SYMULACJA SIECI

Grupa: Rafał Siniewicz, Mateusz Płatek i Maciej Stępień

Modelem w zadaniu "Symulacja sieci" jest linia produkcyjna składająca się z danych, podstawowych elementów: ramp rozładunkowych, robotników oraz magazynów. Tworzą one podstawowe klasy, na których będzie się opierać projekt.
Kolejną klasą będzie produkt, czyli ta część projektu, która łączy wszystkie elementy (pracownika, rampę i magazyn) poprzez przemieszczanie się między nimi wedle ustalonych zasad.
Stworzyliśmy klasę abstrakcyjną Receiver z której dziedziczą Worker i Magazine - zarówno pracownicy jak i magazyny są odbiorcami produktów. Taka struktura umożliwia uproszczenie struktury odbiorców w klasie Worker, ponieważ mogą być oni zarówno innymi pracownikami jak i magazynami.
Dodatkowo utworzyliśmy typ wyliczeniowy enum QueueType, zawierający informację o dwóch rodzajach sposobów przetwarzania półproduktów przez pracowników: FIFO (first in first out) oraz LIFO (last in first out)

Omówienie poszczególnych klas:
- **Klasa Worker:** jest to najbardziej rozbudowana klasa, ze względu na liczbę zadań do wykonania przez pracownika, a także z powodu licznych funkcjonalności i parametrów. Dziedziczy po klasie abstrakcyjnej Receiver.

Pola klasy Worker:

- **workTime(float)** - jest to czas, który robotnik potrzebuje na przetworzenie danego produktu. WorkTime jest typu float, gdyż daje to więcej możliwości i bardziej oddaje rzeczywisty proces
- **endWork(float)** - pokazuje czas, przy którym pracownik powinien zakończyć przetwarzanie produktu
- **qType(QueueType)** - jest to zmienna typu wyliczeniowego QueueType (który może przyjmować wartości FIFO lub LIFO) i określa wedle jakiej kolejności pracownik przetwarza nadchodzące produkty
- **products(list<Product*>)** - jest to kolejka do której są wstawiane produkty, gdy oczekują na zakończenie wykonywania przez robotnika aktualnego zadania
- **workerReceiver(vector<Receiver*>)** - każdy robotnik przekazuje produkt do dalszej obróbki, jest to wektor odbiorców danego produktu
- **workerReceiverPref(vector<float>)** - każdy robotnik ma swoje preferencje odnośnie tego komu przekaże produkt, tworzony jest wektor wartości zmiennoprzecinkowych do przechowywania tych preferencji

Metody klasy Worker:

- **Worker(float time, QueueType queue)** - konstruktor tworzący obiekt typu Worker. Parametrami jakie podajemy jest czas, który robotnik potrzebuje na przetworzenie produktu oraz typ kolejki według której będzie przetwarzał produkty

- **addReceiver(Receiver* r, float pref = -1)** - jest to metoda dodająca kolejnego odbiorcę- robotnika, która domyślnie przyjmuje preferencję -1 co oznacza, że jeśli preferencja nie zostanie podana będzie wyliczany równy rozkład pomiędzy odbiorcami. Metoda nie zwraca niczego
- **work(float time)** - jest to metoda, która aktualizuje aktualny stan pracownika (np. jeśli produkt został przetworzony zostanie on przekazany dalej i rozpocznie się przetwarzanie następnego produktu jeśli jest dostępny).Metoda nie zwraca niczego.
- **addProduct(Product* p)** - metoda dodająca kolejne produkty robotnikowi do przetwarzania, jako argument przyjmuje obiekt typu Product*. Metoda nie zwraca niczego.
- **getQueueType()** - metoda, która zwraca typ kolejki (FIFO lub LIFO)
- **getProducts()** - metoda, która zwraca listę produktów
- **getReceivers()** - metoda, która zwraca listę odbiorców

- **Klasa Magazine:** dziedziczy po klasie abstrakcyjnej Receiver, klasa ta zawiera tylko trzy składowe - wektor produktów jakie trafiają do magazynu, dwie metody: addProduct, która dodaje gotowe produkty do magazynu oraz getProducts, która pokazuje produkty w magazynie.

- **products(vector<Product*>)** - jest to wektor produktów, które znajdują się w magazynie

Metody klasy Magazine:
- **addProduct(Product* p)** - metoda dodająca gotowe produkty do magazynu
- **getProducts()** - metoda pokazująca listę gotowych produktów w magazynie

- **Klasa Ramp:** klasa opisująca rampy. Charakteryzują ją: częstotliwość dostaw produktów, lista pracowników, którzy mają otrzymać produkt oraz ich preferencje.

Pola klasy Ramp:
- **frequency(float)** - jest to częstotliwość z jaką odbywają się dostawy półproduktów do ramp
- **rampReceiver(vector<Worker*>)** - wektor pracowników, którzy otrzymają produkt do przetwarzania
- **rampReceiverPref(vector<float>)** - wektor wartości typu zmiennoprzecinkowego opisujący preferencje dostarczania produktów do poszczególnych pracowników

Metody klasy Ramp:
- **Ramp(float freq)** - jest to konstruktor tworzący obiekt typu Ramp, który jako parametr przyjmuje liczbę określająca częstotliwość
- **addReceiver(Worker* w, float pref = -1)** - metoda dodająca odbiorcę - pracownika, jako wartość domyślną preferencji otrzymuje -1, w celu łatwego zbadania czy preferencje są ustawione czy mają zostać podzielone równo między robotników

- **update(float time)** - metoda aktualizująca - sprawdza czy już jest czas aby rampa otrzymała nowy produkt i jeśli tak to przekazuje go od razu do odpowiedniego pracownika
- **getFrequency()** - metoda, która zwraca częstotliwość
- **getReceivers()** - metoda, która zwraca listę odbiorców - robotników

- **Klasa Product:** klasa opisująca produkt. Posiada tylko trzy składowe: identyfikator produktu, konstruktor i metodę zwracającą identyfikator produktu

Pola klasy Product:

- **_id(float)** - każdy półprodukt dostarczany do fabryki ma swój unikalny identyfikator

Metody klasy Product:

- **Product(int id)** - konstruktor tworzący obiekt typu Product. Jako argument przyjmuje identyfikator.
- **getID()** - metoda, która zwraca identyfikator produktu