Задание №6. Пользовательские элементы управления

Цель задания:

Освоить создание и использование отдельных пользовательских элементов управления (UserControl) без логики взаимодействия с внешними объектами.

В ходе выполнения предыдущих заданий вы могли почувствовать, что работа в классе MainForm стала неудобной. Класс содержит логику трёх вкладок, стал большим, на форме одновременно располагаются до 60 элементов управления с похожим именованием.

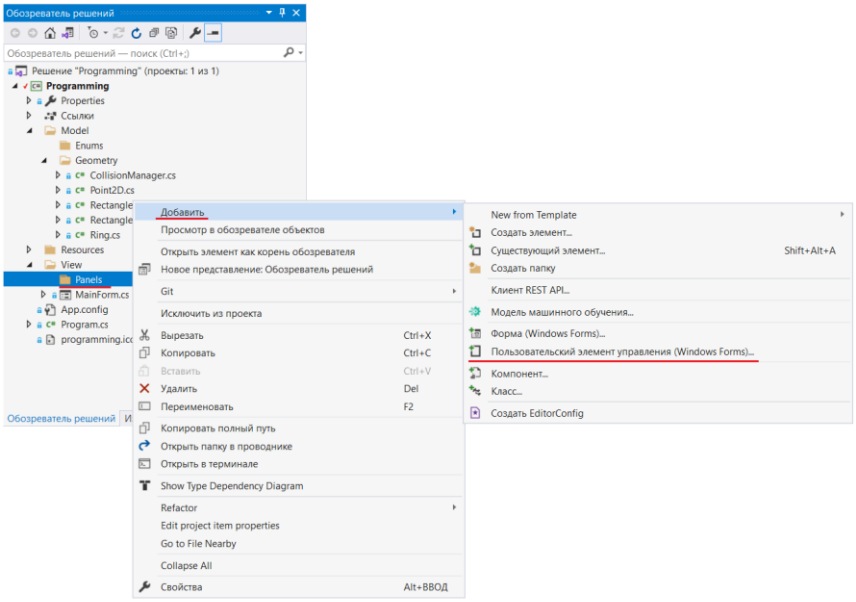
Одна из ключевых задач, которую любой инженер решает в ходе работы, является ***управление сложностью***. Это значит, что инженер должен не просто предоставить работающее решение. Предлагаемое решение должно быть настолько простым в поддержке и отладке, насколько существующие условия позволяют достичь. Без обеспечения простоты реализации со временем любая программа становиться неподдерживаемой, а, значит, неотлаживаемой и нерасширяемой.

Мы уже использовали несколько подходов к упрощению структуры программы. Например, выделение всей валидации в отдельный класс Validator. Работа с небольшими классами, каждый из которых направлен на решение собственной задачи, гораздо удобнее, чем работа с большими классами, которые решают много задач. Разделим главное окно на несколько независимых классов. Для этого создадим пользовательские элементы управления (UserControl).

Вы уже поработали со стандартными элементами управления, такими как кнопка, список, текстовое поле и другие. Windows Forms позволяют создавать собственные элементы управления на основе уже существующих. Это могут быть как небольшие элементы в виде комбинации нескольких стандартных элементов, так и большие элементы, представляющие собой целые вкладки для окон.

Создадим пользовательский элемент управления на примере целой вкладки по определению пересечения прямоугольников.

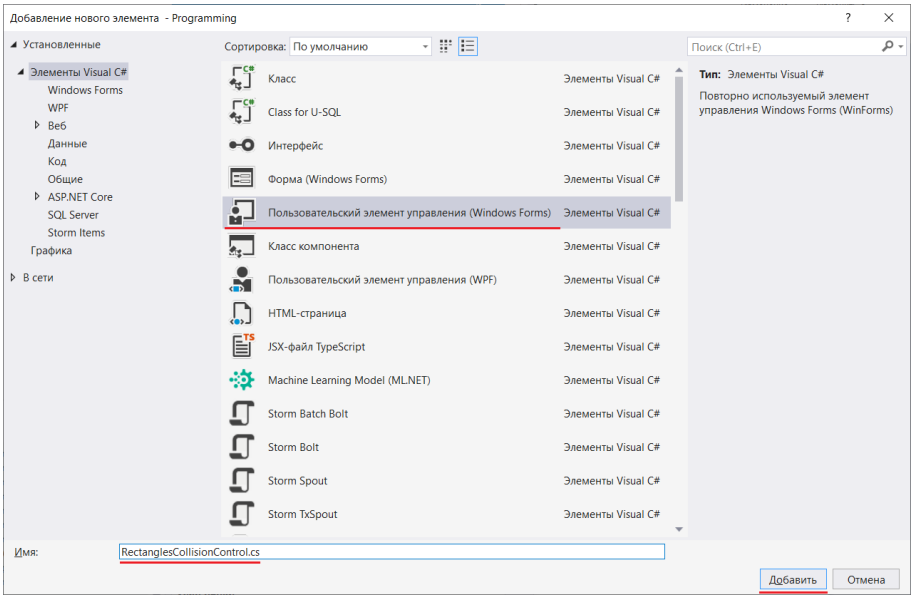
Для создания пользовательского элемента управления необходимо кликнуть правой кнопкой по названию проекта или папки в проекте, в контекстом меню выбрать пункт «Добавить пользовательский элемент управления» («Add UserControl»):



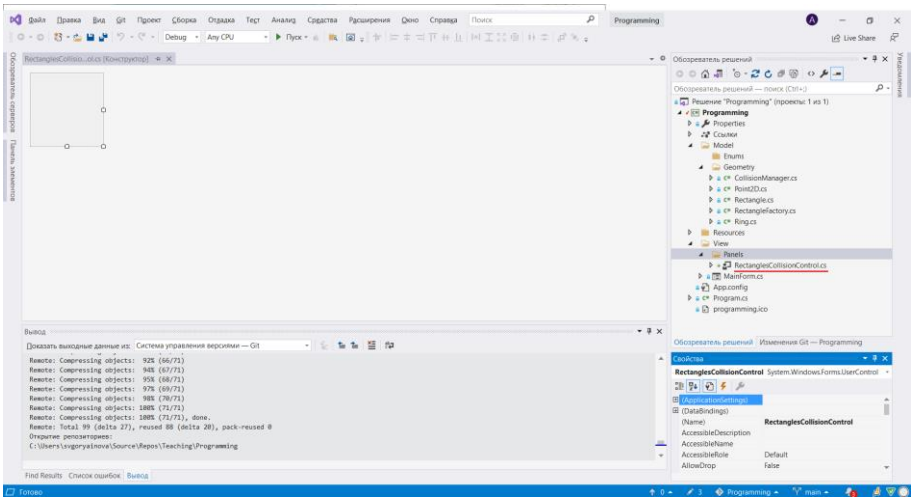
Если вы планируете создание большого количества элементов, сначала создайте для них папку Controls, Tabs, Pages или Panels. В появившемся окне нужно убедиться, что вы создаете именно UserControl, а не какой-либо другой элемент, а ввести его название:

- Название элемента управления, как и все классы, пишется в стиле Pascal – с заглавной буквы каждое новое слово, включая первое.
- Название дается в форме существительного, в конце пишется слово Control (или Panel, TabPage в зависимости от элемента).
- Но, главное, название должно отражать назначение элемента управления.

Если мы хотим создать элемент управления, который будет представлять всю вкладку с пересечением прямоугольников, тогда подходящим названием будет RectanglesCollisionControl:



После нажатия на кнопку «Добавить» в окне добавления нового элемента, аналогично созданию нового окна, среда разработки откроет окно дизайнера для нового элемента управления. Сам элемент управления можно будет найти в дереве решения:

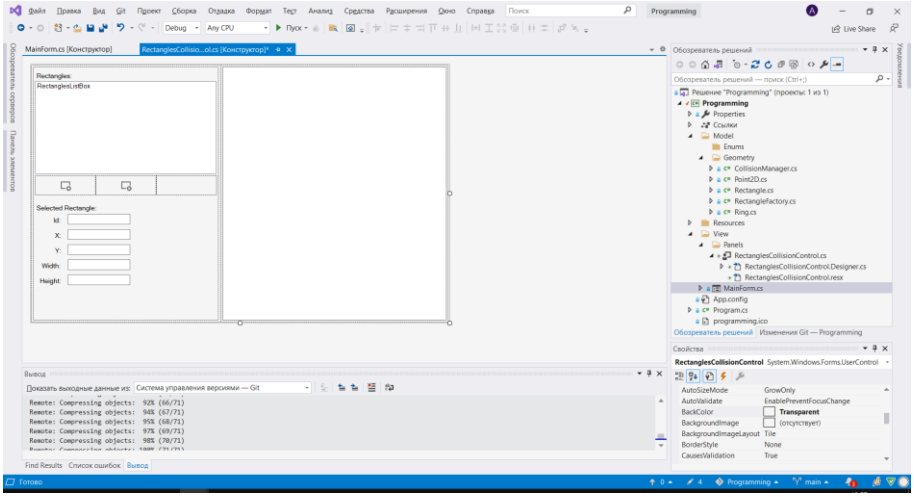


Если посмотреть на дизайнер элемента управления, то можно заметить несколько отличий от дизайнера форм. Во-первых, у элемента не отображается

шапка с названием программы, кнопками «Свернуть» «Развернуть» «Закрыть». Это логично, так как элементы управления могут быть показаны пользователю только в составе окна или другого элемента управления, но никогда не как самостоятельное окно.

Если мы откроем панель «Свойства» для нашего элемента управления, мы не увидим свойств, относящихся к формам, таких как Icon, StartPosition, WindowState, FormBorderStyle – таких свойств нет в элементах управления. Зато здесь есть другие свойства, которые есть у стандартных элементов управления. Это Location, Dock, Anchor и другие. Каждому из этих свойств можно задать значение по умолчанию – значение, которое будет принимать свойство элемента управления, когда его разместят на форме. В подавляющем большинстве случаев их менять не нужно, за исключением MinimumSize, MaximumSize – если вы хотите ограничить размеры элемента управления. Увеличим размер нашего элемента управления и переместим все элементы из вкладки Rectangles главного окна на наш новый элемент управления. Для этого не нужно заново верстать и размещать все элементы. Достаточно открыть главное окно, выделить нужные элементы, скопировать их (Ctrl+C) и вставить их в дизайнере элемента управления (Ctrl+V). Все элементы будут располагаться на тех же местах с теми же свойствами, что и в главном окне. Если при верстке вкладки на главном окне вы размещали все элементы на элементе-контейнере типа TableLayout, GroupBox, Panel и другие, то достаточно скопировать только элемент-контейнер, не выделяя компоненты по отдельности.

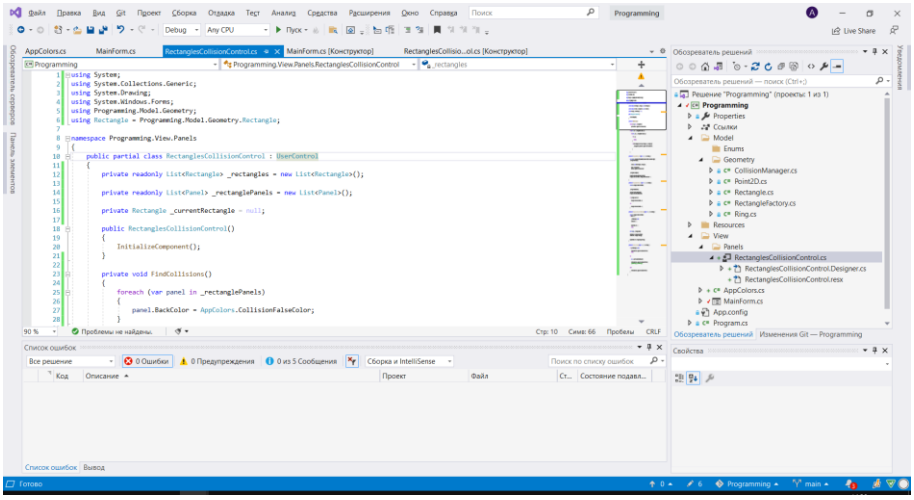
Помните, что при копировании элементов, копируются их названия и все свойства, включая положение, но не копируются обработчики событий:



После перемещения элементов из MainForm в RectanglesCollisionControl, необходимо проверить и при необходимости поправить верстку. В частности, проверить адаптивность верстки.

Если элементы управления в MainForm изначально располагались в контейнере типа TableLayout или др., то перенос адаптивной верстки становится проще – достаточно скопировать TableLayout, а затем задать ему свойство Dock в значение Fill. Контейнер растянется по всему родительскому элементу, а все элементы внутри контейнера сохранят свои адаптивные настройки по отношению к контейнеру.

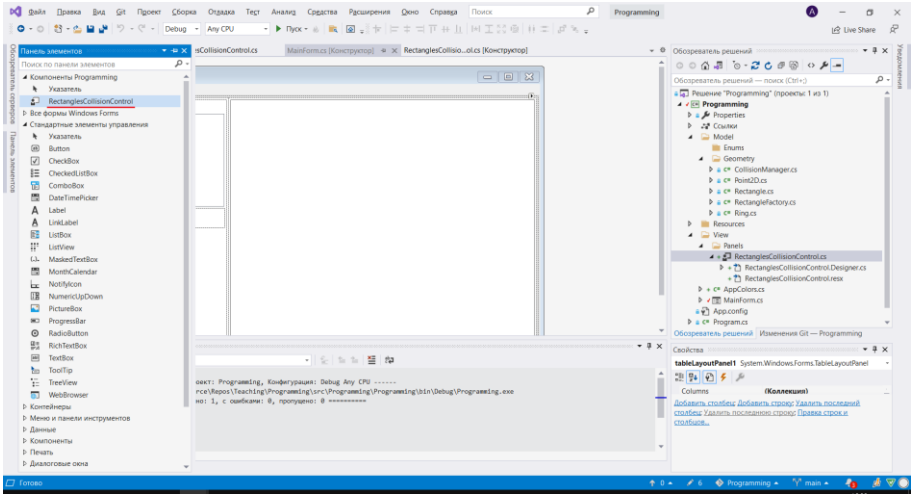
После переноса верстки, перенесем всю связанную логику из MainForm в класс нашего нового элемента. Необходимо перенести всё, что работало с перенесенными элементами управления: поля, константы, открытые и закрытые методы, обработчики событий. Если в конструкторе главного окна был код по инициализации данных на вкладке, его также надо перенести в конструктор нашего нового элемента:



При копировании обработчиков событий, они не подключатся к элементам управления самостоятельно. Необходимо открыть дизайнер нового элемента управления и через панель «Свойства» -> «События» назначить обработчики событий заново.

Чтобы проверить, что перенос был сделан правильно, надо скомпилировать проект. Если при компиляции возникли ошибки, скорее всего, какой-то код был неправильно скопирован.

После успешной компиляции проекта, новый элемент управления RectanglesCollisionControl появится на Панели элементов рядом со стандартными элементами управления:



Все новые элементы управления появляются в отдельном списке по названию проекта. В данном случае, элемент RectanglesCollisionControl будет в списке под названием Programming.

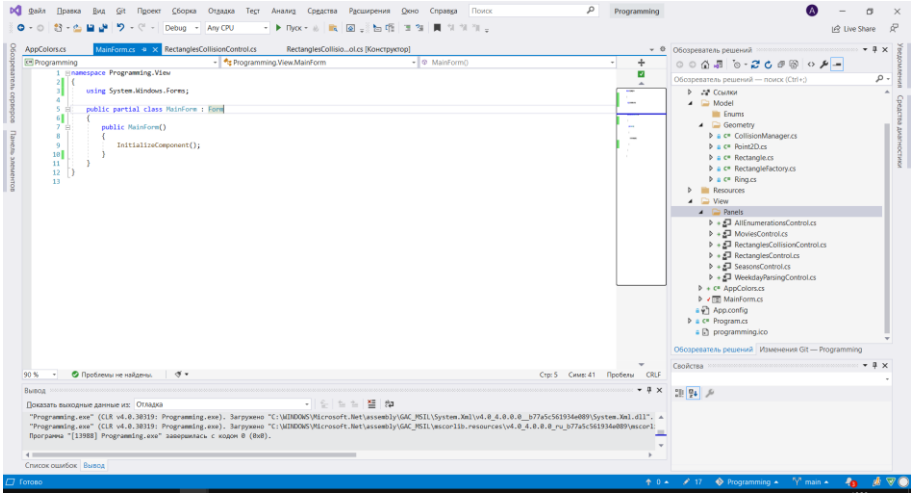
Теперь мы можем открыть дизайнер главного окна и удалить все элементы с вкладки Rectangles. Вместо этих элементов, перенесем на вкладку новый элемент RectanglesCollisionControl с панели элементов. Чтобы элемент занял всю вкладку, зададим свойству Dock значение Dock.Fill.

Запустим программу. Если мы всё сделали правильно, уже в этой версии вкладка Rectangles должна работать как и раньше, но только теперь вся её реализация находится в отдельном классе RectanglesCollisionControl.

После удаления всех элементов вкладке Rectangles из главного окна, все связанные с ними поля и методы стали неиспользуемыми. Остается только удалить весь неиспользуемый код из главного окна, который до этого мы перенесли в новый элемент управления.

По завершению всех изменений в коде следует обязательно запустить программу и тщательно протестировать каждый элемент управления. Часто бывает, что из-за неподключенного обработчика перестает работать валидация, пересчет или обновление данных. Любые ошибки следует сразу же исправить:

Как было сказано ранее, мы можем выносить в отдельные элементы управления не только вкладки, а любые комбинации элементов управления. В рамках нашего приложения, мы можем выделить в отдельные элементы управления группы элементов, располагающихся в GroupBox на первой и второй вкладке приложения. Таким образом, мы создадим три новых элемента для первой вкладке, и два новых элемента для второй вкладки. Работа с интерфейсом в виде отдельных независимых элементов станет проще, а логика в главном окне сократится до 12 строчек кода:



Теперь при модификации любой из вкладки мы будем уверены, что случайно не поломаем что-то в логике другой вкладки. Элементы управления одной вкладки не мешают элементам другой, названия не путаются, разработчику проще ориентироваться в коде. При этом логика каждого отдельного элемента управления не больше двухсот строчек вместо огромного файла главного окна, как это было ранее. Это и есть один из приёмов управления сложностью. А само разделение главного окна на отдельные независимые и легко поддерживаемые классы относится к процессу ***рефакторинга*** – модификации программы с целью повышения её читаемости и возможности поддержки без потери ранее реализованной функциональности.

На практике, разработчики стараются делить интерфейс на отдельные компоненты, а главное окно должно лишь содержать логику по их взаимодействию – брать данные с одного элемента управления и передавать их другим элементам. Так как в нашей программе нет взаимодействия между несколькими вкладками, то логика главного окна пока остается пустой.

Для реализации передачи данных от пользовательских элементов в главное окно и затем другим элементам, необходимо добавить в пользовательский элемент управления открытые свойства нужного типа данных – но это уже не входит в рамки данного задания.

Последовательность выполнения:

1. Создать в ранее созданном проекте Programming Demo новый пользовательский элемент управления RectanglesCollisionControl, как это описано в разделе выше.
2. После создания нового элемента управления RectanglesCollisionControl убедиться, что вёрстка в исполняемом приложении также соответствует первоначальному макету из прошлого задания.
3. После создания нового элемента управления убедиться, что логика в исполняемом приложении также соответствует первоначальным требованиям – валидация данных, добавление и удаление прямоугольников, обновление прямоугольников в случае их редактирования.
4. По аналогии с пользовательским элементом RectanglesCollisionControl, создайте пользовательские элементы для других вкладок:
   - Элемент управления, соответствующий GroupBox Enumerations на вкладке Enums.
   - Элемент управления, соответствующий GroupBox Weekday Parsing на вкладке Enums.

- Элемент управления, соответствующий GroupBox Seasons Handle на вкладке Enums.
- Элемент управления, соответствующий GroupBox Rectangles на вкладке Classes.
- Элемент управления, соответствующий GroupBox Movies на вкладке Classes.

5. Перед созданием любых новых элементов подумайте над их правильным названием.
6. Если вы хотите выделить в отдельный элемент управления содержимое некоторого GroupBox, рекомендуем переносить элементы без самого GroupBox, оставив его в исходной форме. Если новый элемент управления будет использоваться в нескольких местах, то создание элемента без внешнего GroupBox-контейнера даст больше гибкости в расположении элемента в других окнах.
7. После переноса элементов управления на отдельные UserControl, посмотрите их названия. Если в их именовании остались упоминания названий вкладок и пр (т.е. элементов, оставшихся в главном окне), то их следует убрать. Создаваемые пользовательские элементы не должны в своём именовании, верстке или логике содержать какой-либо информации о родительских элементах, где они размещаются. Только так можно обеспечить гибкость в повторном использовании пользовательских элементов управления.
8. При переносе логики для панели Seasons Handle вы можете столкнуться с проблемой, что теперь нельзя обратиться к фону главного окна и изменить его. Создание логики, когда действия пользователя в элементе управления могут влиять на главное окно или другие элементы, реализуется с помощью собственных событий и не входит в данное задание. Поэтому реализуйте другое действие вместо смены цвета фона главного окна.
9. Теперь разные элементы управления занимаются валидацией вводимых данных, но они используют одинаковые цвета для индикации. Создайте отдельный статический класс AppColors и вынесите туда в виде открытых константных полей или свойств все цвета: для валидации, пересечения прямоугольников, смены времен года и др.

*** Дополнительные задания:**

10. В п.8 предлагается заменить смену цвета фона главного окна во время изменения времени года на другое действие. Причина: для реализации данной логики нужно знание так называемых событий. Вы уже знаете, как использовать события стандартных элементов управления, такие как Click, SelectedIndexChanged, TextChanged и др. Но для реализации смены фона в главном окне необходимо реализовать в пользовательском элементе управления собственное событие ButtonClicked и подписаться на него в главном окне. Попробуйте разобраться в вопросе создания собственных событий и реализовать логику смены фона главного окна при смене времени года в пользовательском элементе управления.