Programowanie Obiektowe
Kolekcje, wyjątki i generyki
Zadanie oceniane nr 2c
8-05-2023

Po zakończeniu pracy konieczne jest wgranie zmian do repozytorium (add + commit + push) w katalogu o nazwie w stylu: zadanie_oceniane_2c. Fakt wgrania plików do swojego repozytorium można sprawdzić samodzielnie logując się (via www) na swoje konto i sprawdzając czy pojawiły się tam wszystkie zmiany.

"Konduktor"

W dniu dzisiejszym oprogramowanie które stworzymy symulować będzie process sprawdzania biletów w jednym wagonie składu pasażerskiego. W użyciu będą kolekcje, mapy ale **nie "surowe" tablice**. Odjazd!

Prace do wykonania:

1. **Stworzyć klasy i zorganizować je w sensowne hierarchie odzwierciedlające wspomniane poniżej elementy (nazewnictwo dobrać wedle uznania)**

   - Podróżny (imię, nazwisko, bilet, stopień wygłodzenia (1-10)
     getBilet() - zwraca bilet jaki posiada;

     Podróżny przy instancjonowaniu losuje sobie imię i nazwisko z ogólnie dostępnych (tylko dla obiektów tej klasy) niezmienialnych kolekcji. Powinny zawierać odpowiednio 10 imion i 10 nazwisk. Ma on też dostęp (via konstruktor) do opisanego dalej systemu sprzedaży biletów (w zasadzie do jednej metody) i za jego pośrednictwem kupuje sobie bilet. Jeden obiekt = jeden pasażer.

   - Konduktor
     coś co przechowuje relację Pasażer – Bilet dla biletów które podczas kontroli okazały się nieważne (gdy złapie osobę na takim bilecie to ją tu dodaje); kasownik do biletów – nie wnika jaki dokładanie. Coś co umie sprawdzać bilety, a jak to robi to już konduktor nie wnika. To coś jeśli się trafi na konkretny model może sprawiać problemy i czasem odmawiać współpracy – więc trzeba to brać pod uwagę

   - Wagon (podróżni) – jest to wagon restauracyjny, więc nie ma przedziałów. Są za to stoliki, których zawsze jest tyle samo (8). Podróżni są przechowywani w czymś co trzyma relację nazwy stolika (Stolik_1, Stolik_2, ...) z kolekcją nie pozwalającą na powtórzenia w której podróżni są z automatu sortowani malejąco po stopniu wygłodzenia;Wagon podczas inicjalizacji przyjmuje podróżnych z zewnątrz i rozsadza ich przy stolikach zgodnie z ich biletami.
     getPasazerowiePrzyStoliku(...) - zwraca wszystkich pasażerów siedzących przy danym stoliku o nazwie przekazanej jako parametr. Skoro stolików jest zawsze tyle samo, to zakładamy, że ilość ich nazw zawsze będzie taka sama i się nigdy nie zmieni. Chcielibyśmy łatwo ekstrachować numer z nazwy stolika.

   - Kasownik do biletów (offline)
     sprzedane bilety – indeksowana kolekcja z dużą ilością tanich operacji usuwania;
     sprawdzone bilety – kolekcja nie pozwalająca na powtórzenia;
     boolean walidujBilet(Bilet bilet) – sprawdza czy przekazany jako argument bilet znajduje się w wśród sprzedanych biletów. Jeśli tak to przenosi go z tamtąd do kolekcji sprawdzonych biletów;
     wgrajInformacjeOSprzedanychBiletach(...) – wstawia bilety z przekazanej kolekcji do kolekcji sprzednych biletów usuwając wszystko co było tam wcześniej. Informację o tym czerpie z systemu do sprzedaży biletów do którego ma dostęp.

   - Kasownik do biletów (online)
     Ma dostęp do systemu sprzedaży biletów (wyobraźmy sobie że zdalny), a właściwie do czegoś co potrafi sprawdzić bilet i odpowiedzieć na pytanie czy faktycznie został on wystawiony i oznaczyć go jako użyty.
     boolean walidujBilet(Bilet bilet) – metoda używa do walidacji systemu do sprzedaży biletów do którego ma dostęp. Raz na jakiś czas (z P=0.02) występuje dłuższy problem z połączeniem (może się to zdarzyć bo pociąg czasem jeździ trasami z

bardzo niestabilnym zasięgiem GSM) co zwalnia tą metodę z konieczności kontynuowania procesu online`owej walidacji biletu. Oczywiście powinno to być zasygnalizowane w dobrze znany sposób.

- Bilet
  imię nabywcy,
  nazwisko nabywcy,
  relacja pociągu (WARSZAWA-ZAKOPANE lub ZAKOPANE-WARSZAWA),
  nazwa stolika,
  skasowany?,
  kod zabezpieczający (jakiś nietrywialny hash z połączonego imienia i nazwiska nabywcy)
  Wszystko poza kodem zabezpieczającym pochodzi z zewnątrz.
  W zasadzie to biletem jest bardziej informacja na nim zawarta niż obiekt który ją przechowuje. Zakładamy że dwa bilety są takie same jeśli mają tą samą relację pociągu i tą samą nazwę stolika.

- System sprzedaży biletów
  kolekcja sprzedanych biletów – niepozwalająca na powtórzenia kolekcja pamiętająca kolejność wstawienia biletów,
  Bilet sprzedajBilet(imię, nazwisko, relacja pociągu) – zwraca bilet i zapisuje go w sprzedanych biletach. Losuje nazwę stolika z istniejących i skończonych. Jest to metoda wystawiona tylko dla pasażera;
  boolean zweryfikujBilet(Bilet) - sprawdza czy bilet znajduje się w kolekcji wystawionych biletów. Jeśli tak oznacza go jako skasowanego.
  ... getSprzedaneBilety() - jest to metoda wystawiona tylko dla kasownika offline`owego. Zwraca sprzedane bilety żeby mógł on je weryfikować.

- ✔ Działania konduktora (metoda go())
  Ma on dostęp do wagonu reastauracyjnego wypełnionego pasażerami. Zna wszystkie nazwy stolików więc stolik po stoliku wyciąga z wagonu kolekcje pasażerów przy nich siedzących. Od każdego pasażera pobiera on jego bilety i sprawdza je za pomocą urządzenia jakie mu się w dniu dzisiejszym trafiło. W przypadku gdy urządzenie odmówi współpracy, kontynuje swoje zadania pozostawiając stosowny komentarz na konsoli. Kasownik może zasygnalizować, iż bilet nie jest ważny. Wtedy taki pasażer jest usuwany z kolekcji wraz ze stosownym komunikatem na konsoli. Po zakończonej kontroli konduktor wypisuje na konsolę informaję o ilości nieważnych biletów.

2. **Uczynić Kasownik online`owy parametryzowalnym**
Umożliwić sparametryzowanie kasownika online`owego poprzez przekazywany typ przyjmowany w konstruktorze, którym jest coś co generuje komunikat (jako swoją reprezentację tekstową) pojawiający się w momencie gdy bilet okazje się nieważny. Tekst to: "To jest szwindel!".

3. **Zademonstrować działanie**

Stworzyć potrzebne komponenty (w tym kasownik ofline`owy zsynchronizawany z systemem biletowym oraz 50-ciu pasażerów. Zapakować ich do wagonu i poprosić konduktora żeby robił swoje.