Programowanie obiektowe – zadanie oceniane (2/4)

23-04-2018

Należy wysłać rozwiązanie w 90 minut po rozpoczęciu.
Liczba punktów do zdobycia: 25.

1. Wysyłanie rozwiązania

Należy spakować projekt w archiwum zip i nadać mu nazwę: login.zip
Należy wysłać email:
To: bednarzm@student.mini.pw.edu.pl
Subject: [PO] Zadanie2/4 2018
Załączniki: login_studenta.zip

2. Opis ogólny

Należy stworzyć aplikację symulującą statek pasażerski. Do stworzenia będzie hierarchia klas będąca prostym modelem tego typu środka komunikacja. Do tego należy dostarczyć elementy symulujące procesy zachodzące na statku.

3. Opis szczegółowy

Część A (17 pkt).
Celem tego etapu jest stworzenie aplikacji posiadającej hierarchię potrzebnych klas będącej prostym modelem statku, zdolnej do zainicjowania samej siebie i wypełnienia pasażerami.

Statek zawiera następujące elementy:

- Statek
  Klasa zawierająca wszystkie elementy takie jak:
  - ➔ Listę pasażerów – jest to lista, która powinna zawierać wszystkich pasażerów, który wsiedli na statek.
  - ➔ Kapitan statku
  - ➔ Kilku oficerów – nie ma znaczenia ich kolejność, nie mogą się dublować
  - ➔ Pewną ilość pięter – na których znajdują się kajuty. Jest ich 7.
  - ➔ Pokład – mogą podróżować na nim podróżni nie mający biletu do kajut.
  - ➔ Maszynownię – nie wiadomo co to za klasa i jaki napęd będzie miał on teraz i w przyszłości. Jest to opcja konfigurowalna, zatem Statek ma być klasą generyczną, która jest parametryzowana odpowiednim interfejsem. Wiadomo jednak, że jest to coś, co implementuje interfejs dziedziczący po interfejsie zawierającym dwie metody:
    - plyn()
    - stoj()

- Maszynownia
  Jest to obiekt, który implementuje interfejs zawierający dwie metody:
  - ➔ plyn()
  - ➔ stoj()

  W tej części zadania w przypadku zawołania metody: plyn() na konsoli wypisywany jest tekst: "Płynę". Metod stoj() wypisuje tekst: "Stoję".
- Piętro
  Zawiera elementy takie jak:
  - ➔ Pewna ilość przedziałów (10-20) w których siedzieć będą pasażerowie, konieczna jest możliwość dostępu do danego przedziału po indeksie. Pojemność każdego przedziału to 2 lub 4 miejsca. To piętro przy tworzeniu tych obiektów ma decydować, ile osób wejdzie do każdego z nich. Wszystkie kajuty na danym piętrze mają taką samą pojemność.

  Powinien umożliwiać:
  - ➔ Dostęp do kajut
  - ➔ Informacje o ilości pasażerów aktualnie nim podróżujących
    Informacje o ilości bagażów znajdujących się w kajutach
- Kajuta
  Jest to element piętra.
  Zawiera:
  - ➔ od 2 do 4 miejsc. Miejsca mają być podatne na indeksowanie, tzn. musi istnieć możliwość pobrania pasażera po danym indeksie.

- Pokład
  Jest w stanie pomieścić nieograniczoną liczbę Pasażerów. Używa kolekcji nie pozwalającej

na dodawanie zdublowanych Pasażerów i zapamiętującej ich kolejność.

- Kapitan
  Obiekt ten posiada następujące cechy:
  - ➔ Ma dostęp do pięter, kajut oraz pasażerów.
  - ➔ Dokonuje załadunku pasażerów (losuje piętro, daje pasażerowi referencję na nie i wywołuje motodę szukaj miejsca)
  - ➔ Każdy pasażer z wyjątkiem co dziesiątego zostaje wpisany na listę pasażerów

- Oficer
  Chodzi po pietrach (listuje je) i sprawdza, czy każdy napotkany pasażer znajduje się na liście pasażerów. Jeśli znajdzie jednego, którego nie ma na tej liście, to jest to sytuacja, której nie oczekiwano i musi być odpowiednio obsłużona.

- Pasażer
  Jest to obiekt, który podróżuje statkiem.

- Nabrzeże
  Obiekt zawiera 500 pasażerów, który chcą płynąć statkiem. Nie może być tam zdublowanych pasażerów ale musi być zachowana kolejność ich dodania do nabrzeża. Nabrzeże samo produkuje pasażerów.

Akcja:

- Należy zainicjować statek zgodnie z w/w wytycznymi (statek ma Kapitana, kilku oficerów (kolejność dodania nieistotna), piętra, na których są kajuty. Kapitan listuje kolejno wszystkich pasażerów, którzy są na nabrzeżu w kolejności w której zostali do niego dodani. Nie może być zdublowanych pasażerów.
- Kapitan pobiera kolejno pasażerów z nabrzeża i każe im poszukać sobie miejsca poprzez wywołanie metody na każdym z pasażerów, której argumentem jest referencja na wylosowane wcześniej Piętro i Pokład. Każdego pasażera dodaje do Listy Pasażerów. Niestety co 20-ty przez pomyłkę nie zostaje tam dodany.
- Pasażer, któremu Kapitan kazał poszukać sobie miejsca (po przekazaniu mu referencji na piętro statku):
  - listuje Koje na piętrze i w każdej sprawdza, czy jest tam wolne miejsce (liczba siedzących w przedziale jest mniejsza niż jej pojemność).
  - Jeżeli znalazła się koja w którym jest wolne miejsce, to "dodaje się" do niej.
  - Jeżeli przelistuje wszystkie koje na tym piętrze i nie znajdzie żadnej wolnej, to dodaje się do Pokładu i tam przebywa.

Część B (4 pkt).
Celem niniejszego fragmentu jest ożywienie statku i implementacja naturalnych procesów, które mu towarzyszą. Powinien być on już załadowany i gotowy do rejsu.

Kapitan (po inicjalizacji i załadowaniu statku pasażerami) wykonuje następujące akcje:

- Uruchamia statek wywołując metodę plyn() na interfejsie implementowanym przez coś, co ma być Maszynownią.
  - Raz na 20 wywołań metody plyn() zdarza się sytuacja, która nie powinna się wydarzyć (której Kapitan normalnie nie oczekuje, aczkolwiek się jej spodziewa):
    - Awaria
    - Operator Maszynowni (ten obiekt nie musi istnieć) nie jest w stanie jej uruchomić. Chodzi tylko o komunikat.

Wtedy Kapitan wypisuje na konsoli tekst: "Koniec rejsu! Dryfujemy.".

- Jeżeli udało się uruchomić Statek, to losuje oficera, któremu daje dostęp do pięter i Listy Pasażerów, wywołując na nim metodę: sprawdzajPasazerow. Oficer chodzi po wszystkich piętrach (listuje piętra), a na każdym piętrze odwiedza wszystkie Koje. Sprawdza, czy napotkani pasażerowie znajdują się na Liście Pasażerów. Jeżeli znajdzie się pasażer, którego nie ma na Liście, to jest to sytuacja, która nie jest oczekiwana. Sterowanie wraca do Kapitana, który obsługuje tą niestandardową sytuację: dodaje pasażera do Listy Pasażerów. Następnie ponownie wysyła on Oficera na kontrolę (od początku). Kapitan zapamiętuje gdzieś fakt, że takiego a takiego Pasażera nie było na Liście Pasażerów. Jak Oficer odwiedzi wszystkie piętra, to to samo robi na Pokładzie.

- Na zakończenie na konsoli pojawia się podsumowanie ilu Pasażerów nie było na Liście Pasażerów i zostali do niej dodani.
- Kapitan wywołuje metodę stoj()

Część C (4 pkt).
Udziwnienia.

Istnieje rodzina Oficerów: Pracowity Oficer i Leniwy Oficer. Jest ich pięciu (dwóch leniwych i trzech pracowitych). Znajdują się w kolekcji samosortowalnej – jest ona cały czas posortowana. Dodanie kolejnego obiektu powoduje, że porządek dalej zostaje zachowany. Kolejność posortowania jest taka: najpierw leniwi, potem pracowici oficerowie. Leniwy oficer od pracowitego różni się tym, iż co prawda odwiedza wszystkich pasażerów ale sprawdza czy znajdują się oni na Liście Pasażerów z prawdopodobieństwem 0.1.