В этом упражнении будет изменен интерфейс IMeasuringDevice и добавлено событие NewMeasurementTaken. Событие будет вызвано когда прибор обнаруживает изменение и принимает новое измерение.

Вы измените абстрактный класс MeasureDataDevice и реализуете это событие. Событие NewMeasurementTaken произойдет после того, как устройство заполнит внутренний буфер новым измерением и зарегистрирует его.

Для фиксации новых измерений будет использоваться компонент BackgroundWorker. Фиксация новых измерений будет происходить в событии DoWork, и событие ProgressReported вызовет событие NewMeasurementTaken, чтобы уведомить clientapplication о том, что новое измерение было принято.

Вы сможете запустить фоновый поток выполняется с помощью метода RunWorkerAsync, и устройство будет поддерживать отмену фонового потока с помощью метода Cancel Worker Async.

Новые функциональные возможности будут протестированы с помощью существующего приложения WPF, которое создает экземпляр класса MeasureMassDevice и захватывает события, вызываемые с помощью делегата. Приложение WPF должно иметь возможность приостановить, а затем повторно запустить класс MeasureMassDevice.

Задача 1. Откройте решение \Ex1 \ Starter.

Задание 2: создать новый интерфейс, который расширяет интерфейс IMeasuringDevice

1. В проекте MeasuringDevice добавьте новый интерфейс с именем IEventEnabledMeasuringDevice в файл namedIEventEnabledMeasuringDevice.cs.

2. Измените интерфейс, так чтобы IEventEnabledMeasuringDevice расширял IMeasuringDeviceinterface.

3. В IEventEnabledMeasuringDevice интерфейс, добавить событие NewMeasurementTaken с помощью baseEventHandler делегата.

4. Создайте приложение, чтобы включить Microsoft IntelliSense для отражения изменений.

Задание 3: Добавить событие NewMeasurementTaken к классу MeasureDataDevice

1. Просмотрите список задач.

2. Найдите **TODO - Modify the class definition to implement the extended interface** , а затем дважды щелкните на задачу. Эта задача находится в файле класса MeasureDataDevice.

3. Удалите **TODO - Modify the class definition to implement the extended interface** , а затем измените определение класса для реализации интерфейса IEventEnabledMeasuringDevice вместо IMeasuringDeviceinterface.

4. В списке задач найдите задачу события Todo-Add The NewMeasurementTaken и дважды щелкните эту задачу. Эта задача находится в конце класса MeasureDataDevice.

5. Удалите **TODO - Add the NewMeasurementTaken event** , а затем объявите событие namedNewMeasurementTaken, используя ту же подпись, что и интерфейс.

6. Ниже событие, снять **TODO - Add an OnMeasurementTaken** **method** , а потом добавить метод protected virtual c именем OnNewMeasurementTaken. Метод не должен принимать параметры и иметь тип возвращаемого значения void. Класс MeasureDataDevice будет использовать этот метод для создания события NewMeasurementTaken.

7. В методе OnNewMeasurementTaken, добавить код, чтобы проверить, что есть подписчик на событие theNewMeasurementTaken; если это так, то вызывать событие. Сигнатура делегата EventHandler определяет два параметра: параметр объекта, указывающий объект, вызвавший событие, и параметр EventArgs, предоставляющий дополнительные данные, передаваемые обработчику событий. Выберите параметр объекта this.

Задача 4. Добавление члена BackgroundWorker в класс MeasureDataDevice

1. В списке задач найдите **TODO** **- Declare a BackgroundWorker to generate data** , а затем дважды щелкните на задаче. Эта задача расположена в верхней части класса MeasureDataDevice.

2. Удалите **TODO** **- Declare a BackgroundWorker to generate data** , а затем добавьте член privateBackgroundWorker с именем dataCollector в класс.

Задание 5: Добавить метод GetMeasurements к классу MeasureDataDevice

Метод GetMeasurements будет инициализировать член BackgroundWorker dataCollector для опроса для нового получения тесто поднять событие NewMeasurementTaken каждый раз, когда он обнаруживает новое измерение.

1. В списке задач найдите задачу **TODO - Implement the GetMeasurements method**, а затем дважды щелкните эту задачу.

2. Удалите **TODO - Implement the GetMeasurements method** , а затем добавьте новый закрытый метод namedGetMeasurements в класс. Этот метод не должен принимать параметры и возвращать значение.

3. В методе GetMeasurements добавьте код для выполнения следующих действий:

а. Создайте экземпляр элемента **dataCollector** члена класса BackgroundWorker.

b. Укажите, что элемент Datacollector BackgroundWorker поддерживает отмену.

c. Укажите, что элемент Datacollector BackgroundWorker сообщает о ходе выполнения во время выполнения.

Совет: задайте свойства WorkerSupportsCancellation и WorkerReportsProgress.

4. Добавьте следующий код для создания экземпляра делегата DoWorkEventHandler, ссылающегося на метод calleddataCollector\_DoWork. Присоедините делегат к свойству события DoWork члена dataCollector. Объект dataCollector вызовет метод dataCollector\_DoWork при возникновении события DoWork.

Подсказка: использовать intellisense, чтобы создать заглушку код для метода dataCollector\_DoWork. Для этого введите первую часть строки кода, вплоть до операторов+=, а затем дважды нажмите клавишу TAB. Visual Studio использует встроенный фрагмент кода для завершения строки кода и добавления заглушки метода. Это можно сделать каждый раз, когда вы подключаете обработчик на событие к событию с помощью оператора += .

...

dataCollector.WorkerReportsProgress = true;

dataCollector.DoWork + =

новый DoWorkEventHandler (dataCollector\_DoWork);

} ...

5. Используя тот же метод, что и на предыдущем шаге, создайте экземпляр делегата ProgressChangedEventHandler, который ссылается на метод amethod с именем dataCollector\_ProgressChanged. Присоедините этот делегат к свойству события ProgressChanged члена dataCollector. Объект dataCollector будет вызывать метод dataCollector\_ProgressChanged при возникновении события ProgressChanged.

6. Добавьте код для запуска объекта Datacollector BackgroundWorker, работающего асинхронно.

Задача 6: реализация метода dataCollector\_DoWork

1 Под методом GetMeasurements найдите метод dataCollector\_DoWork.

Этот метод был создан во время предыдущей задачи. Он работает на фоновом потоке, и его целью является сбор и хранение данных измерений.

2 в методе dataCollector\_DoWork, удалите оператор, который вызывает исключение **NotImplementedException** и добавьте код, выполняющий следующие действия:

а. Создайте экземпляр dataCaptured как новый целочисленный массив, содержащий 10 элементов.

b. Инициализируйте целое число i с нулевым начальным значением. Вы будете использовать эту переменную для отслеживания текущей позиции в массиве dataCaptured.

c. Добавьте цикл while, который выполняется пока **dataCollector**.**CancellationPending**  имеет значение false.

3 в цикле while добавьте код для выполнения следующих действий:

1. Вызовите метод **controller**.**TakeMeasurement** , и сохранение результата в массив dataCaptured в позицию, которую определяет i. Метод TakeMeasurement контроллела блокируется, пока доступно новое измерение.
2. Обновите свойство mostRecentCapture, которое содержит значение из массива dataCaptured на той позиции, на которую указывает i.

c. если значение удаляемой переменной равно true, завершите цикл while. Этот шаг гарантирует, что останавливается при уничтожении объекта MeasureDataDevice.

4 добавьте код в цикл while после инструкций, добавленных на предыдущем шаге, чтобы выполнить следующие действия:

проверьте, имеет ли свойство loggingFileWriter значение null.

b если свойство loggingFileWriter не равно null, вызовите loggingFileWriter.Метод writeline, минуя stringparameter формата "измерения - mostRecentMeasure", где mostRecentMeasure является значением переменной themostRecentMeasure.

Примечание: собственность loggingFileWriter-это простой объект streamwriter, который пишет в текстовый файл. Thisproperty инициализируется в startcollecting метод. Для записи в объект aStreamWriter можно использовать метод WriteLine.

5 Добавить строку кода в конец цикла while для вызова dataCollector.Метод ReportProgress, передающий нулевой параметр.

Метод ReportProgress вызывает событие ReportProgress и обычно используется, чтобы вернуть процент completionof задачи, возложенные на объект BackgroundWorker. Событие ReportProgress можно использовать для обновления оценок времени выполнения barsor в пользовательском интерфейсе. В этом случае, поскольку задача будет выполняться бесконечно до отмены, событие theReportProgress будет использоваться в качестве механизма для запроса пользовательского интерфейса на обновление отображения с новым измерением.

6 Добавить код в конец цикла while для выполнения следующих действий:

приращение целого числа i.

b если значение целого числа больше девяти, сбросьте i до нуля.

Вы используете целое число i в качестве указателя на следующую позицию для записи в массиве dataCaptured. Это массив hasspace для 10 измерений. Когда элемент 9 будет заполнен, устройство начнет перезаписывать данные, начиная с element0.

Задание 7: реализовать метод dataCollector\_ProgressChanged

1 найдите метод dataCollector\_ProgressChanged.

Этот метод был создан во время более ранней задачи. Он выполняется при возникновении события ProgressChanged. В этом упражнении это событие происходит, когда метод dataCollector\_DoWork принимает и сохраняет новое измерение.

2 В обработчике событий, удалить код исключения, а затем вызвать метод OnNewMeasurementTaken, передает noparameters.

Метод OnNewMeasurementTaken вызывает событие NewMeasurementTaken, определенное ранее. Вы willmodify интерфейс, чтобы подписаться на это событие, так что когда он вырос, пользовательский интерфейс может обновить отображаемую информацию.

Задание 8: вызвать метод GetMeasurements начать собирать измерений

1. В списке задач найдите задачу TODO-вызов метода GetMeasurements, а затем дважды щелкните эту задачу. Эта задача находится в методе StartCollecting.

2. Удалите TODO-вызовите метод GetMeasurements комментарий и добавьте строку кода для вызова метода theGetMeasurements.

Задание 9: вызвать метод CancelAsync, чтобы остановить сбор измерений

1 в списке задач найдите задачу сборщика данных Todo - Отмена, а затем дважды щелкните эту задачу. Эта задача находится в методе StopCollecting.

2 удалите TODO-отмените комментарий сборщика данных и добавьте код для выполнения следующих действий:

проверить, что член dataCollector не NULL.

б если член dataCollector не NULL, вызывается метод CancelAsync, чтобы остановить работу thedataCollector объект BackgroundWorker.

Задание 10: dispose объекта BackgroundWorker, когда объект MeasureDataDevice разрушается

1. В списке задач найдите задачу сборщика данных Todo-Dispose и дважды щелкните ее. Эта задача находится в методе Dispose класса MeasureDataDevice.

2. Удалите комментарий сборщика данных TODO - Dispose и добавьте код для выполнения следующих действий:

a. Убедитесь, что элемент dataCollector не равен null.

b. Если элемент dataCollector не равен null, вызовите метод Dispose для удаления экземпляра dataCollector.

Задание 11: обновление пользовательского интерфейса для обработки измерений событий

1 в списке задач найдите задачу Todo-Declare a delegate to reference NewMeasurementEvent, а затем дважды щелкните эту задачу. Эта задача находится в коде за MainWindow.окна xaml.

2 удалите комментарий и добавьте код для определения делегата типа EventHandler с именем newMeasurementTaken.

3 в методе startCollecting\_Click удалите комментарий TODO-используйте делегат для ссылки на обработчик событий, код andadd для инициализации делегата newMeasurementTaken с новым делегатом EventHandler, который основан на методе amethod с именем device\_NewMeasurementTaken. В задаче thenext будет создан метод device\_NewMeasurementTaken.

Примечание. IntelliSense нельзя использовать для автоматического создания заглушки для device\_NewMeasurementTakenmethod, как в предыдущих задачах.

4 в методе startCollecting\_Click, снять ТОДО - подключить обработчик события к комментарию событие, и addcode подключить newMeasurementTaken делегат к событию NewMeasurementTaken объекта устройства. Объект Thedevice является экземпляром класса MeasureMassDevice, который наследует от класса abstractclass MeasureDataDevice.

Подсказка: для подключения делегат к событию, используйте += составной оператор присваивания на событие.

Задача 12: осуществлять device\_NewMeasurementTaken метод обработки событий

1 в списке задач найдите метод todo-Add The device\_NewMeasurementTaken Event handler для обновления UIwith новой задачи измерения, а затем дважды щелкните эту задачу.

2 удалите TODO-добавьте метод device\_NewMeasurementTaken обработчик событий для обновления пользовательского интерфейса с комментарием измерения thenew и добавьте частный метод обработчика событий с именем device\_NewMeasurementTaken. Метода, который не возвращает значение, но следует принять следующие параметры:

a. Объект с именем sender.

b. Объект EventArgs с именем e.

3 в методе device\_NewMeasurementTaken добавьте код, чтобы проверить, что член устройства не равен null. Если devicemember не null, выполните следующие задачи:

обновление свойства Text текстового поля mostRecentMeasureBox со значением thedevice.Свойство MostRecentMeasure.

Подсказка: используйте метод ToString для преобразования значения устройства.Собственность MostRecentMeasure returnsfrom целое число в строку.

B обновите свойство Text текстового поля metricValueBox, указав значение устройства.MetricValue метод.

C обновите свойство Text текстового поля imperialValueBox, указав значение устройства.ImperialValue метод.

D сбросить rawDataValues.Свойство itemssource к нулю.

E Установите rawDataValues.Свойство itemssource значению устройства.Возвращает метод GetRawData.

Примечание: последние два шага необходимы для обеспечения того, чтобы механизм привязки данных, используемый Необработанным Databox в окне WPF обновляет отображение правильно.

Задача 13: отсоединить обработчик события

1. В списке задач найдите задачу Todo-Disconnect обработчика событий и дважды щелкните эту задачу. Эта задача расположена в методе stopCollecting\_Click, который запускается, когда пользователь нажимает кнопку остановить сбор.

2. Удалите комментарий обработчика событий Todo-Disconnect и добавьте код для отключения newMeasurementTakendelegate от устройства.Событие NewMeasurementTaken.

Совет: чтобы отключить делегат от события, используйте оператор - = compound assignment для события.

Задача 14: протестировать решение

1 постройте проект и исправьте все ошибки.

2 Запустите приложение.

3 Нажмите кнопку Начать Сбор и убедитесь, что значения измерений начинают отображаться в поле необработанные данные.

Объект MeasureMassDevice используемого приложения метрических измерений и сохраняет их, прежде чем поднимать событие theNewMeasurementTaken. Событие вызывает код, который обновляет UI последней информации. Продолжайте смотреть список необработанных данных, чтобы увидеть заполнение буфера данными, а затем начать перезаписывать более ранние значения.

4 Нажмите кнопку остановить Сбор и убедитесь, что пользовательский интерфейс не обновляется.

5 нажмите начать сбор снова. Убедитесь, что поле списка необработанные данные очищено и что новые данные измерений отображаются capturedand.

6 Нажмите Остановить Сбор.

7 закройте приложение и вернитесь в Visual Studio