PROI 3. Sprawozdanie

Treść zadania:

Proszę zaimplementować system zarządzający zatowarowaniem oraz asortymentem sklepu-drogerii. Sklep musi obsługiwać co najmniej 100 produktów. (Uwaga: liczba klas powinna być mniejsza niż liczba produktów. Np. szampon męski X i szampon damski Y mogą być obiektami klasy Szampon z różnymi wartościami pól typu: firma, nazwa, płeć, objętość, cena, itp.). Należy opracować hierarchię klas w taki sposób, by można było opisać cechy każdego z produktów uwzględniając ich specyfikę, czyli np. dla jednych produktów istotna będzie objętość, inny będzie na sztuki, itp., dla jednych produktów istotny będzie podział na męski i damski lub dla dzieci, inne produkty są unisex. System ma umożliwiać przegląd stanu sklepu oraz operacje kupienia lub sprzedaży produktów (w tym hurtowo).

Opis rozwiązania:

Zamiast definiowania 100 klas produktów, stworzono klasy dla szczególnych grup produktów, z możliwością zmiany wartości pól odpowiednich klas.

Klasa ***Store*** jest główną klasą programu.

Prywatne składowe:

1)string name_of_the_store –objekt, zawiera nazwę sklepu,

2)vector<Product *> products – wektor, zawiera wskaźniki na obiekty klasy Product.

Klasa ***Product***:

Prywatne składowe:

1)string description – nazwa produktu

2)string firm – nazwa firmy lub producenta,

3)double price – cena produktu

4)bool out_of_date – informuje o tym, czy wygasa data ważności produktu

Składowa protected:

1)double availible – liczba dostępnych w sklepie produktów.

Hierarchia dziedziczenia:

1)klasa bazowa ***Product***

2)klasy pochodne: ***Drink, Vegetable, Cheese, Fruit, Cookie, Jerky,***

W pochodnych klasach były stworzone składowe, zawierające informację o cechach produktów, takich jak objętość, waga, etc.

Konstruktory, zdefiniowane w każdej klasie pochodnej:

• Konstruktor tworzący nowy obiekt przy użyciu już istniejącego obiektu klasy bazowej.

• Konstruktor tworzący nowy obiekt od początku.

Interfejs:

• Przeglądanie stanu sklepu za pomocą funkcji:

void show_store_status() const,

• Operacja kupna produktów za pomocą funkcji:

bool buy(Product * p(wskaźnik na produkt), double n(liczba kupionych produktów)),

• Operacja sprzedaży produktów za pomocą funkcji:

bool sell(Product * p(wskaźnik na produkt), double n(liczba kupionych produktów)),

Klasa ***Product*** posiada dwie funkcje służące do zmiany liczby dostępnych produktów:

• void add(double n=0) – dodaje n produktów,

• bool takeoff(double n=0) - odejmuje n produktów(o ile można odjąć taką liczbę),

• virtual void show() const - metoda wirtualna do wypisywania danych o produkcie.