Projekt: System obsługi parkingu

Autor: Kamila Szydełko

Rzeszów 2022

Spis treści

1. Streszczenie

Sprawozdanie przedstawia kolejne etapy budowy aplikacji wspomagającej obsługę parkingu: od postawienia wymagań, które ma spełniać aplikacja, poprzez projekt aplikacji i sekwencje czynności po działającą aplikację.

2. Wstęp

Od kilkunastu lat komputery pomagają nam w codziennym życiu ułatwiając je. Temat, który wybrałam ma za zadanie pomoc w zarządzaniu parkingiem. Celem projektu jest utworzenie programu, który umożliwi użytkownikowi zarządzanie parkingiem oraz sterowanie przyjazdami oraz odjazdami poszczególnych pojazdów. Użytkownik posiada także możliwość sprawdzenia stanu parkingu oraz jego wizualizacji. Aby pomóc mu w zarządzaniu, może także wyszukać pojazd po numerze rejestracyjnym oraz posortować pojazdy według dwóch podanych w programie kryteriów.

3. Wymagania funkcjonalne i niefunkcjonalne

Projektowany system powinien umożliwiać realizację następujących działań i posiadać poniżej wymienione cechy:

- Zarządzanie przyjazdami oraz odjazdami zaparkowanych pojazdów (motocykl, samochód, autobus)
- Wizualizacja parkingu
- Lista, która mówi o stanie parkingu (jakie pojazdy przyjechały, a jakie odjechały)
- Wyświetlanie informacji o pojeździe (typ pojazdu, numer rejestracyjny, pojemność silnika, rok produkcji, numer rejestracyjny, nazwisko właściciela, data parkowania, zajmowane pola)
- Dodanie nowego pojazdu, gdy miejsce parkingowe jest wolne
- Wyszukiwanie pojazdu po numerze rejestracyjnym
- Sortowanie pojazdów według pojemności silnika lub numeru rejestracyjnego

4. Projekt aplikacji

Aplikacja to są dane i działania wykonywane na danych. W tym rozdziale zostaną opisane klasy, które będą reprezentować dane i działania wykonywane na nich.

Aplikacja

Klasa zawiera metodę main oraz wszystkie potrzebne metody i zmienne potrzebne do zarządzania aplikacją. Tutaj inicjuje się okienko aplikacji oraz przyciski. Dodatkowo znajduje się tu ArrayList, który służy jako baza danych do zapisywania przyjazdów oraz odjazdów pojazdów

Pojazd

Za pomocą tej klasy można tworzyć kolejne pojazdy. Klasy: motocykl, samochod, autobus dziedziczą po niej.

GUI

Oknem głównym aplikacji jest Aplikacja.form. Znajdują się tam wszystkie potrzebne wizualne obiekty, które są potrzebne do zarządzania aplikacją przez użytkownika.

Parking

Za pomocą tej klasy można utworzyć parking oraz zarządzać na nim przyjazdami i odjazdami pojazdów. Znajduje się tu także metoda do wyświetlania aktualnego wyglądu parkingu.

Rysunek 1 Opisanie działania aplikacji

Baza danych

Dane o przyjazdach i odjazdach znajdują się jako dwie osobne listy, aby lepiej było zarządzać przyjazdami oraz odjazdami poszczególnych pojazdów. Każdy pojazd, który przyjeżdża zostaje wpisane na listę „lista" (*rys 1*)., każdy pojazd, który odjeżdża zostaje wpisany na listę „lista_wyjazd" (*rys 2*). Baza danych znajduje się w klasie Aplikacja.

Przyjazd
Marka
Pojemnosc_silnika
Rok_produkcji
Numer_rejestracyjny
Nazwisko
Data_parkowania
Zajmowane_miejsce

Rysunek 1 Wygląd bazy danych przyjazdów

Odjazd
Marka
Pojemnosc_silnika
Rok_produkcji
Numer_rejestracyjny
Nazwisko
Data_parkowania

Rysunek 2 Wygląd bazy danych odjazdów

Klasa Aplikacja

Zawiera wszystkie najważniejsze funkcje służące do tego, aby aplikacja działała poprawnie. Znajdują się tu wszystkie przyciski oraz umieszczone w nich odpowiednie funkcje. Znajduje się tu także klasa main.

Klasa Pojazd

Atrybuty:

- Marka
- Pojemność silnika
- Rok produkcji
- Numer rejestracyjny
- Nazwisko
- Data parkowania

Znajdują się tu także gettery i settery oraz metoda toString(), która służy do wyświetlania informacji o pojeździe.

Klasa Motocykl

Dziedziczy po klasie Pojazd

Atrybuty:

- Marka
- Pojemność silnika
- Rok produkcji
- Numer rejestracyjny
- Nazwisko
- Data parkowania

Znajdują się tu także gettery i settery oraz metoda toString(), która służy do wyświetlania informacji o motocyklu.

Klasa Samochód

Dziedziczy po klasie Pojazd

Atrybuty:

- Marka
- Pojemność silnika
- Rok produkcji
- Numer rejestracyjny

- Nazwisko
- Data parkowania

Znajdują się tu także gettery i settery oraz metoda toString(), która służy do wyświetlania informacji o samochodzie.

Klasa Autobus

Dziedziczy po klasie Pojazd

Atrybuty:

- Marka
- Pojemność silnika
- Rok produkcji
- Numer rejestracyjny
- Nazwisko
- Data parkowania

Znajdują się tu także gettery i settery oraz metoda toString(), która służy do wyświetlania informacji o autobusie.

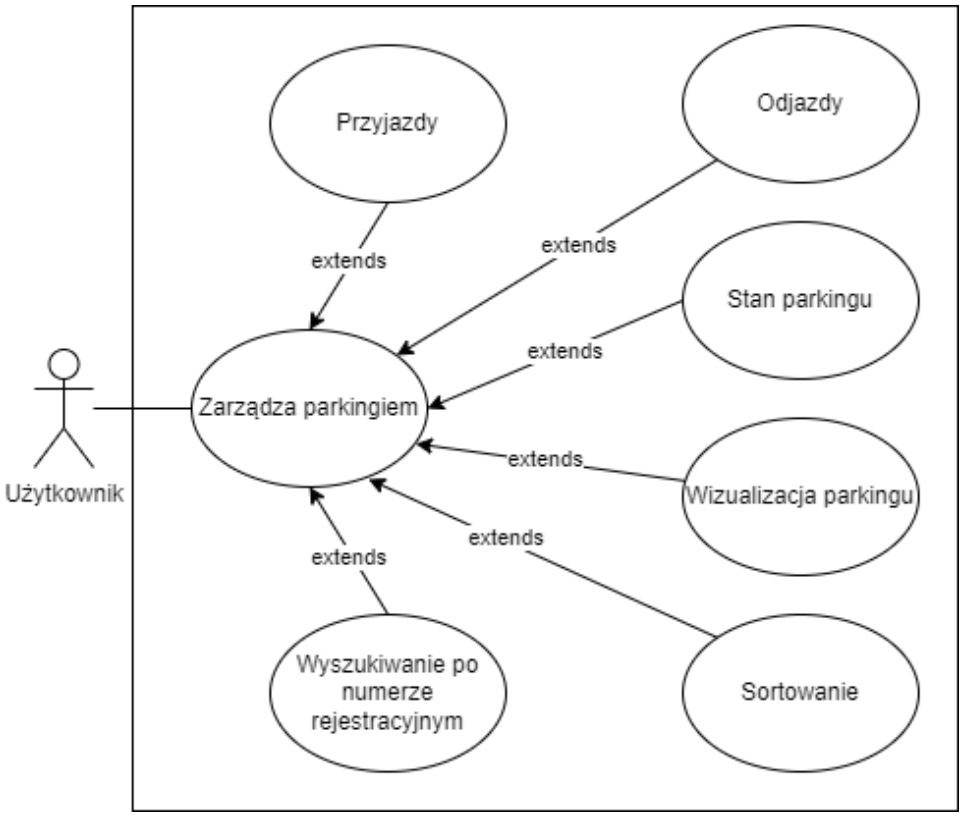
Klasa Parking

Znajduje się tu konstruktor do tworzenia parkingu o wymiarach 11x20 z uwzględnieniem przejazdów co drugi wiersz. Konstruktor tworzy pusty parking, na którym wolne miejsca oznaczone są literą „x”, przejazdy literą „p”,. Dodane pojazdy są oznaczane w następujący sposób: motocykl literą „m”, samochód literą „s”, autobus literą „a”. Dodatkowo znajdują się tutaj metody Przyjazd, Odjazd i Wyglad_parkingu.

5. Diagramy UML

5.1 Diagram przypadków użycia

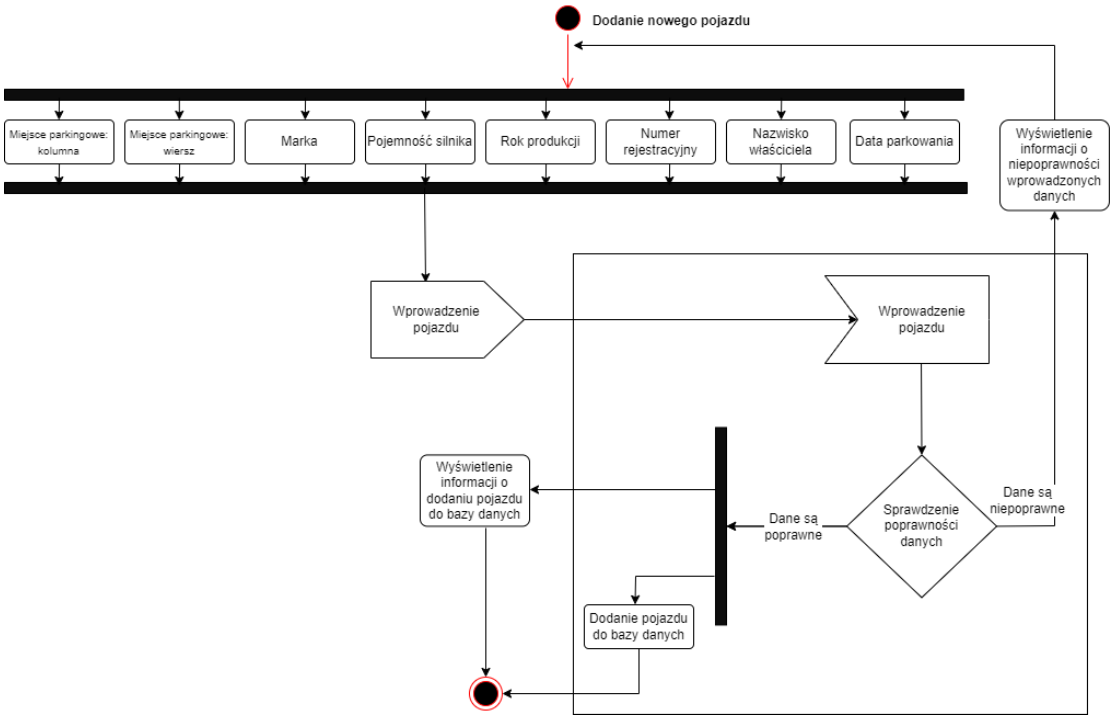
Na rysunku poniżej (*rys.1*) przedstawiono diagram przypadków użycia projektowanej aplikacji.



Rysunek 1 Diagram przypadków użycia

5.2 Diagram sekwencji czynności

Zadaniem tego rozdziału jest pokazanie sekwencji kroków jakie są wykonywane przez modelowany fragment systemu. Oto przykładowe działania, które trzeba wykonać aby wprowadzić nowy pojazd na parking (*rys.1*).



Rysunek 1 Wprowadzanie danych o pojazdach na parking

6. Opis strony użytkowej projektu

Po uruchomieniu aplikacji pokazuje się okno prezentujące Menu funkcji (*rys 1*) oraz menu przyjazdy oraz odjazdy pojazdów (*rys 3*). Na *rysunku 1* zaprezentowano wygląd całej aplikacji.

Rysunek 1 Wygląd aplikacji

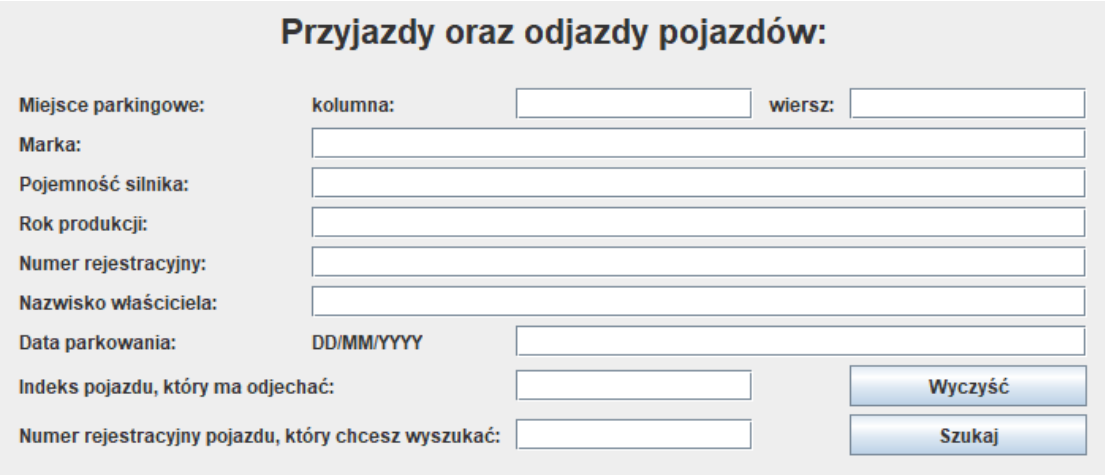
W „Menu funkcji", które znajduje się po prawej stronie dodane są odpowiednie przyciski do przyjazdów, odjazdów oraz do sortowania aut znajdujących się na parkingu według numeru rejestracyjnego i pojemności silnika. Aktualny stan parkingu oraz jego wygląd wizualny można sprawdzić za pomocą odpowiednio podpisanych przycisków (*rys 2*).

Rysunek 2 Wygląd menu

Po prawej stronie znajduje się menu „Przyjazdy oraz odjazdy pojazdów" (*rys 3*). Jeśli użytkownik chce dodać nowe auto, wypełnia puste miejsca odpowiednimi danymi (miejsce parkingowe, markę, pojemność silnika, rok produkcji, numer rejestracyjny, nazwisko właściciela, datę zaparkowania), a następnie w Menu funkcji (*rys 1*) wybierze, który pojazd ma przyjechać. Pojazd zostanie wtedy dodany do listy przyjazdów. Jeśli użytkownik chce, aby jakiś pojazd wyjechał z parkingu, wystarczy że wpisze numer pojazdu z listy, oraz miejsce

parkingowe, na którym się znajduje, a następnie w Menu Funkcji (*rys 1*) wybierze, który pojazd ma odjechać. Pojazd zostanie wtedy dodany do listy odjazdów.

Znajduje się tu także przycisk „Wyczyść", który pozwala użytkownikowi w szybki sposób wyczyścić pola tekstowe w aplikacji. Dodatkowo na samym dole usytuowana jest opcja wyszukiwania pojazdu po numerze rejestracyjnym.



Rysunek 3 Wygląd menu wprowadzania danych

7. Podsumowanie i wnioski

Projekt udało się zrealizować zgodnie z postawionymi wymaganiami. Od strony użytkowej można poprawić następujące elementy:

- Bardziej przejrzysty wizualnie parking, dodanie dokładnej numeracji kolumn i wierszy
- Osobne wyskakujące okna dla przyjazdów i odjazdów, które wyświetlałyby się po naciśnięciu odpowiedniego przycisku. W wspomnianych oknach użytkownik wybierałby jaki typ pojazdu chce dodać i wpisać odpowiednie dane
- Brak możliwości wpisania na miejsce parkingowe pojazdu, jeśli nie są wypełnione wszystkie pola identyfikujące pojazd
- Wybór przez użytkownika za pomocą jakich wartości chce posegregować lub znaleźć pojazdy znajdujące się na parkingu

W przyszłości istnieje możliwość dodania większej ilości pojazdów, które będą mogły parkować na parkingu (np. rowery, hulajnogi).

8. Literatura:

- https://www.w3schools.com/java/default.asp