Symulacja Supermarketu

1.Autorzy:

-Wojciech Pobocha, nr indeksu: 318399

-Mikołaj Sendybył, nr indeksu: 318402

2.Założenia:

-symulacja obiektu odbywa się w pętli while o określonej przez użytkownika liczbie iteracji,

-każda iteracja głównej pętli symbolizuje jedną chwile czasu, w której wykonywane są różne metody przez różne klasy i potrzebują na to różnych ilość czasu,

-klienci mogą wchodzić i wychodzić z supermarketu, wybierać produkty z półki, a następnie wkładać je do swojego koszyka, gdy produktu nie ma na półce mogą pytać pracownika o jego dostępność w magazynie, magazynier może przynieść im ten produkt jeśli znajduje się on na magazynie, lub też poinformować o jego niedostępności, podczas gdy klient czeka na odpowiedź,

-po skończonych zakupach klienci podchodzą do jednej z otwartych kas, wybierając tę o najmniejszej kolejce,

-kasy otwierają się gdy liczba klientów stojących aktualnie w kolejkach przypadających na liczbę otwartych kas będzie większa od pewnej określonej liczby [new_checkout_condition],

-kas jest tyle ilu kasjerów. Każda kasa do otwarcia potrzebuje jednego kasjera,

-każda kasa posiada swoją własną kolejkę klientów reprezentowaną przez wektor klientów,

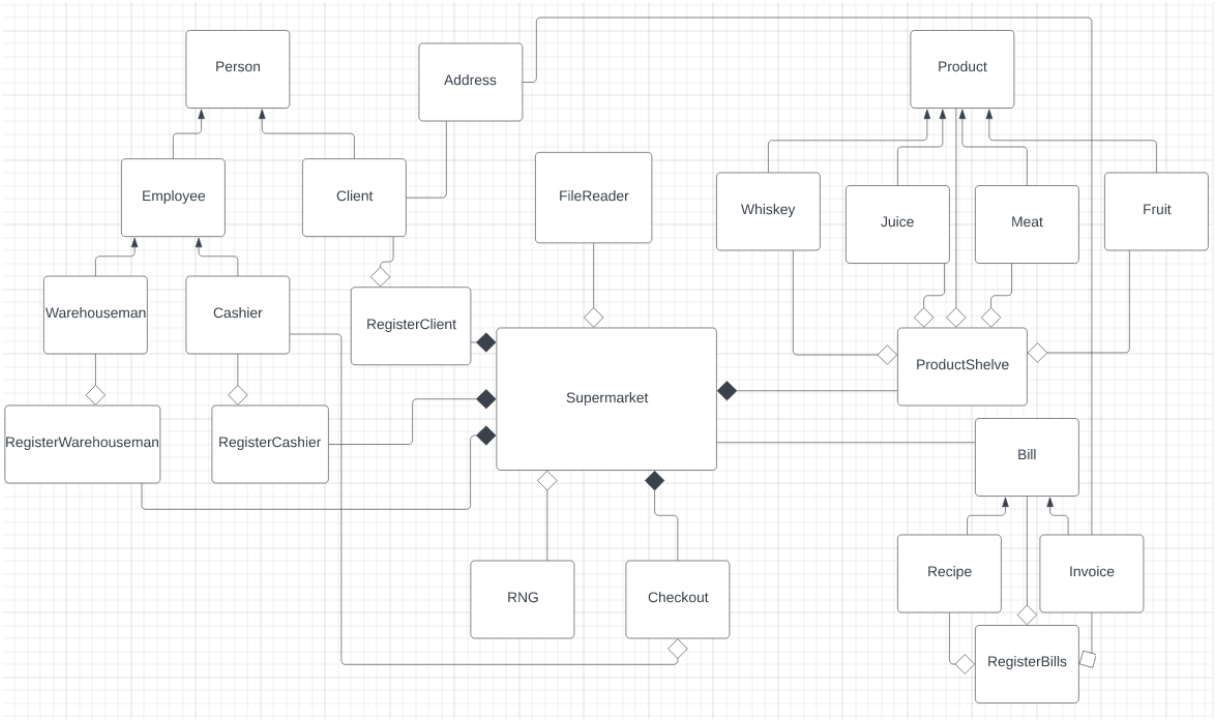
-kasa posiada opcje wydawania paragonu lub faktury w ramach rachunku za zakupy,

-ilość kasjerów, magazynierów i klientów w trakcie symulacji jest generowana losowo w określonym przedziale,

-w każdej iteracji pętli jest losowo generowana szansa na przyjście kolejnych klientów do sklepu,

-kasy posiadają adaptacyjną prędkość skanowania uzależnioną od ilości klientów w ich kolejce.

3.Podział programu na klasy według hierarchii i relacji:



4.Opis działania symulacji:

Argumentami wywołania programu są ścieżka(string) do pliku do wczytania imion i nazwisk, kolejny argument to ścieżka do wczytania produktów, kolejny to ścieżka do wczytania adresów, następnym argumentem jest ścieżka do zapisu całej symulacji. Ostatnim argumentem jest liczba iteracji pętli supermarketu i jest ona typu int.

Na początku działania symulacji wczytują się przy użyciu klasy FileReader podstawowe dane tak jak baza imion, nazwisk i adresów do tworzenia losowych klientów i pracowników, dane produktów, z których tworzą się obiekty, następnie losowo tworzeni są pracownicy, klienci, listy zakupów klientów, a także półki sklepowe z losowymi liczbami produktów na nich. Na końcu fazy wczytywania

W kolejnej fazie 5 funkcji [do_shopping, go_to_magazine, give_prd_to_client, opening_checkouts, scan_prodcut] wykonuje się w pętli, o określonej przez użytkownika długości.

Opis poszczególnych funkcji:

I. do_shopping – iteruje po bazie danych klientów znajdujących się na sklepie i w zależności od ustawionej na nim flagi :

1) is_done – klient skończył zakupy, funkcja wyszukuje mu najlepszą dostępną kasę (o najmniejszej kolejce), klient przechodzi do kasy i jest usuwany z bazy danych klientów znajdujących się na sklepie.
2) is_waiting – klient czeka na informacje zwrotną od magazyniera
3) Brak falgi - klient, w zależności od tego czy poszukiwany przez niego produkt znajduje się na półce sklepowej, wykonuje następujące akcje:

    - jeśli produkt znajduje się na półce, klient wkłada go swojego koszyka (tj. do jego wektora produktów dodaje produkt, a w półce sklepowej (zorganizowanej na mapie) wartość odpowiadająca danemu produktowi jest dekrementowana

    - jeśli produktu nie ma na półce, klient przeszukuje bazę danych magazynierów, i jeśli któryś nic nie robi zadaje mu pytanie, czy produkt jest na magazynie. Jeśli nie ma żadnego wolnego pracownika, klient czeka jedną turę i ponownie sprawdza próbuje zadać pytanie.

Klient z każdym następnym szukanym produktem ma ponownie, losowo ustawiany atrybut is_busy (czas oczekiwania – w tym przypadku czas szukania produktu na półce sklepowej)

II. go_to_magazine – zapytany przez klienta pracownik idzie do magazynu i szuka produktu (zajmuje mu to losowo generowaną liczbę tur). Jeśli produkt jest na magazynie, to pracownik wkłada go sobie do kieszeni(vektor), natomiast jeśli nie, wkłada do kieszeni „produkt duch” – specjalnie generowany obiekt klasy product, służący informowaniu klienta, że produktu nie ma na magazynie.

III. give_prd_to_client – pracownik wraca do klienta i przekazuje mu produkt z magazynu. Jeśli był to „produkt duch”, klient nie dodaje go do swojego koszyka i zaczyna szukać następnego produktu ze swojej listy zakupów. W innym

przypadku dodaje produkt do koszyka i idzie szukać następnego produktu. Magazynier po przekazaniu produktu opróżnia swoją kieszeń.

IV. opening_checkouts – funkcja sprawdza, czy suma klientów we wszystkich kasach przypadających na liczbę otwarty kas przekracza określoną wartość. Jeśli wskaźnik przekracza określoną wartość otwiera się nowa kasa.

V. scan_products – iteruje po kasach; jeśli w kasie stoi chociaż jedna osoba, odpala się funkcja checkout_action (skanowanie produktów [odbywa się ze zmiennym krokiem , tak aby automatycznie dostosować jego długość do liczby pozostałych produktów do zeskanowania]).

Dodatkowo w każdej iteracji istnieje szansa, że w sklepie pojawi się nowy kilent.

5.Zastosowanie elementów biblioteki STL:

-vector jest wykorzystywany do wszystkich klas Register jako pojemnik na obiekty, korzystaliśmy z niego ze względu na możliwość wyboru elementu o wybranym indeksie, a także na możliwość łatwej iteracji po jego elementach.

-mapa, która jest podstawą półki sklepowej, której każdy element składa się z produktu, a także ilości danego produktu – każda para to <Product, int>.

6.Sytuacje wyjątkowe

Jedyną sytuacją, gdy projekt może zadziałać niepoprawnie jest moment gdy podane pliki są złe, lub ich dane są zniekształcone, program rzuca wtedy wyjątek FileReadError, który jako argument przyjmuje linię, w której jest błąd, a także nazwę błędu, jego obsługa polega na złapaniu wyjątku, a następnie wypisaniu jego komunikatu do terminala, co bardzo ułatwia znalezienie błędu w plikach.

7.Podział obowiązków

Wojciech Pobocha:

- wczytywanie do plików,
- zapisywanie do plików, klasy pracowników, klasy rachunków,
- generacja pracowników, produktów i klientów,
- stworzenie baz danych dla pracowników, klientów i produktów
- stworzenie klasy pearson i bill i wszystkich ich pochodnych (z wyjątkiem client)

Mikołaj Sendybył:

- stworzenie klas client, checkout oraz klasy product i wszystkich jej pochodnych
- modyfikacja klasy warehouseman, tak by odpowiadała potrzebą symulacji
- przeciążanie operatorów pracowników, kas i klientów
- wypisywanie w terminalu
- metody supermarketu odpowiadające za przebieg symulacji