**МІНІСТЕРСТВО ОСІТИ ТА НАУКИ УКРАЇНИ ОДЕСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ**

**ІМЕНІ І.І.МЕЧНИКОВА**

**Індивідуальна робота з теми “Програмування(Об’єктні технології та мови) “**

**Варіант 21б(Гіпермаркет)**

Зробив:

Студент 1 курсу спеціальності комп’ютерна інженерія

Денисенко Олександр Володимирович

Викладач:

Антонено Олександр Сергійович

**ПЛАН**

1. **Завдання.**
2. **Опис програми. Характеристика класів та діаграма.**

**3. Висновок.**

**Завдання**

Створити класи: "Товар" (фірма, найменування, ціна, максимальний відсоток знижки), "Побутовий прилад", "Пилосос" (спадкоємець Побутового приладу), "Фотоапарат", "Дзеркальний Фотоапарат" (спадкоємець Фотоапарата), "Ноутбук" (розмір діагоналі, вага, кількість ядер процесора, пам'ять). Також реалізувати клас "Покупець" (поле: кількість грошей), і його спадкоємця "Постійний покупець" (поле: ПІБ, загальна вартість куплених товарів). Реалізувати віртуальний метод "індивідуальна знижка покупця" - для звичайного покупця повертає 0%, для постійного = ПІБ, загальна вартість куплених товарів ділити на 1000, але не більше 15%. Також реалізувати метод купити товар, який обчислює ціну товару, враховуючи знижку покупця (якщо знижка покупця більша за максимальну знижку для цього товару, то застосовується максимальний відсоток знижки товару). При цьому перевіряється, чи вистачить покупцеві грошей, і якщо так, то від його грошей віднімається вартість товару (враховуючи знижку). Для постійного покупця після покупки також відповідно збільшується загальна вартість куплених товарів.

В основному проєкті передбачити можливість вводити інформацію про товари і покупців. Також передбачити можливість здійснювати купівлю товарів покупцями.

**ОПИС ПРОГРАМИ**

**Клас** Product

Клас Product є базовим класом для представлення товарів у цьому коді. Він інкапсулює таку інформацію про продукт, як:

1. company: назва компанії-виробника товару (тип std::string).

2. title: назва продукту (тип std::string).

3. price: ціна продукту (тип double).

4. maxDiscount: максимальна знижка, яку можна застосувати до цього товару (тип double).

**Конструктор**:

Цей клас містить конструктор, який приймає всі вищевказані параметри і проводить валідацію вхідних даних. Наприклад, назва компанії та товару не повинні бути порожніми, ціна та максимальна знижка не повинні бути від'ємними. Конструктор:

Product(const std::string& company, const std::string& title, double price, double maxDiscount)

: company(company), title(title), price(price), maxDiscount(maxDiscount) {

if (company.empty()) {

throw std::invalid\_argument("Company cannot be empty");

}

if (title.empty()) {

throw std::invalid\_argument("Title cannot be empty");

}

if (price < 0) {

throw std::invalid\_argument("Price cannot be negative");

}

if (maxDiscount < 0 || maxDiscount > 1) {

throw std::invalid\_argument("Invalid maximum discount");

}

}

**Методи**:

Клас Product також надає методи для доступу до цих полів (getCompany, getTitle, getPrice, getMaxDiscount), які дають змогу витягувати відповідні значення.

Особливістю класу Product є наявність віртуального методу calculateDiscount(), який обчислює підсумкову ціну товару з урахуванням застосованої знижки. Цей метод може бути перевизначений у похідних класах для врахування специфічної логіки розрахунку знижок для різних типів товарів.

virtual double calculateDiscount(double customerDiscount) const {

double applicableDiscount = std::min(customerDiscount, maxDiscount);

return price \* (1 - applicableDiscount);

}

Також клас Product має віртуальний деструктор, що важливо для правильного керування пам'яттю під час використання поліморфізму.

virtual ~Product() {}

**Класи які наслідуються від Product:**

Класи HouseholdAppliance, VacuumCleaner, Camera, і DSLRСamera всі є спадкоємцями класу Product, який представляє загальну концепцію товару в цій програмі. Кожен із цих класів успадковує основні властивості продукту, як-от назва, ціна та максимальна знижка, і використовує їх для представлення більш специфічних типів товарів. HouseholdAppliance і VacuumCleaner представляють побутову техніку і пилососи відповідно, тоді як Camera і DSLRCamera представляють камери і DSLR-камери відповідно. Усі вони використовують конструктори базового класу для ініціалізації своїх властивостей і не додають нових полів або методів, за винятком Laptop, який додає додаткові поля, специфічні для ноутбуків, як-от розмір екрана, вагу, кількість ядер процесора та об'єм пам'яті.

**Класи Customer та RegularCustomer:**

Клас Customer представляє клієнта, який здійснює покупки

Його елементи:

Зміна money: це приватне поле, яке представляє кількість грошей, якими володіє покупець. Значення цього поля зменшується при кожній успішній покупці.

Конструктор:

Customer(double money): цей конструктор приймає кількість грошей, якими володіє покупець. Якщо вхідне значення негативне, генерується виняток std::invalid\_argument.

Customer(double money) : money(money) {

if (money < 0) {

throw std::invalid\_argument("Money cannot be negative");

}

}

**Методи:**

**getMoney():** цей метод повертає поточну суму грошей покупця.

**calculateIndividualDiscount():** це віртуальний метод, який призначений для перевизначення в класах-нащадках. У класі Customer цей метод просто повертає 0, оскільки передбачається, що у звичайного покупця немає індивідуальної знижки.

**buyProduct(const Product& product):** цей метод дозволяє покупцеві купувати товари. Він приймає на вхід посилання на об'єкт типу Product і обчислює вартість цього товару з урахуванням максимально можливої знижки та індивідуальної знижки покупця. Якщо у покупця достатньо грошей, він купує товар, і вартість товару списується з рахунку покупця. Якщо грошей недостатньо, виводиться повідомлення "Insufficient funds." і метод повертає false.

**Деструктор**

**~Customer():** цей віртуальний деструктор дає змогу коректно знищити об'єкти класів-нащадків, коли вони представлені як покажчики на об'єкти базового класу Customer.

Клас RegularCustomer

Клас RegularCustomer представляє постійного покупця і успадковується від класу Customer. Він містить такі додаткові елементи:

**fullName**: це приватне поле, яке зберігає повне ім'я постійного покупця.

**totalPurchaseCost**: це приватне поле, що зберігає загальну суму покупок, здійснених цим покупцем.

**Конструктор:**

Конструктор RegularCustomer(const std::string& fullName, double money): этот конструктор принимает полное имя и количество денег, которыми владеет покупатель. Если поле fullName пусто, генерируется исключение std::invalid\_argument**.**

RegularCustomer(const std::string& fullName, double money)

: Customer(money), fullName(fullName), totalPurchaseCost(0.0) {

if (fullName.empty()) {

throw std::invalid\_argument("Full name cannot be empty");

}

}

**Методи:**

**getFullName():** цей метод повертає повне ім'я постійного покупця**.**

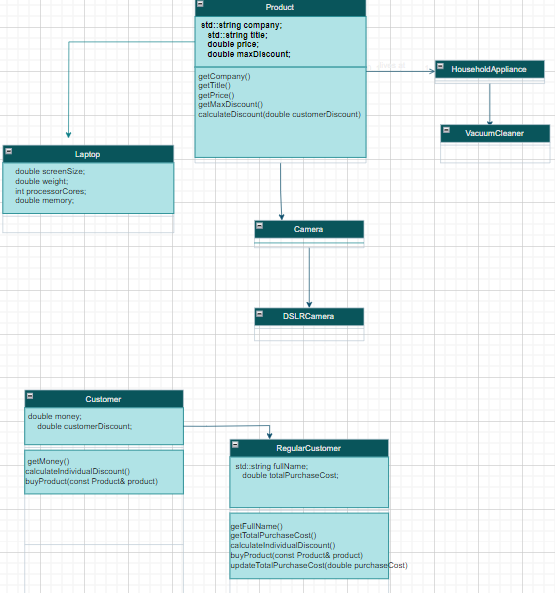
**getTotalPurchaseCost():** цей метод повертає загальну вартість усіх покупок, здійснених постійним покупцем.

**calculateIndividualDiscount():** перевизначений метод, який обчислює індивідуальну знижку на основі загальної вартості всіх покупок.

**buyProduct(const Product& product):** цей метод дозволяє постійному покупцеві купувати товари. Він працює так само, як і в базовому класі Customer, але також оновлює загальну суму покупок після кожної успішної покупки.

**updateTotalPurchaseCost(double purchaseCost):** цей метод оновлює загальну суму покупок постійного покупця після кожної успішної покупки. Він приймає вартість покупки як параметр.

**Діаграма класів**

****

**Висновок**

Ця програма представляє систему моделювання процесу покупки різних товарів. Вона використовує різні класи для представлення елементів системи. В основу програми закладено принципи об'єктно-орієнтованого програмування, які допомагають забезпечити структурованість, гнучкість та масштабованість коду.

Класи Product, HouseholdAppliance, VacuumCleaner, Camera, DSLRCCamera, Laptop представляють собою моделі продуктів. Вони інкапсулюють властивості товарів, а також дії, які можна з ними виконувати. Ці класи ілюструють такі принципи ООП, як наслідування та поліморфізм.

Класи Customer та RegularCustomer використовуються для моделювання поведінки покупців. Особливість цих класів полягає в тому, що вони відображають не тільки статичні характеристики покупців, але й їхню поведінку під час покупки товарів.

Функція main() координує взаємодію між цими класами. Вона створює об'єкти класів, викликає методи цих об'єктів, а також виводить результати взаємодії між об'єктами.

В загальному плані програма виконує своє завдання, моделюючи процес покупки різних товарів різними покупцями. Код написано в структурований і гнучкий спосіб, що дозволяє додавати нові класи продуктів або покупців, а також змінювати правила обчислення знижок.