Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский государственный университет аэрокосмического приборостроения»

ОСНОВЫ ПРОГРАММИРОВАНИЯ

Методические указания к курсовой работе

Санкт-Петербург

2022 г.

Составители: Богословская Н.В.

В методических указаниях рассмотрены цели и задачи курсового проектирования по дисциплине «Основы программирования». Описаны структура и требования к оформлению пояснительной записки. Приведены варианты индивидуальных заданий. Рассмотрены следующие вопросы:
—описание целей и постановка задач курсовой работы;
—разработка архитектуры библиотеки классов;
—использование основных отношений между классами: наследование, реализация, ассоциация, композиция, агрегация;
—проектирование абстрактных классов и интерфейсов.
Методические указания предназначены для студентов, обучающихся по направлению 09.03.02 «Информационные системы и технологии».
Методические указания подготовлены кафедрой 42 информационных систем и технологий.

Цель курсового проектирования как учебной дисциплины: использование полученных знаний и навыков объектно-ориентированного программирования для разработки пользовательской библиотеки классов, предназначенной для программирования приложений пользователя в конкретной предметной области.

Библиотека должна учитывать особенности области приложения и располагать необходимой функциональностью для построения интерфейса пользователя, структурирования, записи, чтения данных в различных форматах, обработки, вычисления и визуализации данных.

Курсовая работа позволяет:

- систематизировать теоретические знания и закрепить полученные практические умения по дисциплине «Основы программирования» в соответствии с требованиями к уровню подготовки по направлению 09.03.02 «Информационные системы и технологии»;
- сформировать умения работы с учебной литературой и иными информационными источниками;
- развить профессиональную письменную и устную речь;
- развить системное мышление, творческую инициативу, самостоятельность, организованность и ответственность за принимаемые решения;
- сформировать навыки планомерной регулярной работы над решением поставленных задач.

1. Этапы выполнения курсовой работы

1.1 Описание целей и постановка задачи

Результатом выполнения данного пункта является подробное описание функциональности разрабатываемой библиотеки классов и области ее применения.

Необходимо в свободной форме представить архитектуру библиотеки, структуру разрабатываемых классов: какие данные могут быть описаны с помощью этих классов, какими методами обработки классы должны обладать. Также необходимо перечислить категорию информационных систем, прикладных программных решений или интерфейсов пользователей, в которых разработанные классы могут быть использованы как готовые компоненты.

Обязательно структурировать текст и расписать все требования по пунктам. Рекомендуемый объем: не менее одной страницы А4.

Пример очень краткого описания:

Требуется разработать библиотеку классов для эффективного программирования векторных графических примитивов: точки, линии, треугольника, прямоугольник, квадрата, многоугольника.

Объекты классов должны хранить следующую информацию:

- *координаты фигуры,*
- *размеры фигуры,*
- *толщину контура обводки,*
- *цвет заливки.*

Объекты классов должны иметь методы для:

- *изменения всех перечисленных свойств,*
- *размещения фигур,*
- *перемещения,*
- *удаления,*
- *изменения размеров.*

Фигуры могут иметь заливку указанным фоном или оставаться не закрашенными.

Библиотека графических примитивов может использоваться как при дизайне любых интерфейсов пользователя (форм, страниц), так и для построения графиков или чертежей при визуализации аналитической обработки данных в информационных системах и системах поддержки принятия решений.

В классах необходимо обеспечить:

- *Хранение условно-постоянной информации статическими полями или константами.*
- *Все необходимые конструкторы с параметрами и конструкторы по умолчанию;*
- *Свойства для установки и получения значений всех координат, а также для изменения цвета и получения текущего цвета;*
- *Для линий – методы изменения угла поворота линий относительно первой точки;*
- *Для многоугольника – метод масштабирования.*
- *Для всех фигур – методы вывода на заданную пользователем поверхность и методы удаления (уничтожения, стирания) фигур с поверхности.*

Библиотека графических примитивов будет использована для оформления интерфейсов пользователя информационных систем.

1.2 Описание архитектуры библиотеки

В этом пункте необходимо продумать: какие классы должны быть разработаны и какие отношения существуют между этими классами. Можно выделить несколько основных отношений: наследование, реализация, ассоциация, композиция и агрегация.

Описание архитектуры должно содержать текст и диаграммы UML.

Наследование является базовым принципом ООП и позволяет одному классу (наследнику) унаследовать функционал другого класса (родительского). Нередко отношения наследования еще называют генерализацией или обобщением. Наследование определяет отношение IS A, то есть "является". Например, на рис. 1 показано наследование класса Student от класса Employee.

```
public class Employee
{
    //Поля класса
    #region
    //Конструкторы класса
    #region
    //Свойства класса
    #region
    //Методы класса
    #region
}
```

```
public class Student : Employee
{
    //Поля
    public int NumberGr { get; set; }
    private int[] ocenka;
    private string email;
    private string phone;
    //Конструкторы
     public Student(string fio, int[] ocenki) : base(
    { this.FullName = fio; ocenka = ocenki; }
```

Рисунок 1 – Реализация наследования

В данном случае используется наследование, а объекты класса Student также являются и объектами класса Employee.

С помощью диаграмм UML отношение наследования между классами выражается в незакрашенной стрелке от класса-наследника к классу-родителю – рис.2.

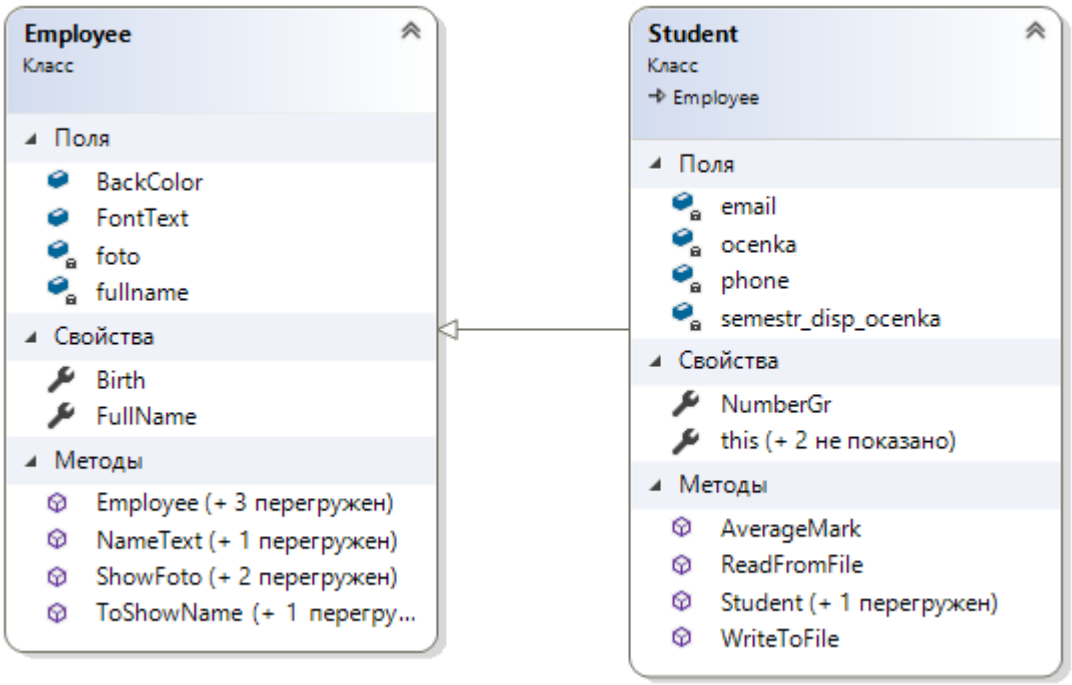

Рисунок 2 – UML-диаграмма для отношения наследования

Реализация предполагает определение интерфейса и его реализация в классах. Например, имеется интерфейс IBankAccount с методом PayIn, который реализуется в классе GoldBank как видно из рис. 3.

```
interface IBankAccount
{
   //метод для внесения денег на счет
    void PayIn(decimal amount);
```

```
class GoldBank : Bank, IBankAccount
{
    private decimal balance;
    public void PayIn(decimal amount)
    {
        balance += amount;
    }
```

Рисунок 3 – Отношение реализации

С помощью диаграмм UML отношение реализации выражается или незакрашенной стрелкой от класса к интерфейсу, только линия теперь пунктирная или специальным символом – рис.4.

Рисунок 4 – UML-диаграмма для отношения реализации

Ассоциация — это отношение, при котором объекты одного типа неким образом связаны с объектами другого типа. Например, объект одного типа содержит или использует объект другого типа. Например, студент учится в определенной группе как видно из рис. 5.

```
public class Group
{

}
public class Student
{
    public Group Group { get; set;}
```

Рисунок 5 – Отношение ассоциации

Класс Student связан отношением ассоциации с классом Group. На схемах UML ассоциация обозначается в виде обычной стрелки – рис. 6.

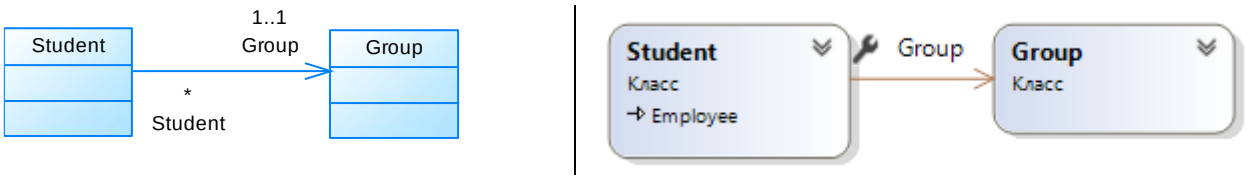

Рисунок 6 – UML-диаграмма для ассоциации

Нередко при отношении ассоциации указывается кратность связей. В данном случае единица у Group и звездочка у Student на диаграмме отражает связь 1 ко многим.

Отношения агрегации и композиции являются частными случаями отношения ассоциации.

Композиция определяет отношение HAS A, то есть отношение "имеет". Например, класс Клиент (Client) содержит объект класса Счет (Account) – рис.7.

```
class Account { }
class Client
{
    Account account;
    public Client()
    {
        account = new Account();
    }
```

Рисунок 7 – Отношение композиции

При этом класс Клиент полностью управляет жизненным циклом объекта класса Счет. При уничтожении объекта класса Клиент в области памяти вместе с ним будет уничтожен и объект класса Счет. И в этом плане объект Клиента является главным, а объект Счета - зависимым.

На диаграммах UML отношение композиции проявляется в обычной стрелке от главной сущности к зависимой, при этом со стороны главной сущности, которая содержит, объект второй сущности, располагается закрашенный ромбик – рис. 8.

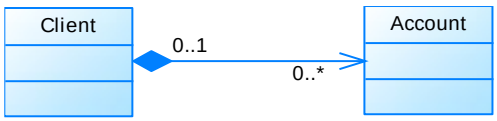

Рисунок 8 – UML-диаграмма для композиции

От композиции следует отличать ***агрегацию***. Она также предполагает отношение HAS A, но реализуется она иначе – рис. 9.

```
public abstract class Account { }
class Client
{
    Account account;
    public Client(Account acc)
    {
        account = acc;
    }
```

Рисунок 9 – Отношение агрегации

При агрегации реализуется слабая связь, то есть в данном случае объекты Client и Account будут равноправны. В конструктор класса Client передается ссылка на уже имеющийся объект класса Account. И, как правило, определяется ссылка не на конкретный класс, а на абстрактный класс или интерфейс, что увеличивает гибкость программы.

Отношение агрегации на диаграммах UML отображается также, как и отношение композиции, только теперь ромбик будет не закрашенным – рис. 10.

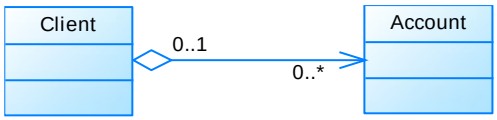

Рисунок 10 – UML-диаграмма для агрегации

Необходимо описать все классы, которые планируется реализовывать в библиотеке и построить отношения между ними. Чем больше будет продумано на этом этапе, тем легче будет идти разработка. Описание классов и отношений лучше делать используя UML 2.5.1.

1.3 Описание интерфейсов

После того как список классов сформирован нужно определить абстрактный класс и классы возможных интерфейсов – как они будут взаимодействовать между собой.

Обязательно описать назначение интерфейсов как технологии взаимодействия, а не только как возможность реализации множественного наследования.

Интерфейсы — это обычные абстрактные классы, имена которых обычно начинаются с заглавной буквы “I”, например: IGame, ILogic, IJournal, IEvenHandler и т.д. От интерфейсов наследуются реальные классы, которые имплементирует все виртуальные функции интерфейса.

Проектирование интерфейсов –важная и ответственная задача. Как правило, интерфейсы проектируется один раз на длительное время, а реальные объекты за интерфейсами могут меняться. Например пусть имеется интерфейс IJournal, в котором объявлен метод Save() для записи журнала. Интерфейс определяет возможность сохранения (записи) журнала, но это может быть FileJournal, NetJournal, DBJournal, то есть любой объект для сохранения (записи). Главное чтобы в классе, который наследуется от интерфейса, был определен метод Save() иначе, интерфейс не будет работать.

У абстрактных классов и интерфейсов много общего. Однако все же они имеют некоторые отличия.

Когда следует использовать абстрактные классы:

• Если надо определить общую функциональность для родственных объектов.

Когда следует использовать интерфейсы:

• Если нам надо определить функциональность для группы разноплановых объектов, которые могут быть никак не связаны между собой.

Выбор между абстрактным классом и интерфейсом можно свести к следующему принципу: если классы относятся к единой системе классификации, то выбирается абстрактный класс. Иначе выбирается интерфейс.

Говоря об использовании абстрактных классов и интерфейсов, можно рассмотреть такие аспекты, как состояние и действие. Как правило, абстрактные классы фокусируются на общем состоянии классов-наследников. В то время как интерфейсы строятся вокруг какого-либо общего действия (действий).

Таким образом, можно сформулировать принципы использования интерфейсов и абстрактных классов:

- если разноплановые классы обладают какими-то общими действиями, то это действия следует выносить в интерфейс;
- для родственных классов, которые имеют общее состояние, лучше определять абстрактный класс.

1.4 Написание тестов на функциональность библиотеки

Необходимо протестировать не только основную функциональность разработанных классов, но и все их отношения, показать, как описанные связи реализуются. Описание тестов должно присутствовать в тексте записки.

При тестировании обязательно разрабатывайте понятный интерфейс клиента: подписывайте элементы управления, демонстрируйте начальное и конечное состояние тестируемых объектов, делайте выводы.

1.5 Демонстрация результатов практического использования классов

Подойдя к этому этапу, вы должны иметь готовую библиотеку классов. Все классы и взаимодействия между ними описаны, у каждого класса определены его методы и остается только разработать конкретный пример формы (страницы), в которой используется функциональность разработанной

библиотеки. На этом этапе могут использоваться не все разработанные классы, а только те, которые решают поставленную задачу.

1.6 Требования к пояснительной записке

Обязательные позиции:

1. Анализ поставленной задачи: функциональность разрабатываемой библиотеки и назначение ее использования

2. Диаграмма классов

3. Описание программного кода классов

4. Тестирование функциональности классов

5. Пример использования библиотеки для построения интерфейса прикладного решения

6. Руководство пользователя-программиста при использовании созданной библиотеки как ресурса

Заключение

Библиографический список

Приложения

Объем пояснительной записки не менее 15 страниц, без учета титульного листа, который не нумеруется. Диаграммы классов необходимо разработать, используя UML 2.5.1.

Пояснительная записка к курсовой работе оформляется в соответствии с требованиями, описанными на сайте ГУАП в разделе Нормативная документация, URL: https://guap.ru/standart/doc.

2. Оценка результатов курсовой работы

На оценку 3:

1. Архитектура библиотеки должна иметь не менее 4-х классов, включая абстрактный класс.
2. Должно быть реализовано отношение наследования.
3. Обязательна реализация полиморфизма: перегруженные и переопределенные методы и свойства классов.

4. Данные класса должны быть описаны полями и свойствами.
5. Представлены тесты и продемонстрированы результаты их выполнения в соответствии с п. 1.4.
6. Показан результат практического использования библиотеки для создания приложения пользователя в конкретной предметной области в соответствии с п. 1.5
7. Обязательны развернутые комментарии к программному коду.

На оценку 4:

1. Все, что на оценку 3.
2. Реализация событий классов.
3. Реализация отношений ассоциации, композиции, агрегации.

На оценку 5:

1. Все, что на оценку 4.
2. Использование интерфейсов.

Анализ и постановка задачи в п. 1.1 являются основой для тестирования заявленной функциональности – учитывайте этот факт и пишите текст раздела ответственно.

3. Варианты заданий

Тема курсовой работы выбирается самостоятельно из табл. 1. Кроме того, студент может самостоятельно придумать себе задание, отсутствующее в предложенном списке, предварительно согласовав его с преподавателем.

Таблица 1 – Варианты тем для курсовой работы

№ п/п	Примерный перечень тем для выполнения курсовой работы / выполнения курсового проекта
1	В приложении пользователь может создать объект класса Текст, используя классы Предложение, Слово. Классы должны иметь методы для создания, изменения, чтения из файла, сохранения в файл, удаления.
2	В приложении пользователь может создать объект класса Автомобиль, используя классы Колесо, Двигатель. Методы классов должны фиксировать следующие состояния: движение, заправка, замена колеса. Информация об автомобиле должна включать все технические особенности.
3	В приложении пользователь может создать объект класса Самолет, используя классы Крыло, Шасси, Двигатель. Реализовать методы для сборки самолета, описания маршрута, заправки.

4	В приложении пользователь может создать объект класса Государство, используя классы Область, Район, Город. Реализовать методы чтения, записи необходимой информации о государстве, обеспечить поиск областей, районов, городов с соответствующей информацией о каждом объекте.
5	В приложении пользователь может создать объект класса Планета, используя классы Материк, Океан, Остров. Методы для чтения, записи всей необходимой информации о планете, поиск и сортировку всех объектов, изменение информации.
6	В приложении пользователь может создать объект класса Звездная система, используя классы Планета, Звезда, Луна. Необходимы методы чтения, записи, просмотра списков объектов, изменения информации.
7	В приложении пользователь может создать объект класса Компьютер, используя классы Винчестер, Дисковод, Оперативная память, Процессор. Методы для проверки на вирусы, сборки, чтения, записи полной технической информации о собранном компьютере.
8	В приложении пользователь может создать объект класса Квадрат, используя классы Точка, Отрезок. Необходимы методы для задания размеров, цвета заливки и прорисовки контуров фигур, создания, удаления, перемещения, изменения цвета.
9	В приложении пользователь может создать объект класса Круг, используя классы Точка, Окружность. Методы для работы с фигурами должны обеспечить создание, удаление, изменение размеров и координат, определения принадлежности точки, перемещения.
10	В приложении пользователь может создать объект класса Щенок, используя классы Животное, Собака. Методы для работы с информацией о щенке должны позволять читать, сохранять всю информацию о щенке, обеспечить просмотр фотографий щенка с возможностью чтения, сохранения.
11	В приложении пользователь может создать объект класса Текстовый файл, используя классы Файл, Директория. Методы для работы с файлами должны полностью обеспечить пользователя возможностями создания, удаления, изменения, переименования файлов.
12	В приложении пользователь может создать объект класса Одномерный массив, используя классы Массив, Элемент. Методы для работы с массивами должны обеспечивать их создание, изменение, выполнение операций (сложить, вычесть, перемножить).
13	В приложении пользователь может создать объект класса Простая дробь, используя класс Число. Методы для работы с объектами должны обеспечить создание, выполнение операций сложения, вычитания, умножения и деления.
14	В приложении пользователь может создать объект класса Дом, используя классы Окно, Дверь. Методы должны позволять сохранить, прочитать, изменить всю необходимую информацию об объекте, показать графическое

	представление объекта с возможностью сохранения и редактирования.
15	В приложении пользователь может создать объект класса Цветок, используя классы Лепесток, Бутон. Методы для работы с объектами должны сохранять, изменять всю необходимую информацию как в текстовом, так и в графическом виде.
16	В приложении пользователь может создать объект класса Дерево, используя классы Лист, Ветка. Методы для описания объектов должны позволять сохранять, изменять всю необходимую информацию об объекте: посадке, обрезке, прививке как в текстовом, так и в графическом виде.
17	Создать базовый класс Point. В приложении пользователь может на его основе создать классы ColoredPoint и Line. На основе класса Line создать классы ColoredLine и PolyLine. Методы для работы с графическими объектами должны позволять создание, заливку, удаление, изменение цвета контура, перемещение.
18	В приложении пользователь может создать объект класса Фотоальбом, используя классы Фотография, Страница. Методы для работы с графическими объектами должны обеспечить создание, удаление, редактирование и просмотр.
19	В приложении пользователь может описать класс Комната, содержащий сведения о метраже, высоте потолков и количестве окон. Описать методы для создания, изменения, вычисления параметров (например, площади), записи, поиска и сортировки в каталоге.
20	В приложении пользователь может создать класс Поезд, содержащий следующую информацию: пункт назначения, номер поезда (может содержать буквы и цифры, время и дату отправления, количество вагонов различного типа (плацкартных, общих, купе, СВ). Предусмотреть свойства для получения состояния объекта. Описать класс Вокзал, содержащий массив поездов. Обеспечить следующие возможности: вывод информации о поезде по указанному номеру, вывод информации о поездах, отправляющихся после указанного времени, или в указанном направлении, или по совокупности параметров времени и направления.
21	В приложении пользователь может создать объект класса Самолет, содержащий следующую информацию: пункта назначения; номер рейса; название авиакомпании; дату и время отправления; стоимость полета. Предусмотреть свойства для получения состояния объекта. Описать класс Аэропорт, содержащий массив самолетов. Обеспечить следующие возможности: вывод информации о самолете по указанному номеру рейса; вывод информации о самолетах, отправляющихся после указанного времени, или в указанном направлении, или по совокупности параметров времени и направления.
22	В приложении пользователь может создать объект класса Товар, содержащий следующие закрытые поля: название товара; название фирмы-изготовителя; стоимость товара. Предусмотреть свойства для получения состояния объекта. Описать класс Склад, содержащий массив товаров. Обеспечить следующие возможности: вывод информации о товаре по введенному наименованию;

	поиск подходящего товара по наименованию и цене.
23	В приложении пользователь может создать объект класса Туристическая путевка. Сформировать набор предложений клиенту по выбору туристической путевки различного типа (отдых, экскурсии, лечение, шопинг, круиз и т. д.) для оптимального выбора. Учитывать возможность выбора транспорта, питания и числа дней. Реализовать выбор и сортировку направлений.
24	Создать абстрактный класс Figura. В приложении пользователь может на его основе создать классы Triangle и Square. На основе класса Square создать класс ManySided. Методы для работы с фигурами должны обеспечивать установку и получение значений всех координат, а также для изменения цвета и получения текущего цвета; для треугольника – метод, вычисляющий возможность построения треугольника; для всех фигур – методы масштабирования; для квадрата и многоугольника – методы заливки внутреннего пространства; для всех фигур – методы вывода на заданную пользователем поверхность и методы удаления (уничтожения, стирания) фигур с поверхности. Класс Triangle объявить закрытым классом, не имеющим наследников.
25	В приложении пользователь может Создайте класс Phone, который содержит описание его технических параметров и реализует методы для ответа на звонок, совершения звонка, приема и отправки сообщения, работы с фото и видео и т.д. Определить иерархию Телефоны, Мобильные телефоны, Проводные телефоны, Смартфоны, Кнопочные телефоны, Фитнес-часы, Фитнес-браслеты.
26	В приложении пользователь может Транспорт. Определить иерархию подвижного состава железнодорожного транспорта. Создать пассажирский поезд. Подсчитать общую численность пассажиров и багажа. Провести сортировку вагонов поезда на основе уровня комфортности. Найти в поезде вагоны, соответствующие заданному диапазону параметров числа пассажиров.
27	В приложении пользователь может Авиакомпания. Определить иерархию самолетов. Провести сортировку самолетов компании по дальности полета. Найти самолет в компании, соответствующий заданному диапазону параметров потребления горючего.
28	В приложении пользователь может Таксопарк. Определить иерархию легковых автомобилей. Провести сортировку автомобилей парка по расходу топлива. Найти автомобиль в компании, соответствующий заданному диапазону параметров скорости.
29	В приложении пользователь может создать объект класса Кредит. Сформировать набор предложений клиенту по целевым кредитам различных банков для оптимального выбора. Учитывать возможность досрочного погашения кредита и/или увеличения кредитной линии. Реализовать выбор и поиск кредита.
30	В приложении пользователь может создать объект класса Счет и выполнить все необходимые действия. Необходимо учитывать следующие возможности: клиент может иметь несколько счетов в банке; счет может быть открыт, закрыт; счет открыт с определенными условиями (ставка, срок). Требуется обеспечить начисление и списывание сумм по счетам, начисление процентов в

	соответствие с условиями счета (процентная ставка и сроки начисления), списывание оплаты за ведение счета. Реализовать необходимые отношения с классами Клиент, Банк,
31	<Тема самостоятельно сформулирована студентом и обязательно утверждена у руководителя>