Treść zadania

1. **Utwórz klasę** Adres z polami:

• ulica,

• kodPocztowy,

• miasto.

Dodaj konstruktor do inicjalizacji wszystkich pól oraz metodę wyswietlAdres(), która wypisze te dane w przyjaznym formacie.

2. **Zaimplementuj klasę** Klient, która zawiera:

Pola:

• imie

• nazwisko

• Adres adres (czyli obiekt klasy Adres)

• Konstruktor przyjmujący wszystkie niezbędne dane.

• Metodę daneKlienta(), która wyświetli imię, nazwisko oraz odwoła się do adres.wyswietlAdres().

3. **Przygotuj klasę abstrakcyjną** Zamowienie**:**

Pola wspólne:

• numerZamowienia

• dataZlozenia

• Klient klient

Metody wirtualna:

• obliczKoszt() const = 0;

• wyswietlSzczegoly() const;

• Konstruktor przyjmujący numer, datę oraz **obiekt** Klient.

4. **Klasy pochodne:**

ZamowienieProdukt**:**

• Pola:

• nazwaProduktu

• cenaJednostkowa

• ilosc

• Metody:

• double obliczKoszt().

• wyswietlSzczegoly() const – wyświetla najpierw wspólne dane (metoda bazowa), potem szczegóły produktu.

ZamowienieUsluga**:**

• Pola:

• rodzajUslugi

• stawkaZaGodzine

• liczbaGodzin

Metody:

• obliczKoszt()

• wyswietlSzczegoly() const override – wyświetla dane wspólne, następnie szczegóły usługi.

5. **Obsługa VAT lub rabatów (wybierz jedną z opcji lub obie):**

• Możesz dodać do każdej klasy pochodnej (lub do klasy bazowej) pole stawkaVAT (np. 23%) i w metodzie obliczKoszt() zwracać koszt wraz z podatkiem.

• dodać **rabat** (np. 10% dla zakupów powyżej pewnej kwoty) i uwzględnić go w obliczKoszt().

• Obliczenia z VAT/rabatem mogą wymagać dodatkowych parametrów w konstruktorach albo w polach klasy.

6. **Klasa** MagazynZamowien (lub KoszykZamowien):

• Przechowuje listę zamówień.

• Metody:

• dodajZamowienie(...) – dodaje zamówienie do kontenera

• obliczKosztCalkowity() – iteruje po wszystkich zamówieniach, sumuje wyniki obliczKoszt().

• Może się przydać obsługa wyjątków – jeśli w trakcie obliczania kosztu któregoś zamówienia wystąpi błąd walidacji, należy go obsłużyć.

• wyswietlWszystkieZamowienia() – wywołuje wyswietlSzczegoly() dla każdego obiektu.

7. **Klasa** Faktura:

• Posiada listę wybranych zamówień

• Metody:

• dodajZamowienieDoFaktury(...) – np. przekazać informację konkretnego zamówienia.

• generujFakture() – wyświetla listę zamówień na fakturze, z podsumowaniem kosztów.

• dodać numer faktury, datę wystawienia

8. **Funkcja** main()**:**

• Utwórz kilka obiektów Adres i na ich bazie – obiekty Klient.

• Utwórz obiekty ZamowienieProdukt oraz ZamowienieUsluga z różnymi wartościami (kilka z nich może świadomie zawierać błędne dane, żeby przetestować obsługę wyjątków).

• Dodaj zamówienia do MagazynZamowien.

• Wyświetl szczegóły wszystkich zamówień (wywołując wyswietlWszystkieZamowienia()).

• Spróbuj obliczyć łączny koszt (obliczKosztCalkowity()), obsługując ewentualne wyjątki.

• Utwórz obiekt klasy Faktura, dodaj do niego wybrane zamówienia (np. kilka z nich) i wywołaj generujFakture(), aby wyświetlić czytelne podsumowanie.

Wymagania funkcjonalne

1. **Każde zamówienie musi mieć przypisanego klienta**, więc w konstruktorze Zamowienie należy wymagać obiektu Klient.

2. **Walidacje**:

• Jeśli ilosc <= 0 lub cenaJednostkowa < 0 w ZamowienieProdukt, należy rzucić wyjątek (np. std::runtime_error("Błędne dane produktu!")).

• Jeśli liczbaGodzin <= 0 lub stawkaZaGodzine < 0 w ZamowienieUsluga, również należy rzucić wyjątek.

3. **Obsługa wyjątków**:

• W funkcji main(), podczas dodawania bądź obliczania kosztu w MagazynZamowien, użyj try-catch, aby przechwycić ewentualne błędy i wyświetlić komunikat informujący o problemie.

• W przypadku wystąpienia wyjątku dla konkretnego zamówienia, można np. pominąć je w dalszych obliczeniach lub podjąć inną decyzję biznesową (zgodnie z logiką systemu).

4. **VAT lub rabaty** – wprowadź stawkę VAT (np. 23%) lub rabat (np. 5–10% przy przekroczeniu określonego progu) i zastosuj ją w metodach obliczKoszt(). Możesz:

• Zrobić to na poziomie klas ZamowienieProdukt i ZamowienieUsluga,

• Lub w innej klasie (np. Faktura), w której przy sumowaniu kosztów dokonuje się dopiero naliczania VAT/rabatu.

5. **Klasa** Faktura:

• Może mieć pole numerFaktury, dataWystawienia i listę/wskaźniki do zamówień, które mają być na fakturze.

• Metoda generujFakture() wyświetla informacje w czytelnej formie:

• Numer faktury, data, dane zamówień (najlepiej: numer zamówienia, klient, koszt netto, ew. koszt brutto), sumę końcową.

• Możesz zadecydować, czy poszczególne zamówienia mają być dołączone w wersji brutto, czy brutto obliczane jest przy generowaniu faktury.