "Samolot został skierowany na lądowanie awaryjne na najbliższe lotnisko", gdy ilość paliwa spadnie do wartości wynoszącej minimum 50,
        - konstruktor parametryczny, do którego przekazujemy wartości wszystkich zmiennych. Wyrzuca wyjątek WyjatekTransportu gdy ilość załogi składa się z mniej niż dwóch osób (informacja: „Wczoraj były urodziny kapiutana, nie polecimy”) lub WyjatekBrakuPaliwa, gdy ilość paliwa jest mniejsza niż 200 (info: „Mamy wyciek paliwa, bez paniki proszę”).
2. SamolotPasazerski:
    - pola: maxIloscMiejsc, liczbaPasazerow,
    - metody:
        - odpowiednie gettery oraz toString(),
        - konstruktor parametryczny, do którego przekazujemy wartości wszystkich zmiennych. Wyrzuca wyjątek WyjatekBiznesowy z odpowiednią informacją, gdy liczba pasażerów jest mniejsza niż połowa dostępnych miejsc.

3. KomputerKontrolny:
    - metody:
        - główna metoda aplikacji, czyli main().

W funkcji main() należy:

1. Utworzyć tablicę typu String zawierającą nazwy kilku miast.
2. Utworzyć kolekcję Map, której kluczem jest kolejny nr samolotu, a wartością sam samolot.
3. Utwórz kolekcję (lista), w której elementami będą samoloty.
4. Włożyć do kolekcji Map 10 samolotów pasażerskich (pętla).
   Uwagi:
    - ilość paliwa w samolocie wynosi max 500,
    - ilość załogi max 10,10
    - miejsce odlotu wylosować z tablicy utworzonej w podpunkcie 2,
    - czas przylotu ma się mieścić w przedziale czasowym 20:00 - 24:00,
    - maxIloscMiejsc wynosi 300
    - iloscPasazerow losowa (do 300);
    - W sekcji catch() obsługi wyjątków należy wyświetlić odpowiedni komunikat i nie tworzyć danego samolotu (pamiętając, że samolotów **łącznie ma być 10**).
5. Elementami z kolekcji Map uzupełnij kolekcję utworzoną w podpunkcie 3. Użyj do tego **foreach**.
6. W pętli przeprowadzić symulację przyjmowania samolotów (Zagnieżdżona pętla: dopóki jakiś samolot jest jeszcze na liście, przetwarzamy wszystkie)
   Należy (używając utworzonej listy i iteratora listowego):
    a. w każdej iteracji,
    - dla wszystkich samolotów sprawdzić aktualny stan paliwa i obniżyć go o losową wartość od 50 do 200. Gdy komuś zabraknie paliwa (wyjątek) należy usunąć go z listy,
    - wypisać informację o samolotach

Punktacja:

- Przygotowanie hierarchii wyjątków i interfejsu - 1 pkt
- Przygotowanie klas - 4 pkt
- Funkcja main - 5 pkt