# Symulacja windy

Denis Lyakhov, II rok, Informatyka, gr. 2a

Styczeń 2021

## 1 Cel programu

Program ma na celu zademonstrować działanie windy lub kilku wind w postaci symulacji napisanej w języku Ada. Użytkownik będzie miał możliwość ręcznej konfiguracji przeprowadzanej symulacji. Między innymi, wprowadzenie dowolnej liczby wezwań windy lub kilku wind, wybranie pięter startowych oraz końcowych.

W przypadku jednej windy, symulacja zademonstruje w jaki sposób winda otrzymuje wezwania i ich wykonuje, podejmując decyzje o dalszych piętrach docelowych, gdzie należy wpuścić osobę do windy, a gdzie wypuścić.

W dodatku do powyżej opisanych wymagań, symulacja musi wziąć pod uwagę wariant z działaniem kilku wind w tym samym czasie. Co oznacza to, że system zarządzający tymi windami, musi być w stanie sensownie rozdzielić wezwania pomiędzy wszystkimi windami, znajdującymi w tym systemie.

## 2 Opis i schemat struktury zadaniowej programu

Pojedyncza winda w naszym programie jest przedstawiona w postaci zadania (task). Umożliwia on nam na implementację współbieżności w języku Ada. Zadanie posiada następujące argumenty:

1. elevatorNum - numer porządkowy windy do identyfikacji w systemie
2. currentFloor - przechowuje informację o tym, na jakim piętrze winda się znajduje.
3. direction - pokazuje kierunek ruchu windy
4. currentDestination - wskazywa na docelowe piętro
5. currentCall (i) - obsługiwane wezwanie

W ciele zadania "Elevator" znajduję się algorytm odbierający wezwanie z dostępnych, podejmujący dalej decyzję o dalszym celu (w tym, kierunek ruchu

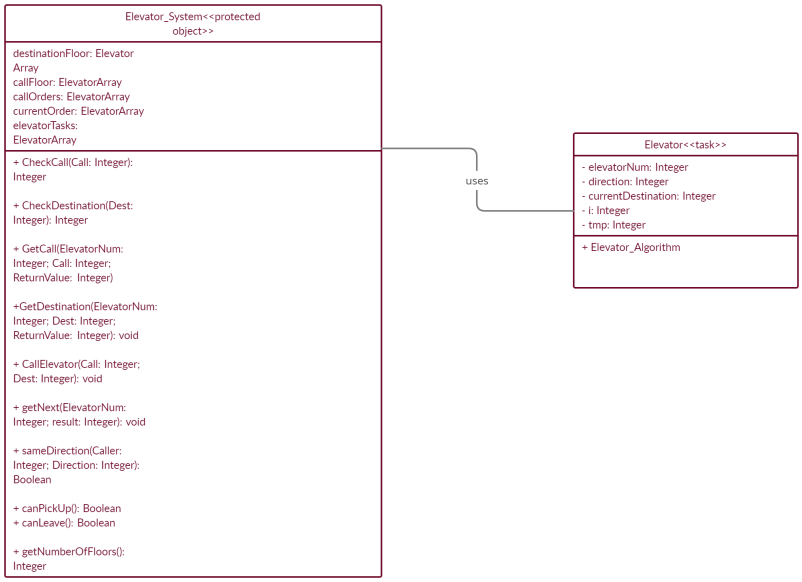
oraz docelowe piętro) oraz w pętli wykonujące przemieszczenie windy do aktualnego celu, w międzyczasie podejmujący decyzje o wpuszczenie i wypuszczenie osoby do oraz z windy.

Rolę systemu zarządzającego windami będzie wypełniać jeden obiekt chroniony, zawierający argumenty:

1. callFloor - tablica wezwań wind
2. destinationFloor - tablica pięter docelowych
3. callOrders - kolejka wezwań
4. elevatorTasks - rozdzielone wezwania pomiędzy windami

System też posiada metody, umożliwiające na komunikowanie z windami, z których najważniejsze są:

1. GetCall - otrzymanie następnego wezwania
2. GetDestination - otrzymanie następującego celu




Rysunek 1: Elevator UML Diagram

# 3 Instrukcja obsługi

Program posiada dwa pliki główne - "MainOneElevator.adb" oraz "MainTwoElevators.adb". Do przeprowadzania symulacji w wariantach z jedną oraz dwoma windami.

Jak jest wyżej napisane, użytkownik ma możliwość przeprowadzenia własnej symulacji. W tym celu, w metodach głównych plików glównych użytkownik ma możliwość wpisania żądanej sekwencji wezwań windy. Robi się to za pomocą funkcji "callElevator". Jako pierwszy argument przyjmuje Integer z piętrem, z którego wezwanie zostało przeprowadzone, drugi argument - piętro docelowe dla tego wezwania, trzeci argument - czas opożnienia (czas w postaci Float, po którym wezwanie musi nastąpić, domyślnie jest ustwiono na "0.0").

Dodatkowo, można zmienić liczbę pięter oraz kabin w systemie za pomocą podania odpowiedniego argumentu obiektowi "Elevator Shaft" na początku programu oraz tworzenie nowego obiektu "Elevator" w sposób jak jest napisane w programie.

Do kompilacji plików głównych należy wykonać komendę: "gnatmake -P Elevator_Main.gpr"

# 4 Testy

Testy znajdują się w folderze "Tests". Do ich kompilacji należy wykonać komendę: "gnatmake -P Elevator_Tests.gpr". Po tym testowe symulacje można zobaczyć poprzez uruchamianie odpowiednich plików wykonywalnych (np. "one_elevator_test_1", "two_elevator_test_4", itd.)

# 5 Ograniczenia programu

Głównym ograniczeniem programu jest obsługa dużej ilości kabin. Program zakładana istnienie kabin w liczbie nie większej niż liczba pięter.