Лабораторная работа № 4

Обработка исключений

1. ЦЕЛИ РАБОТЫ

Приобрести навыки использования механизма исключений для сигнализации об ошибочных ситуациях и их обработки.

2. ВОПРОСЫ ДЛЯ ИЗУЧЕНИЯ

Повторить использование перечислений [4, гл. 6, раздел «Enums»], [3, гл. 15].

Механизм исключений в C# и .NET Framework 4.5 [4, гл. 4, раздел «try Statements and Exceptions»)], [3, гл. 20].

Правила работы с исключениями [2], [3, гл. 20].

3. ЗАДАНИЕ

3.1. Варианты №№ 1, 6. «Служба доставки»

Заказ может находиться в одном из следующих состояний: «*формируется*», «*в обработке*», «*доставляется*», «*доставлен*».

В состоянии «*формируется*» заказ оказывается сразу после создания, в это время в него могут добавляться/удаляться позиции, меняться атрибуты «Срочная доставка», «Адрес доставки».

После окончательного согласования с клиентом заказа, он переводится в состояние «*в обработке*». В это время заказ комплектуется и готовится к доставке. У заказа, находящегося «*в обработке*», может изменяться только атрибут «Адрес доставки».

После того, как заказ сформирован и передан в службу доставки, он переводится в состояние «*доставляется*». Начиная с этого момента атрибуты и состав заказа редактироваться не могут.

После передачи клиенту и оплаты, заказ переходит в состояние «*доставлен*».

В рамках данной лабораторной работы необходимо внести следующие изменения в модель, разработанную в ходе выполнения лабораторных работ №№ 2-3:

- добавить в сущность «Заказ» атрибуты, содержащие текущий статус заказа, дату/время подтверждения заказа клиентом, дату/время передачи в службу доставки и дату/время доставки заказа клиенту;
- реализовать операции перехода заказа по состояниям.

Кроме того, необходимо для всех операций модели реализовать проверку предусловий:

- корректности переданных параметров;
- допустимости выполнения операции в текущем состоянии объекта.

Например, в качестве заказчика для заказа или товара для позиции заказа не может быть задано значение «`null`», в качестве количества товара в позиции заказа или стоимости товара не могут указываться отрицательные величины, заказ в состоянии «*доставляется*» нельзя перевести в состояние «*формируется*», у заказа в состоянии «*формируется*» нельзя удалить позицию и т. п.

Приложение не должно аварийно завершаться в случае ввода пользователем некорректных значений даты, чисел и т. п., а должно адекватно реагировать на ошибку. Например, выводить понятное сообщение и предлагать повторить ввод. (*Указание*: Ввод некорректных данных пользователем – это ситуация нормальная, а значит приводить к генерации исключений она не должна. Поэтому при реализации этого требования не следует ждать исключения и обрабатывать его в операторе `try/catch`, а заранее проверять корректность вводимых значений и на основе этой проверки действовать. В данной ситуации можно использовать методы `TryParse`)

Общий порядок работы с приложением остается, как и был реализован в лабораторных работах № 2 и № 3.

3.2. Варианты №№ 2, 7. «Управление задачами»

Задача может находиться в одном из следующих состояний: «*утверждается*», «*реализуется*», «*проверяется*», «*закрыта*».

После создания и до полного согласования с заказчиком задача находится в состоянии «*утверждается*». У задачи в этом состоянии могут редактироваться все атрибуты, кроме номера, даты завершения работ (`CloseDate`) и потраченного времени.

После утверждения задача переходит в состояние «*реализуется*»; в это время разрешается редактировать описание задачи и затраченное время (`HoursSpent`).

После реализации отдел тестирования выполняет проверку задачи (состояние «*проверяется*»), при этом может редактироваться только затраченное время (`HoursSpent`).

Если ошибок в реализации не обнаружено, то задача закрывается (состояние «*закрыта*»), при закрытии может быть указана дата завершения работ (`CloseDate`). Если были обнаружены ошибки в реализации, то задача возвращается в состояние «*реализуется*».

Задача в состоянии «*закрыта*» редактироваться не может.

В рамках данной лабораторной работы необходимо внести следующие изменения в модель, разработанную в ходе выполнения лабораторных работ №№ 2-3:

- добавить в сущность «Задача» атрибуты, содержащие текущее состояние задачи, дату/время утверждения задачи и дату/время завершения реализации;
- реализовать операции перехода задачи по состояниям.

Кроме того, необходимо для всех операций модели реализовать проверку предусловий:

- корректности переданных параметров;
- допустимости выполнения операции в текущем состоянии объекта.

Например, в качестве заказчика для проекта или должности сотрудника не может быть задано значение «`null`», в качестве затраченного на выполнение задачи времени или почасовой ставки сотрудника не могут указываться отрицательные величины, разряд сотрудника может быть только целым числом от 1 до 5, задачу в состоянии «*закрыта*» нельзя перевести в состояние «*реализуется*», у задачи в состоянии «*реализуется*» нельзя редактировать атрибут «Отдельно оплачивается заказчиком» и т. п.

Приложение не должно аварийно завершаться в случае ввода пользователем некорректных значений даты, чисел и т. п., а должно адекватно реагировать на ошибку. Например, выводить понятное сообщение и предлагать повторить ввод. (*Указание*: Ввод некорректных данных пользователем – это ситуация нормальная, а значит приводить к генерации исключений она не должна. Поэтому при реализации этого требования не следует ждать исключения и обрабатывать его в операторе

`try/catch`, а заранее проверять корректность вводимых значений и на основе этой проверки действовать. В данной ситуации можно использовать методы `TryParse`)

Общий порядок работы с приложением остается, как и был реализован в лабораторных работах № 2 и № 3.

3.3. Варианты №№ 3, 8. «Учебная программа»

Индивидуальная учебная программа студента может находиться в одном из следующих состояний: «*черновик*», «*составлена*», «*утверждена*», «*в архиве*».

После создания программа находится в состоянии «*черновик*». Студент осуществляет выбор курсов, формирование и проверку программы.

Когда программа готова, студент переводит ее в состояние «*составлена*» (в это состояние не может быть переведена недопустимая программа). Ответственный преподаватель осуществляет проверку, при необходимости корректирует программу, и либо утверждает ее, переводя в состояние «*утверждена*», либо отклоняет, переводя в состояние «*черновик*».

После завершения обучения студента программа переводится в состояние «*в архиве*».

В рамках данной лабораторной работы необходимо внести следующие изменения в модель, разработанную в ходе выполнения лабораторных работ №№ 2-3:

- добавить в сущность «Учебная программа» атрибуты, содержащие текущее состояние программы, дату/время составления и дату/время утверждения;
- реализовать операции перехода программы по состояниям.

Кроме того, необходимо для всех операций модели реализовать проверку предусловий:

- корректности переданных параметров;
- допустимости выполнения операции в текущем состоянии объекта.

Например, в качестве студента или получаемой степени в программе не может быть задано значение «`null`», в качестве часов лекций и практики не могут указываться отрицательные величины, программу в состоянии «*в архиве*» нельзя перевести в состояние «*черновик*», у задачи в состоянии «*утверждена*» нельзя добавлять и удалять курсы и т. п.

Приложение не должно аварийно завершаться в случае ввода пользователем некорректных значений даты, чисел и т. п., а должно адекватно реагировать на ошибку. Например, выводить понятное сообщение и предлагать повторить ввод. (*Указание*: Ввод некорректных данных пользователем – это ситуация нормальная, а значит приводить к генерации исключений она не должна. Поэтому при реализации этого требования не следует ждать исключения и обрабатывать его в операторе `try/catch`, а заранее проверять корректность вводимых значений и на основе этой проверки действовать. В данной ситуации можно использовать методы `TryParse`)

Общий порядок работы с приложением остается, как и был реализован в лабораторных работах № 2 и № 3.

3.4. Варианты №№ 4, 9. «Начисление зарплаты»

Табель рабочего времени может находиться в одном из следующих состояний: «*заполняется*», «*заполнен*», «*закрыт*».

После создания табель находится в состоянии «*заполняется*». В этом состоянии в табель вносятся записи о рабочем времени (таймшиты), могут редактироваться атрибуты «Начало заполнения» и «Конец заполнения».

После завершения заполнения табель переходит в состояние «*заполнен*». В этом состоянии осуществляется проверка введенных данных и расчет заработной платы.

После расчета заработной платы табель переводится в состояние «*закрыт*». В этом состоянии с табелем не могут выполняться никакие операции.

В рамках данной лабораторной работы необходимо внести следующие изменения в модель, разработанную в ходе выполнения лабораторных работ №№ 2-3:

- добавить в сущность «Табель рабочего времени» атрибуты, содержащие текущее состояние табеля, дату/время фактического завершения и дату/время закрытия;
- добавить в сущность «Табель рабочего времени» атрибут «Величина заработной платы» (`SalaryAmount`);
- реализовать операции перехода табеля по состояниям.

Кроме того, необходимо для всех операций модели реализовать проверку предусловий:

- корректности переданных параметров;
- допустимости выполнения операции в текущем состоянии объекта.

Например, в качестве сотрудника для табеля или должности для сотрудника не может быть задано значение «`null`», в качестве часовой ставки должности и отработанного времени не могут указываться отрицательные величины, табель в состоянии «*закрыт*» нельзя перевести в состояние «*заполняется*», в табель в состоянии «*заполнен*» нельзя добавлять и удалять записи о рабочем времени и т. п.

Приложение не должно аварийно завершаться в случае ввода пользователем некорректных значений даты, чисел и т. п., а должно адекватно реагировать на ошибку. Например, выводить понятное сообщение и предлагать повторить ввод. (*Указание*: Ввод некорректных данных пользователем – это ситуация нормальная, а значит приводить к генерации исключений она не должна. Поэтому при реализации этого требования не следует ждать исключения и обрабатывать его в операторе `try/catch`, а заранее проверять корректность вводимых значений и на основе этой проверки действовать. В данной ситуации можно использовать методы `TryParse`)

Общий порядок работы с приложением остается, как и был реализован в лабораторных работах № 2 и № 3.

3.5. Варианты №№ 5, 10. «Качество работы»

С точки зрения приложения, каждое изделие может находиться в одном из следующих состояний: «*изготавливается*», «*дорабатывается*», «*выпущена*» и «*утилизация*».

Сразу после создания сущность «Изделие» находится в состоянии «*изготавливается*».

После завершения изготовления изделие либо выпускается (осуществляется переход в состояние «*выпущена*» и фиксируется дата/время выпуска), либо признается бракованным.

Бракованное изделие либо утилизируется (осуществляется переход в состояние «*утилизация*» и указывается значение атрибута «Брак»), если дефекты неисправимы, либо направляется на доработку (осуществляется переход в состояние «*дорабатывается*» и указывается значение атрибута «Брак»).

Изделие в состоянии «*дорабатывается*» обрабатывается приложением по тем же правилам, что и изделие в состоянии «*изготавливается*» (т. е. может быть либо выпущено, либо отправлено на дальнейшую доработку, либо отправлено на утилизацию).

Для изделия в состояниях «*изготавливается*» и «*дорабатывается*» могут вносится записи в журнал рабочего времени.

В рамках данной лабораторной работы необходимо внести следующие изменения в модель, разработанную в ходе выполнения лабораторных работ №№ 2-3:

- добавить в сущность «Изделие» атрибут, содержащий текущее состояние изделия;
- реализовать операции перехода изделия по состояниям.

Кроме того, необходимо для всех операций модели реализовать проверку предусловий:

- корректности переданных параметров;
- допустимости выполнения операции в текущем состоянии объекта.

Например, в качестве номенклатуры изделия не может быть задано значение «`null`», в качестве затраченного времени в журнале не может указываться отрицательная величина, изделие в состоянии «*утилизация*» нельзя перевести в состояние «*изготавливается*», для изделия в состоянии «*выпущено*» нельзя добавлять и удалять записи о затраченном рабочем времени и т. п.

Приложение не должно аварийно завершаться в случае ввода пользователем некорректных значений даты, чисел и т. п., а должно адекватно реагировать на ошибку. Например, выводить понятное сообщение и предлагать повторить ввод. (*Указание*: Ввод некорректных данных пользователем – это ситуация нормальная, а значит приводить к генерации исключений она не должна. Поэтому при реализации этого требования не следует ждать исключения и обрабатывать его в операторе `try/catch`, а заранее проверять корректность вводимых значений и на основе этой проверки действовать. В данной ситуации можно использовать методы `TryParse`)

Общий порядок работы с приложением остается, как и был реализован в лабораторных работах № 2 и № 3.

4. ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое перечисления (enums)? Для чего они могут использоваться? Как с ними работать?
2. Опишите механизм обработки ошибок посредством исключений.
3. Приведите синтаксис генерации исключения. Назовите правила, по которым выбирается класс генерируемого исключения.
4. Приведите синтаксис конструкции `try/catch/finally`. Каково назначение ее блоков?
5. Назовите и охарактеризуйте основные стратегии обработки исключений в блоке `catch`.
6. Сформулируйте основные правила обработки исключений.
7. Как и для чего организуются цепочки исключений?
8. Опишите иерархию исключений библиотеки .NET Framework BCL.

5. СПИСОК ОСНОВНОЙ ЛИТЕРАТУРЫ

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

2. Design Guideliness for Exceptions. — Visual Studio. — http://msdn.microsoft.com/en-us/library/vstudio/ms229014(v=vs.100).aspx

3. Jeffrey Richter. CLR via C#, 4th Edition. — Microsoft Press, 2012

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