Государственный университет – Высшая школа экономики

Факультет Бизнес-Информатики

Кафедра Основ информатики
и прикладного программного обеспечения

C#

Объектно-ориентированный язык программирования

Пособие к практическим занятиям - №8

Проф. Забудский Е.И.

Москва 2006

Тема 8. Объектно-ориентированный подход к разработке программного обеспечения.

System.Windows.Forms. Графический Интерфейс Пользователя – Graphical User Interface (GUI).

Два практических занятия
(4 часов)

Разработан Проект Расчет оценки студента в консольном и оконном вариантах

(**см. cs-файлы Листинги 1 – 8** и UML-диаграммы классов на **рис. 2 и рис. 3**)

За компонентно-ориентированным программированием – будущее
(см. **листинги 4.5 - 4.10** – Материалы к Практич. занятию № 4)

// **КОММЕНТАРИЙ**. На сайте http://www.firststeps.ru/ “Первые шаги” представлено много интересных обучающих материалов по различным интегрированным средам и языкам программирования, в том числе представлены **C#** и платформа **.NET** (step by step).

Содержание

1. Методология разработки

Компьютерных моделей реальных или концептуальных систем

Она включает шесть этапов:

1) этап предпроектных исследований. Учет технических, временных и финансовых ограничений. Определение целесообразности продолжения работы над приложением.

2) этап анализа. Сбор информации, необходимой для продолжения работы.

3) этап проектирования. Создание схемы интерфейса и структуры программы, без создания каких–либо фактических программ.

4) этап разработки. Создание приложения, включая все элементы интерфейса и программного кода.

5) этап внедрения. Применение и тестирование программы (не рассматривается).

6) этап опытной эксплуатации. Совершенствование продукта, призванное устранить возможные проблемы или удовлетворить новые запросы (не рассматривается).

Применение методологии помогает выявлять возникающие в процессе разработки проблемы, а также решать их на раннем этапе проектирования, чтобы не вносить изменения в готовый программный продукт.

На каждом этапе создается некий вещественный, или *передаваемый* продукт. Передаваемым материалом может служить: 1) техническое задание, 2) письмо, информирующее клиента о невозможности выполнить проект с учетом накладываемых ограничений на затраты времени и средств. На каждом этапе принимается обоснованное решение — продолжать или прекращать работу над проектом.

2. Компьютерная модель концептуальной системы РАСЧЕТ ОЦЕНОК

2.1. Первый этап: предпроектные исследования

Первый этап может начаться с телефонного разговора с клиентом, со служебной записки начальнику отдела разработки систем от вице–президента либо с письма клиента, предлагающего обсудить возможность доработки существующей системы. В нашем случае декан факультета предложил разработать *Проект расчета оценок* студентов для факультета английского языка. Задание расширилось, чтобы удовлетворить потребности математического и естественного факультетов.

Цель предпроектных исследований состоит не в самой разработке системы, а в определении реальной необходимости доработки существующей системы либо работы над новым проектом.

Продолжительность предпроектных исследований обычно совсем невелика: для крупных проектов она составляет один или два дня, а в нашем случае — примерно час.

Конечный результат фазы предпроектных исследований — решение о том, целесообразно ли продолжать работу, или следует отказаться от нее. Что может вызвать решение отказаться от проекта? Существует три фактора, обычно называемых *ограничениями,* от которых зависит решение о продолжении работы над проектом:

1) технический фактор. Проект невозможно выполнить с применением современных технологических возможностей.

2) фактор времени. Проект можно выполнить, но не в сроки, установленные заказчиком. Этот фактор часто служит причиной отказа от проекта после этапа предпроектных исследований.

3) финансовый фактор. Проект можно выполнить в установленные заказчиком сроки, но выделяемые средства этого не позволяют.

От *Проекта расчета оценок* отказываться не будем, и принимаем его к разработке. Он целесообразен как для студентов, так и для преподавателя. Оплата в размере $450 производится после окончательной сдачи проекта.

2.2. Второй этап: анализ

Этот этап иногда называют этапом сбора данных.

Этот этап посвящен детальному изучению задачи, недостатков существующей системы или новых требований. В зависимости от размеров проекта, этот этап может быть таким же коротким, как предпроектные исследования, либо растянуться на несколько месяцев.

Для *Проекта расчета оценок* этот этап подразумевает еще одно посещение деканата с целью сбора более полной информации о задании либо обсуждения сведений, полученных во время предпроектных исследований.

Разработчики склонны считать, что в результате предпроектных исследований имеются все необходимые сведения о проекте. Однако дополнительная встреча с заказчиком может принести много новой информации.

В данный момент самая большая ошибка — начать уже теперь писать текст программы. Почему? Необходимо получить у Заказчика больше информации о задании, поскольку осталось еще несколько невыясненных вопросов.

До встречи с Заказчиком я отправил ему по электронной почте сообщение. Его содержательная часть заключается в следующем**:**

"…Отмечу наиболее важные моменты нашего обсуждения *Проекта расчета оценок.* Мы разработаем для Вас программный продукт, выполняемый на персональном компьютере в среде **Windows** и доступный на **Web**. Программа будет осуществлять следующие основные функции:

1. Предоставлять простой и удобный графический интерфейс (**GUI** – **G**raphical **U**ser **I**nterface) для расчета оценок студентов.

2. Пользователю будет предложено указать факультет, на котором обучается студент — английского языка, математики или естественных наук.

3. Если пользователь указывает, что желает вычислить оценку студента факультета АНГЛИЙСКОГО языка, программа предложит ему указать оценки: **а)** за экзамен в середине семестра, **б)** заключительный экзамен, **в)** реферат и **г)** выступление в аудитории. Итоговая оценка будет вычислена, как сумма 25% от оценки за экзамен в середине семестра, 25% от оценки за заключительный экзамен, 30% от оценки за реферат и 20% от оценки за выступление в аудитории.

4. Если пользователь укажет, что желает вычислить оценку студента факультета ЕСТЕСТВЕННЫХ наук, программа предложит ему указать оценки: **а)** за экзамен в середине семестра, **б)** заключительный экзамен и **в)** реферат. Итоговая оценка будет вычислена, как сумма 40% от оценки за экзамен в середине семестра, 40% от оценки за заключительный экзамен и 20% от оценки за реферат.

5. Если пользователь укажет, что желает вычислить оценку студента факультета МАТЕМАТИКИ, программа предложит ему указать оценки: **а)** за экзамен в середине семестра, а также **б)** за заключительный экзамен. Итоговая оценка будет рассчитана, как сумма 50% от оценки за экзамен в середине семестра и 50% от оценки за заключительный экзамен.

6. После расчета оценка будет отображена на экране.

Это сообщение, по существу, станет техническим заданием **…"**

В ответ Заказчик направил таблицу, в которой приведено соответствие между буквенными и цифровыми оценками на разных факультетах:

Таблица 1. Соответствие буквенных и цифровых оценок

Буквенная оценка	Цифровая оценка		
	Факультет английского языка	Факультет математики	Факультет естественных наук
A	>= 93	>= 90	>= 90
B	(>= 85)…(< 93)	(>= 83)…(< 90)	(>= 80)…(< 90)
C	(>= 78)…(< 85)	(>= 76)…(< 83)	(>= 70)…(< 80)
D	(>= 70)…(< 78)	(>= 65)…(< 76)	(>= 60)…(< 70)
F	< 70	< 65	< 60

По итогам встречи с Заказчиком установлено: все расчеты, выполняемые на факультетах при выставлении оценок, производятся вручную, что является рутинным трудом и приводит к многочисленным ошибкам. Считаем, что Программа Расчета Оценок довольно быстро оправдает себя.

2.3 Третий этап: проектирование

Проектирование охватывает множество различных элементов. Ниже приводится список компонентов, "разрабатываемых" на этапе проектирования:

1. Ввод.
2. Вывод.
3. Обработка.
4. Файл.

Обычно этапу проектирования уделяется слишком мало времени. Программисты склонны немедленно приступать к написанию программного кода. Этому есть оправдание: написание программы — увлекательное занятие. К сожалению, **немедленный переход к написанию кода –** большая ошибка.

Одаренные (и безрассудные – «КРУТЫЕ users») программисты часто приступают к написанию своего кода без тщательной проработки модели. В процессе работы над проектом им приходится изменять фрагменты уже готового программного кода. На полпути может быть найден метод, который неплохо было бы применить уже в начале разработки. Это потребует модификации готового кода. В худшем случае готовая программа может не выполнять ожидаемых от нее функций, в результате чего нужно будет переделывать ее буквально с самого начала.

Хорошая проработка модели позволит существенно снизить вероятность такого развития событий. В конечном счете, программа будет работать ожидаемым образом, а общее время ее разработки уменьшится («спешите МЕДЛЕННО»).

Приступаем к этапу проектирования. После его завершения мы получим: **1)** оформленное техническое задание и **2)** набросок интерфейса пользователя **- GUI**.

Начинающие программисты часто не могут решить, с чего начать этап проектирования. В большинстве случаев сначала определяется вывод программы. Дело в том что, зная выводимые данные, несложно определить, какую информацию пользователь должен ввести для получения желаемого результата. Когда известны вводимые и выводимые данные, можно определить, какие вычисления и операции потребуются, чтобы преобразовать одно в другое.

2.3.1. Проектирование вывода

Удачная особенность Проекта Расчет оценок — выводимые данные можно описать просто: рассчитанная оценка.

Результат расчета оценки будем выводить на экран дисплея в окно **Windows**

Наряду с числовым значением оценок будет отображаться их буквенный эквивалент.

2.3.2. Проектирование ввода

Должны вводиться следующие данные (это компоненты итоговой оценки):

- оценка за экзамен в середине семестра,
- оценка за заключительный экзамен,
- оценка за реферат,
- оценка за выступление.

Состав вводимых данных зависит от факультета, на котором учится студент. Поэтому название факультета следует также ввести в состав указываемых пользователем данных.

Для студентов факультета АНГЛИЙСКОГО языка необходимы **все четыре** оценки. Для студентов факультета ЕСТЕСТВЕННЫХ наук необходимы **три оценки** — за экзамен в середине семестра, за заключительный экзамен и за реферат. Для студентов МАТЕМАТИЧЕСКОГО факультета достаточно **две оценки** за экзамен в середине семестра и заключительный экзамен.

Как исходные данные будут вводиться в программу? Как пользователь сообщит программе название факультета, на котором учится студент?

2.3.2.1. Обозначение типа студента

Наш девиз — пользователь должен как можно меньше работать с клавиатурой.

Почему? Если программа предусматривает ввод названия факультета с клавиатуры, необходимо, чтобы все пользователи печатали названия одинаково, на что не приходится рассчитывать. Одни будут писать названия полностью, другие сокращенно, а третьи использовать отличающиеся названия.

Для того чтобы указать факультет на котором обучается студент (это и определяется тип студента) воспользуемся объектом **C#**, который называется переключатель (**RadioButton**)

Для сведения – переключатели имеют круглую форму,

Другим типом объектов интерфейса, подобных переключателям, являются флажки (**check boxes**). Они имеют квадратную форму. Флажок имеет два состояния — установленное и сброшенное. Пользователь выбирает опцию, выполняя щелчок на флажке мышкой. При этом состояние флажка меняется на противоположное.

Более целесообразно использование переключателей. В группе переключателей одновременно может быть установлен лишь один из них. Именно это и нужно. Студент не может одновременно принадлежать нескольким факультетам. Если пользователь указывает, что оценка вычисляется для студента факультета английского языка, не желательно, чтобы программа предоставляла возможность выбора еще какого–либо факультета. Флажки же допускают одновременную установку нескольких элементов (выбора *нескольких* опций).

Для получения переключателя мы фактически порождаем экземпляр объекта по готовому шаблону переключателя, называемому классом. Этот класс обладает встроенными свойствами (поведением). В данном случае, эти свойства не допускают одновременную установку нескольких переключателей.

Итак, в техническое задание включается пункт использовать переключатели для представления факультета университета.

2.3.2.2. Обозначение компонентов итоговой оценки

Для ввода пользователем данных, на основании которых рассчитывается итоговая оценка в системе **Windows** предусмотрен элемент называемый окном редактирования; в языке **C#** для данного объекта используется обозначение **TextBox**. Этот объект идеально подходит для нашего приложения, поскольку дает пользователю возможность вводить в него текст (данные) с клавиатуры.

Будем использовать четыре таких объекта — по одному для каждого компонента оценки.

Примечательно, что можно избирательно скрывать некоторые из этих объектов (т. е. окон редактирования), в частности те, которые неприменимы для выбранного пользователем факультета университета. Другими словами, если пользователь выбирает факультет МАТЕМАТИКИ, то будут видны только Два Окна Редактирования. Для факультета ЕСТЕСТВЕННЫХ наук будут отображены Три Окна Редактирования, а для факультета АНГЛИЙСКОГО языка – Четыре,

Теперь возможно (и необходимо) изобразить общий вид интерфейса пользователя на бумаге еще до создания программы.

2.3.3. Проектирование графического интерфейса

Изображение на бумаге помогает представить внешний вид интерфейса (рис. 1). Создание подобного чертежа значительно упрощает программирование интерфейса.

Графический интерфейс программы будет выполнен в стиле операционной системы **Windows**. Язык **C#** позволяет создавать программы, имеющие оконное представление. Однако можно создавать и так называемые консольные приложения.

Многооконный интерфейс операционных систем Microsoft Windows, Linux и Macintosh разработан для упрощения работы пользователей. Язык **C#** оперирует на значительно более низком уровне, чем интерфейс пользователя. Поэтому он позволяет создавать программы, которые называются консольными и не обладают графическим интерфейсом.

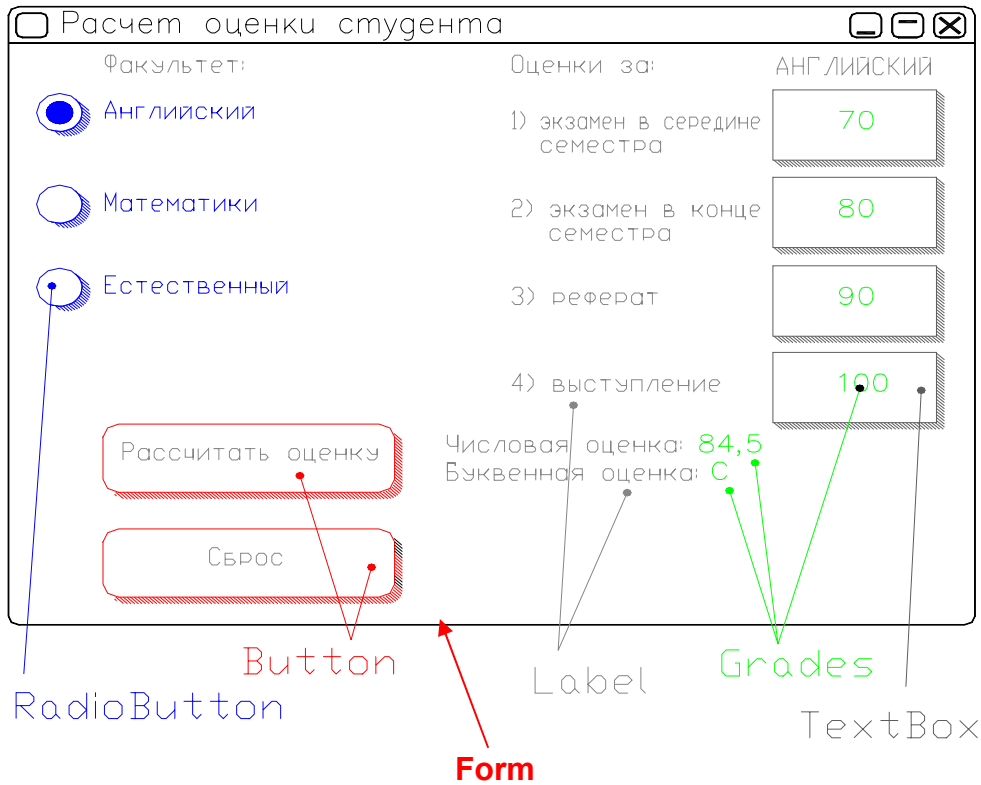


Рис. 1. Набросок GUI Проекта Расчет оценки

Программу Расчета Оценок будем писать, как консольную. Затем выполним ее тестирование, а после этого добавим графический интерфейс (**GUI**).

На схеме, представленной на рис. 1, изображены**:** по одному переключателю для каждого факультета**;** окна редактирования для ввода компонентов оценок**;**. все элементы снабжены надписями (ярлыками – объект **Label**)**…** Надписи возле переключателей являются их составной частью. Когда будем изучать объекты **RadioButton**, узнаем, как указывать эти надписи.

Однако, как уже отмечалось, не все окна редактирования должны быть видимыми. Это зависит от установленного пользователем переключателя.

В графическом интерфейсе предусмотрена кнопка (**Button**) "Рассчитать оценку". В результате нажатия на кнопку будет выполнен расчет и отображение итоговой оценки.

Соображения относительно способа и места отображения итоговой оценки студента**:** можно отображать оценку в самом окне программы, либо выводить сообщение.

Окно сообщения хорошо привлекает внимание. Кроме того, окно сообщения продолжает висеть на экране, пока пользователь не щелкнет на кнопке ОК.

Также можно воспользоваться для отображения результата расчета окном редактирования (объект **TextBox**) или ярлыком (объект **Label**).

После отображения итоговой оценки студента в окне сообщения следует автоматически удалять содержимое окон редактирования, чтобы перейти к следующему студенту. Для этого лучше всего предусмотреть еще одну кнопку (сброс) (**Button**) (см. рис. 1), которая будет сбрасывать установленный пользователем переключатель, делать видимыми все четыре окна редактирования и очищать их содержимое.

На схеме отображена строка заголовка окна. Кроме того, окно содержит пиктограмму Системного меню (**Control Menu**), а также стандартные кнопки Минимизировать (**Minimize**), Максимизировать (**Maximize**) и Закрыть (**Close**).

Замечательная особенность проектирования на бумаге состоит в том, что можно легко внести изменения, если что–либо упущено. При этом впоследствии не придется переделывать программный код.

2.3.4. Проектирование обработки данных

Обработкой называют преобразование вводимых данных в выводимые или преобразование информации. На данном этапе разработки мы должны были определить все выводимые данные (результат расчета оценки) и вводимые данные (компоненты оценки) .

Программа Расчет Оценок включает главную линию — вычисление оценки студента, а также несколько побочных "линий" обработки информации, к ним относятся:

- определение факультета, на котором занимается студент,
- выборочное сокрытие и отображение необходимых окон редактирования,
- обеспечение ввода в окна редактирования допустимых данных – чисел в интервале от **0** до **100**,
- выполнение расчетов,
- отображение рассчитанной оценки.

На этапе проектирования обработки данных мы не пишем код программы. Это еще предстоит. Пока же мы определяем, какие процессы необходимо выполнить, чтобы преобразовать вводимые данные в выводимую информацию.

Программирование выполняется на следующем (*четвертом*) этапе – этапе разработки. На этапе проектирования обработки данных (это *третий* этап) устанавливается, **что** делать, но не **как. Как** – это часть этапа разработки.

На этапе проектирования обработки данных имеет большое значение документирование правил обработки, поскольку их произвольное толкование может приводить к ошибкам. Разработчики систем издавна пользуются различными средствами, помогающими документировать проекты**:**

1) некоторые разработчики применяют *блок–схемы* (**flowcharts**). В блок–схемах для графического представления правил обработки используются специальные символы – блоки;

2) другие разработчики предпочитают псевдокод. **Псевдокод** представляет собой близкое к разговорной речи описание хода выполнения программы в неграфическом виде.

Обе методики получили распространение во времена процедурного программирования. Процедурной называется программа, которая выполняется с начала и до конца *почти* без прерываний. Такая программа строго предписывает пользователю способ взаимодействия с ней. Например, в программе расчета оценок пользователь выбирает факультет, а затем вводит значения компонентов оценок. Процедурное программирование (с применением таких языков, как Basic, Fortran и COBOL) напоминает поездку рейсовым автобусом, когда все остановки предопределены и имеют жестко заданный порядок следования.

Программы для Windows управляются **событиями**. Управляемые **событиями** программы (написанные на языках вроде Visual Basic, C++ и **C#**) не требуют от пользователя жестко заданного поведения. Вместо этого программа реагирует на действия пользователя. **Для взаимодействия с программой пользователю предоставляется графический интерфейс**. Это больше напоминает посещение городка аттракционов: купив входной билет, посетитель может выбирать развлечения по своему усмотрению. Управляемая **событиями** программа должна сохранять работоспособность и реагировать на любые неожиданные **события**.

Без составленного на бумаге плана процесс программирования может развиваться неудачно.
Уделяйте время разработке решения на бумаге, и вы об этом не пожалеете.

Продолжаем обсуждение проектирования обработки данных.

Проектирование – процесс итеративный (**iteratio** /лат./ – **повторение**). Редко бывает, что разработчик попадает в точку с первого раза.

Главная цель процесса обработки данных в Проекте Расчет Оценки: *расчет оценки студента*. Для этого необходимо знать факультет, на котором студент занимается, а также промежуточные значения – составляющие итоговой оценки.

Итоговая оценка студента факультета АНГЛИЙСКОГО языка, если он получил **88** баллов за экзамен в середине семестра, **90** баллов за заключительный экзамен, **85** баллов за реферат и **75** баллов за выступление определяется по следующему выражению:

Итоговая оценка = 88 x **0,25** + 90 x **0,25** + 85 x **0,30** + 75 x **0,20** = **85** (1)

Значения весовых коэффициентов 0,25 ; 0,25 ; 0,30 ; 0,20 (их сумма должна быть равна единице) – см. на **с. 5 пункты 3, 4, 5**. Согласно **табл. 1** числовая оценка **85** соответствует буквенной оценке **B**.

Если пользователь выбирает факультет АНГЛИЙСКОГО языка, а затем вводит значения за экзамен в середине семестра, заключительный экзамен и реферат, но сделал ошибку, например, при вводе значения за выступление, то программа должна отреагировать на это – **выводить сообщение об ошиб-**

ке? Такое сообщение представляет собой еще одну форму выводимых данных.

Подчеркнем, что не все пользователи будут вводить данные в одном и том же порядке (то есть взаимодействие с объектами окна может осуществляться в произвольном порядке).

Предположение, что пользователь будет выполнять действия в определенном порядке, характерно для процедурного программирования.

Следует избегать использования принципов процедурного программирования. Процедурная программа часто создает у пользователя ощущение, что не он управляет программой, а она им. Когда мы пишем программу под **Windows**, имеет смысл дать пользователю почувствовать, что он управляет **событиями**.

Именно поэтому введена в графический интерфейс пользователя кнопка (**Button**) "Рассчитать оценку", с помощью которой пользователь может сообщить программе, что все готово для расчета оценки. При этом программа не должна выполнять расчеты, пока пользователь не щелкнет на этой кнопке.

Итак – этап проектирования завершен.

Далее приводится окончательный вариант технического задания.

2.3.5. ТЕХНИЧЕСКОЕ ЗАДАНИЕ Проект Расчет оценок

Общие положения

Графический интерфейс программы содержит следующие элементы:

➢ Три переключателя (**RadioButton**), представляющие три факультета, на которых могут обучаться студенты.

➢ Четыре окна редактирования (**TextBox**) с надписями (**Label**) "промежуточный экзамен", "заключительный экзамен", "реферат" и "выступление", в которые пользователь будет вводить компоненты оценки, необходимые для вычисления итоговой оценки.

➢ Одну кнопку (**Button**) с надписью "Вычислить оценку", В результате щелчка на ней выводится окно сообщения с числовым и буквенным представлением оценки.

➢ Одну кнопку (**Button**) с надписью "Сброс", с помощью которой очищается содержимое этих четырех окон редактирования.

Вывод программы

Итоговая оценка студента в числовом виде и оценка в буквенном виде — в окне сообщения.

Ввод в программу

Пользователь указывает следующие данные:

✓ Факультет, на котором учится студент; он устанавливается с помощью группы переключателей (**RadioButton**).

✓ Если выбран факультет АНГЛИЙСКОГО языка, то в соответствующие ЧЕТЫРЕ окна редактирования вводятся ЧЕТЫРЕ оценки: за экзамен в середине семестра, заключительный экзамен, реферат и выступление.

✓ Если выбран факультет МАТЕМАТИКИ, то в соответствующие ДВА окна редактирования вводятся ДВЕ оценки за промежуточный экзамен, а также за заключительный экзамен.

✓ Если выбран факультет ЕСТЕСТВЕННЫХ наук, то в соответствующие ТРИ окна редактирования вводятся ТРИ оценки за экзамен в середине семестра, заключительный экзамен и реферат.

Правила обработки

- Оценка студента факультета АНГЛИЙСКОГО языка вычисляется как сумма 25% от оценки за экзамен в середине семестра, 25% от оценки за заключительный экзамен, 30% от оценки за реферат и 20% от оценки за выступление.

- Оценка студента факультета МАТЕМАТИКИ вычисляется, как сумма 50% от оценки за экзамен в середине семестра и 50% от оценки за заключительный экзамен.

- Оценка студента факультета ЕСТЕСТВЕННЫХ наук вычисляется как сумма 40% от оценки за экзамен в середине семестра, 40% от оценки за заключительный экзамен и 20% от оценки за реферат.

На каждом факультете принято свое соотношение между числовой оценкой студента и ее буквенным эквивалентом (см. табл. 1).

Напоминаю, что этап проектирования представляет собой итеративный процесс, и что часто приходится неоднократно возвращаться к одному и тому же месту.

2.4. Четвертый этап: разработка

Этап разработки во многих отношениях наиболее увлекательная часть методологии разработки Компьютерных моделей реальных или концептуальных систем. Фактически мы приступим к написанию и анализу кода на языке **C#**.

В течение фазы **разработки** нам придется постоянно обращаться к техническому заданию, чтобы не отступать от него ни на шаг. Любые отклонения от технического задания должны быть одобрены руководителем проекта (преподавателем) или Заказчиком (деканатом).

Этап разработки будет разбит на две части. Сначала необходимо разработать **консольную** программу, которая будет выполнять те же расчеты, что и окончательная версия приложения под **Windows**. Различие между ними состоит в том, что консольная программа не будет обладать графическим интерфейсом (**GUI**).

Консольное приложение позволит убедиться в работоспособности внутреннего алгоритма программы. После этого будем заниматься построением графического интерфейса пользователя и внедрением готового алгоритма в версию, работающую под **Windows**.

2.4.1. Определение составляющих систему классов в консольном варианте Проекта

После того как все требования к системе (Проект Расчет Оценок) зафиксированы в Техническом задании, можно приступать к определению составляющих систему классов. Одним из способов выявления классов является выделение всех СУЩЕСТВИТЕЛЬНЫХ в требованиях к системе, т. е. ОБЪЕКТОВ (например, людей), мест и предметов. Не слишком старайтесь определить все необходимые классы с первого раза. Вы можете исключать, добавлять и изменять классы на различных этапах работы над проектом. Для начала важно сформировать хоть какую то основу. Проектирование является итерационным процессом – пользуйтесь этим. Как и всегда при использовании мозгового штурма, сформируйте первоначальный вариант системы, имея при этом в виду, что окончательный результат может выглядеть совсем иначе.

Для Проекта Расчет Оценок в консольном варианте очевидны следующие классы**:** **Student**, **EnglishStudent**, **MathStudent**, **ScienceStudent**, **DisplayGrade**, **Grades*Console***.

Для Проекта Расчет Оценок в оконном варианте часть классов сохраняется *не*изменными. К ним относятся классы**:** **Student**, **EnglishStudent**, **MathStudent**, **ScienceStudent**. Также вводятся два новых класса**:** **DrawGUI**, **GradesGUI**.

Определение обязанностей каждого класса в консольном варианте Проекта

После выявления составляющих систему классов необходимо определить обязанности каждого из них. Какие данные будет содержать класс, и какие действия он должен выполнять?

2.4.1.1. Базовый класс Student. Листинг 1, файл Student.cs

Класс **Student** принят в качестве базового для классов **EnglishStudent, MathStudent, ScienceStudent** (эти три класса являются производными от базового). Базовый класс содержит шесть переменные экземпляра и свойства доступа к ним, которые являются общими для всех производных классов.

Кроме того, класс **Student** объявлен абстрактным (строка **005**, сл.), так как содержит абстрактный метод **Calculate()** (строка **014**). Этот метод переопределяется в производных классах в связи с тем, что расчет итоговой оценки для студентов каждого из трех различных факультетов имеет свои особенности. Класс **Student** содержит шесть Свойств для переменных **midterm, finalExamGrade, research, presentation, finalNumericGrade, finalLetterGrade,** которые имеют спецификатор доступности **protected** (строки **007…012**). Этот спецификатор определяет доступ к переменным не только из базового, но и из производных классов. Производные классы также имеют доступ к свойствам базового класса. В четырех свойствах осуществляется проверка значений переменных **midterm, finalExamGrade, research, presentation** (это промежуточные оценки). Если их значения выходят за установленный диапазон **0…100 (см. табл. 1)**, то метод **Show()** пространства имен **Windows.Forms** выводит информацию в окно сообщения (message box) и программа прекращает работу.

Листинг 1, файл Student.cs

```
001:  using System;
002:  using System.Windows.Forms;
003:  //  Класс Student содержит один абстрактный метод и шесть свойств
004:  //  абстрактный класс содержит хотя бы один абстрактный метод
005:  abstract class Student // базовый класс
006:  {// protected переменные экз-ра - доступны из базового и производных классов
007:  protected int midterm = 0;
008:  protected int finalExamGrade = 0;
009:  protected int research = 0;
010:  protected int presentation = 0;
011:  protected float finalNumericGrade = 0;
012:  protected String finalLetterGrade = "";
013:
014:  public abstract void Calculate(); // абстр-й м-д - без тела  и с " ; ".
015:  // М-д переопр-ся в произв-х кл-х: EnglishStudent, MathStudent, ScienceStudent
016:
017:  public int Midterm  // Св-во Midterm устан-т/возв-т зна-е пер-й экз-ра midterm
018:  {
019:  get
020:  {
021:  return midterm;
022:  }
023:  set
024:  {
025:  if (value < 0 | value > 100)
026:  {// Оценка должна быть в диапазоне 0…100
027:  MessageBox.Show("Ошиб-я оценка за экз-н в середине сем-ра (" + value + ") " +
028:  "Программа прекратила работу", "Расчет оценки",
029:  MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
030:  Environment.Exit (0); // Арг-т "0" м-да Exit() указ-т на норм-е завер-е прог-ы
031:  }
032:  else
033:  midterm = value;
034:  }
035:  } // окончание свойства Midterm
```

```
036:
037:  // Св-во FinalExamGrade устан-т/возв-т знач-е пер-й экз-ра finalExamGrade
038:  public int FinalExamGrade
039:  {
040:  get
041:  {
042:  return finalExamGrade;
043:  }
044:  set
045:  {
046:  if (value < 0 | value > 100)
047:  {// Оценка должна быть в диапазоне 0…100
048:   MessageBox.Show("Ошиб-я оценка за экз-н в конце семестра (" + value + ") " +
049:  "Программа прекратила работу", "Расчет оценки",
050:  MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
051:  Environment.Exit (0);
052:  }
053:  else
054:  finalExamGrade = value;
055:  }
056:  } // окончание свойства FinalExamGrade
057:
058:  // Св-во Research устан-т/возв-т значение пер-й экз-ра research
059:  public int Research
060:  {
061:  get
062:  {
063:  return research;
064:  }
065:  set
066:  {
067:  if (value < 0 | value > 100)
068:  {// Оценка должна быть в диапазоне 0…100
069:   MessageBox.Show("Ошибочная оценка за реферат (" + value + ") " +
070:   "Программа прекратила работу", "Расчет оценки",
071:   MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
072:   Environment.Exit (0);
073:  }
074:  else
075:  research = value;
076:  }
077:  }// окончание свойства Research
078:
079:  // Св-во Presentation устан-т/возв-т знач-е пер-ной экз-ра presentation
080:  public int Presentation
081:  {
082:  get
083:  {
084:  return presentation;
085:  }
086:  set
087:  {
088:  if (value < 0 | value > 100)
089:  {// Оценка должна быть в диапазоне 0…100
090:  MessageBox.Show("Ошибочная оценка за выступление (" + value + ") " +
091:  "Программа прекратила работу", "Расчет оценки",
092:  MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
093:  Environment.Exit(0) ;
094:  }
095:  else
096:  presentation = value;
097:  }
098:  }// окончание свойства Presentation
099:
```

```
100:  // Св-во FinalNumericGrade возв-т знач-е пер-ной экз-ра finalNumericGrade
101:  public float FinalNumericGrade
102:  {
103:  get
104:  {
105:  return finalNumericGrade;
106:  }
107:  }// окончание свойства FinalNumericGrade
108:
109:  // Св-во FinalLetterGrade возвр-т знач-е пер-ной экз-ра finalLetterGrade
110:  public string FinalLetterGrade
111:  {
112:  get
113:  {
114:  return finalLetterGrade;
115:  }
116:  } окончание свойства FinalLetterGrade
117:  }
```

2.4.1.2. Производный класс EnglishStudent. Листинг 2, файл EnglishStudent.cs

В этом классе осуществляется расчет числовой оценки (**finalNumericGrade**) студента факультета АНГЛИЙСКОГО языка, а также определение ее буквенного эквивалента (**finalLetterGrade**). Для расчета итоговой оценки используются ЧЕТЫРЕ промежуточных**:** оценка за экзамен в середине семестра (**midterm),** оценка за заключительный экзамен (**finalExamGrade),** оценка за реферат (**research),** оценка за выступление (**presentation)**. Также используются весовые коэффициенты, учитывающие долю промежуточных оценок в итоговой, соответственно (см. выше выражение (1)) (см. строки **011...014**)**:**

```
ENGLISH_MIDTERM_PERCENTAGE = 0,25; ENGLISH_FINALEXAM_PERCENTAGE = 0,25;
ENGLISH_RESEARCH_PERCENTAGE = 0,30; ENGLISH_PRESENTATION_PERCENTAGE = 0,20.
```

Класс **EnglishStudent** является производным от базового класса **Student.** Класс **EnglishStudent** имеет доступ к ШЕСТИ свойствам **Midterm, FinalExamGrade, Research, Presentation, FinalNumericGrade, FinalLetterGrade** базового класса и переопределяет его метод **Calculate()** без параметров (строки **054…081**). Абстрактный (переопределя**емый**) метод **Calculate()** базового класса **Student** (строка **014**) и переопределя**ющий** метод **Calculate()** производного класса **EnglishStudent** (строки **054…081**) имеют одинаковые сигнатуры, о чем свидетельствует ключевое слово **override.** Этот метод **Calculate()** используется при реализации программы в консольном варианте. Четыре промежуточные оценки, используемые этим методом для расчета итоговой оценки, вводятся с клавиатуры в соответствии с оператором **ReadLine()**.

В классе **EnglishStudent** содержится еще один метод **Calculate()** с **четырьмя параметрами** (строки **026…048**), он перегружает метод **Calculate()** без параметров (строки **054…081**). Метод **Calculate()** с четырмя параметрами (строки **026…048**) используется при реализации программы в варианте **GUI** (**G**raphical **U**ser **I**nterface). Промежуточные оценки, используемые этим методом для расчета итоговой оценки, передаются из окон редактирования (**TextBox**), реализованных в классе **DrawGUI,** при кликании мышкой на кнопке (**Button**) «Расчет оценки»**.**

Листинг 2, файл EnglishStudent.cs

```
001:  /* В классе EnglishStudent два метода Calculate():
002:   один для консольного варианта программы Расчета оценок,
003:   другой - для варианта GUI. Это позволяет исполь-ть класс EnglishStudent
004:   в неизменном виде для обоих вариантов программы Расчета оценок*/
005:  using System;
006:  using System.Windows.Forms;
007:
008:  // класс EnglishStudent производный;  класс Student базовый
```

```
009:  class EnglishStudent : Student
010:  {// четыре весовых коэфф-та, учитывающие долю промеж-й оценки в итоговой
011:  const float ENGLISH_MIDTERM_PERCENTAGE = .25F;
012:  const float ENGLISH_FINALEXAM_PERCENTAGE = .25F;
013:  const float ENGLISH_RESEARCH_PERCENTAGE = .30F;
014:  const float ENGLISH_PRESENTATION_PERCENTAGE = .20F;
015:
016:  public EnglishStudent() // конструктор класса
017:  {
018:  Console.WriteLine("** Расчет оценки для студента факультета АНГЛИЙСКОГО языка **");
019:  }
020:   /* Перегружающий м-д Calculate() с четырмя параметрами.
021:    Этот м-д используется при реализации программы в варианте GUI.
022:    Эти четыре пар-ра передаются методу из класса DrawGUI.
023:    Они вводятся в окна редактирования (TextBoxes).
024:    Метод осуществляет расчет итоговой оценки студента фак-та англ-го языка
025:    в числовом варианте и ее преобразование в буквенный эквивалент*/
026:  public void Calculate(int midterm, int finalExamGrade, int research, int
      presentation)
027:  { // осуществляется обращение к свойствам базового класса Student
028:  Midterm = midterm;
029:  FinalExamGrade = finalExamGrade;
030:  Research = research;
031:  Presentation = presentation;
032:  // расчет оценки в числовом варианте
033:  finalNumericGrade =
034:  (Midterm * ENGLISH_MIDTERM_PERCENTAGE) +
035:  (FinalExamGrade * ENGLISH_FINALEXAM_PERCENTAGE) +
036:  (Research * ENGLISH_RESEARCH_PERCENTAGE) +
037:  (Presentation * ENGLISH_PRESENTATION_PERCENTAGE);
038:   // определение буквенного эквивалента числовой оценки
039:  if (finalNumericGrade >= 93) finalLetterGrade = "A";
040:  else
041:  if ((finalNumericGrade >= 85) & (finalNumericGrade < 93)) finalLetterGrade = "B";
042:  else
043:  if ((finalNumericGrade >= 78) & (finalNumericGrade < 85)) finalLetterGrade = "C";
044:   else
045:    if ((finalNumericGrade >= 70) & (finalNumericGrade < 78)) finalLetterGrade = "D";
046:  else
047:  if (finalNumericGrade < 70) finalLetterGrade = "F";
048:  } // окончание 1-го метода Calculate()
049:
050:  /* Переопределяющий м-д  Calculate() без параметров. Этот м-д используется
051:   при реализации программы в консольном варианте.
052:   Метод осуществляет расчет итоговой оценки студента факультета анг-го языка
053:   в числовом варианте и ее преобразование в буквенный эквивалент*/
054:  override public void Calculate()
055:  {
056:  // осуществляется обращение к свойствам базового класса Student
057:  Console.Write("Введите оценку за экзамен в середине семестра: ");
058:  Midterm = int.Parse(Console.ReadLine());
059:  Console.Write("Введите оценку за заключительный экзамен: ");
060:  FinalExamGrade = int.Parse(Console.ReadLine());
061:  Console.Write("Введите оценку за реферат: ");
062:  Research = int.Parse(Console.ReadLine());
063:  Console.Write("Введите оценку за выступление: ");
064:  Presentation = int.Parse(Console.ReadLine());
065:  // расчет оценки в числовом варианте
066:  finalNumericGrade =
067:  (midterm * ENGLISH_MIDTERM_PERCENTAGE) +
068:  (finalExamGrade * ENGLISH_FINALEXAM_PERCENTAGE) +
069:  (research * ENGLISH_RESEARCH_PERCENTAGE) +
070:  (presentation * ENGLISH_PRESENTATION_PERCENTAGE);
071:  // определение буквенного эквивалента числовой оценки
```

```
072:  if (finalNumericGrade >= 93) finalLetterGrade = "A";
073:  else
074:  if ((finalNumericGrade >= 85) & (finalNumericGrade < 93)) finalLetterGrade = "B";
075:  else
076:  if   ((finalNumericGrade >= 78) & (finalNumericGrade < 85)) finalLetterGrade = "C";
077:  else
078:  if ((finalNumericGrade >= 70) & (finalNumericGrade < 78)) finalLetterGrade = "D";
079:  else
080:  if (finalNumericGrade < 70) finalLetterGrade = "F";
081:  }// окончание 2-го метода Calculate()
082:  }
```

2.4.1.3. Производный класс MathStudent. Листинг 3, файл MathStudent.cs

В этом классе осуществляется расчет числовой оценки (**finalNumericGrade**) студента факультета МАТЕМАТИКИ, а также определение ее буквенного эквивалента (**finalLetterGrade**). Для расчета итоговой используются ДВЕ промежуточные оценки: оценка за экзамен в середине семестра (**midterm),** оценка за заключительный экзамен (**finalExamGrade)**. Также используются весовые коэффициенты (их ДВА), учитывающие долю промежуточных оценок в итоговой, соответственно (см. строки **010...011**):

```
MATH_MIDTERM_PERCENTAGE = 0,50; MATH_FINALEXAM_PERCENTAGE = 0,50.
```

Класс **MathStudent** является производным от базового класса **Student.** Класс **MathStudent** имеет доступ к ЧЕТЫРЕМ свойствам **Midterm, FinalExamGrade, FinalNumericGrade, FinalLetterGrade** базового класса и переопределяет его метод **Calculate()** без параметров (строки **046…065**). Абстрактный (переопределяемый) метод **Calculate()** базового класса **Student** (строка **014**) и переопределяющий метод **Calculate()** производного класса **MathStudent** (строки **046…065**) имеют одинаковые сигнатуры, о чем свидетельствует ключевое слово **override.** Этот метод **Calculate()** используется при реализации программы в консольном варианте. Две промежуточные оценки, используемые этим методом для расчета итоговой оценки, вводятся с клавиатуры в соответствии с оператором **ReadLine()**.

В классе **MathStudent** содержится еще один метод **Calculate()** с **двумя параметрами** (строки **023…040**), он перегружает метод **Calculate()** без параметров (строки **046…065**). Метод **Calculate()** с двумя параметрами (строки **023…040**) используется при реализации программы в варианте **GUI** (**G**raphical **U**ser **I**nterface). Промежуточные оценки, используемые этим методом для расчета итоговой оценки, передаются из окон редактирования (**TextBox**), реализованных в классе **DrawGUI,** при кликании мышкой на кнопке (**Button**) «Расчет оценки».

Листинг 3, файл MathStudent.cs

```
001:  /* В классе MathStudent два метода Calculate():
002:   один для консольного варианта программы Расчета оценок,
003:   другой - для варианта GUI. Это позволяет исполь-ть класс MathStudent
004:   в неизменном виде для обоих вариантов программы Расчета оценок*/
005:  using System;
006:  using System.Windows.Forms;
007:  // класс MathStudent производный;  класс Student базовый
008:  class MathStudent : Student
009:  {// два весовых коэфф-та, учитывающие долю промеж-й оценки в итоговой
010:  const float MATH_MIDTERM_PERCENTAGE = .50F;
011:  const float MATH_FINALEXAM_PERCENTAGE = .50F;
012:
013:  public MathStudent()
014:  {
015:  Console.WriteLine ("** Расчет оценки для студента факультета МАТЕМАТИКИ **");
016:  }
017:  /* Перегружающий м-д  Calculate() с двумя параметрами.
018:   Этот м-д используется при реализации программы в варианте GUI.
```

```
019:  Эти два пар-ра передаются методу из класса DrawGUI.
020:  Они вводятся в окна редактирования (TextBoxes).
021:  Метод осуществляет расчет итоговой оценки студента факультета математики
022:  в числовом варианте и ее преобразование в буквенный эквивалент*/
023:  public void Calculate(int midterm, int finalExamGrade)
024:  { // осуществляется обращение к свойствам базового класса Student
025:  Midterm = midterm;
026:  FinalExamGrade = finalExamGrade;
027:  // расчет оценки в числовом варианте
028:  finalNumericGrade =
029:  (midterm * MATH_MIDTERM_PERCENTAGE) +
030:  (finalExamGrade * MATH_FINALEXAM_PERCENTAGE);
031:  // определение буквенного эквивалента числовой оценки
032:   if (finalNumericGrade >= 90) finalLetterGrade = "A";
033:  else
034:  if ((finalNumericGrade >= 83) & (finalNumericGrade < 90)) finalLetterGrade = "B";
035:  else
036:  if ((finalNumericGrade >= 76) & (finalNumericGrade < 83)) finalLetterGrade = "C";
037:  else
038:  if ((finalNumericGrade >= 65) & (finalNumericGrade < 76)) finalLetterGrade = "D";
039:  else if (finalNumericGrade < 65) finalLetterGrade = "F";
040:  }// окончание 1-го метода Calculate()
041:
042:  /* Переопределяющий м-д  Calculate() без параметров. Этот м-д используется
043:   при реализации программы в консольном варианте.
044:   Метод осуществляет расчет итоговой оценки студента факультета математики
045:   в числовом варианте и ее преобразование в буквенный эквивалент*/
046:  public override void Calculate()
047:  {
048:  Console.Write("Введите оценку за экзамен в середине семестра: ");
049:  Midterm = int.Parse(Console.ReadLine());
050:  Console.Write("Введите оценку за заключительный экзамен: ");
051:  FinalExamGrade = int.Parse(Console.ReadLine());
052:  // расчет оценки в числовом варианте
053:  finalNumericGrade =
054:  (midterm * MATH_MIDTERM_PERCENTAGE) +
055:  (finalExamGrade * MATH_FINALEXAM_PERCENTAGE);
056:  // определение буквенного эквивалента числовой оценки
057:   if  (finalNumericGrade >= 90) finalLetterGrade = "A";
058:  else
059:  if ((finalNumericGrade >= 83) & (finalNumericGrade < 90)) finalLetterGrade = "B";
060:  else if ((finalNumericGrade >= 76) &
061:  (finalNumericGrade < 83)) finalLetterGrade = "C";
062:  else
063:  if ((finalNumericGrade >= 65) & (finalNumericGrade < 76)) finalLetterGrade = "D";
064:  else if (finalNumericGrade < 65) finalLetterGrade = "F";
065:  }// окончание 2-го метода Calculate()
066:  }
```

2.4.1.4. Производный класс ScienceStudent. Листинг 4, файл ScienceStudent.cs [1]

В этом классе осуществляется расчет числовой оценки (**finalNumericGrade**) студента факультета ЕСТЕСТВЕННЫХ наук, а также определение ее буквенного эквивалента (**finalLetterGrade**). Для расчета итоговой используются ТРИ промежуточные оценки: оценка за экзамен в середине семестра (**midterm),** оценка за заключительный экзамен (**finalExamGrade),** оценка за реферат (**research)**. Также используются весовые коэффициенты, учитывающие долю промежуточных оценок в итоговой, соответственно (см. строки **010...012**):

```
SCIENCE_MIDTERM_PERCENTAGE = 0,40; SCIENCE_FINALEXAM_PERCENTAGE = 0,40;
```

[1] Расчет итоговых оценок для разных факультетов отличается количеством учитываемых промежуточных оценок, а также количеством и значениями весовых коэффициентов (см. Техническое задание).

SCIENCE_RESEARCH_PERCENTAGE = 0,20.

Класс **ScienceStudent** является производным от базового класса **Student.** Класс **ScienceStudent** имеет доступ к ПЯТИ свойствам **Midterm, FinalExamGrade, Research, FinalNumericGrade, FinalLetterGrade** базового класса и переопределяет его метод **Calculate()** без параметров (строки **050…073**). Абстрактный (переопределя**емый**) метод **Calculate()** базового класса **Student** (строка **014**) и переопределя**ющий** метод **Calculate()** производного класса **ScienceStudent** (строки **050…073**) имеют одинаковые сигнатуры, о чем свидетельствует ключевое слово **override.** Этот метод **Calculate()** используется при реализации программы в консольном варианте. Три промежуточные оценки, используемые этим методом для расчета итоговой оценки, вводятся с клавиатуры в соответствии с оператором **ReadLine()**.

В классе **ScienceStudent** содержится еще один метод **Calculate()** с **тремя параметрами** (строки **024…044**), он перегружает метод **Calculate()** без параметров (строки **050…073**). Метод **Calculate()** с четырмя параметрами (строки **024…044**) используется при реализации программы в варианте **GUI** (**G**raphical **U**ser **I**nterface). Промежуточные оценки, используемые этим методом для расчета итоговой оценки, передаются из окон редактирования (**TextBox**), реализованных в классе **DrawGUI,** при кликании мышкой на кнопке (**Button**) «Расчет оценки».

Листинг 4, файл ScienceStudent.cs

```
001:  /* В классе SciencehStudent два метода Calculate():
002:   один для консольного варианта программы Расчета оценок,
003:   другой - для варианта GUI. Это позволяет исполь-ть класс SciencehStudent
004:   в неизменном виде для обоих вариантов программы Расчета оценок*/
005:  using System;
006:  using System.Windows.Forms;
007:  // класс MathStudent производный;  класс Student базовый
008:  class ScienceStudent : Student
009:  {// три весовых коэфф-та, учитывающие долю промеж-й оценки в итоговой
010:  const float SCIENCE_MIDTERM_PERCENTAGE = .40F;
011:  const float SCIENCE_FINALEXAM_PERCENTAGE = .40F;
012:  const float SCIENCE_RESEARCH_PERCENTAGE = .20F;
013:
014:  public ScienceStudent()
015:  {// конструктор класса
016:  Console.WriteLine("** Расчет оценки для студента фак-та ЕСТЕСТВЕННЫХ наук **");
017:   }
018:   /* Перегружающий м-д Calculate() с тремя параметрами.
019:    Этот м-д используется при реализации программы в варианте GUI.
020:    Эти три пар-ра передаются методу из класса DrawGUI.
021:    Они вводятся в окна редактирования (TextBoxes).
022:    Метод осуществляет расчет итоговой оценки студента фак-та ЕСТЕСТВЕННЫХ наук
023:    в числовом варианте и ее преобразование в буквенный эквивалент*/
024:   public void Calculate(int midterm, int finalExamGrade, int research)
025:  { // осуществляется обращение к свойствам базового класса Student
026:  Midterm = midterm;
027:  FinalExamGrade = finalExamGrade;
028:  Research = research;
029:  // расчет оценки в числовом варианте
030:  finalNumericGrade =
031:  (Midterm * SCIENCE_MIDTERM_PERCENTAGE) +
032:  (FinalExamGrade * SCIENCE_FINALEXAM_PERCENTAGE) +
033:  (Research * SCIENCE_RESEARCH_PERCENTAGE);
034:  // определение буквенного эквивалента числовой оценки
035:  if  (finalNumericGrade >= 90) finalLetterGrade = "A";
036:  else
037:  if ((finalNumericGrade >= 80) & (finalNumericGrade < 90)) finalLetterGrade = "B";
038:  else
039:  if ((finalNumericGrade >= 70) & (finalNumericGrade < 80)) finalLetterGrade = "C";
```

```
040:  else
041:   if ((finalNumericGrade >= 60) & (finalNumericGrade < 70)) finalLetterGrade =
      "D";
042:  else
043:  if (finalNumericGrade < 60) finalLetterGrade = "F";
044:  }// окончание 1-го метода Calculate()
045:
046:  /* Переопределяющий м-д  Calculate() без параметров. Этот м-д используется
047:  при реализации программы в консольном варианте.
048:  Метод осуществляет расчет итоговой оценки студента фак-та ЕСТЕСТВЕННЫХ наук
049:  в числовом варианте и ее преобразование в буквенный эквивалент*/
050:  override public void Calculate()
051:  {
052:  Console.Write("Введите оценку за экзамен в середине семестра: ") ;
053:  Midterm = int.Parse(Console.ReadLine());
054:  Console.Write("Введите оценку за заключительный экзамен: ");
055:  FinalExamGrade = int.Parse(Console.ReadLine());
056:  Console.Write("Введите оценку за реферат: ");
057:  Research = int.Parse(Console.ReadLine());
058:  // расчет оценки в числовом варианте
059:  finalNumericGrade =
060:  (midterm * SCIENCE_MIDTERM_PERCENTAGE) +
061:  (finalExamGrade * SCIENCE_FINALEXAM_PERCENTAGE) +
062:  (research * SCIENCE_RESEARCH_PERCENTAGE);
063:  // определение буквенного эквивалента числовой оценки
064:  if (finalNumericGrade >= 90) finalLetterGrade = "A";
065:  else
066:  if ((finalNumericGrade >= 80) & (finalNumericGrade < 90)) finalLetterGrade = "B";
067:  else
068:  if ((finalNumericGrade >= 70) & (finalNumericGrade < 80)) finalLetterGrade = "C";
069:  else
070:  if ((finalNumericGrade >= 60) & (finalNumericGrade < 70)) finalLetterGrade = "D";
071:  else
072:  if (finalNumericGrade < 60) finalLetterGrade = "F";
073:  }// окончание 2-го метода Calculate()
074:  }
```

2.4.1.5. Класс DysplayGrade. Листинг 5, файл DysplayGrade.cs

В этом классе содержатся три перегруженных конструктора **DysplayGrade()**, которые выводят итоговую оценку студентов факультетов английского языка, математики и естественных наук в окно сообщения (message box) **MessageBox.Show()**. Объект **MessageBox** содержит несколько перегруженных методов **Show()**, которые отличаются сигнатурами. Параметрами у конструкторов **DysplayGrade()** являются переменные экземпляра, хранящие значения промежуточных оценок, необходимых для расчета итоговой оценки. Эти конструкторы имеют различные сигнатуры, т.к. количество промежуточных оценок для каждого из факультетов различно (см. Техническое задание).

Листинг 5, файл DysplayGrade.cs

```
001:  using System; // Осуществляется вывод в окно сообщения результатов расчета оценки
002:  using System.Windows.Forms;
003:  /* В классе DisplayGrade содержится три перегруженных конструктора:
004:  они имет разную сигнатуру (то есть разное кол-во пар-ов).
005:   Версия метода Show() принимает четыре аргумента:
006:   Первый аргумент представляет надпись, отображаемую в окне сообщения.
007:   Второй аргумент представляет собой надпись в строке заголовка (title bar) окна.
008:   Третий аргумент определяет номер и название каждой из кнопок, отображаемых в окне.
009:   В данном случае применена константа C#, указывающая на то,
010:   что должна отображаться кнопка OK.
011:   Четвертый аргумент указывает пиктограмму, отображаемую в окне сообщения.
012:   Здесь применена константа, обозначающая пиктограмму справки (Information).*/
013:  class DisplayGrade
014:  {
```

```
015:  public DisplayGrade(int midterm, int finalExamGrade, int research,
      int presentation, float finalNumericGrade, string finalLetterGrade)
016:   {// конст-р имеет 6 параметров
017:   MessageBox.Show ("*** Студент факультета АНГЛИЙСКОГО языка ***\n\n" +
018:  "Оценка за экзамен в середине семестра: " + midterm + "\n" +
019:  "Оценка за экзамен в конце семестра: " + finalExamGrade + "\n" +
020:  "Оценка за реферат: " + research + "\n" +
021:  "Оценка за выступление: " + presentation + "\n\n" +
022:  "Итоговая числовая оценка: " + finalNumericGrade + "\n" +
023:  "Итоговая буквенная оценка: " + finalLetterGrade,
024:  "Расчет оценки", MessageBoxButtons.OK, MessageBoxIcon.Information);
025:  }        // конец конст-ра DisplayGrade с 6 параметрами
026:
027:  public DisplayGrade(int midterm, int finalExamGrade,
      float finalNumericGrade, string finalLetterGrade)
028:  { // конст-р имеет 4 параметра
029:  MessageBox.Show("*** Студент факультета МАТЕМАТИКИ ***\n\n" +
030:  "Оценка за экзамен в середине семестра: " + midterm + "\n" +
031:   "Оценка за экзамен в конце семестра: " + finalExamGrade + "\n\n" +
032:   "Итоговая числовая оценка: " + finalNumericGrade + "\n" +
033:  "Итоговая буквенная оценка: " + finalLetterGrade,
034:  "Расчет оценки", MessageBoxButtons.OK, MessageBoxIcon.Information);
035:  }        // конец конст-ра DisplayGrade с 4 параметрами
036:
037:  public DisplayGrade(int midterm, int finalExamGrade,
      int research, double finalNumericGrade, string finalLetterGrade)
038:    {// конст-р имеет 5 параметров
039:  MessageBox.Show("*** Студент факультета ЕСТЕСТВЕННЫХ наук ***\n\n" +
040:  "Оценка за экзамен в середине семестра: " + midterm + "\n" +
041:  "Оценка за экзамен в конце семестра: " + finalExamGrade + "\n" +
042:  "Оценка за реферат: " + research + "\n\n" +
043:  "Итоговая числовая оценка: " + finalNumericGrade + "\n" +
044:  "Итоговая буквенная оценка: " + finalLetterGrade,
045:  "Расчет оценки", MessageBoxButtons.OK, MessageBoxIcon.Information);
046:  }        // конец конст-ра DisplayGrade с 5 параметрами
047:  }        // конец класса DisplayGrade
```

2.4.1.6. Начальный класс GradesConsole – консольный вариант Проекта Расчет оценки. Листинг 6, файл GradesConsole.cs

В этом классе находится метод **Main(),** который содержит цикл **while** (строка **015**, сл.), позволяющий рассчитывать оценку для *не*ограниченного количества студентов. Цикл выполняется всегда, если переменная **moreGradesToCalculate** принимает значение **Yes** (строки **040, 041, 042** и строка **015**). В теле цикла **while** содержится оператор **switch** (строка **018**, сл.).

Весь программный код, необходимый для создания объектов классов **EnglishStudent**, **MathStudent** или **ScienceStudent** (объектами этих классов являются студенты конкретных факультетов) содержится в операторе **switch**. Оператор **switch** проверяет (строка **018**) введенное пользователем в блоке запроса число (строки **017 и 048…069**), которое может принимать значение **1**, **2** или **3**. Это значение присваивается переменной **response** (строки **054, 055, 008** и **018**). Если пользователь ввел число **1** (это соответствует выбору студента факультета АНГЛИЙСКОГО языка – строка **054**), то объявляется объектная переменная **eStudent** (строка **021**), представляющая объект класса **EnglishStudent**:

```
021:  EnglishStudent  eStudent = new EnglishStudent();
```

После чего мы обращаемся к методу **Calculate()** объекта класса **EnglishStudent** (см. строки **054…081** в этом классе) следующим образом:

```
0.22:  eStudent.Calculate();
```

Метод **Calculate() (**строки **054…081)** класса **EnglishStudent** *переопределяет* абстрактный метод **Calculate()** класса **Student** (строка **014**) и рассчитывает оценку для студента факультета английского языка при реализации консольного варианта программы Расчет оценок.

За этим следует создание объекта **x** класса **DisplayGrade**. При этом в зависимости от количества и типов аргументов вызывается один из трех созданных перегруженных конструкторов (соответственно трем факультетам). Для **отображения** шести оценок (это результат работы программы Расчет оценки) в окне сообщения (message box), которые получил студент факультета английского языка вызывается конструктор с шестью параметрами (строки **015…025** в классе **DisplayGrade**)

```
023:  DisplayGrade x = new DisplayGrade (eStudent.Midterm,
                                         eStudent.FinalExamGrade,
                                         eStudent.Research,
                                         eStudent.Presentation,
                                         eStudent.FinalNumericGrade,
                                         eStudent.FinalLetterGrade);
```

Значения шести аргументов передаются конструктору посредством обращения к одноименным свойствам, которые расположены в базовом классе **Student**. Имена этих свойств указаны после оператора уточнения "**.**" (см. выше строку **023**).

В классе **GradesConsole** кроме метода **Main()** имеется также метод **WhatKindOfStudent()** (строки **048…069,** это блок запроса) в котором пользователь задает тип студента (строки **054, 055**), соответствующий факультету на котором студент обучается.

Листинг 6, файл GradesConsole.cs

```
001:  using System;                          // Консольный вариант Проекта Расчет оценок
002:  using System.Windows.Forms;
003:
004:  class GradesConsole
005:  {
006:  public static void Main(string [] args)
007:  {
008:  string response;  // в этой переменной хранится признак типа студента
009:  string moreGradesToCalculate;  // в этой пер-й хр-ся ответ на запрос о необ-ти произ-ть расчет оценок
010:
011:  Console.Write("Вы желаете рассчитать оценку? ");
012:  moreGradesToCalculate = Console.ReadLine();
013:  moreGradesToCalculate = moreGradesToCalculate.ToUpper();
014:
015:  while(moreGradesToCalculate == "YES")
016:   {// в цикле реализован расчет для неограниченного количества студентов
017:   response = WhatKindOfStudent();
018:   switch(int.Parse(response))
019:   {// осуществляется переход к расчету оценки студента соответствующего факультета
020:   case 1:
021:  EnglishStudent eStudent = new EnglishStudent();
022:  eStudent.Calculate();          // обращение к м-ду Calculate() без парам-в класса EnglishStudent
023:  DisplayGrade x = new DisplayGrade (eStudent.Midterm,
                                         eStudent.FinalExamGrade,
                                         eStudent.Research,
                                         eStudent.Presentation,
                                         eStudent.FinalNumericGrade,
                                         eStudent.FinalLetterGrade); //обр-ние к кон-ру с 6-ю
024:  break;            // параметрами в классе DisplayGrade для вывода рез-в расчета в окно сообщения
025:  case 2:
026:  MathStudent mStudent = new MathStudent();
```

```
027:  mStudent.Calculate();              // обращение к м-ду Calculate() без парам-в класса MathStudent
028:  DisplayGrade y = new DisplayGrade (mStudent.Midterm,
                                         mStudent.FinalExamGrade,
                                         mStudent.FinalNumericGrade,
                                         mStudent.FinalLetterGrade);// обр-ние к конс-ру с 4-мя
029:  break;             // параметрами в классе DisplayGrade для вывода рез-в расчета в окно сообщения
030:  case 3:
031:  ScienceStudent sStudent = new ScienceStudent();
032:  sStudent.Calculate();          // обращение к м-ду Calculate() без парам-в класса ScienceStudent
033:  DisplayGrade z = new DisplayGrade (sStudent.Midterm,
                                         sStudent.FinalExamGrade,
                                         sStudent.Research,
                                         sStudent.FinalNumericGrade,
                                         sStudent.FinalLetterGrade);// обр-ние к конс-ру с 5-ю
034:  break;             // параметрами в классе DisplayGrade для вывода рез-в расчета в окно сообщения
035:  default:
036:  MessageBox.Show (response + " - неправильный тип студента ");
037:  break;
038:  }      // конец switch
039:
040:  Console.Write("Вы желаете рассчитать другую оценку? ");
041:  moreGradesToCalculate = Console.ReadLine();
042:  moreGradesToCalculate = moreGradesToCalculate.ToUpper();
043:  }      // конец цикла while
044:
045:  Console.WriteLine("Благодарим за использование " + "программы расчета оценок!");
046:  }      // конец метода Main
047:
048:  public static string WhatKindOfStudent()
049:  { // В этом методе задается тип студента (то есть фак-тет на котором он обучается)
053:  string response;
054:  Console.Write("Введите тип студента: " + "(1 = English, 2 = Math, 3 = Science): ");
055:  response = Console.ReadLine();
056:  if
057:  (response == "")
058:   {
059:  MessageBox.Show("Вы должны выбрать тип студента! ");
060:  Environment.Exit(0);
061:  }
062:  else
063:  if (int.Parse(response) < 1 |  int.Parse(response) > 3)
064:  {
065:  MessageBox.Show (response + " - неправильный тип студента! ");
066:  Environment.Exit(0);
067:  }
068:  return response;
069:  }         // конец метода WhatKindOfStudent
070:  }         // конец класса GradesConsole
```

2.4.1.7. Создание выполнимого файла, реализующего консольный вариант проекта Расчет оценок

Есть два способа создания выполнимого файла.

1-й способ. С помощью команды,
запускаемой из командной строки **Visual Studio .NET 2003 Command Promt** ,

```
csc GradesConsole.cs  EnglishStudent.cs MathStudent.cs ScienceStudent.cs DisplayGrade.cs Student.cs
```

создаем выполнимый файл **GradesConsole.exe.** Его запуск обеспечивает выполнение консольного варианта Проекта Расчет оценок.

2-й способ. Другая возможность создания **консольного** варианта проекта**:** запустить среду **MS VS .NET;** реализовать **ConsoleApplication;** скопировать содержимое файлов **GradesConsole.cs, EnglishStudent.cs, MathStudent.cs, ScienceStudent.cs, DisplayGrade.cs, Student.cs** в окно среды**;** подключить пространство **System.Windows.Forms** и нажать на клавишу **F5** (естественно, что при копировании содержимого файлов разумно удалить строки – using System; using System.Windows.Forms; – из всех файлов кроме файла **GradesConsole.cs**).

2.4.1.8. Взаимодействие классов в консольном варианте проекта Расчет оценок

Консольный вариант **проекта Расчет оценок** реализуется в рассмотренных выше классах:

- ✓ **GradesConsole.cs** – начальный класс (см. разд. **2.4.1.6**);
- ✓ **EnglishStudent.cs** – производный класс (см. разд. **2.4.1.2**);
- ✓ **MathStudent.cs** – производный класс (см. разд. **2.4.1.3**);
- ✓ **ScienceStudent.cs** – производный класс (см. разд. **2.4.1.4**);
- ✓ **DisplayGrade.cs** (см. разд. **2.4.1.5**);
- ✓ **Student.cs** – базовый класс (см. разд. **2.4.1.1**).

Большинство классов не являются изолированными. Хотя класс должен выполнять определенные обязанности, часто он вынужден взаимодействовать с другими классами**:** *1)* для получения необходимых данных или *2)* для доступа к методам другого класса. Для этого используются **сообщения** между классами. Когда какому–либо классу потребуется некоторая информация или понадобится выполнение определенных действий, он может послать соответствующее сообщение другому классу.

Проследим взаимодействие между классами в **консольном** варианте проекта Расчет оценок (*не останавливаясь на деталях*). Это взаимодействие реализуется при запуске на выполнение **PE**-файла **GradesConsole.exe** (см. разд. **2.4.1.7**). Для определенности будем прослеживать взаимодействие между классами в предположении расчета оценки для студента факультета МАТЕМАТИКИ *(взаимодействие между классами для расчета оценки студента других факультетов выполняется аналогично*).

После запуска файла **GradesConsole.exe** среда **.NET** обращается к статическому методу **Main()** (класс **GradesConsole**), который (метод) обеспечивает роль диспетчера при выполнении программы. Пользователь присваивает переменной **moreGradesToCalculate** значение **Yes** и начинает выполняться цикл **while.** В соответствии с методом **WhatKindOfStudent()** выбирается тип студента (как условились это студент факультета МАТЕМАТИКИ) и переменной **response** присваивается значение **2.** В соответствии с этим значением оператор **switch** обеспечивает объявление объектной переменной **mStudent** (строка **026**), представляющей объект класса **MathStudent**:

```
026:  MathStudent mStudent = new MathStudent();
```

После чего осуществляется обращение к методу **Calculate()** объекта класса **MathStudent** (см. строки **046…065** в этом классе) следующим образом:

```
0.27: mStudent.Calculate();
```

Метод **Calculate()** (строки **054…081)** класса **MathStudent** *переопределяет* абстрактный метод **Calculate()** базового абстрактного класса **Student** (строка **014**) и осуществляет расчет оценки для студента факультета МАТЕМАТИКИ при реализации консольного варианта программы Расчет оценок.

Далее создается объект **y** класса **DisplayGrade**

```
028:  DisplayGrade y = new DisplayGrade (mStudent.Midterm,
                                         mStudent.FinalExamGrade,
                                         mStudent.FinalNumericGrade,
                                         mStudent.FinalLetterGrade);  .
```

При этом вызывается перегруженный конструктор класса **DisplayGrade** с четырьмя аргументами (строки **027…035**) и в соответствии с методом **Show()** пространства **Windows.Forms** **отображаются** четыре оценки в окне сообщения (message box), которые получил студент факультета МАТЕМАТИКИ (это результат работы программы).

Значения четырех аргументов передаются конструктору **DisplayGrade()** посредством обращения к *одно*именным свойствам, которые расположены в базовом классе **Student**. Имена этих свойств указаны после оператора уточнения "**.**" (см. выше строку 028).

Далее пользователь присваивает переменной **moreGradesToCalculate** новое значение **Yes (**или **No)** и продолжает выполняться цикл **while** (или осуществляется выход из цикла и прекращение работы программы), то есть рассчитывается оценка для другого студента так, как описано выше и т.д..

2.4.2. Оконный вариант проекта Расчет оценок, реализующий GUI

Разработаем **Г**рафический **И**нтерфейс **П**ользователя (**GUI**) для проекта Расчет оценок. При этом классы **Student** (базовый)**, EnglishStudent, MathStudent, ScienceStudent** (производные) консольного варианта проекта Расчета оценок используются без изменений. Дополнительно разработаны: класс **DrawGUI** – производный от класса **Form** и начальный класс **GradesGUI,**

2.4.2.1. Производный класс DrawGUI. Листинг 7, файл DrawGUI.cs

Обычно, программный код для создания **GUI** (**G**raphical **U**ser **I**nterface – **Г**рафический **И**нтерфейс **П**ользователя) размещают в отдельном классе. Для формирования **GUI** создадим класс с именем **DrawGUI**.

Рассмотрим рисунок **GUI** (рис.1), определим элементы **GUI** и перечислим в табл. 2 объекты и классы **C#** из **System.Windows.Forms**, которые потребуются для создания этих элементов.

Таблица 2. Перечень элементов управления на **форме (см. рис. 1)**

Класс C# из System.Windows.Forms	**Элемент GUI**	**Имя объекта класса**
Button	кнопка Рассчитать оценку кнопка Сброс	**btnCalculate** **btnReset**
RadioButton	переключатель Английский переключатель Математики переключатель Естественный	**radEnglish** **radMath** **radScience**
Label	Факультет Оценки за английского языка, математики, естественных наук экзамен в середине семестра: экзамен в конце семестра: реферат: выступление: Числовая оценка:	**lblTypes** **lblGrades** **lblStudentType** **lblMidterm** **lblFinalExam** **lblResearch** **lblPresentation** **lblFinalGrade**
TextBox	окно редактирования (экзамен в середине семестра) окно редактирования (экзамен в конце семестра) окно редактирования (реферат) окно редактирования (выступление)	**txtMidterm** **txtFinalExam** **txtResearch** **txtPresentation**
GroupBox	Контейнер, объединяющий переключатели в логическую группу	**grpStudentType**
Form	**всё окно, то есть форма**	**DrawGUI**

A. Еще раз о процессе создания GUI

Графический интерфейс пользователя состоит из одного класса. Назовем этот класс **DrawGUI**, и поместим в него объекты, созданные из классов **System.Windows.Forms** (см. табл. 2). Затем в новом начальном классе **GradesGUI** (см. далее **Листинг 7, файл DrawGUI.cs**) создадим один экземпляр **x** класса **DrawGUI**, который и будет выводить нарисованный нами **GUI.** Когда **GUI** будет выведен на дисплей, пользователь сможет взаимодействовать с начальным классом **GradesGUI** при помощи обработчиков событий.

Обработчики событий дают знать начальному классу **GradesGUI**, к какому именно объекту **формы** (**всё окно**) обращался пользователь. С их помощью будем знать, что пользователь выбрал переключателем студента факультета математики**;** какое значение он ввел в окно оценки за экзамен в середине семестра или когда он нажал кнопку "Рассчитать оценку". Без обработки событий можно только вывести **GUI** (сама **форма**) и ничего более.

Далее дадим **пояснение к основным составляющим класса DrawGUI** в порядке их следования в Листинге **DrawGUI.cs** (см. также рис. 1).

В строке **009** создан объект **btnCalculate** класса **Button** – кнопка «Расчет оценки»

В строке **010** создан объект **btnReset** класса **Button** – кнопка «Сброс». При ее нажатии **форма** очищается от результатов расчет предыдущей оценки и принимает вид «по умолчанию», который заложен в классе **DrawGUI.**

В строках **012…013** созданы объекты **radEnglish, radMath, radScience** класса **RadioButton** – это переключатели, посредством которых будет выбираться факультет, соответственно, Английского языка, Математики или Естественных наук на котором обучается студент, для которого необходимо рассчитать оценку.

В строках **016…023** созданы объекты **lblTypes, lblGrades, lblStudentType, lblMidterm, lblFinalExam, lblResearch, lblPresentation, lblFinalGrade** класса **Label** – это ярлыки (надписи) , соответственно: «Факультет», «Оценки за», «английского языка, математики, естественных наук», «экзамен в середине семестра:», «экзамен в конце семестра:», «реферат:», «выступление:», «Числовая оценка:» .

В строках **025…029** созданы объекты **txtMidterm, txtFinalExam, txtResearch, txtPresentation** класса **TextBox** – это окна редактирования в которые пользователь будет вводить значения промежуточных оценок студента.

В строке **030** создан объект **grpStudentType** класса **GroupBox** – это рамка-контейнер, которая будет охватывать три переключателя **radEnglish, radMath, radScience,** то есть объединять их в логическую группу. Результатом этого является возможность активизировать только один из переключателей, так как предполагается, что студент может заниматься только на одном факультете.

В строках **035…239** реализован Обработчик событий

B. Что такое событие?

Созданный класс **DrawGUI** отвечает за вывод **Г**рафического **И**нтерфейса **П**ользователя (**GUI**) для Проекта расчета оценок. **GUI** представляет собой практически пустую оболочку. Сам он не реагирует на те события, которые инициирует пользователь при взаимодействии с программой. И как результат, программа не в состоянии рассчитать оценку.

Что конкретно означает термин событие?

Событие — это нечто, инициируемое действиями пользователя при работе с **GUI**.

Например, созданный **GUI** содержит форму, рамку группы, две кнопки, три переключателя, четыре окна редактирования и несколько ярлыков (см. рис. 1). Когда пользователь запускает программу на выполнение, выводится **GUI**, пользователь мышкой выбирает один из трех переключателей, то есть факультет. В зависимости от выбранного факультета выводятся два, три или четыре окна редактирования (см. Техническое задание). Затем пользователь с помощью клавиатуры вводит в появившиеся окна редактирования значения соответствующих промежуточных оценок, затем щелкает на кнопке "Рассчитать оценку", что должно привести к выводу итоговой оценки в числовом и буквенном виде. После этого, если пользователь пожелает рассчитать другую оценку, ему нужно будет щелкнуть на кнопке "Сброс" и повторить весь процесс еще раз.

Все эти действия не являются событиями, но каждое из них инициирует событие. Можно попытаться представить событие, как круги на воде, вызванные броском камня, то есть действиями пользователя. Событием является не бросок камня, а круги на воде, вызванные этим действием. Каждое взаимодействие пользователя с объектами **GUI** вызывает круги на воде. Программа может реагировать на эти круги на воде, если, написать так называемый обработчик событий. Обработчики событий должны *отслеживать* возникновение событий или кругов на воде.

Генерация события встроена в сами объекты (см. табл. 2). Необходимо создать обработчик, который обнаруживает появление события, когда оно произойдет.

C. Что такое обработчик событий?

Имеется несколько способов реализации обработчика событий:

- обработчики событий могут задаваться в виде методов того же класса, который представляет **GUI**. Это самый простой способ, он реализован в классе **DrawGUI** (см. **Листинг 7, файл DrawGUI.cs** , строки **035…239**);

- обработчики событий могут быть также реализованы в своем собственном классе. Многие программисты на **C#** так и поступают, но этот способ более сложен.

В классе **DrawGUI** создан**:** один обработчик событий-метод для всех трех переключателей**;** А также два обработчики событий - отдельные методы – для кнопок "Вычислить оценку" и "Сброс". Разделение обработчиков событий соответствует принципам модульного программирования.

D. Реализация простого обработчика событий

Реализация алгоритма реагирования на события состоит из **четырех** шагов, которые заключаются в следующем.

1. Объявить объект **EventHandler**.

2. Создать объект класса **EventHandler**, который определяет имя обработчика событий в качестве единственного аргумента конструктора.

3. Зарегистрировать объект класса **EventHandler** для того объекта **GUI**, чьи события он должен обрабатывать.

4. Запрограммировать обработчик событий **EventHandler**, используя объекты издателя и **EventArgs** для реализации в методе задуманных решений.

Далее **рассматриваем эти шаги для трех обработчикой событий**, реализованных в классе **DrawGUI** (см. **Листинг 7, файл DrawGUI.cs**).

В строках **040…042** реализован **1-й шаг** – объявлены три объекта **RadioButtonHandler, CalculateButtonHandler, ResetButtonHandler** класса **EventHandler.** Для примера приводится одна строка **040**.

040: **EventHandler RadioButtonHandler**; .

Существует реальный класс **EventHandler**. Сама **форма** и помещенные в нее объекты (**см. табл. 2**) вызывают или инициируют события, а появление этих событий осуществляется в форме объектов.

В строках **047…049** реализован **2-й шаг** (см. конструктор **DrawGUI**() – строки **044…153)** – созданы экземпляры каждого объекта **RadioButtonHandler, CalculateButtonHandler, ResetButtonHandler** с вызовом конструктора класса **EventHandler** и передачей единственного аргумента, ссылающегося на имя метода класса **DrawGUI**, который будет обрабатывать события, инициируемые данным объектом на форме. Это следующие аргументы, соответственно: **RadioButton_Click, btnCalculate_Click, btnReset_Click.** Для примера приводится одна строка **047**.

047: **RadioButtonHandler = new EventHandler(RadioButton_Click);**

Имя в скобках – это имя метода **RadioButton_Click** класса **DrawGUI,** который будет обрабатывать события переключателей **radEnglish, radMath, radScience**.

В строках **056…060** реализован **3-й шаг** (см. конструктор **DrawGUI**() – строки **044…153)** – он предназначен для регистрации объекта класса **EventHandler** у объекта **GUI**, события которого мы хотим обрабатывать. Для примера приводится одна строка **056**.

056: **radEnglish.Click** += **RadioButtonHandler**;

При регистрации объекта **EventHandler** указываются имя объекта-переключателя **radEnglish**, затем точка и идентификатор обрабатываемого события **Click**, после чего указывается соответствующий обработчик событий, то есть метод **RadioButtonHandler**. Другими словами, при возникновении события **Click** для переключателя **radEnglish** необходимо вызвать метод **RadioButtonHandler**.

Далее, перед выполнением 4-го шага , в конструкторе **DrawGUI**() (строки **044…153),** выполняется следующее:

В строке **063** –

063: **grpStudentType.Location = new Point(20,30);**

с помощью объекта **Point** заданы координаты расположения верхнего правого угла объекта **grpStudentType** – рамка-контейнер – внутри которой будут находиться три переключателя класса **RadioButton.** Координаты **x = 20, y = 30** (в пикселях) измерены относительно верхнего правого угла всей **формы**. **Location** – это имя свойства с помощью которого точно указывается расположение объектов. Свойству **Location** присваивается объект **Point**.

В строке **065** –

065: **grpStudentType.Size = new Size(105,160);**

создан объект **Size** из пространства имен **System.Drawing** и определено свойство **Size** объекта **grpStudentType.** Ширина и высота объекта **grpStudentType** – рамка-контейнер – заданы в пикселях – **105 и 160.**

В строках **067…075** заданы положение и размер подписей к переключателям, причем, например, в строке **069** установлено свойство **Text** объекта **radEnglish**, которое выводит подпись к переключателю **radEnglish.**

В строках **077…099** заданы аналогично положение и размер ярлыков (объект **Label**). Таким же образом выводится ярлык (то есть текст).

В строках **102…109** заданы положение и размер окон редактирования. В этих окнах пользователь приводит промежуточные оценки студента.

В строках **112…117** заданы положение и размер объектов-кнопок **btnCalculate** и **btnReset**, а также выводится их название: «Рассчитать оценку» и «Сброс».

В строке **120** свойство **Checked** устанавливает переключатель **radEnglish** в положение «Вкл.». Это эквивалентно щелчку мышкой по этому переключателю.

В строках **126…128** три переключателя **radEnglish, radMath, radScience** помещаются в рамку-контейнер **grpStudentType.** Это позволяет устанавливать только один перключатель, так как студент по соглашению может заниматься только на одном факультете.

В строках **133…147** все элементы **GUI** помещаются в контейнер верхнего уровня, каковым является сама **форма**.

В строках **149…150** задаются размеры (в пикселях) и заголовок **формы.**

В строке **151** запрещено изменение размеров **формы.**

Переходим к анализу **4-го шага** реализации алгоритма реагирования на события, то есть рассматриваем:

❖ программный код обработчика событий-метода **RadioButton_Click()** – щелчки мышкой для трех переключателей (выбор одного из трех факультетов) – (строки **158…185**);

❖ программный код обработчика событий-метода **btnCalculate_Click()** – Расчет оценки – (строки **189…219**);

❖ программный код обработчика событий-метода **btnReset_Click()** – Сброс – (строки **223…239**).

4-й шаг. Обработчик событий-метод RadioButton_Click()

Рассмотрим операторы метода **RadioButton_Click()**.

В этом методе проверяется свойство **Text** аргумента **sender** типа **Object** с помощью серии операторов **if** для выяснения, какой из трех переключателей инициировал событие:

```
158:   private void RadioButton_Click(Object sender,  System.EventArgs e)
159:   {
160:    if  ((sender as RadioButton).Text =  "Английский")
161:    {
```

В зависимости от выбранного факультета устанавливаем свойства **Visible** объектов класса **TextBox** нашего **GUI**, делая их тем самым видимыми “**true**” или *не*видимыми “**false**” (это окна редактирования):

```
163:    txtMidterm.Visible = true;
164:    txtFinalExam.Visible = true;
165:    txtResearch.Visible = true;
166:    txtPresentation.Visible = true;
```

И, наконец, последним оператором блока **if** устанавливается надпись **lblStudentType** объекта типа Label нашего **GUI**. Эта надпись будет использоваться обработчиком событий кнопки "Рассчитать оценку" для определения (строки **191, 202** или **211**) – экземпляр какого именно класса **Student** необходимо создать для правильного вычисления оценки:

```
168:    lblStudentType.Text =  "АНГЛИЙСКИЙ ";
```

Такая же процедура повторяется для проверки других факультетов.

4-й шаг. Обработчик событий-метод btnCalculate_Click()

В отличие от обработчика событий переключателя **RadioButton**, который должен обрабатывать события всех трех переключателей, обработчик событий кнопки "Рассчитать оценку" зарегистрирован для обработки только одного объекта типа **Button** — кнопки "Рассчитать оценку", следовательно, не требуется определять, которая из кнопок была нажата. Однако необходимо знать, какой переключатель выбран к тому моменту, когда пользователь щелкнет на кнопке "Рассчитать оценку", и это осуществляется с помощью операторов **if**. Для этого используется надпись объекта ярлыка **lblStudentType** в обработчике событий переключателя. В обработчике событий переключателя есть операторы для запоминания в надписи ярлыка **lblStudentType** (строки **168, 176, 184**) названия факультета, выбранного переключателем. Для определения того, какой именной объект **Student** нужно создать, проверяется надпись ярлыка, например, "АНГЛИЙСКИЙ" с помощью оператора **if** следующим образом:

```
189:   private void btnCalculate_Click(Object sender,   System.EventArgs e)
190:   {
191:    if   (lblStudentType.Text =   "АНГЛИЙСКИЙ")
192:   {
```

Если условие выполняется, то создается экземпляр класса **EnglishStudent**:

```
EnglishStudent  eStudent = new EnglishStudent();
```

После этого вызывается перегруженный метод **Calculate()** нового объекта **eStudent** класса **EnglishStudent** (см. строки **026…048** в этом классе) и ему передаются в качестве четырех аргументов полученные из окон редактирования нашего **GUI** значения составляющих для расчета оценки. Для преобразования полученных из окон редактирования строковых значений в значения типа **int** применен метод **Parse()** объекта типа **int**. Метод **Calculate()** ожидает значения именно этого типа:

```
194:   eStudent .Calculate(int.Parse(txtMidterm.Text),
                           int.Parse(txtFinalExam.Text),
                           int.Parse(txtResearch.Text),
                           int.Parse(txtPresentation.Text));
```

После этого метод **Calculate()** выводит в поле надписи ярлыка **lblFinalGrade** нашего **GUI** значения свойств **FinalNumericGrade** и **FinalLetterGrade** объекта **EnghshStudent**:

```
196:   lblFinalGrade.Text =   ("Числовая оценка:   " + eStudent. FinalNumericGrade +
                              ". Буквенная оценка:   " + eStudent. FinalLetterGrade);
```

Условные операторы для проверки равенства надписи ярлыка **lblFinalGrade** значениям "МАТЕМАТИКА" и "ЕСТЕСТВЕННЫЕ" (строки **202** и **211**), а также операторы для создания обоих объектов из классов **MathStudent** и **ScienceStudent** (строки **204** и **213**).

4-й шаг. Обработчик событий-метод btnReset_Click()

рассмотрим программный код обработчика событий для кнопки "Сброс". *Назначение этого кода заключается в восстановлении исходного состояния формы*. Для начала установим в свойстве **Checked** переключателя "Английский" значение **true**:

```
223:   private void btnReset_Click(Object sender, System.EventArgs e)
224:   {
227:   radEnglish.Checked = true;
```

Должны быть выведены все четыре окна редактирования, для этого устанавливаем в их свойствах **Visible** значение **true**:

```
228:  txtMidterm.Visible = true;
229:  txtFinalExam.Visible = true;
230:  txtResearch.Visible = true;
231:  txtPresentation.Visible = true;
```

Очистка окон редактирования от введенных пользователем значений осуществляется следующим образом:

```
232:  txtMidterm.Text = "";
233:  txtFinalExam.Text = "";
234:  txtResearch.Text = "";
235:  txtPresentation.Text = ""; .
```

То же самое со свойством **Text** ярлыка **lblFinalGrade**:

```
236:  lblFinalGrade.Text = ""; .
```

Также восстановлено значение "АНГЛИЙСКИЙ" для надписи ярлыка **lblStudentType**, которое было установлено при запуске программы (строка **085**):

```
238:  lblStudentType.Text =  "АНГЛИЙСКИЙ";
```

Листинг 7, файл DrawGUI.cs

```
001:  using System; // Граф-й Интерфейс Пользователя (GUI) для Проекта расчета оценок
002:  using System.Windows.Forms;
003:  using System.Drawing;
004:  // Окно (Window) - элемент GUI - и класс DrawGUI создаются от базового класса Form.
005:  // Объект DrawGUI класса Form – это все ОКНО
006:  // Окно (Window) –  является контейнером для других элементов GUI, а именно:
007:  class DrawGUI : Form //класс DrawGUI - производный; класс Form - базовый
008:  {              // 1) кнопки (Buttons) - расчет оценки; сброс вычисления (очистка)
009:  Button btnCalculate = new Button();
010:  Button btnReset = new Button();
011:
012:  RadioButton radEnglish = new RadioButton(); // 2) перекл-ли (RadioButtons) фак-ов
013:  RadioButton radMath = new RadioButton();
014:  RadioButton radScience = new RadioButton();
015:
016:  Label lblMidterm = new  Label(); // 3) ярлыки (Labels): - экз-н в середине семестра
017:  Label lblFinalExam = new Label();     //         - экз-н в конце семестра
018:  Label lblResearch = new Label ();     //         - реферат
019:  Label lblPresentation = new Label(); //         - выступление
020:  Label lblFinalGrade = new Label ();  //         - числовая оценка
021:  Label lblTypes = new Label();        //         - Факультет
022:  Label lblGrades = new Label();       //         - Оценки за
023:  Label lblStudentType = new Label();  // - англ-го языка, матем-ки, естест-х наук
024:                    // 4) окна редакт-ния (TextBoxes), в них указ-ся соотв-я оценка
025:  TextBox txtMidterm = new TextBox();
026:  TextBox txtFinalExam = new TextBox();
027:  TextBox txtResearch = new TextBox();
028:  TextBox txtPresentation = new TextBox();
029:  // рамка группы (GroupBox) - внутри рамки м-т быть выбран только один из перекл-лей
030:  GroupBox grpStudentType = new GroupBox();
031:
032:  /* Реализация ОБРАБОТЧИКА СОБЫТИЙ.
033:  Реал-ция алгоритма реагирования на события состоит из след-х четырех ШАГОВ:
034:
035:  1-й ШАГ - объявление объектов  класса EventHandler;
036:  класс EventHandler - управление событием ...
037:  Событие - это нечто (круги на воде), иниц-мое действиями польз-ля при работе с GUI.
038:  Соб-е - это не бросок камня, а "круги на воде". Щелчок мышкой - это "броска камня".
```

```
039:  Каждое взаимодействие польз-ля с объектами GUI вызывает событие (круги на воде)*/
040:  EventHandler RadioButtonHandler;
041:  EventHandler CalculateButtonHandler;
042:  EventHandler ResetButtonHandler;
043:
044:  public DrawGUI() // конструктор производного класса DrawGUI, создает ВСЁ ОКНО
045:  {/* 2-й ШАГ -  создание объектов (экземпляров) класса EventHandler. В круглых ск-х
046:  () арг-ты - это имена м-дов класса EventHandler, кот-е будут обраб-вать события */
047:  RadioButtonHandler = new EventHandler(RadioButton_Click);
048:  CalculateButtonHandler = new EventHandler(btnCalculate_Click);
049:  ResetButtonHandler = new EventHandler(btnReset_Click);
050:   /* 3-й ШАГ - регистрация объекта класса EventHandler (z.B., RadioButtonHandler)
051:   у объекта GUI (z.B., radEnglish), события которого необходимо обработать.
052:   Click - это идентиф-ор обраб-го события. М-д RadioButtonHandler – обраб-к события
053:  Вызов м-да RadioButtonHandler для обработки события Click для перкл-ля radEnglish.
054:  В программе обработчик событий только один - его имя Click.
055:  Др-ми словами, на 3-м ШАГЕ рег-ся обр-к соб-й Click для кажд-го из 5-ти объектов*/
056:  radEnglish.Click += RadioButtonHandler;
057:  radMath.Click += RadioButtonHandler;
058:  radScience.Click += RadioButtonHandler;
059:  btnCalculate.Click += CalculateButtonHandler;
060:  btnReset.Click += ResetButtonHandler;
061:
062:  // положение и размер объекта grpStudentType: "рамка группы - GroupBox".
063:  grpStudentType.Location = new Point(20,30);
064:  // Эта рамка охватывает три переключателя -  RadioButton
065:  grpStudentType.Size = new Size(105,160);
066:
067:  radEnglish.Location = new Point(3,10); // положение и размер "Text".
068:  radEnglish.Size = new Size (85,40) ;   // Это подписи трех перекл-й RadioButton,
069:  radEnglish.Text = "Английский";  // они распол-ны внутри рамки группы - GroupBox
070:  radMath.Location = new Point(3,60);
071:  radMath.Size = new Size(88,40);
072:  radMath.Text = "Математики";
073:  radScience.Location = new Point(3,110);
074:  radScience.Size = new Size(98,40);
075:  radScience.Text = "Естественный";
076:
077:  lblTypes.Location = new Point(35, 0); // полож-е и размер объекта "Label" (Text).
078:  lblTypes.Size = new Size(100,20);
079:  lblTypes.Text = "Факультет:";
080:  lblGrades.Location = new Point(150,0);
081:  lblGrades.Size = new Size(120,20);
082:  lblGrades.Text = "Оценки студента за:";
083:  lblStudentType.Location = new Point(300,0);
084:  lblStudentType.Size = new Size(100,20);
085:  lblStudentType.Text = "АНГЛИЙСКИЙ";
086:  lblMidterm.Location = new Point(150,50);
087:  lblMidterm.Size = new Size (118,40);
088:  lblMidterm.Text = "1) экзамен в середине семестра --";
089:  lblFinalExam.Location = new Point(150,100);
090:  lblFinalExam.Size = new Size(110,40);
091:  lblFinalExam.Text = "2) заключительный экзамен --";
092:  lblResearch.Location = new Point(150,150);
093:  lblResearch.Size = new Size (100,40) ;
094:  lblResearch.Text = "3) реферат --";
095:  lblPresentation.Location = new Point(150,200);
096:  lblPresentation.Size = new Size (100,40);
097:  lblPresentation.Text = "4) выступление --";
098:  lblFinalGrade.Location = new Point(225,250);
099:  lblFinalGrade.Size = new Size (250,40);
100:
101:  // положение и размер объекта - окна редактирования (TextBoxes)
```

```
102:  txtMidterm.Location = new Point(300,50);
103:  txtMidterm.Size = new Size(50,70);
104:  txtFinalExam.Location = new Point(300,100);
105:  txtFinalExam.Size = new Size(50,70);
106:  txtResearch.Location = new Point(300,150);
107:  txtResearch.Size = new Size (50,70);
108:  txtPresentation.Location = new Point(300,200);
109:  txtPresentation.Size = new Size(50,70);
110:
111:  // положение и размер "Text" на кнопках Button (объекты btnCalculate и btnReset)
112:  btnCalculate.Location = new Point(50,250);
113:  btnCalculate.Size = new Size(160, 40 );
114:  btnCalculate.Text = "Рассчитать оценку";
115:  btnReset.Location = new Point(50,300);
116:  btnReset.Size = new Size (160,40);
117:  btnReset.Text = "Сброс";
118:
119:  //св-во Checked уст-т перек-ль radEnglish в пол-е "Вкл." - это анал-но щелчку мышкой
120:  radEnglish.Checked = true;
121:
122:  /* св-во Controls - это массив объектов: кнопки (Buttons), ярлыки (Labels), перекл-
123:  ли (RadioButtons) и др. С помощью м-да Add добавляются в массив три переключателя -
124:  RadioButton. При этом рамке группы - GroupBox указывается, что она будет служить
125:  "контейнером" для них */
126:  grpStudentType.Controls.Add (radEnglish);
127:  grpStudentType.Controls.Add (radMath);
128:  grpStudentType.Controls.Add (radScience);
129:
130:   /* С помощью м-да Add добавляются в массив объекты: окна редактирования,
131:  ярлыки, кнопки и рамка группы   При этом форме (всему окну) указывается,
132:  что она будет служить "контейнером" для них */
133:  Controls.Add(txtMidterm);
134:  Controls.Add(txtFinalExam);
135:  Controls.Add(txtResearch);
136:  Controls.Add(txtPresentation);
137:  Controls.Add(lblTypes);
138:  Controls.Add(lblMidterm);
139:  Controls.Add(lblFinalExam);
140:  Controls.Add(lblResearch);
141:  Controls.Add(lblPresentation);
142:  Controls.Add(lblStudentType);
143:  Controls.Add(lblGrades);
144:  Controls.Add(lblFinalGrade);
145:  Controls.Add(btnCalculate);
146:  Controls.Add(btnReset);
147:  Controls.Add(grpStudentType);
148:
149:  Text = "Расчет оценки";                    // Имя (заголовок) формы (всё Окно)
150:  Size = new Size(500,450);                  // Размеры формы
151:  // Запрет пользователю изменять размеры формы мышкой
152:  FormBorderStyle = FormBorderStyle.FixedSingle;
153:  }
154:  /* 4-й ШАГ - прогр-й код обработчика первых трех событий - щелчки для всех трех
155:  переключателей. Обработчик оформлен в виде метода RadioButton_Click:
156:  sender - это 1-й арг-т типа Object, sender - это объект, иниц-щий событие;
157:  e - это 2-й арг-т типа System.EventArgs*/
158:  private void RadioButton_Click(Object sender, System.EventArgs e)
159:  {
160:   if ((sender as RadioButton).Text == "Английский")
161:  {// Св-во Visible объектов класса TextBox,
162:  // имея значение "true", делает видимыми  ОКНА РЕДАКТИРОВАНИЯ
163:  txtMidterm.Visible = true;
164:  txtFinalExam.Visible = true;
165:  txtResearch.Visible = true;
```

```
166:  txtPresentation.Visible = true;
167:  // установка ярлыка lblStudentType в состояние  "АНГЛИЙСКИЙ"
168:  lblStudentType.Text = "АНГЛИЙСКИЙ";
169:   }
170:   if ((sender as RadioButton).Text == "Математики")
171:  {
172:  txtMidterm.Visible = true;
173:  txtFinalExam.Visible = true;
174:  txtResearch.Visible = false;
175:  txtPresentation.Visible = false;
176:  lblStudentType.Text = "МАТЕМАТИКА";
177:  }
178:  if ((sender as RadioButton).Text == "Естественный")
179:  {
180:  txtMidterm.Visible = true;
181:  txtFinalExam.Visible = true;
182:  txtResearch.Visible = true;
183:  txtPresentation.Visible = false;
184:  lblStudentType.Text = "ЕСТЕСТВЕННЫЕ";
185:  }
186:  }
187:   // 4-й ШАГ - программный код обработчика события - Расчет оценки,
188:   // который оформлен в виде метода btnCalculate_Click
189:  private void btnCalculate_Click(Object sender, System.EventArgs e)
190:  {
191:  if (lblStudentType.Text == "АНГЛИЙСКИЙ")
192:  {// метод Parse() объекта int преобразует строковое значение в целочисленное
193:  EnglishStudent eStudent = new EnglishStudent();
194:  eStudent.Calculate (int.Parse(txtMidterm.Text),
195:                      int.Parse(txtFinalExam.Text),
196:                      int.Parse(txtResearch.Text),
197:                      int.Parse(txtPresentation.Text));
198:  lblFinalGrade.Text = ("Числовая оценка: "  + eStudent.FinalNumericGrade +
199:                      ". Буквенная оценка: " + eStudent.FinalLetterGrade);
200:  }
201:
202:  if (lblStudentType.Text == "МАТЕМАТИКА")
203:  {
204:  MathStudent mStudent = new MathStudent();
205:  mStudent.Calculate(int.Parse(txtMidterm.Text),
206:                     int.Parse(txtFinalExam.Text));
207:  lblFinalGrade.Text = ("Числовая оценка: "  + mStudent.FinalNumericGrade +
208:                      ". Буквенная оценка: " + mStudent.FinalLetterGrade);
209:  }
210:
211:  if (lblStudentType.Text == "ЕСТЕСТВЕННЫЕ")
212:  {
213:  ScienceStudent sStudent = new ScienceStudent();
214:  sStudent.Calculate(int.Parse(txtMidterm.Text),
215:                     int.Parse(txtFinalExam.Text),
216:                     int.Parse (txtResearch.Text));
217:  lblFinalGrade.Text = ("Числовая сценка: "  + sStudent.FinalNumericGrade +
218:                      ". Буквенная оценка: " + sStudent.FinalLetterGrade);
219:  }
220:  }
221:   // 4-й ШАГ - программный код обработчика события Сброс - очистка,
222:      //который оформлен в виде метода btnReset_Click
223:  private void btnReset_Click(Object sender, System.EventArgs e)
224:  {/* св-во Checked устанавл-т переключатель radEnglish в полож-е "Вкл." -
225:  это аналог-но щелчку мышкой. Св-во Visible объектов класса TextBox,
226:  имея значение "true", делает видимыми  окна редакт-я*/
227:  radEnglish.Checked = true;
228:  txtMidterm.Visible = true;
229:  txtFinalExam.Visible = true;
```

```
230:  txtResearch.Visible = true;
231:  txtPresentation.Visible = true;
232:  txtMidterm.Text = "";    // Окна редактирования очищаются
233:  txtFinalExam.Text = "";
234:  txtResearch.Text = "";
235:  txtPresentation.Text = "";
236:  lblFinalGrade.Text = "";
237:  // установка ярлыка lblStudentType в исходное
238:  lblStudentType.Text = "АНГЛИЙСКИЙ"; состояние
239:  }
240:  }
```

2.4.2.2. Начальный класс GradesGUI. Листинг 8, файл GradesGUI.cs

Здесь содержится метод **Main()**, являющийся диспетчером выполнения всей программы. Создается объект **x** класса **DrawGUI** (строка **009** в **Листинге 8, файл GradesGUI.cs**). Далее посредством вызова метода **Run()** класса **Application**, **создается выводящий форму** объект. Обращение к м-ду **Run()** сообщает C#, что **форма** должна оставаться видимой, до ее закрытия пользователем.

Листинг 8, файл GradesGUI.cs

```
001:  // Проект Расчет оценок в варианте GUI (Графический Интерфейс Пользователя)
002:  using System;
003:  using System.Windows.Forms;
004:
005:  class GradesGUI
006:  {
007:  public static void Main(string[] args)
008:  { // Создание экз-ра x класса DrawGUI
009:  DrawGUI x = new DrawGUI();
010:  /* посредством вызова метода Run() класса Application, создается выводящий
011:  форму объект. Обращение к м-ду Run() сообщает C#, что форма должна оставаться
012:  видимой, до ее закрытия пользователем*/
013:  Application.Run(x);
014:  }
015:  }
```

2.4.3. Создание выполнимого файла, реализующего проект Расчет оценок в варианте GUI

Возможны два способа создания выполнимого файла.

1-й способ. С помощью команды,
запускаемой из командной строки **Visual Studio .NET 2003 Command Promt** ,

```
csc GradesGUI.cs DrawGUI.cs EnglishStudent.cs MathStudent.cs ScienceStudent.cs DisplayGrade.cs Student.cs
```

создаем выполнимый файл **GradesGUI.exe.** Его запуск обеспечивает выполнение Проекта Расчет оценок в варианте **GUI**.

2-й способ. Другая возможность создания **оконного** варианта проекта**:** запустить среду **MS VS .NET;** реализовать **WindowsApplication;** скопировать содержимое файлов **DrawGUI.cs** (1-й), **GradesGUI.cs, EnglishStudent.cs, MathStudent.cs, ScienceStudent.cs, Student.cs** в окно среды**;** нажать на клавишу **F5** (естественно, что при копировании содержимого файлов разумно удалить строки – using System; using System.Windows.Forms; – из всех файлов кроме файла **DrawGUI.cs**).

2.4.4. Взаимодействие классов проекта Расчет оценок в варианте GUI

Оконный вариант проекта **Расчет оценок** реализуется в рассмотренных выше классах:

- ✓ **DrawGUI.cs** – производный класс (от класса **Form**) (см. разд. **2.4.2.1**);
- ✓ **GradesGUI.cs** – начальный класс (см. разд. **2.4.2.2**);

- ✓ **EnglishStudent.cs** – производный класс (см. разд. **2.4.1.2**);
- ✓ **MathStudent.cs** – производный класс (см. разд. **2.4.1.3**);
- ✓ **ScienceStudent.cs** – производный класс (см. разд. **2.4.1.4**);
- ✓ **Student.cs** – базовый класс (см. разд. **2.4.1.1**).

После запуска файла **GradesGUI.exe** среда **.NET** обращается к статическому методу **Main()** (класс **GradesGUI**), который (метод) обеспечивает роль диспетчера при выполнении программы.

Создается объект **x** класса **DrawGUI** (строка **009** в **Листинге 8, файл GradesGUI.cs**).

009: **DrawGUI x = new DrawGUI();**

Далее посредством вызова метода **Run()** класса **Application**, создается выводящий **форму** объект.

013: **Application.Run(x);**

Обращение к м-ду **Run()** сообщает **C#**, что **форма** должна оставаться видимой, до ее закрытия пользователем.

Итак, **форма** на экране дисплея. То, что на ней отображено (это состояние по умолчанию) заложено в коде файла **DrawGUI.cs** (см. выше разд. **2.4.2.1**).

Проанализируем наше взаимодействие с **формой** и реакцию программы на это взаимодействие на примере расчета оценки для студента факультета ЕСТЕСТВЕННЫХ наук. Как отмечалось выше программы для Windows управляются **событиями**. Для взаимодействия с программой пользователю предоставляется графический интерфейс (**наша форма – она на экране дисплея**). Причем, управляемые **событиями** программы не требуют от пользователя жестко заданного поведения. Вместо этого программа реагирует на действия пользователя.

Это напоминает посещение городка аттракционов: купив входной билет, посетитель может двигаться куда желает и выбирать развлечения по своему усмотрению.

Итак, наши действия и реакция программы:

1. Щелкаем (**Click**) мышкой по переключателю **radScience** (“Естественный”).
Алгоритм реагирования на это состоит из четырех шагов **первого** обработчика событий:

1-й шаг – объявлен объект класса **EventHandler**

040: **EventHandler RadioButtonHandler;**

2-й шаг – создан объект (экземпляр) **RadioButtonHandler** класса **EventHandler**

047: **RadioButtonHandler = new EventHandler(RadioButton_Click);**

Аргумент **RadioButton_Click –** это имя метода класса **EventHandler**, который будет обрабатывать событие.

3-й шаг – регистрация объекта **RadioButtonHandler** класса **EventHandler** у объекта **GUI** (**radScience**), события которого необходимо обработать. **Click** - это идентификатор обрабатываемого события.

058: **radScience.Click += RadioButtonHandler;**

4-й шаг – Первый обработчик оформлен в виде метода **RadioButton_Click**

158: **private void RadioButton_Click(Object sender, System.EventArgs e)**

/* **sender** - это **1-й** аргумент типа **Object**, **sender** - это объект, инициирующий событие;
e - это **2-й** аргумент типа **System.EventArgs. Далее два if-блока в коде метода не показаны** */

179: {

……

178: **if ((sender as RadioButton).Text == "Естественный")**

179: {

```
180:    txtMidterm.Visible = true;
181:    txtFinalExam.Visible = true;
182:    txtResearch.Visible = true;
183:    txtPresentation.Visible = false;
184:    lblStudentType.Text = "ЕСТЕСТВЕННЫЕ";
185:    }
185:    }
```

В соответствии с выбранным факультетом **"**Естественный**"** устанавливается свойство **Visible** объектов класса **TextBox** нашего **GUI**, так что три окна редактирования видимы, а одно невидимо.

И, наконец, последним оператором блока **if** (строка **184**) устанавливается надпись **"**ЕСТЕСТВЕННЫЕ**"** ярлыка **lblStudentType** объекта типа Label нашего **GUI**. Эта надпись будет использоваться обработчиком событий кнопки "Рассчитать оценку" для определения (строка **211**) – экземпляр какого именно класса **Student** необходимо создать для правильного вычисления оценки.

Итак, разошлись круги по воде в результате щелчка (**Click**) по переключателю **radScience "**Естественный**"** и **форма** изменилась (проделайте это и убедитесь сами).

2. Теперь вводим промежуточные оценки в три окна редактирования (пусть это значения **70**, **80** и **90**).

3. Щелкаем (**Click**) мышкой по кнопке “Рассчитать оценку”. Опять будут выполняться четыре шага, но уже **второго** обработчика событий. Три первых шага не комментируем, но приведем соответствующий код:

```
041:    EventHandler CalculateButtonHandler;                          // 1-й шаг
048:    CalculateButtonHandler = new EventHandler(btnCalculate_Click); // 2-й шаг
059:    btnCalculate.Click += CalculateButtonHandler;                 // 3-й шаг
```

4-й шаг – Второй обработчик оформлен в виде метода **btnCalculate_Click**

```
189:     private void btnCalculate_Click(Object sender, System.EventArgs e)
         /* Далее два if-блока в коде метода не показаны */
190:    {
           ……..
211:    if (lblStudentType.Text == "ЕСТЕСТВЕННЫЕ")
212:    {
213:    ScienceStudent sStudent = new ScienceStudent();
214:    sStudent.Calculate(int.Parse(txtMidterm.Text),
215:                       int.Parse(txtFinalExam.Text),
216:                       int.Parse (txtResearch.Text));
217:    lblFinalGrade.Text = ("Числовая сценка: "  + sStudent.FinalNumericGrade +
218:                ". Буквенная оценка: " + sStudent.FinalLetterGrade);
219:    }
220:    }
```

Так как ярлык **lblStudentType** содержит надпись **"**ЕСТЕСТВЕННЫЕ**",** то выполняется вышеприведенный **if**-блок: 1) создается объект **sStudent** класса **ScienceStudent;** 2) для этого объекта вызывается перегружающий метод **Calculate()** с тремя параметрами в классе **ScienceStudent (**строки **024…044** в **Листинге 4, файл ScienceStudent.cs).** Согласно методу рассчитывается числовая оценка (**78**) и ее буквенный эквивалент (**C**). В процессе выполнения этого метода осуществляется обращение к свойствам класса **Student.**

Итак, программа выполнила свою функциональность, то есть рассчитала оценки и вывела их на **форме** справа от кнопки “Рассчитать оценку”. Теперь можно подготовиться к расчету оценки другого студента.

4. Щелкаем (**Click**) мышкой по кнопке “Сброс”. Опять будут выполняться четыре шага, но уже **третьего** обработчика событий. Три первых шага не комментируем, но приведем соответствующий код:

```
042:    EventHandler ResetButtonHandler;                    // 1-й шаг
049:    ResetButtonHandler = new EventHandler(btnReset_Click); // 2-й шаг
060:    btnReset.Click += ResetButtonHandler;               // 3-й шаг
```

4-й шаг – Третий обработчик оформлен в виде метода **btnReset_Click**

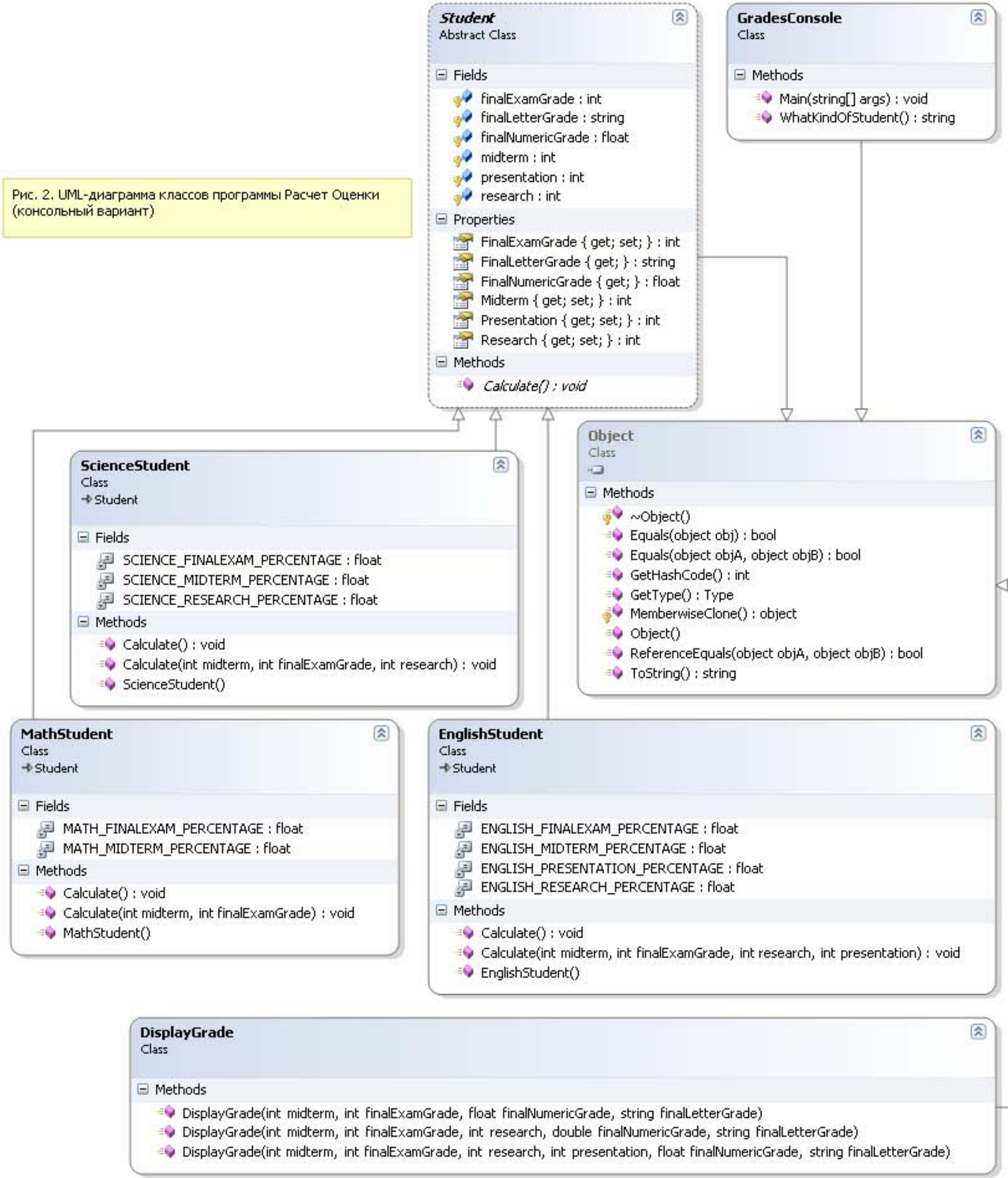
```
223:    private void btnReset_Click(Object sender, System.EventArgs e)
224:    {
.....
239:    }
```

Согласно этому методу восстанавливается первоначальный вид **формы** (по умолчанию), заданный в классе **DrawGUI.** Далее повторяются действия согласно пунктам **1.** , **2.** , **3.** и **4.** для студента другого (или того же) факультета. Если в пункте **3.** будут введены оценки не во все окна, то среда **MS VS .NET** выдаст сообщение "Входная строка имела неверный формат" и для продолжения работы необходимо программу запустить на выполнение заново. Если в пункте **3.** будут введены оценки выходящие за допустимый диапазон **0…100**, то будет выведена информация в окно сообщения и программа прекратит работу.

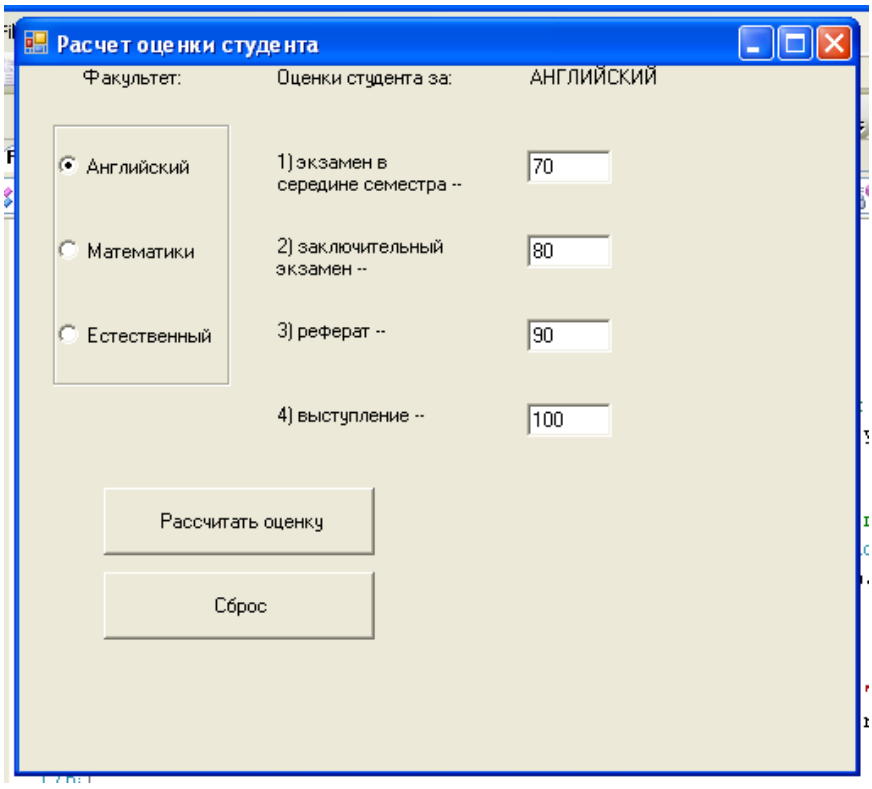
3. Разработка модели классов, описывающей систему

Когда состав классов определен и перечислены их обязанности и способы взаимодействия, можно приступать к разработке общей модели классов, описывающей всю систему целиком. Общая модель иллюстрирует взаимоотношения различных классов в рамках системы. Для моделирования систем используется язык **UML (Unified Modeling Language)**. В настоящее время на рынке представлены различные средства моделирования, использующие **UML** и предоставляющие пользователю удобную среду для создания и поддержки **UML**–моделей, например, пакет визуального моделирования **Rational Rose** или **MS VS .NET 2005**. На рис. 2 показана UML-диаграмма классов проекта Расчет Оценки в консольном варианте. На рис. 3 представлена UML-диаграмма классов этого проекта в оконном варианте – реализован GUI (Graphical User Interface). Моделирование крупных систем фактически невозможно без применения хорошей нотации и средств моделирования.

Рис. 2. UML-диаграмма классов программы Расчет Оценки (консольный вариант)



4.1. Результаты работы программы в варианте GUI - Graphical User Interface – Графический Интерфейс Пользователя



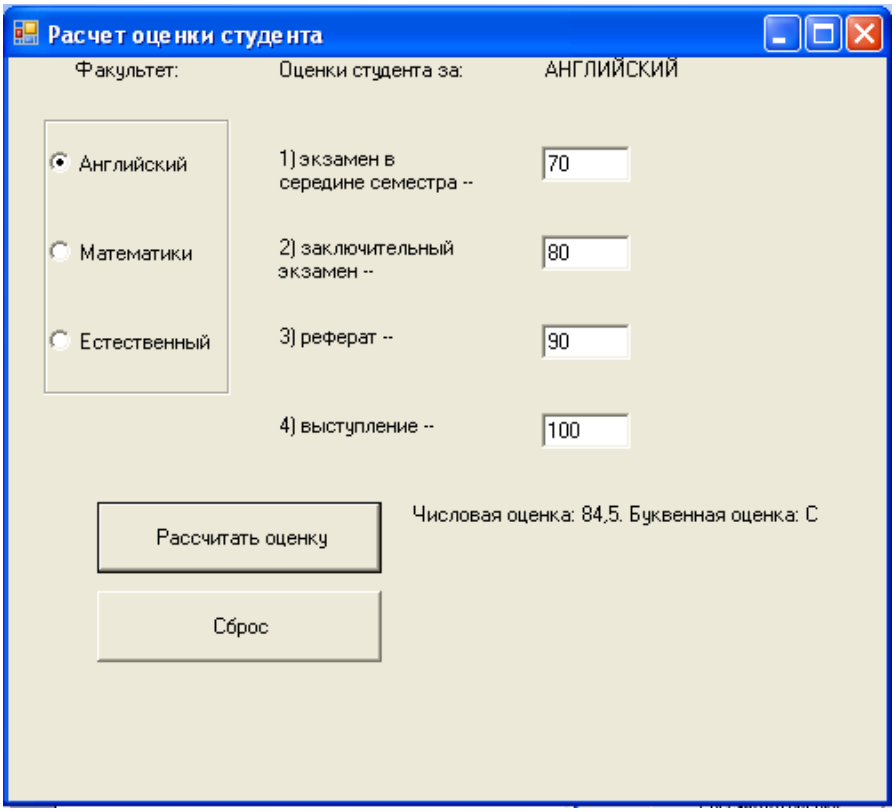
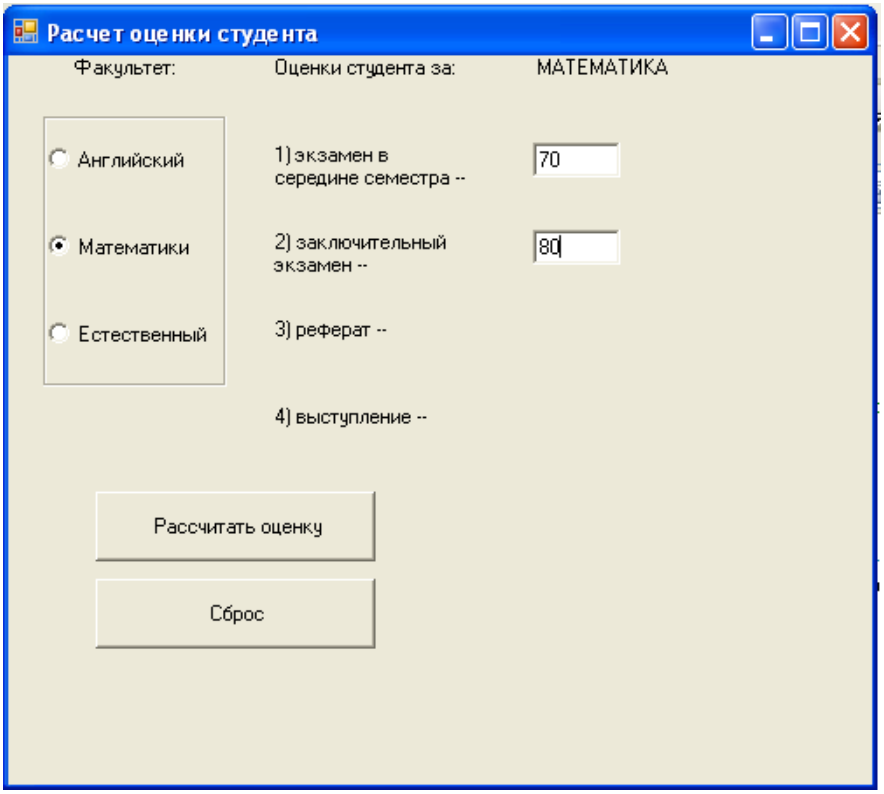


Рис. 4. Вид диалоговых окон при расчете оценки студента факультета английского языка



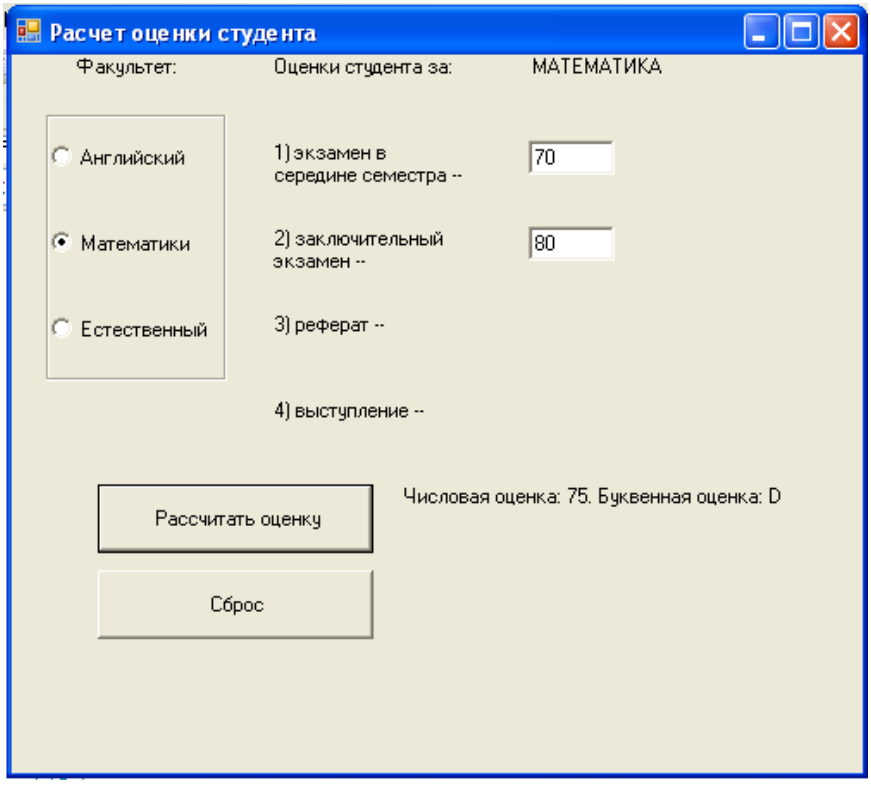
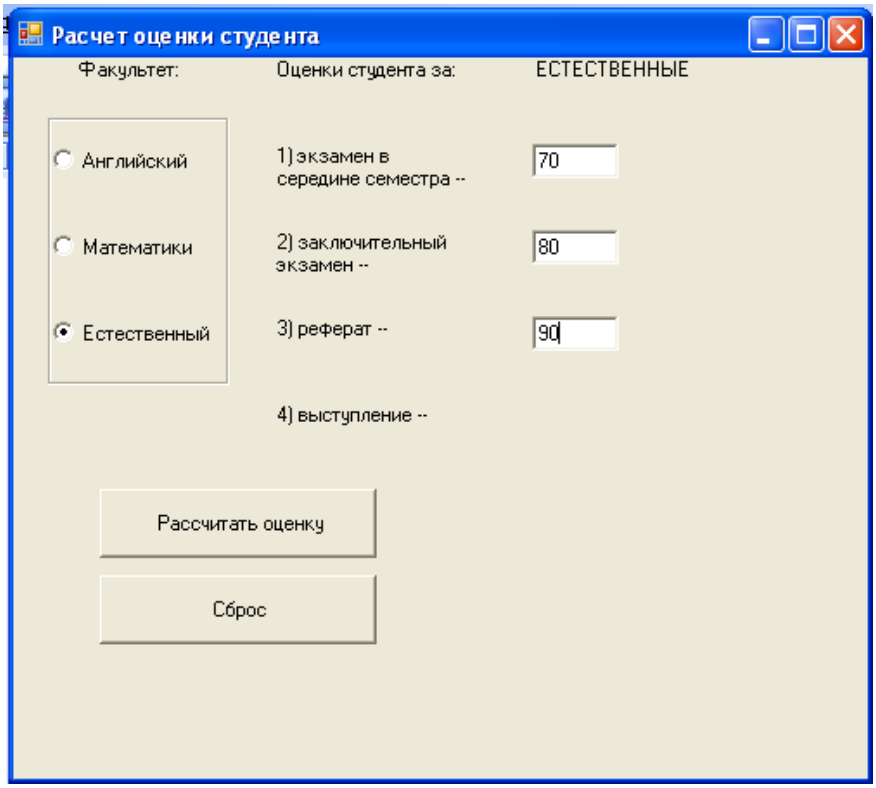


Рис. 5. Вид диалоговых окон при расчете оценки студента факультета математики



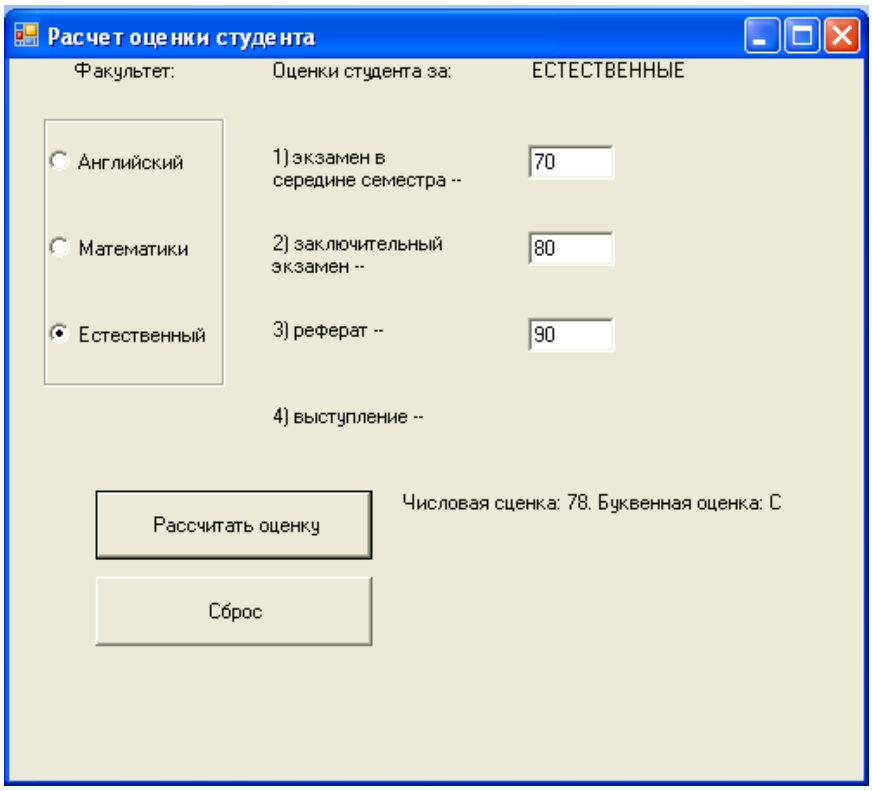


Рис. 6. Вид диалоговых окон при расчете оценки студента факультета естественных наук

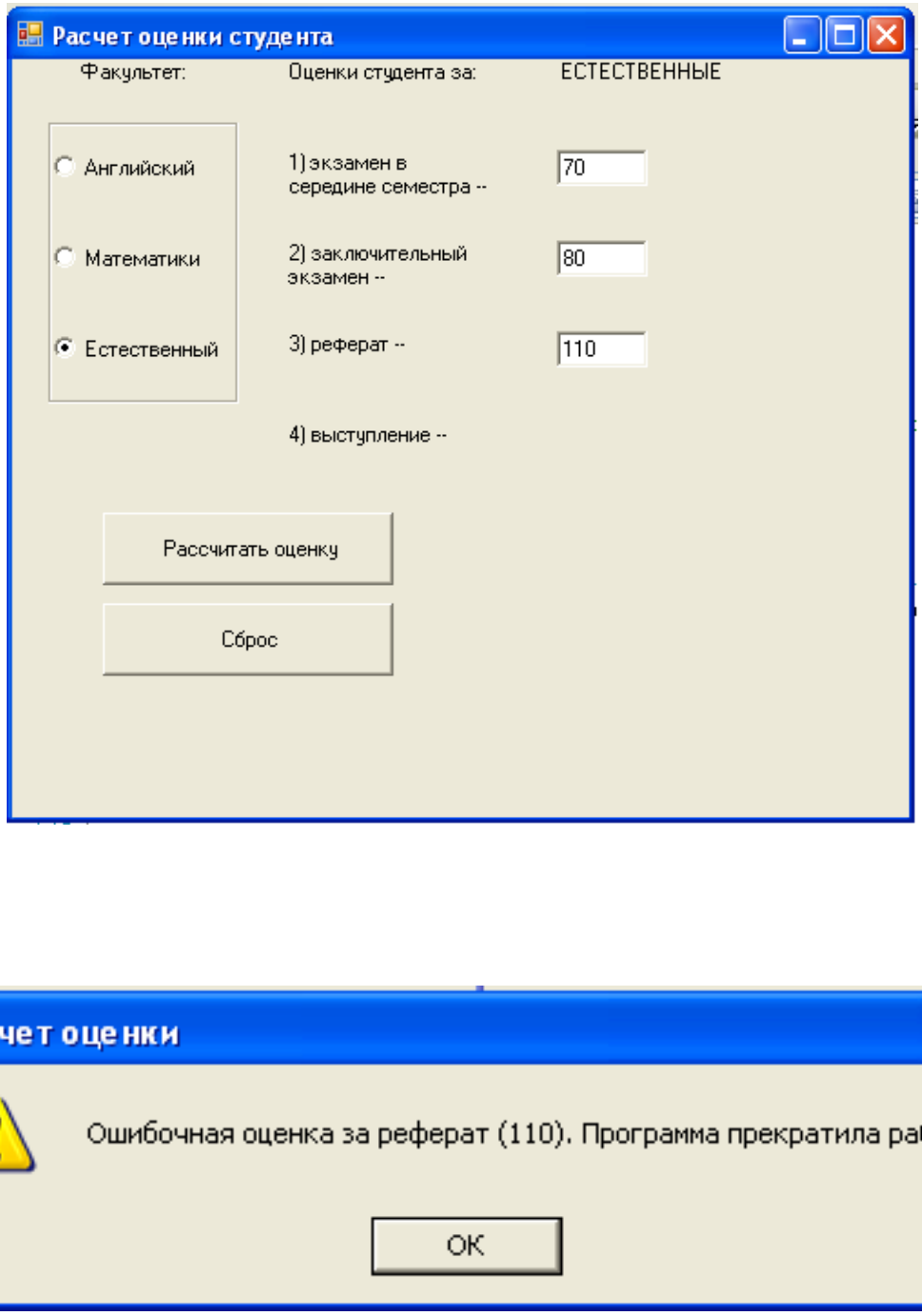


Рис.7. Вид диалоговых окон при расчете оценки
студента факультета естественных наук.
Оценка введена неправильно, так как выходит за диапазон **0…100**

4.2. Результаты работы программы в консольном варианте

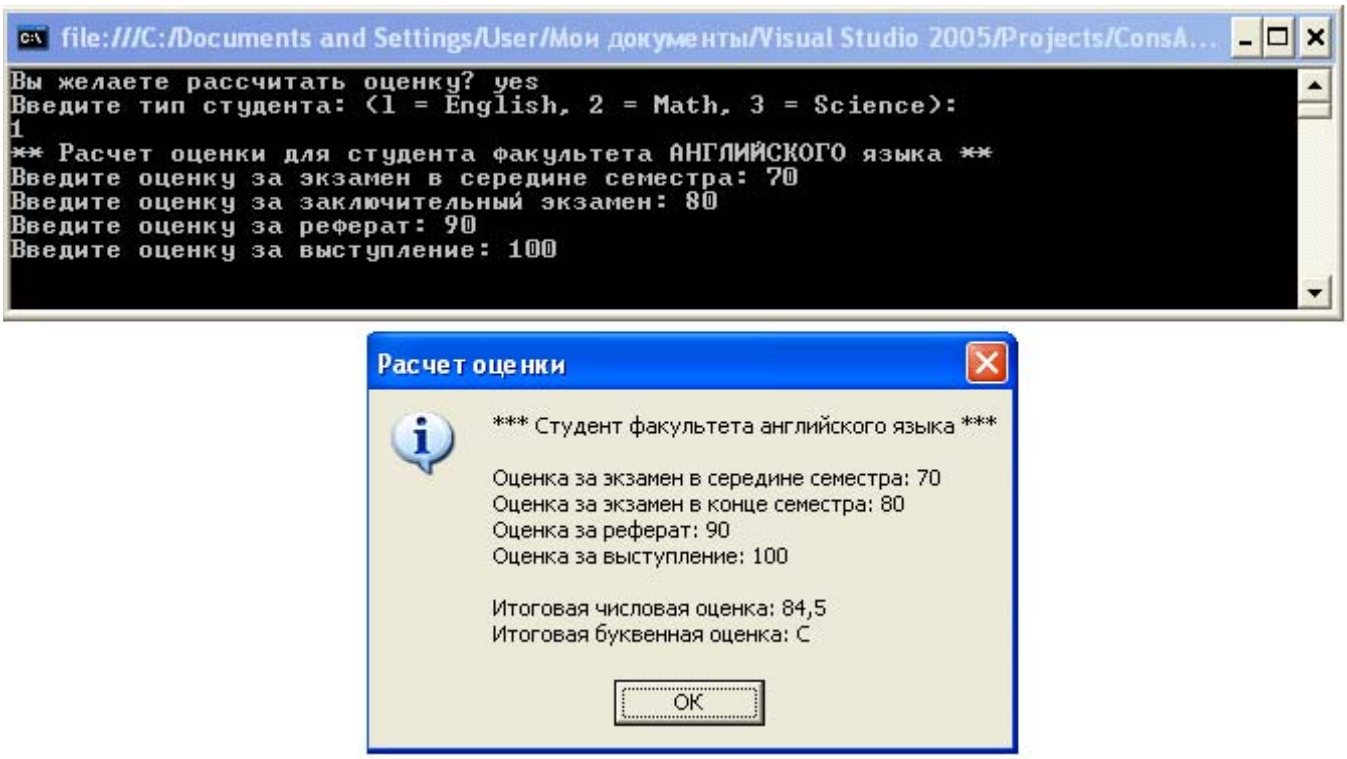


Рис. 8. Вид диалоговых окон при расчете оценки студента факультета английского языка. Консольный вариант

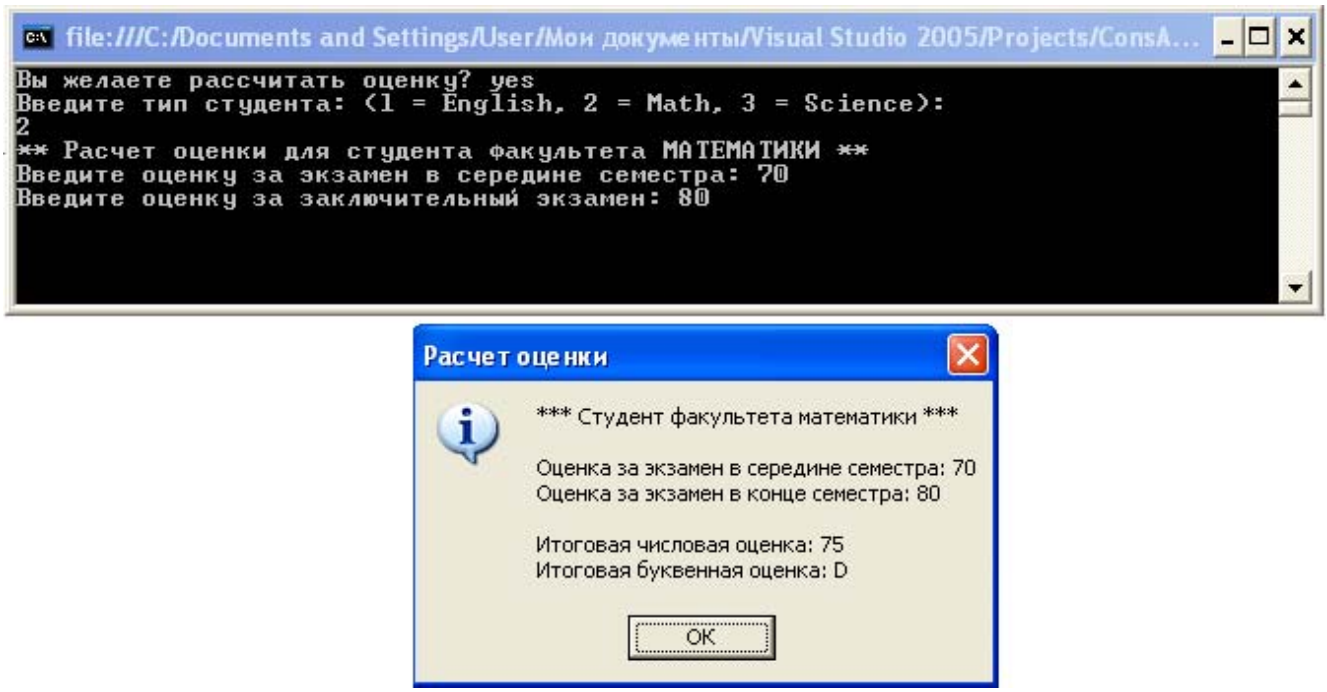


Рис. 9. Вид диалоговых окон при расчете оценки студента факультета математики. Консольный вариант

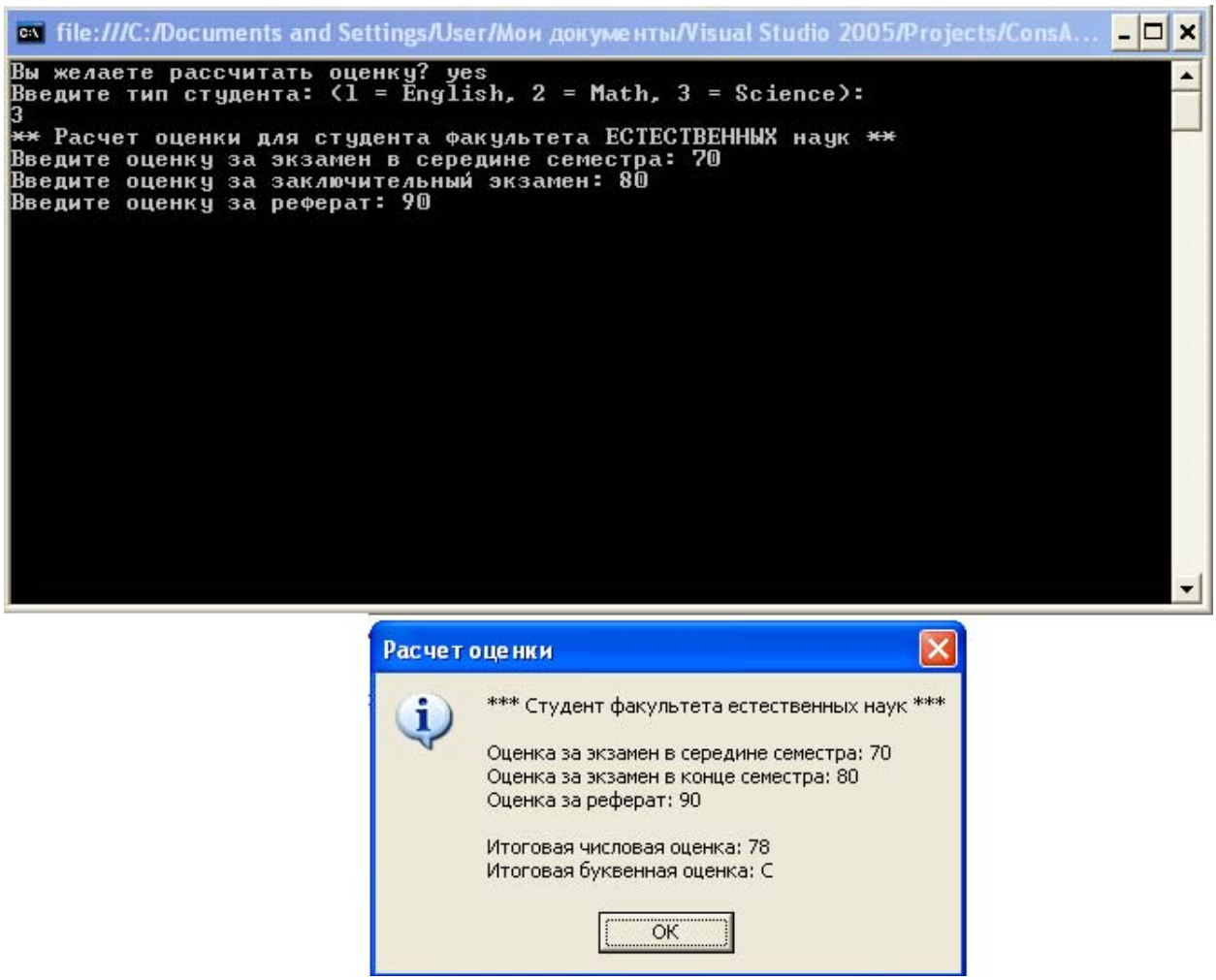


Рис. 10. Вид диалоговых окон при расчете оценки студента факультета естественных наук. Консольный вариант

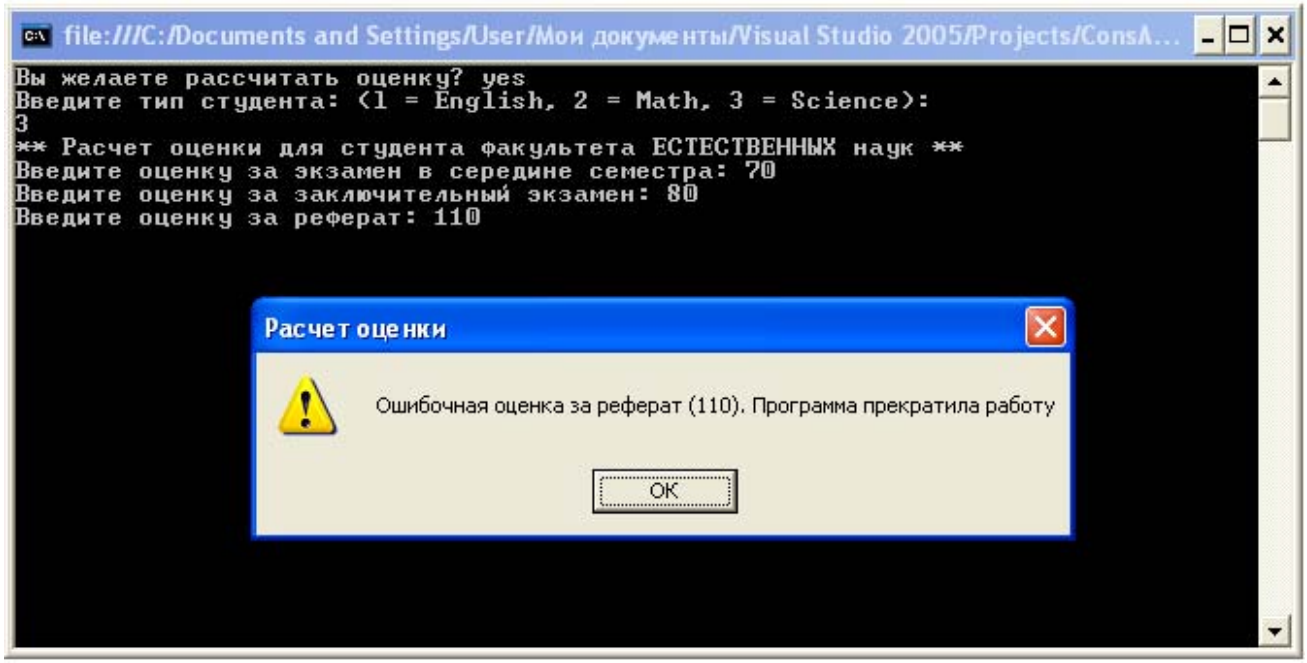


Рис.11. Вид диалоговых окон при расчете оценки
студента факультета естественных наук.
Оценка введена неправильно, так как выходит за диапазон **0…100.**
Консольный вариант

Список литературы

1. К. Нейгел, Б. Ивьен, Дж. Глинн. **C# 2005 для профессионалов** (+CD-ROM) Издательство: Диалектика, 200**6**. Твердый переплет, 1376 с. ISBN 5-8459-1052-8, 0-7645-7534-1 Тираж: 3000 экз. Формат: 70x100/16. http://www.ozon.ru/?context=detail&id=2764114&from=autocategory

2. Р. Лафоре. **Объектно-ориентированное программирование в C++.** Серия: Классика Computer Science .Издательство: Питер , 200**6**.Твердый переплет , 923 с. ISBN: 5-94723-302-9 Тираж: доп. 2000 экз. http://www.biblio-globus.ru/description.aspx?product_no=535640

3. Мэтт Вайсфельд. **Объектно-ориентированный подход: Java, .Net, C++** . Второе издание / Пер. с англ. - М: КУДИЦ-ОБРАЗ, 200**5**. - 336 с. http://www.biblio-globus.ru/description.aspx?product_no=8735431

Что значит освоить объектно-ориентированное программирование? Для этого недостаточно выучить синтаксис языка C#, Java или C++. Нужно разобраться в принципиальных положениях объектного подхода, понять, чем он отличается от других. И предлагаемая книга будет в этом отличным помощником. В ней на конкретных примерах разбираются все основные понятия объектно-ориентированного подхода. Советую прочесть эту книгу [**3**].