PROGRAMOWANIE OBIEKTOWE - PROJEKT

OGÓLNY OPIS PROJEKTU

Celem projektu jest stworzenie aplikacji do zarządzania restauracją dowożącą obiady do klientów. Podstawowymi bytami w tym świecie powinny być *zamówienia, dostawcy, klienci* i *posiłki.*

Klienci dzielą się na okazjonalnych, stałych i firmowych. Klient okazjonalny ma imię/nazwę/kod, numer telefonu, adres dostawy, przypisane zamówienie, godzinę zamówienia i opcjonalnie email. Klienci stali mają dodatkowo punkty lojalnościowe i stałą zniżkę na zamówienie. Przy przekroczeniu zadanego progu liczby punktów lojalnościowych, stały klient dostaje zniżkę 100 zł przy następnym zamówieniu i zebrane punkty zmniejszają się o zadany próg. Klienci firmowi oprócz adresu dostawy mają stały adres korespondencyjny, indywidualny numer konta bankowego i REGON.

Posiłki są ułożone w menu. Każdy posiłek ma nazwę, listę składników, cenę, kategorię, do której należy (np. Pizze, Makarony, Desery) oraz opcjonalnie rozmiar porcji (np. dziecięca/zwykła lub 32cm/46cm.) Posiłki można łączyć w zestaw obiadowy, a w ramach zestawu można ustawić zniżkę (cena zestawu = cena posiłków składowych*zniżka.) Zbiór posiłków i/lub zestawów tworzy zamówienie. Poniżej zadanej kwoty do ceny zamówienia powinna być dodawana cena za dowóz. Cena za dowóz powinna być również dodawana, jeśli adres dostawy leży dalej niż zadana odległość.

Każdy dostawca ma imię, nazwisko, PESEL, dni i godziny pracy, listę kategorii prawa jazdy. Dostawca może korzystać tylko z tych pojazdów, na które ma uprawnienia. Pojazdy dzielą się na skutery i samochody. Każdy pojazd ma ładowność, prędkość, numer rejestracyjny, pojemność baku.

Klienci zamawiają listę posiłków i zestawów, które po odpowiednim czasie przygotowania są przekazywane dostawcom, którzy rozwożą je pojazdami. Jeśli więcej niż jeden posiłek powstanie w podobnym czasie, jeden dostawca w ramach jednego kursu może zabrać i rozwieźć kilka posiłków. W takim przypadku dostawca odwiedza kolejne punkty dostawy a następnie wraca do restauracji.

WYMAGANIA

DOSTĘPNE FUNKCJE I PODSTAWOWE ZAŁOŻENIA:

- użytkownik wprowadza do systemu posiłki, klientów i dostawców poprzez **panel kontrolny**;
- posiłki, klienci i dostawcy są tworzeni tylko na polecenie użytkownika, zamówienia mają pojawiać się automatycznie proporcjonalnie do liczby zaewidencjonowanych klientów;
- tworzeni posiłek/klient/dostawca/zamówienie powinny mieć automatycznie wylosowane wszystkie wartości pól (nawet jeśli są tworzone na polecenie użytkownika);
- wszyscy klienci/dostawcy są **rysowani na mapie** zgodnie z ich aktualnym adresem/położeniem;

- każdy klient, dostawca to **osobny wątek;**
- wątek klienta odpowiada za generowanie w losowych odstępach czasu zamówień;
- jeśli dostawca jest w restauracji, wątek dostawcy sprawdza cyklicznie czy są nieobsłużone zamówienia i, jeśli są, zaczyna je rozwozić;
- oprócz zamówień powstających automatycznie, powinna istnieć możliwość ręcznego stworzenia zamówienia w panelu kontrolnym;
- użytkownik może oglądać podstawowe informacje o obiekcie (kliencie, dostawcy) w osobnym **oknie lub panelu informacyjnym** po kliknięciu na narysowany na mapie obiekt;
- korzystając z przycisków w oknie informacyjnym obiektu, użytkownik może **usunąć** klienta/dostawcę z systemu, zmusić dostawcę do **awaryjnego powrotu do restauracji**;
- podczas jazdy pojazd traci paliwo; informacja ta powinna być widoczna w oknie informacyjnym dostawcy; należy założyć, że po powrocie do restauracji pojazd jest zawsze tankowany do pełna;
- pojazdy dostawców nie powinny najeżdżać na siebie na mapie; jeśli trasy dostawców się skrzyżują dostawcy powinni się wyminąć lub poczekać aż jeden z nich przejedzie pierwszy;
- do rozwiązania problemów z wielowątkowością należy wykorzystać **semafor lub monitor**;
- powinna istnieć możliwość zapisania i odtworzenia stanu symulacji poprzez **serializację**;
- podstawowymi kryteriami oceny są jakość kodu i funkcjonalność zgodna z wymaganiami; walory estetyczne wizualizacji działają na plus oddającego, ale brak wymyślnej grafiki w żaden sposób nie obniża oceny;
- podobnie jak walory estetyczne, choć niewymagane, na plus będą działać nawiązania do kultury popularnej (np. Pizza Tycoon, Top Chef, Master Chef, Bake Off).

DODATKOWE WYMAGANIA:

- w terminie najpóźniej do 28. listopada 2016 r. należy przesłać prowadzącemu swoją wizję modelowanego świata w postaci **diagramu klas UML**;
- w terminie najpóźniej do 5. grudnia 2016 r. należy przedstawić prowadzącemu szkielety **klas Java** napisane zgodnie z zaproponowanym diagramem klas UML;
- wszystkie niestatyczne pola klas powinny być prywatne, a dostęp do nich powinien być realizowany poprzez **settery** i **gettery**;
- kod projektu powinien być udokumentowany tak by umożliwić wygenerowanie dokumentacji w formacie HTML przy pomocy narzędzia **javadoc**;
- nadesłany projekt powinien zawierać aplikację w postaci **uruchamialnego** pliku *.jar, folder ze źródłami oraz prosty plik *readme* zawierający:
  - imię, nazwisko, numer indeksu, grupę i dzień zajęć,
  - krótką instrukcję obsługi programu;
- kod źródłowy programu wraz z dodatkowymi plikami powinien zostać przesłany na adres prowadzącego **do 22.01.2017 do 23:59**; po tym terminie maksymalna ocena z projektu będzie co tydzień obniżana o ocenę w dół lub wymagane będzie rozbudowanie projektu;
- istnieje możliwość pertraktacji niektórych punktów wymagań funkcjonalnych, ale wszelkie zmiany w projekcie muszą być konsultowane z prowadzącym;

- zastrzega się możliwość wezwania studenta w celu osobistego wyjaśnienia wątpliwości dotyczących programu zaliczeniowego;
- **wszelkie plagiaty będą skutkowały oceną 2,0 zarówno dla osoby udostępniającej jak i kopiującej kod; wykrycie plagiatu po sprawdzeniu projektu i wystawieniu z niego oceny powoduje zmianę oceny na 2,0.**