Programowanie Obiektowe
Kolekcje, wyjątki i generyki
Zadanie oceniane nr 2a
25-04-2023

Po zakończeniu pracy konieczne jest wgranie zmian do repozytorium (add + commit + push) w katalogu o nazwie w stylu: zadanie_oceniane_2a. Fakt wgrania plików do swojego repozytorium można sprawdzić samodzielnie logując się (via www) na swoje konto i sprawdzając czy pojawiły się tam wszystkie zmiany.

"Boarding"

W dniu dzisiejszym oprogramowanie które stworzymy symulować będzie process obsługi samolotu pasażerskiego. W użyciu będą kolekcje, mapy ale **nie "surowe" tablice**. Zatem lecimy...

Prace do wykonania:

1. **Stworzyć klasy i ich hierarchie odzwierciedlające wspomniane poniżej elementy (nazewnictwo dobrać wedle uznania)**

   - Pilot (staż pracy 5-20, nalot 2000-5000)
   - Steward (staż pracy 1-3)
   - Pasażer (id, bilet)

     oraz

   - Bagaż główny (masa 5-23)
   - Bagaż podręczny (masa 5-23)

     Bilet posiada informacje o miejscu: rząd (np. 19) oraz miejsce (np. "D").

     Należy stworzyć generator biletów używany podczas inicjalizacji pasażera, dla którego stworzony zostanie bilet wskazujący na losowe miejsce z puli dostępnej w samolocie (opisane dalej). Oczywiście dwa wygenerowane bilety nie mogą wskazywać na to samo siedzisko i to jest ich unikalną cechą, więc trzeba o to jakoś zadbać. Losowanie kolejnej wejściówki do skutku, aż uzyskamy miejsce też nie jest tu porządanym rozwiązaniem więc trzeba to zrobić lepiej.

     Swóch pasażerów jest tożsamych ze sobą jeśli mają to samo id. W pozostałych przypadkach zakładamy że dwa fizyczne obiekty reprezentują zawsze dwie różne osoby. Podobnie z bagażami.

   - Samolot
     Jest on zbudowany w sposób następujący:
     - ma 25 sześcioosobowych rzędów (dostępnych po indeksie, bez usuwania). Każde miejsce w rzędzie na którym można ulokować pasażera jest powiązane z jedną z liter: "A", "B", "C", "D", "E", "F".
     - dwa miejsca dla pilotów jako zwykłe pola
     - cztery miejsca dla stewardów którzy są dostępni po indeksie i będą często usuwani
     - ciasny i ciężko dostępny luk do którego trafiają bagaże wszystkich rodzajów (w przypadku rozładowywania luku pierwszy jest wyjmowany bagaż który był tam załadowany jako ostatni).

     Samolot samodzielnie obsadza na miejscach pilotów i stewardów oraz inicjalizuje miejsca do siedzenia.

     Należy w dobrze znany sposób umożliwić wystawienie na zewnątrz następujących metod:

     - ✔ wstawiaBagaz
       Wstawia bagaż jednocześnie zapamiętując sumaryczną wagę wszystkich wstawionych do tej pory, umożliwiając jej odczyt innym klasom.

     - ✔ boardPassenger
       przyjmuje jako argument pasażera, rząd i miejsce w samolocie, a następnie usadza tam tego człowieka wypisując na konsoli informację kogo i gdzie

posadzono. Zakładamy że przypadek w którym miejsce jest zajęte to jest sytuacja która może mieć miejsce ale metoda nie poczuwa się (bo w tej sytuacji nie musi) do kontynuacji działania i powinna to konkretnie w dobrze znany sposób zasygnalizować.

- ✔ containsPassenger
  Sprawdza czy pasażer przekazany jako argument jest w samolocie

- Terminal
  Posiada w sobie kolekcje pasażerów oraz bagaży, które nie mogą się powtarzać, a w przypadku tych pierwszych liczy się jeszcze kolejność ich dołączenia. Umożliwia też dodawanie podróżnych i bagaży z zewnątrz. Oprócz tego jest tam Humanistyczny inżynier załadunku i zwykły inżynier załądunku opisani dalej.

- Inżynier załadunku
  Siedzi w terminalu i ma dostęp do kolekcji bagaży.
  Wstawia bagaże główne jak i podręczne do luku w kolejności od najcięższego do najlżejszego po wywołaniu na nim metody: zaladujBagaze(). Projektant Terminala powinien swoimi decyzjami maksymalnie ułatwić pracę inżynierowi załadunku. Oczywiście w przyrodzie nic ginie, bagaż załadowany do samolotu znika w terminalu.

- Humanistyczny inżynier załadunku
  Ma dostęp do kolekcji pasażerów czekających na boarding oraz do czegoś co pewnie jest samolotem (ale w to nie wnika). Interesuje go jedynie że to coś ma metodę boardPassenger, którą wykorzysta. Reszta jest nieistotna.
  Potrafi zrobić to samo co Inżynier Załadunku + posiada metodę rozsadzLudzi(), która ma dostęp do kolekcji pasażerów będących w terminalu, iteruje ją, sprawdza każdemu bilet i na jego podstawie wywołuje metodę boardPassenger, przekazując do niej wszystkie potrzebne elementy. Wie że metoda ta czasem może w wiadomy sposób odmówić pracy, więc obsługuje tą sytuację i na konsolę wypisuje detale dotyczące trefnego biletu. Sprawa jest zamknięta na miejscu i nie propaguje się dalej. Pasażer, który wsiadła do samolotu znika w Terminalu.

2. **Uczynić Samolot parametryzowalnym**
Dodać do Samolotu pole z obiektem środka przymusu bezpośredniego o klasie/typie który nie może być znany w momencie implementacji ale będzie on wskazany dopiero podczas tworzenia maszyny. W momencie instancjonowania Samolotu zakładamy że typem tym będzie obiekt klasy stworzonej klasy Gazrurka spoza hierarchii, który to utworzy konstruktor samolotu. Poprawić odpowiednie miejsca w kodzie (łącznie ze światem).

3. **Zademonstrować działanie**
Stworzyć uruchamialną klasę Świat posiadającą metodę go(), która tworzy Samolot, Terminal oraz dodaje do niego 120 pasażerów oraz 50 bagaży podręcznych i 50 głównych. Następnie uruchamia inżynierów którzy wykonują swoją robotę.