Дисциплина «Программирование корпоративных систем» Рабочая тетрадь 5

ПО "Система складского учета"

|  |
| --- |
| **Теоретический материал** |
| C# является полноценным объектно-ориентированным языком. Это значит, что программу на C# можно представить в виде взаимосвязанных взаимодействующих между собой объектов.  Описанием объекта является класс, а объект представляет экземпляр этого класса.  Класс может хранить некоторые данные. Для хранения данных в классе применяются поля. По сути поля класса - это переменные, определенные на уровне класса.  Кроме того, класс может определять некоторое поведение или выполняемые действия. Для определения поведения в классе применяются методы.  Для обращения к функциональности класса - полям, методам (а также другим элементам класса) применяется точечная нотация точки - после объекта класса ставится точка, а затем элемент класса:  **объект.поле\_класса;**  **объект.метод\_класса(параметры\_метода);**  После определения класса мы можем создавать его объекты. Для создания объекта применяются конструкторы. По сути конструкторы представляют специальные методы, которые называются так же как и класс, и которые вызываются при создании нового объекта класса и выполняют инициализацию объекта. Общий синтаксис вызова конструктора:  **new конструктор\_класса(параметры\_конструктора);**  Ключевое слово this представляет ссылку на текущий экземпляр/объект класса. В каких ситуациях оно нам может пригодиться?  **public СlassPrimer(string str, int a) { this.str = str; x = a; }**    Первая часть - this.str означает, что str - это поле текущего класса, а не название параметра str.  Если бы у нас параметры и поля назывались по-разному, то использовать слово this было бы необязательно. Также через ключевое слово this можно обращаться к любому полю или методу.  Обычно определяемые классы и другие типы в .NET не существуют сами по себе, а заключаются в специальные контейнеры - пространства имен. Пространства имен позволяют организовать код программы в логические блоки, позволяют объединить и отделить от остального кода некоторую функциональность, которая связана некоторой общей идеей или которая выполняет определенную задачу.  Для определения пространства имен применяется ключевое слово namespace, после которого идет название пространства имен:  **namespace имя\_пространства\_имен**  **{**  **// содержимое пространства имен**  **}**  Все поля, методы и остальные компоненты класса имеют модификаторы доступа. Модификаторы доступа позволяют задать допустимую область видимости для компонентов класса. То есть модификаторы доступа определяют контекст, в котором можно употреблять данную переменную или метод.  В языке C# применяются следующие модификаторы доступа:  **private**: закрытый или приватный компонент класса или структуры. Приватный компонент доступен только в рамках своего класса или структуры.  **private** **protected**: компонент класса доступен из любого места в своем классе или в производных классах, которые определены в той же сборке.  **file**: добавлен в версии C# 11 и применяется к типам, например, классам и структурам. Класс или структура с такми модификатором доступны только из текущего файла кода.  **protected**: такой компонент класса доступен из любого места в своем классе или в производных классах. При этом производные классы могут располагаться в других сборках.  **internal**: компоненты класса или структуры доступен из любого места кода в той же сборке, однако он недоступен для других программ и сборок. **protected** **internal**: совмещает функционал двух модификаторов protected и internal. Такой компонент класса доступен из любого места в текущей сборке и из производных классов, которые могут располагаться в других сборках.  **public**: публичный, общедоступный компонент класса или структуры. Такой компонент доступен из любого места в коде, а также из других программ и сборок.    Кроме обычных методов в языке C# предусмотрены специальные методы доступа, которые называют свойства. Они обеспечивают простой доступ к полям классов и структур, узнать их значение или выполнить их установку.  Стандартное описание свойства имеет следующий синтаксис:  **[модификаторы] тип\_свойства название\_свойства**  **{**  **get { действия, выполняемые при получении значения свойства}**  **set { действия, выполняемые при установке значения свойства}**  **}** |

|  |  |
| --- | --- |
|  | |
| **Задание 1** | |
| ***Задача:*** | |
|  | **Система складского учета**  Разработать ПО со следующей архитектурой классов и функционалом:  **Класс «Склад»:**  Хранимая информация:   * + - Id склада (целое число);     - Тип (тип строка (холодный, сортировочный, общий, утилизация));     - Объем склада (вещественное число);     - Адрес (строка);     - Массив хранимых товаров (массив объектов класса «Товар»)   Методы:   * + - Создание склада;     - Редактирование склада;     - Вывод информации о складе;   **Класс «Товар»:**  Хранимая информация:   * + - Id Товара (целое число);     - Id Поставщика (целое число);     - Название (строка);     - Объем одной единицы товара (вещественное число);     - Цена одной единицы товара (вещественное число);     - Количество дней до окончания срока годности (целое число).   Методы:   * + - Создание товара;     - Изменение товара;     - Вывод информации о товаре;     - Удаление товара.     **Общие требования к функционалу:**   * Метод поставки товаров (в одной поставке может быть набор разных товаров), поставка должна подвергаться автоматической логистической оптимизации: 1) Если у всех товаров поставки срок хранения >=30 дней, то поставка направляется на общий склад/склады с подходящим свободным объемом. 2) Если у всех товаров поставки срок хранения <30 дней, то поставка направляется на холодный склад/склады с подходящим свободным объемом. 3) Если в поставке есть товары с кратким (<30) и длительным (>=30) сроком хранения, то поставка направляется на сортировочный склад/склады с подходящим свободным объемом. * Метод внутреннего оптимизационного перемещения товаров, должен проводить анализ всех сортировочных складов/конкретного склада, после чего товары с этих складов/склада должны быть распределены по складам в соответствии со своим сроком хранения. * Метод внутреннего перемещения товаров должен производить перемещение указанных товаров с одного выбранного склада на другой выбранный склад. * Метод внутреннего перемещения товаров с истекшим сроком годности, должен проводить анализ всех складов/конкретного склада и производить перемещения товаров с истекшим сроком годности (<=0) на склад/склады утилизации. * Метод анализа складской сети должен выдавать статус каждого склада (нарушений нет/ нарушения есть) с информацией о необходимости проведения для него: одного или нескольких конкретных методов перемещения. * Метод подсчета стоимости товара на конкретном складе.   Все методы перемещение и поставки товаров должны выдавать итоговый результат в формате логирования: «Товар», «объем», «от куда», «куда».  Итоговый проект должен содержать> =3 файла классов.  Разрабатываемый программный продукт должен содержать использование свойств, и не содержать нарушения принципов ООП с точки зрения применения модификаторов доступа. |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |