

Sprawozdanie – Restauracja Kebabowa

Tomasz Potoczko



Projekt zaliczeniowy z Laboratorów Programowania Obiektowego

prowadzone przez kmdr. Maksymiliana Września

Zawartość

[Program 4](#_Toc105605945)

[Analiza 4](#_Toc105605946)

[Identyfikacja problemu 4](#_Toc105605947)

[Cechy problemu które skłaniają do rozwiązań komputerowych 4](#_Toc105605948)

[Research w podobne problemy 4](#_Toc105605949)

[Podobne problemy I ich rozwiązania 4](#_Toc105605950)

[Niezbędne cechy rozwiązania cyfrowego 4](#_Toc105605951)

[Limitacje proponowanego rozwiązania 4](#_Toc105605952)

[Specyfikacja problemu 4](#_Toc105605953)

[Wymagania rozwiązania (config) 4](#_Toc105605954)

[Kryteria sukcesu rozwiązania 4](#_Toc105605955)

[Design 5](#_Toc105605956)

[Dekompozycja problemu 5](#_Toc105605957)

[Rozbicie problemu na mniejsze części 5](#_Toc105605958)

[Opis rozwiązania 5](#_Toc105605959)

[Struktura rozwiązania 5](#_Toc105605960)

[Algorytmy 6](#_Toc105605961)

[Kluczowe zmienne, struktury oraz klasy 11](#_Toc105605962)

[Development 12](#_Toc105605963)

[Iteracyjny proces postępu 12](#_Toc105605964)

[Opisane dowody rozwoju 12](#_Toc105605965)

[Dowody rozwiązań prototypowych 12](#_Toc105605966)

[Testy informujące developera 12](#_Toc105605967)

[Dowody testowania 12](#_Toc105605968)

[Naprawy zaradcze kodu 13](#_Toc105605969)

[Ewaluacja I ocena końcowa 14](#_Toc105605970)

[Testy informujące ewaluacje 14](#_Toc105605971)

[Dowody testowania ‘solidności’ rozwiązania 14](#_Toc105605972)

[Testy użyteczności (usability) /User feedback 19](#_Toc105605973)

[Zmierzyć mniej więcej jak dobrze rozwiązany został problem względem designu oraz kryteriów sukcesu 19](#_Toc105605974)

[Opis końcowy rozwiązania 20](#_Toc105605975)

[Dowody cechy użytkowych (usability features) z designu, komentując ich skuteczność 20](#_Toc105605976)

[Utrzymanie i możliwy rozwój 20](#_Toc105605977)

[Łatwość utrzymania rozwiązania 20](#_Toc105605978)

[Możliwy dalszy rozwój 20](#_Toc105605979)

# Program

link

# Analiza

## Identyfikacja problemu

Jest to rozwiązanie na czysto teoretyczny, wręcz wirtualny, problem, czyli zbudowania prostej symulacji *restauracji*.

### Cechy problemu które skłaniają do rozwiązań komputerowych

* Potrzeba zarządzania potencjalnie dużą ilością pracowników oraz zamówień (które, muszą być archiwizowane).
  + Jest to o wiele wygodniejsze oraz czytelniejsze w formie cyfrowej, czyli w bazach danych niż na papierze.
* ???

## Research w podobne problemy

### Podobne problemy I ich rozwiązania

* ???

### Niezbędne cechy rozwiązania cyfrowego

* Możliwość operowania na dużych ilościach danych w krótkim czasie
* Możliwość utrzymywania baz danych przy minimalnym wkładzie od użytkownika
* Czytelność oraz łatwość szukania danych
* Możliwość ograniczenia dostępu do danych w zależności od pracownika

### Limitacje proponowanego rozwiązania

* ???

## Specyfikacja problemu

### Wymagania rozwiązania (config)

* ???

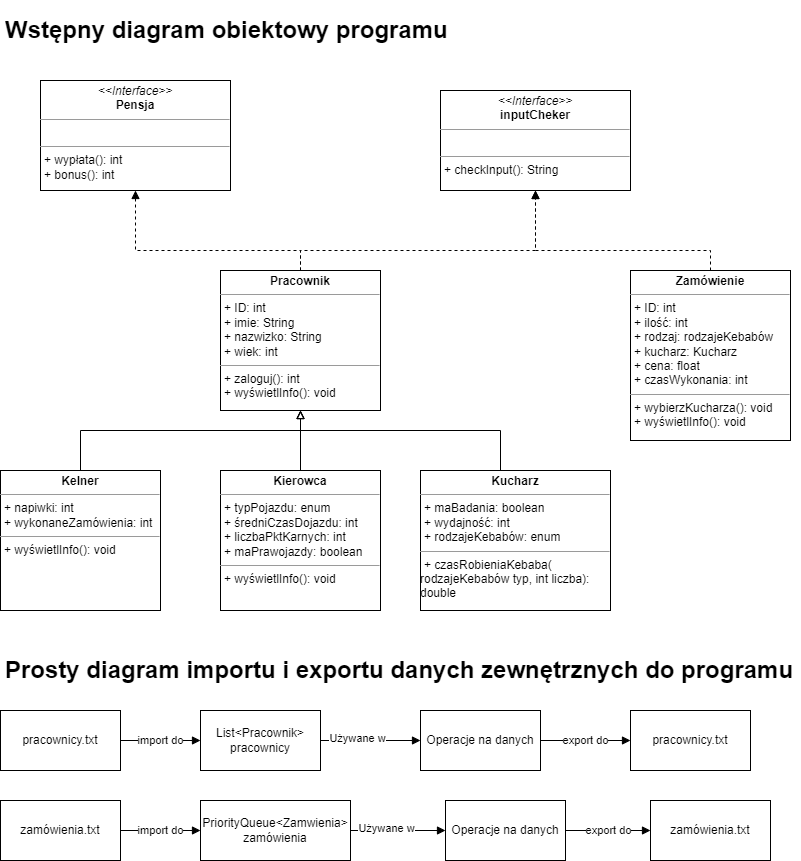
### Kryteria sukcesu rozwiązania

* Podstawowe cechy rozwiązania:
  + Program będzie wczytywał dane pracowników z zewnętrznego pliku do listy, operował na nich oraz nadpisywał/dopisywał do pliku
  + Program będzie wczytywał dane o zamówieniach z zewnętrznego pliku do kolejki priorytetowej, operował na nich oraz nadpisywał/dopisywał do pliku
  + Program będzie pozwalał zalogować się każdemu pracownikowi, nie każdy o równych uprawnieniach
  + Program będzie napisany obiektowo
* Opcjonalne cechy rozwiązania:
  + Część zmiennych w klasach będzie niezależna od danych wprowadzonych od użytkownika
  + Użytkownik będzie miał kontakt z programem poprzez menu
  + Niezależnie od wejść podanych od użytkownika, kod się nie wysypie

# Design

## Dekompozycja problemu

### Rozbicie problemu na mniejsze części

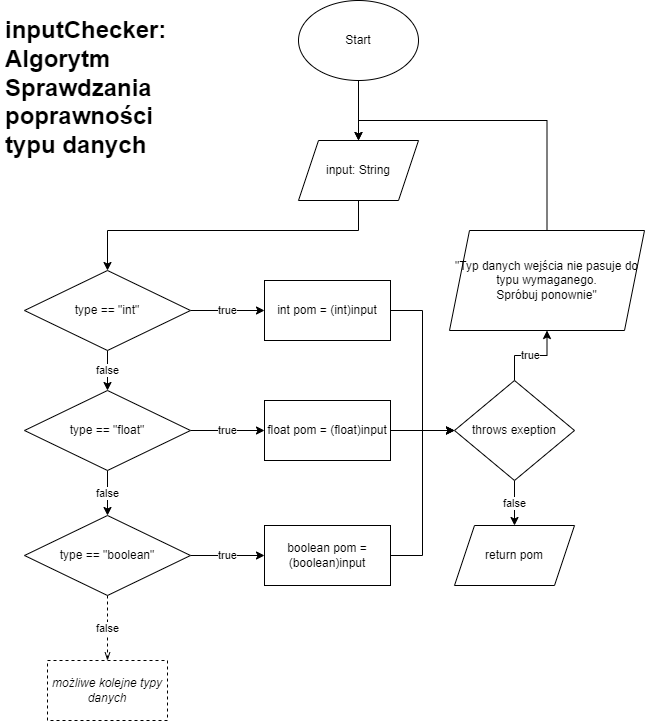


## Opis rozwiązania

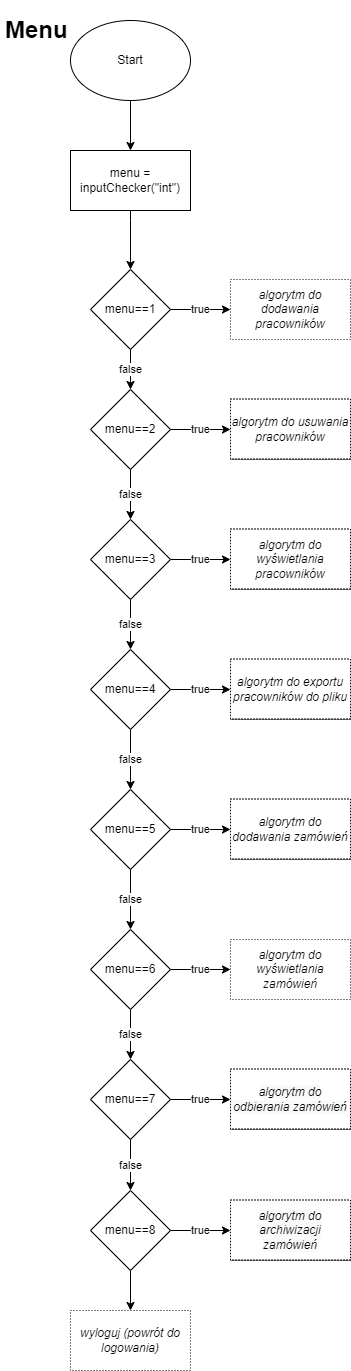
### Struktura rozwiązania

### Algorytmy

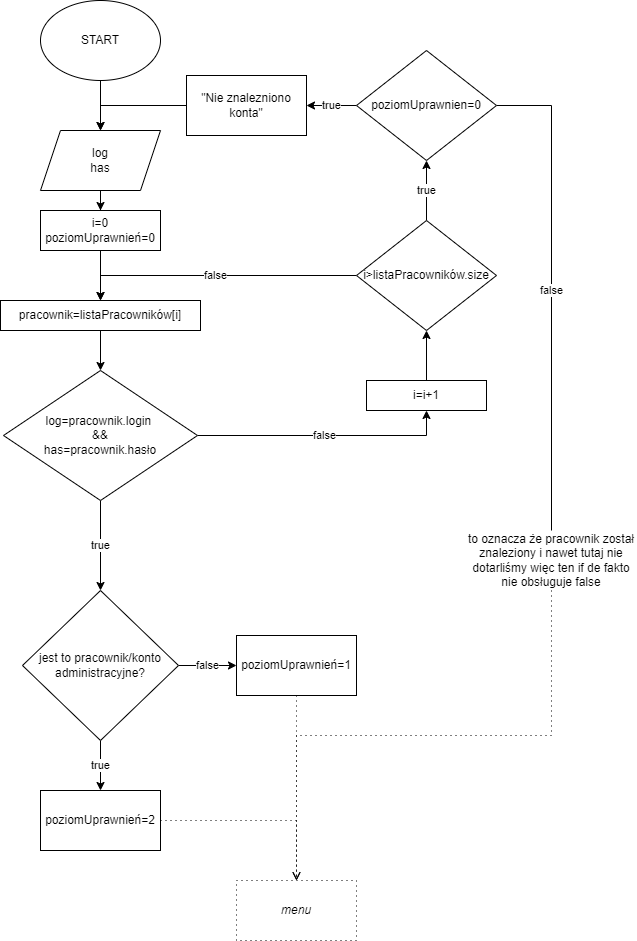
#### inputChecker



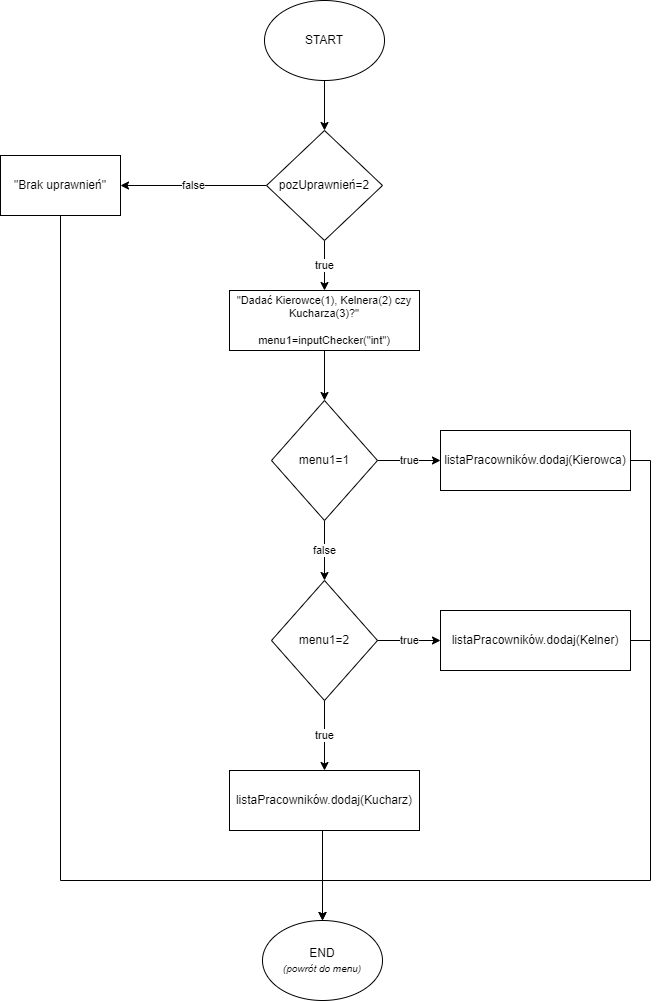
#### menu



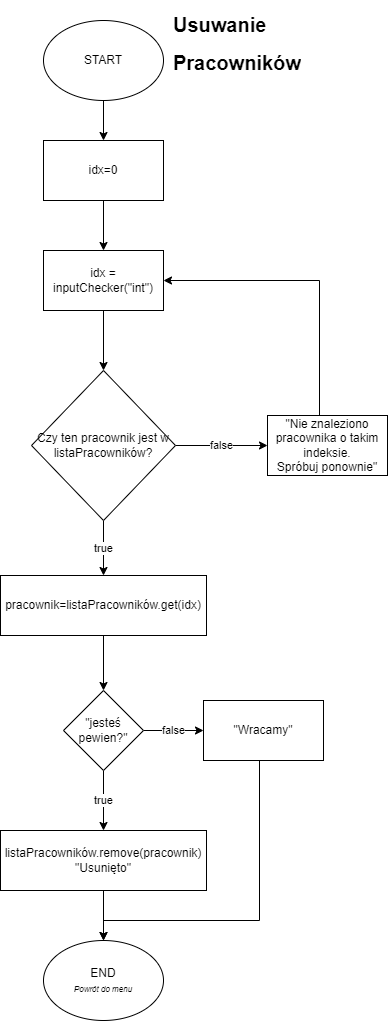
#### Log



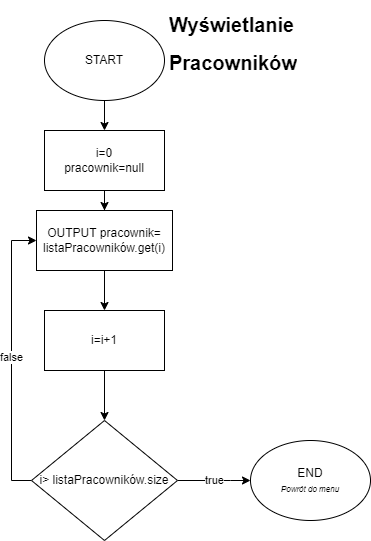
#### Dodawanie pracowników



#### Usuwanie pracowników



#### Wyświetlanie pracowników



#### Zamówienia

Algorytm do dodawania, wypisywania, odbierania oraz archiwizowania zamówień są identyczne do tych dla pracowników.

### Kluczowe zmienne, struktury oraz klasy

|  |  |  |  |
| --- | --- | --- | --- |
| **Typ** | **Lokacja** | **Nazwa** | **Opis** |
| Klasa | N/A | Pracownik | Superklasa z której dziedziczą Kelner, Kierowca, Kucharz. Implementuje inputChecker. Działa jako podstawa, gdyż każdy użytkownik jest pracownikiem przez co każdy ma wspólne cechy, które są zawarte w tej klasie. |
| Klasa | N/A | Kelner | Podklasa Pracownik, bardziej specjalistyczna wariacja pracownika. Zawiera liczbaWykonanychZamówień, i tylko obiekty tej klasy mogą odbierać zamówienia. |
| Klasa | N/A | Kierowca | Podklasa Pracownik, bardziej specjalistyczna wariacja pracownika.??? |
| Klasa | N/A | Kucharz | Podklasa Pracownik, bardziej specjalistyczna wariacja pracownika. ??? |
| Klasa | N/A | Zamówienie | Obiekty tej klasy przechowują wszystkie potrzebne zmienne dla zamówienia. |
| Interface | N/A | inputChecker | Zawiera metodę, która sprawdza czy dane wejście od użytkownika jest poprawne. Jako argument przyjmuje ciąg znaków który wyznacza jaki typ jest oczekiwany od użytkownika. |
| Interface | N/A | Pensja | Zawiera metody do liczenia pensji dla pracowników, która dla różnych podklas może się różnić. |
| Lista | Main | Pracownicy | Przechowuje wszystkie obiekty (nowe czy importowane) podklas klasy Pracownik, które są używane w programie. |
| Kolejka Priorytetowa | Main | Zamówienia | Przechowuje wszystkie obiekty klasy Zamówienia, które są posortowane są od tych z najkrótszym czasem wykonania do najdłuższych. |
| String | Pracownik | imie |  |
| String | Pracownik | nazwisko |  |
| int | Pracownik | ID |  |
| int | Kelner | liczba  Wykonanych  Zamówień |  |
| enum | Kucharz i Zamówienie | rodzaje  Kebabów | Wyznacza rodzaje kebaba które mogą być zamówione. Ważne do wyliczania czasu zrobienia zamówienia, gdyż każdy kucharz może zrobić danego kebaba szybciej lub wolniej. |
| int | Zamówienie | ID | ID zamówienia. |
| double | Zamówienie | cena | Cena zamówienia. |
| Kucharz | Zamówienie | kucharz | Potrzebne do wyznaczenia czasuWykonania. |
| int | Zamówienie | Czas  Wykonania | Czas wykonania zamówienia zależny od kucharza i typu kebaba. |
| int | Main | poziom Uprawnień | Zależne od konta. Wyznacza które funkcje są dostępne dla zalogowanego użytkownika. |

# Development

Normalnie robiłbym tą część wraz z pisaniem programu, ale że dowiedziałem się o sprawozdaniu, gdy kod był w 90% skończony to nic nie zdążyłem tu dać. Zostawiam to, żeby pokazać co normalnie bym zrobił do jakiegoś większego projektu.

## Iteracyjny proces postępu

### Opisane dowody rozwoju

Zrzuty ekranu nowych cząstek kodu i jego zmiany, napotkane problemy z implementacją idą do Testów informujących developera.

### Dowody rozwiązań prototypowych

Zrzuty ekranu pokazujące implementacje algorytmów z Design‘u, potencjalne zmiany w działaniu algorytmów itp.

## Testy informujące developera

### Dowody testowania

### Naprawy zaradcze kodu

Zrzuty ekranu problemów, błędów, glitchy itp. oraz jak zostały one naprawione.

# Ewaluacja I ocena końcowa

## Testy informujące ewaluacje

### Dowody testowania ‘solidności’ rozwiązania

|  |  |  |  |
| --- | --- | --- | --- |
| **Obiekt testowany** | **Sposób testu** | **Odpowiedź programu** | **Zgadza się z kryteriami?** |
| Wczytywanie danych z zewnętrznego pliku | Wyświetlenie elementów w listy |  | Tak |
| Kolejka priorytetowa | Wyświetlenie elementów w kolejce |  | Tak |
| Logowanie | Wpisanie niepoprawnych danych logowania |  | Tak |
|  | Wpisanie poprawnych danych logowania |  | Tak |
| Menu | Wpisanie liczby z menu |  | Tak |
| Menu | Wpisanie liczby z poza menu |  | Tak |
| inputChecker | Wejście typu String w oczekiwany int |  | Tak |
| inputChecker | Wejście typu String w oczekiwany double |  | Tak |
| inputChecker | Wejście typu String w oczekiwany boolean |  | Tak |
| inputChecker | Wejście typu int w oczekiwany float |  | Tak |
| inputChecker | Wejście typu float w oczekiwany int |  | Tak |
| inputChecker | Wejście typu int w oczekiwany boolean |  | Tak, ale dodałem też możliwość wpisania 1 lub 0. |
| inputChecker | Wejście typu float w oczekiwany boolean |  | Tak |
|  |  |  |  |

## Opis końcowy rozwiązania

### Dowody cechy użytkowych (usability features) z designu, komentując ich skuteczność