Лабораторная работа № 2

Ассоциации между классами.
Распределение обязанностей

Время выдачи работы: 3-я неделя.

Срок защиты работы: 6-я неделя (до 16.03.2013).

1. ЦЕЛИ РАБОТЫ

Приобрести навыки реализации объектно-ориентированной модели и распределения обязанностей между объектами модели.

2. ВОПРОСЫ ДЛЯ ИЗУЧЕНИЯ

Изучить интерфейс класса `List<T>` (конструкторы, свойства, методы) [2, гл. 7, раздел «Lists, Queues, Stacks, and Sets» → «List<T> and ArrayList»)].

Изучить часто используемые классы библиотеки FCL [2, гл. 6, разделы «String and Text Handling», «Dates and Times», «Dates and Time Zones», «Formatting and Parsing», «Enums»].

3. ЗАДАНИЕ

3.1. Варианты №№ 1. «Служба доставки»

Модель модуля определения стоимости заказов для приложения службы доставки представлена на рис. 1.

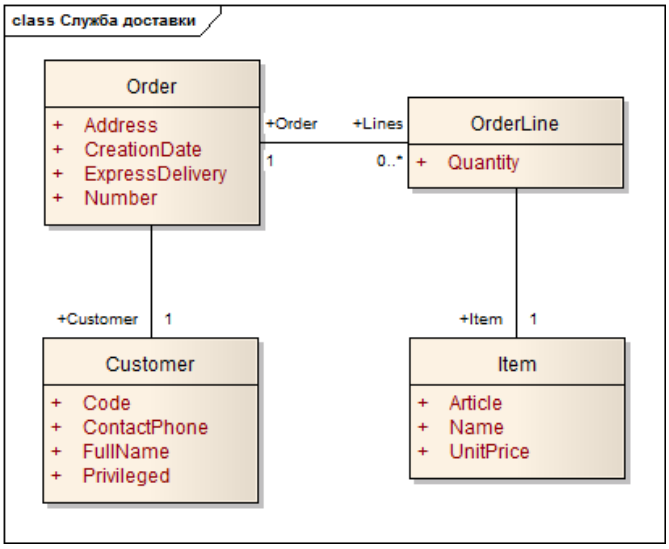

Рис. 1. Модель предметной области приложения «Служба доставки»

Клиент компании (сущность `Customer`) имеет следующие атрибуты: персональный код (`Code`), Ф.И.О. (`FullName`), контактный телефон (`ContactPhone`). Некоторые клиенты являются «привилегированными (`Privileged`)».

Каждый заказ (класс `Order`) идентифицируется номером (атрибут `Number`) и описывается следующими атрибутами: датой поступления (`CreationDate`), адресом доставки (`Address`) и пометкой, является ли заказ срочным (`ExpressDelivery`).

Один заказ подразумевает доставку одного или более товаров. Позиция заказа представлена сущностью `OrderLine` и имеет следующие атрибуты: заказанный товар/услуга (`Item`) и количество (`Quantity`).

Товар (сущность `Item`) описывается артикулом (`Article`), наименованием (`Name`) и ценой единицы товара (`UnitPrice`).

Стоимость позиции заказа определяется как произведение цены единицы товара/услуги на заказанное количество (`Cost = Item.UnitPrice * Quantity`).

Общая стоимость заказа (`TotalCost`) определяется как сумма стоимостей всех его позиций. Срочная доставка увеличивает стоимость заказа на 25%. Если заказчик – «привилегированный» клиент, и общая стоимость превысила 1 500, то предоставляется 15% скидка.

Необходимо реализовать классы `Customer`, `Item`, `Order`, `OrderLine`. Реализованная модель должна позволять:

- создавать заказ, указав клиента, номер, дату, адрес доставки и «срочность»;
- формировать заказ, т. е. добавлять, редактировать и удалять позиции (при добавлении позиции заказа указывается товар и количество);
- определять общую стоимость заказа по описанным выше правилам.

В консольном приложении требуется продемонстрировать использование полученной модели.

Приложение должно запросить номер заказа, дату поступления, предложить выбрать клиента, запросить адрес доставки и ее «срочность». Далее в цикле должны вводиться позиции заказа: пользователь выбирает товар и указывает заказанное количество. После окончания формирования заказа приложение должно выводить его общую стоимость и стоимость с учетом скидки (если она предоставляется).

Для выбора клиентов и товаров должен использоваться предустановленный набор данных (2-3 клиента и 7-10 товаров).

3.2. Варианты №№ 2. «Управление задачами»

Модель приложения для управления задачами в рамках проектов представлена на рис. 2.

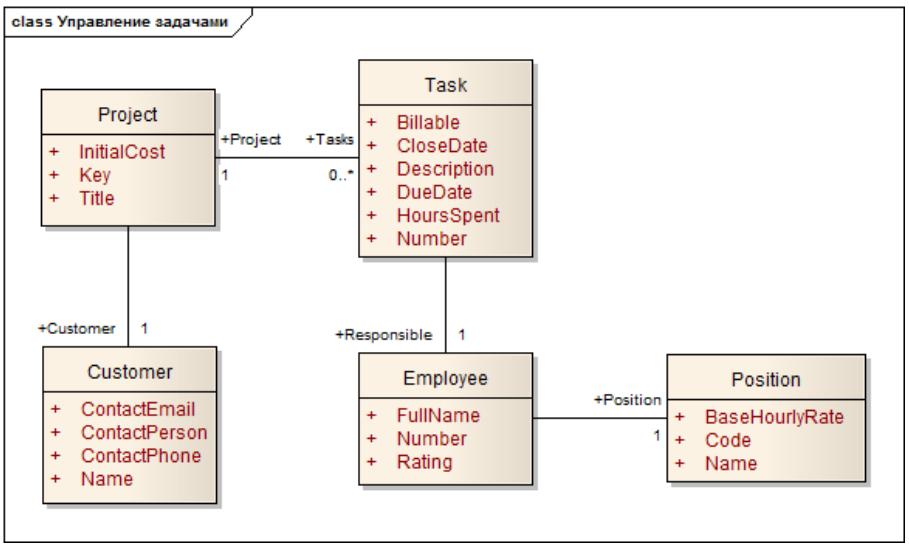

Рис. 2. Модель предметной области приложения «Управление задачами»

Проект (класс `Project`) определяется ключевой строкой (атрибут `Key`) и имеет следующие атрибуты: бюджет (`InitialCost`) и название (`Title`). Клиент, по заказу которого выполняется проект, представлен сущностью `Customer` со следующими атрибутами: наименование (`Name`), Ф.И.О. контактного лица (`ContactPerson`), контактный телефон (`ContactPhone`) и контактный e-mail (`ContactEmail`).

В рамках проекта выделяются задачи (класс `Task`). Задача представляется кратким номером (атрибут `Number`), подробным описанием (атрибут `Description`), сроком исполнения (атрибут `DueDate`), датой завершения работ (атрибут `CloseDate`), затраченным на выполнение временем (в часах; атрибут `HoursSpent`), признаком «Отдельно оплачивается заказчиком» (атрибут `Billable`).

За выполнение каждой задачи отвечает один из сотрудников компании. Сотрудник представлен сущностью `Employee` со следующими атрибутами: табельный номер (`Number`), Ф.И.О. (`FullName`), разряд (`Rating`, целое число от 1 до 5). Должность сотрудника моделируется сущностью `Position`, описываемой атрибутами: шифр (`Code`), название (`Name`) и базовая почасовая ставка (`BaseHourlyRate`). Стоимость часа работ сотрудника (обозначим ее `HourlyRate`) определяется как сумма базовой почасовой ставки, определяемой должностью, и надбавкой за разряд (5% от базовой ставки для 2-го разряда, 10% – для 3-го и т. д.; по 5% за каждый разряд). *Например*:

Захаров Степан Борисович, занимающий должность «ведущий разработчик» (базовая ставка: 300 руб./час) и имеющий 5-й разряд, должен получать 375 руб./час.

Стоимость задачи, отдельно оплачиваемой заказчиком, определяется следующим образом. Если работы выполнены в срок, то стоимость равна произведению стоимости часа работ ответственного сотрудника на затраченное время (`Cost = Responsible.HourlyRate * HoursSpent`). Если согласованный срок превышен, то за каждый день просрочки начисляется пеня 1%, но не более 25%.

Стоимость проекта определяется как сумма бюджета и стоимостей отдельно оплачиваемых задач.

Необходимо реализовать классы `Project`, `Customer`, `Task`, `Employee`. Реализованная модель должна позволять:

- создавать проект, указав клиента, ключевую строку, название и бюджет;
- добавлять, редактировать и удалять задачи (при добавлении задачи указываются ее атрибуты и ответственный сотрудник);
- определять общую стоимость проекта по описанным выше правилам.

В консольном приложении требуется продемонстрировать использование полученной модели.

Приложение должно запросить ключевую строку проекта, название, бюджет и предложить выбрать клиента. Далее в цикле должна вводиться информация о задачах проекта: пользователь вводит номер задачи, ее детальное описание, запланированную и фактическую даты выполнения, затраченное количество часов, выбирает ответственного сотрудника и указывает, оплачивается ли выполнение задачи отдельно. После окончания ввода информации о проекте и его задачах приложение должно выводить его общую стоимость.

Для выбора клиентов и сотрудников должен использоваться предустановленный набор данных (2-3 клиента и 7-10 сотрудников с 3-4 должностями).

3.3. Варианты №№ 3. «Учебная программа»

Фрагмент модели предметной области приложения, управляющего индивидуальными учебными программами студентов, представлен на рис. 3.

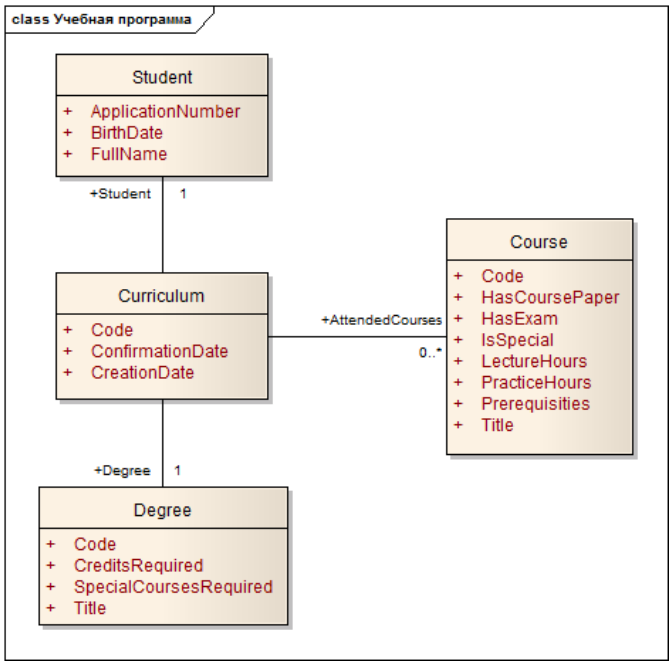

Рис. 3. Модель предметной области приложения «Учебная программа»

Университет предлагает студентам для изучения ряд учебных курсов, некоторые из них являются общеобразовательными, а некоторые – специальными. «Трудоемкость» курса определяется т. н. кредитными единицами, количество которых вычисляется следующим образом:

$$C = \left[\frac{L+1{,}25P}{18{,}00}\right],$$

где C – количество кредитных единиц курса,

L – количество часов лекций,

P – количество часов практических занятий.

Экзамен увеличивает трудоемкость курса на 1 кредитную единицу, а курсовая работа – на 2 единицы.

Кроме того, некоторые курсы могут быть выбраны только вместе с определенными курсами (*например*, если студент выбирает курс «Логика и теория алгоритмов», он обязательно должен прослушать и подготовительные курсы – «Дискретная математика» и «Теория вероятностей»).

Для получения квалификационной степени за все время обучения студент должен прослушать определенное количество спецкурсов, а так же набрать в сумме некоторое количество кредитных единиц. После зачисления в Университет, студент составляет индивидуальную учебную программу, содержащую список посещаемых курсов. Составленная программа является допустимой, если для каждого выбранного курса

были выбраны и все требуемые предварительные курсы, а так же выполняются требования квалификационной степени.

Студент представлен сущностью `Student` и имеет следующие атрибуты: номер заявления (`ApplicationNumber`), Ф.И.О. (`FullName`) и дата рождения (`BirthDate`).

Индивидуальная учебная программа студента представлена сущностью `Curriculum` и имеет следующие атрибуты: регистрационный код (`Code`), дата составления (`CreationDate`) и дата утверждения (`ConfirmationDate`).

Выбранный курс (класс `Course`) описывается следующими атрибутами: регистрационный код (`Code`), наименование (`Title`), общеобразовательный или спецкурс (флаг `IsSpecial`), количество часов лекций (`LectureHours`) , количество часов практики (`PracticeHours`), наличие экзамена (флаг `HasExam`), наличие курсовой работы (`HasCoursePaper`) и список регистрационных кодов курсов, которые должны быть обязательно выбраны вместе с данным курсом (атрибут `Prerequisities`).

Квалификационная степень (класс `Degree`) имеет следующие атрибуты: шифр (`Code`), наименование (`Title`), требуемое общее количество кредитных единиц (`CreditsRequired`) и минимальное количество спецкурсов (`SpecialCoursesRequired`)

Необходимо реализовать классы `Student`, `Degree`, `Curriculum`, `Course`. Реализованная модель должна позволять:

- создавать учебную программу, указав студента, получаемую квалификационную степень, регистрационный номер, дату заполнения;
- добавлять и удалять в программу курсы;
- определять допустимость составленной программы по описанным выше правилам.

В консольном приложении требуется продемонстрировать использование полученной модели.

Приложение должно запросить номер заявления, Ф.И.О. и дату рождения студента, регистрационный номер программы, дату заполнения и предложить выбрать квалификационную степень. Далее в цикле должны выбираться посещаемые учебные курсы. После окончания выбора курсов приложение должно выводить, допустима ли составленная программа.

Для выбора степеней и курсов должен использоваться предустановленный набор данных (2-3 степени и 15-20 курсов).

3.4. Варианты №№ 4. «Начисление зарплаты»

Модель приложения начисления зарплаты сотрудникам представлена на рис. 4.

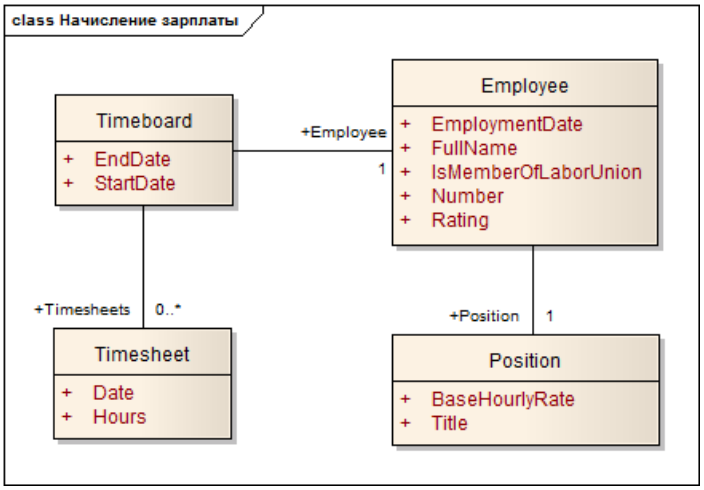

Рис. 4. Модель предметной области приложения «Начисление зарплаты»

Время, отработанное сотрудником компании, учитывается в табеле рабочего времени в виде записи: дата – число отработанных часов.

Стоимость часа работ сотрудника (обозначим ее `HourlyRate`) определяется как сумма базовой почасовой ставки, определяемой должностью, и надбавкой за разряд (10% от базовой ставки для 2-го разряда, 20% – для 3-го и т. д.; по 10% за каждый разряд). *Например*: Захаров Степан Борисович, занимающий должность «ведущий разработчик» (базовая ставка: 300 руб./час) и имеющий 5-й разряд, должен получать 420 руб./час.

Переработки (т. е. превышение 8-часового рабочего дня), а также работу в выходные дни (субботу и воскресенье) компания оплачивает по двойному тарифу.

Итоговая сумма, выплачиваемая сотруднику на руки, складывается из оплаты отработанного им времени, из которой удерживается 13% подоходного налога, а также 2% профсоюзных отчислений (если сотрудник является членом профсоюза).

Пример. Пусть почасовая ставка Захарова Степана Борисовича составляет 420 руб./час. В табеле рабочего времени имеются следующие записи:

01.02 (пятница)	10 час.
02.02 (суббота)	6 час.
04.02 (понедельник)	8 час.
05.02 (вторник)	8 час.
06.02 (среда)	6 час.

Тогда 2 часа переработок в пятницу 01.02, а также 6 часов работы в субботу 02.02 будут оплачиваться по двойному тарифу 840 руб./час. Остальное рабочее время (по 8 часов 01.02, 04.02, 05.02 и 6 часов 06.02) оплачивается по обычному тарифу 420 руб./час. Оплата рабочего времени составляет:

$$(2 + 6) \cdot 840 + (8 + 8 + 8 + 6) \cdot 420 = 6720 + 12600 = 19320 \text{ (руб.)}$$

Из этой суммы удерживается подоходный налог $19320 \cdot 0{,}13 = 2511{,}60$ (руб.) и профсоюзные отчисления (т. к. данный сотрудник является членом профсоюза): $19320 \cdot 0{,}02 = 386{,}40$ (руб.). Итого, «на руки» Степан Борисович должен получить 16 422 руб.

Сотрудник представлен сущностью `Employee` со следующими атрибутами: табельный номер (`Number`), Ф.И.О. (`FullName`), разряд (`Rating`, целое число от 1 до 5), дата приема на работу (`EmploymentDate`) и членство в профсоюзе (`IsMemberOfLaborUnion`).

Должность сотрудника моделируется сущностью `Position`, описываемой атрибутами: шифр (`Code`), название (`Name`) и базовая почасовая ставка (`BaseHourlyRate`).

Табель рабочего времени (сущность `Timeboard`) имеет атрибуты «Начало заполнения» (`StartDate`) и «конец заполнения» (`EndDate`) и содержит записи о рабочем времени – экземпляры сущности `Timesheet` (атрибуты: `Date` – дата и `Hours` – отработанное время).

Необходимо реализовать классы `Employee`, `Position`, `Timesheet`, `Timeboard`. Реализованная модель должна позволять:

- заполнять табель рабочего времени, указывая даты и отработанные часы;
- определять величину зарплаты за рабочее время, указанное в табеле.

В консольном приложении требуется продемонстрировать использование полученной модели.

Приложение должно запросить начальную и конечную даты табеля, табельный номер сотрудника, Ф.И.О., разряд, дату приема на работу, принадлежность профсоюзу и предложить выбрать должность. Далее в цикле должны создаваться записи о рабочем времени. После окончания приложение должно выводить величину заработной платы, которую сотрудник получит «на руки».

Для выбора должностей сотрудников должен использоваться предустановленный набор данных (5-7 должностей).

3.5. Варианты №№ 5. «Качество работы»

Модель приложения оценки качества работы предприятия представлена на рис. 5.

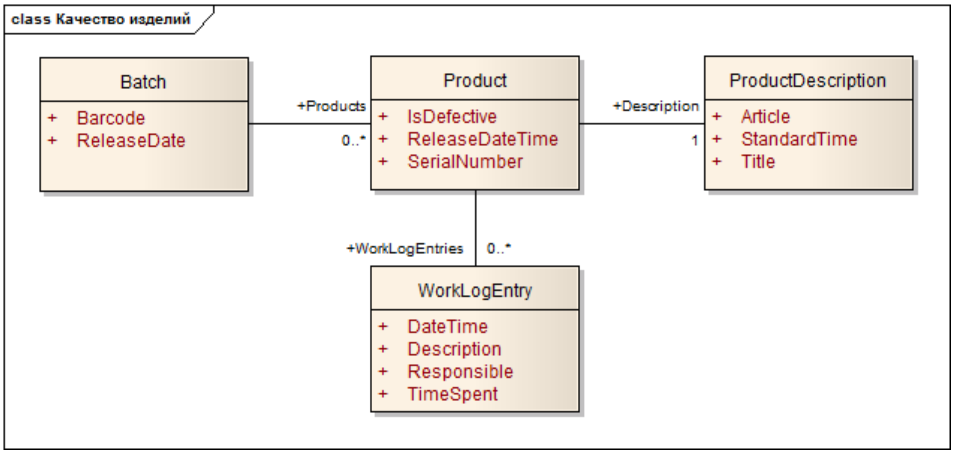

Рис. 5. Модель предметной области приложения «Качество работы»

Завод выпускает партии изделий. После выпуска и присвоения серийного номера каждое изделие проходит контроль качества, по результатам которого признается либо бракованным (и в дальнейшем направляется на утилизацию), либо годным. Все работы, выполняемые с изделием, фиксируются в журнале работ.

После выпуска партии присваивается категория, отражающая качество работы бригады:

- **1-я категория**: не более 3% брака; 80% изделий изготовлены за установленное нормативом время; для остальных изделий среднее время переработки не превышает 5 минут;
- **2-я категория**: не более 5% брака; 75% изделий изготовлены за установленное нормативом время;
- **3-я категория**: в остальных случаях.

Партия представляется классом `Batch` и имеет следующие атрибуты: штрих-код (`Barcode`) и дата выпуска (`ReleaseDate`).

Изделие представляется классом `Product` и имеет следующие атрибуты: серийный номер (`Barcode`), дата/время выпуска (`ReleaseDateTime`) и флаг «Брак» (`IsDefective`).

Сущность `ProductDescription` моделирует номенклатуру изготавливаемых изделий и содержит следующие атрибуты: артикул (`Article`), наименование (`Title`) и норма рабочего времени на изготовление (`StandardTime`).

Записи в журнале рабочего времени, потраченного на изготовление изделия, представлены сущностью `WorkLogEntry`. Эта сущность имеет следующие атрибуты: дата и время внесения записи (`DateTime`), рабочий, выполнявший работу (`Responsible`), время работ (`TimeSpent`), описание работ (`Description`).

Необходимо реализовать классы `Batch`, `Product`, `ProductDescription`, `WorkLogEntry`. Реализованная модель должна позволять:

- добавлять информацию об изделиях, входящих в партию;
- добавлять записи в журнал рабочего времени изделия;
- определять категорию качества партии.

В консольном приложении требуется продемонстрировать использование полученной модели.

Приложение должно запросить штрих-код партии и дату ее выпуска. Далее в цикле должны определяться изделия, входящие в партию: приложение запрашивает серийный номер, дату/время изготовления, является ли изделие бракованным, и просит выбрать номенклатуру изделия. Далее в цикле вводятся записи о рабочем времени, потраченном на изготовление изделия. После окончания приложение должно выводить категорию качества партии.

Для выбора номенклатур изделий должен использоваться предустановленный набор данных (5-7 номенклатур).

4. ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Опишите принцип «Command-Query Separation». Каково его назначение?
2. Что такое обязанность класса? Как следует распределять обязанности?
3. Для чего предназначен класс `List<T>`? Опишите его интерфейс.
4. Как представляются строки в .NET?
5. Как выполнить форматирование строки?
6. В чем особенности сравнения и упорядочения строк?
7. Опишите назначение и интерфейс класса `StringBuilder`.
8. Для чего предназначены классы `DateTime`, `TimeSpan`?

9. Что такое перечисления (enums)? Для чего они могут использоваться? Как с ними работать?

5. СПИСОК ОСНОВНОЙ ЛИТЕРАТУРЫ

1. .NET Framework 4.5. — Библиотека MSDN. — http://msdn.microsoft.com/ru-ru/library/w0x726c2.aspx

2. Joseph Albahari, Ben Albahari. C# 5.0 in a Nutshell. The Defenitive Reference, 5th Edition. — O'Reilly, 2012