Упражнение 1: Обработка исключительных ситуаций

Задача 1. Откройте Switch.cs file  в Visual Studio.

2. Изучите класс.

Обратите внимание, что класс содержит несколько методов, каждый из которых способен выдать хотя бы одно исключение, зависящее от результата генерации случайного числа. В нижней части файла обратите внимание на определения каждого из пользовательских исключений, которые может создать класс Switch. Это очень простые exceptionclasses, которые просто содержат сообщение об ошибке.

Задание 3: Обрабатать исключения класса Switch

Проект SwitchTestHarness содержит ссылку на класс SwitchDevice и вызывает каждый метод в классе Switch для имитации опроса нескольких датчиков и диагностических устройств. В настоящее время проект не обрабатывает ни одного исключения, поэтому при возникновении исключения приложение завершится ошибкой. Для защиты приложения от исключений, создаваемых классом Switch, необходимо добавить код обработки исключений в проект SwitchTestHarness.

1. Откройте главное окно.код XAML.cs-файл в Visual Studio.

2. В классе MainWindow найдите метод Button1\_Click. Этот метод запускается, когда пользователь нажимает кнопку Shutdown.

3. Удалите комментарий **TODO - Add exception handling**, и затем найдите комментарии **Step 1 - disconnect from thePower Generator** и **Step 2 - Verify the status of the Primary Coolant System**. Заключите код между этими комментариями в try / catch блок, который ловит the**SwitchDevices.PowerGeneratorCommsException**. Это исключение, которое может выбросить метод **DisconnectPowerGenerator**.

4. В блоке catch добавьте код для добавления новой строки текста в элемент управления textBlock1 с сообщением "\*\*\*Exception in step 1:" , а затем содержимое свойства Message исключения. Свойство **Message**  содержит сообщение об ошибке, которое определяет объект Switch, при выбрасывании исключения.

Подсказка: чтобы добавить строку текста в элемент управления textblock, использовать оператор += к свойству Text элемента управления.

5. Напишите код между комментариями **Step 2 - Verify the status of the Primary Coolant System** и **Step 3 -Verify the status of the Backup Coolant System** в блоке try / catch, который ловит исключения SwitchDevices.CoolantPressureReadException и SwitchDevices.CoolantTemperatureReadExceptionexceptions. В каждом обработчике исключений, следуя той же схеме, что и Шаг 3, выведите сообщение в новой строке в элементе управления textBlock1 (обратите внимание, что это Шаг 2, а не Шаг 1 процесса завершения).

6. Напишите код между комментариями **Step 3 - Verify the status of the Backup Coolant System** и **Step 4 -Record the core temperature prior to shutting down the reactor**  в блоке try / catch, который ловит исключния SwitchDevices.CoolantPressureReadException и **SwitchDevices.CoolantTemperatureReadException**. В каждом обработчике исключений выведите сообщение в новой строке в элементе управления textBlock1 (это Шаг 3).

7. Напишите код между комментариями **Step 4 - Record the core temperature prior to shutting down the reactor** и **Step 5 - Insert the control rods into the reactor**, который ловит исключение **SwitchDevices.CoreTemperatureReadException** . В обработчике исключений выведите сообщение на новую строку в элементе управления textBlock1 (это Шаг 4).

8. Напишите код между комментариями **Step 5 – Insert the control rods into the reactor** and **Step 6 - Record the core temperature after shutting down the reactor**  в блоке try / catch, который ловит исключение **SwitchDevices.RodClusterReleaseException** . В обработчике исключений напечатайте сообщение на новой строке в элементе управления textBlock1 (это Шаг 5).

9. Напишите код между комментариями **Step 6 - Record the core temperature after shutting down the reactor** и **Step 7 - Record the core radiation levels after shutting down the reactor**  в try/Catch блок,который ловит SwitchDevices.Исключение CoreTemperatureReadException. В обработчике исключений выведите сообщение в новой строке в элементе управления textBlock1 (это шаг 6).

10. Напишите код между комментариями **Step 7 - Record the core radiation levels after shutting down the reactor** и **Step 8 - Broadcast "Shutdown Complete" message**  в блоке try / catch, который ловит **SwitchDevices.CoreRadiationLevelReadException** . В обработчике исключений выведите сообщение в новую строку в элементе управления textBlock1 (это шаг 7).

11. Напишите два оператора после комментария **Step 8 – Broadcast "Shutdown Complete" message**  в блоке try / catch, который ловит **SwitchDevices.SignallingException** . В каждом обработчике исключений напечатайте сообщение на новой строке в элементе управления textBlock1 (это шаг 8).

12. Постройте решение и исправьте все ошибки.

Задание 4: Тест приложения

Запустите приложение и нажмите кнопку завершение работы. Просмотрите сообщения, отображаемые в MainWindowwindow, и убедитесь, что исключения теперь перехватываются и регистрируются.

Примечание: класс коммутатора случайным образом генерирует исключения как раньше, таким образом, Вы не можете видеть никаких сообщений об исключениях первый раз, когда вы нажимаете кнопку. Повторите процесс нажатия кнопки и проверьте Выходные данные, пока не появятся сообщения об исключении.

Упражнение 2. Обнаружение исключения

В этом упражнении метод будет изменен таким образом, чтобы он выбрасывал исключение ArgumentException, если он является аргументом, содержащим ошибочные или недопустимые данные.

Сценарий

Одно из технических устройств, которое производит Fabrikam, Inc. выполняет несколько вычислений, включающих матрицы. Эти матрицы представляют координаты множества точек в границах многомерной сетки. Само устройство собирает данные для этих точек и составляет матрицы. Затем он использует метод C# для их перемножения вместе, чтобы создать новый набор точек данных. При нормальной работе, ни один из элементов матриц не должен быть отрицательным. Однако иногда данные, которые записывает устройство, содержат ошибку - если устройство определяет значение, которое находится вне диапазона, то генерирует значение -1 для точки данных. К сожалению, это код, который перемножитель матриц не может обнаружить и вычисляет результат, который является ошибочным. Вы были providedwith копию этого кода как метод, внедренный в приложение WPF.

Основными задачами данного упражнения являются:

1. Откройте решение MatrixMultiplication.

2. Добавьте код для создания исключений в методе MatrixMultiply.

3. Обработайте исключения, которые создает метод MatrixMultiply.

4. Выполнения тестов и тестирования приложения.

Задача 1. Откройте решение MatrixMultiplication

1. В Visual Studio откройте решение MatrixMultiplication из Ex2 \ Starter

2. Откройте Matrix.cs  а затем найдите метод MatrixMultiply.

Метод MatrixMultiply выполняет умножение двух матриц, которые принимает в качестве параметра и возвращают результат.

В настоящее время метод принимает матрицы любого размера, и выполняет проверку данных, содержащихся в матрицах. Вы добавите проверки, чтобы убедиться, что две матрицы совместимы (Количество столбцов в первой матрице равно количеству строк во второй матрице), и что никакое значение в ни в одной из матриц не является отрицательным числом.

Если матрицы несовместимы, или какая-то из них содержит отрицательное значение, метод должен выбросить исключение.

Задача 2. Добавление кода для создания исключений в методе MatrixMultiply

1. В методе MatrixMultiply, найдите и удалите комментарий **TODO – Evaluate input matrices forcompatibility**. Под блоком комментариев добавьте код для выполнения следующих действий:

а. Сравните количество столбцов в matrix1, чтобы количество строк в matrix2.

b. Выбросите исключение ArgumentException, если их значения не равны. В сообщении должно быть написано, что число столбцов и строк должно совпадать.

Подсказка: вы можете получить количество столбцов в матрице, изучив длину первого измерения. Количество строк в матрице можно получить, изучив длину второго измерения.

2. Найдите и удалите комментарий TODO – Evaluate matrix data points for invalid data. На этом этапе метод перебирает элементы в каждой матрице, умножая значение в каждой ячейке матрицы 1 на значение в соответствующей ячейке матрицы matrix2. Добавьте код под блоком комментариев, чтобы выполнить следующие действия:

a.Убедитесь, что значение в текущем столбце и строке матрицы 1 больше нуля. Переменные ячейки и строки содержат столбец и строку, которые необходимо проверить.

b.Бросьте исключение ArgumentException, если значение не больше нуля. Исключение должно содержать сообщение " Матрица 1 содержит недопустимую запись в ячейке[x, y]."где x и y - значения столбцов и строк ячейки.

3. Добавьте еще один блок кода, чтобы проверить, что значение в текущем столбце и строке matrix2 больше нуля. Если это не так, выбросьте исключение ArgumentException с сообщением " Matrix2 содержит неправильное значение в ячейке[x, y].".

Задание 3: Обработайте исключения, которые выбрасывает метод MatrixMultiply

1. Откройте окно MainWindow WPF в окне конструктора и просмотрите на него.

Это окно предоставляет пользовательский интерфейс, который позволяет пользователю вводить данные для двух матриц, которые будут перемножены. Пользователь нажимает кнопку **Calculate**, чтобы рассчитать и отобразить результат.

2. Откройте файл кода для окна MainWindow WPF.

3. В классе MainWindow найдите метод ButtonCalculate\_Click. Этот метод выполняется, когда нажимает кнопку **Calculate**.

4. В методе ButtonCalculate\_Click найдите строку кода, которая вызывает метод **Matrix**.**MatrixMultiply** и заключите эту строку кода в блок try / catch, который ловит исключение ArgumentException.

5. В блоке catch добавьте инструкцию, которая отображает окно сообщения, содержащее содержимое свойства Message объекта исключения.

Подсказка: Вы можете использовать метод **MessageBox.Show**  для отображения окна сообщения. Укажите сообщение todisplay как строку, передаваемую в качестве параметра для этого метода.

6. Постройте решение и исправьте все ошибки.

7. Запустить приложение без отладки.

8. В файле Mainwindow окне в первом выпадающем списке выберите Матрица 1: 2 колонки, в seconddrop меню окно, выберите Матрица 1: 2 ряда, а затем в третьем раскрывающемся списке, выберите Матрица 2: 2Columns.

Это создает пару матриц 2 × 2, инициализированных нулями.

9. Введите некоторые неотрицательные значения в ячейки обеих матриц и нажмите кнопку вычислить.

Убедитесь, что результат вычисляется и отображается, а исключения отсутствуют.

10. Введите одно или несколько отрицательных значений в ячейки в любой из матриц и нажмите кнопку вычислить еще раз.

Убедитесь, что отображается соответствующее сообщение об исключении и что оно идентифицирует матрицу и ячейку, которая является inerror.

11. Закройте окно MainWindow и вернитесь в Visual Studio.

Приложение создает и перехватывает исключения, поэтому необходимо проверить, что приложение работает должным образом.Хотя с помощью интерфейса приложения можно проверить наличие отрицательных точек данных, пользовательский интерфейс не позволяет создавать массивы различных измерений. Таким образом были предоставлены модульные тестовые случаи, которые будут вызывать theMatrixMultiply метод с данными, которые вызовут исключения. Эти тесты уже созданы; вы просто запустите их, чтобы убедиться, что ваш код работает должным образом.

Упражнение 3: Проверка на переполнение разрядной сетки

В этом упражнении вы изучите, что происходит по умолчанию, если вычисление целого числа вызывает числовое переполнение. Затем приложение будет изменено, чтобы проверить наличие исключений переполнения разрядной сетки и повторить вычисление.

Сценарий

Часть программного обеспечения для измерительного устройства выполняет целочисленное умножение, но целочисленные значения могут получиться очень большими. Необходимо убедиться, что программное обеспечение не создает ошибок, вызванных переполнением разрядной сетки.

Основными задачами данного упражнения являются:

1. Откройте решение IntegerOverflow.

2. Добавьте блок **checked**.

Задача 1. Откройте решение IntegerOverflow

1. Откройте решение IntegerOverflow в Ex3 \Starter folder.

2. Запустите приложение и нажмите кнопку умножить. Наблюдайте результат, который отображается и обратите внимание, что это неправильно.

Приложение 2147483647 умножает на 2, и отображает результат -2. Это происходит потому, что умножение привело к переполнению разрядной сетки. По умолчанию ошибки переполнения такого рода не выбрасывают исключения. Однако, во многих ситуациях, лучше поймать ошибки переполнения, чем позволить приложению продолжить работать с неправильными данными.

3. В Visual Studio в меню отладка выберите команду остановить отладку.

Задача 2. Добавление проверяемого блока

1. В Обозревателе решений откройте окно MainWindow.xaml.cs.

2. Найдите метод DoMultiply\_Click

Этот метод выполняется, когда пользователь нажимает кнопку **Multiply**.

3. Удалите комментарий **TODO - Place the multiplication in a checked block**. Добавьте блок try/catch вокруг строки кода, которая выполняет операцию умножения, и затем перехватите OverflowException.

4. Внутри блока try добавьте проверяемый блок вокруг строки кода, которая выполняет арифметику умножения.

5. Постройте решение и исправьте все ошибки.

Задание 3: тестирование приложения

1. Запустить приложение.

2. Нажмите Кнопку **Multiply**. Убедитесь, что приложение теперь выдает сообщение, информирующее Вас о том, что арифметическая операция привела к переполнению.

3. Нажмите кнопку ОК, закройте окно MainWindow и вернитесь в Visual Studio.

4. Закройте Visual Studio.