МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
федеральное государственное бюджетное образовательное учреждение высшего образования  
**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет информационных систем и технологий  
Кафедра «Информационные системы»  
Дисциплина «Компонентно-ориентированное программирование»

Отчет по лабораторным работам №1-7

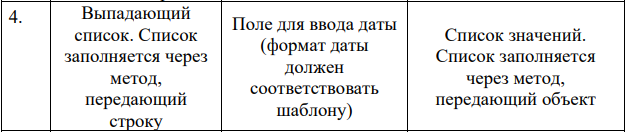
Выполнила:  
студентка гр. ПИбд-32  
Цуканова И.В.  
Проверил:  
доцент кафедры  
Эгов Е.Н.

Ульяновск, 2024 г

**Лабораторная работа 1**

Для каждого компонента описать его назначение, а также публичные методы: что подавать на вход, что будет на выходе и какие могут возникать исключения.

**Задание по варианту:**



1. CustomComboBox - визуальный компонент для выбора из выпадающего списка. Включает в себя ComboBox.

AddItem - метод заполнения выпадающего списка, принимает строку.

ClearList - метод очистки списка.

SelectElement(get; set) - публичное свойство для установки и получения выбранного значения (возвращает пустую строку, если нет выбранного значения).

1. CustomTextBox - визуальный компонент для ввода даты с проверкой по шаблону. Включает в себя TextBox.

SetTip - метод для задания примера правильного ввода. На вход принимает строку.

Template - публичное свойство для задания шаблона.

TextBoxValue - публичное свойство для получения и заполнения значения textbox. Если не задан шаблон, выдаётся ошибка. Если введённая дата не соответствует шаблону, выдаётся ошибка.

1. Компонент CustomListBox - визуальный компонент для вывода списка в виде списка записей. Включает в себя ListBox.

SetLayoutString - метод для заполнения макетной строки и символов начала и конца для имени свойства. На вход принимает строку, символ, символ. Перый символ - символ начала, второй символ - символ конца.

AddElement - параметризованный метод заполнения списка. На вход принимает объект какого-то класса.

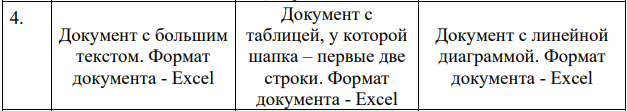
GetSelectedItem - параметризованный метод для получения объекта из выбранной строки.

Index - публичное свойство для установки и получения индекса выбранной строки.

**Лабораторная работа 2**

Для каждого компонента описать его назначение, а также публичные методы: что подавать на вход, что будет на выходе и какие могут возникать исключения.

**Задание по варианту:**



1. ExcelDocument - не визуальный компонент для создания документа с большим текстом. Формат документа - Excel.

CreateExcel - метод создания документа Excel с текстом. На вход принимает:

string path - путь к файлу.

string title - заголовок документа.

List<string> text - абзацы текста.

Исключение возникают в случае, если какой-либо входной параметр не заполнен.

1. ExcelTable - не визуальный компонент для создания документа с таблицей, у которой шапкой является первые две строки и в них есть группировка. Формат документа - Excel.

CreateExcel - метод создания документа Excel с таблицей. На вход принимает:

string path - путь к файлу.

string title - заголовок документа.

List<MergeCells> mergeCells - информация по объединению ячеек (по столбцам). В MergeCells :

string Header - текст для объединённых ячеек.

int Start - начальный столбец.

int Stop - конечный столбец.

List<InfoHeaders> headers - заголовки для шапки с информацией. В InfoHeaders :

string Name - заголовок.

string Reference - какое свойство/поле класса соответствует колонке.

int Width - ширина колонки.

List<TableData> values - данные для таблицы.

Исключения возникают в случаях, если какие-либо входные данные не заполнены, объединённые ячейки накладываются друг на друга, ячейки шапки не заполнены и для каждой колонки не известно свойство/поле класса из которого для него следует брать значение.

1. ExcelDiagram - не визуальный компонент для создания документа с линейной диаграммой. Формат документа - Excel.

CreateExcel - метод создания документа Excel с линейно диаграммой. На вход принимает:

string path - путь к файлу.

string header - заголовок в документе.

string title - заголовок диаграммы.

LegendPosition position - расположение легенды для диаграммы. LegendPosition - перечисление, содержит 4 элемента: Left, Top, Right, Bottom.

List<Data> values - данные для диаграммы. В Data :

string Name - название серии.

List<int> Values - данные серии.

Исключение возникают в случае, если какой-либо входной параметр не заполнен.

**Лабораторная работа 3**

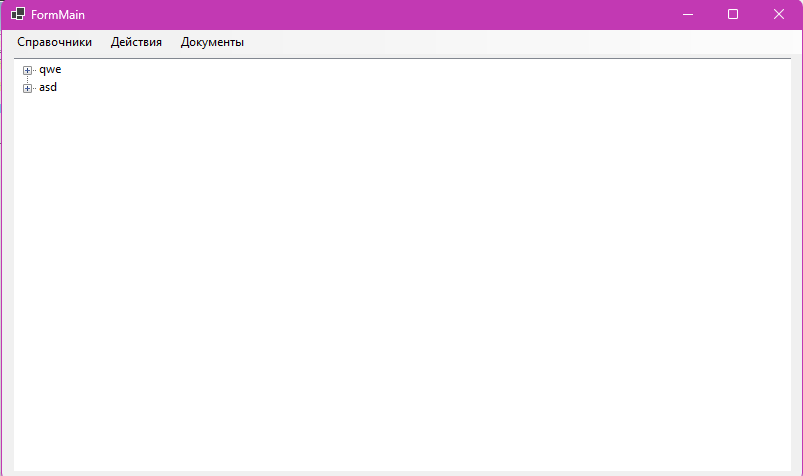
Описать программный продукт: цель, какие действия могут выполняться. Дать описание (лучше таблицей) используемым сторонним (компонентами, разработанными вашими одногруппниками) компонентам: преимущества и недостатки

**Задание по варианту:**

Учет книг в библиотеке. По книге хранить следующую информацию: название, описание, жанр (выпадающий список), стоимость (может быть бесплатной). Выводить в виде дерева (жанр - стоимость - идентификатор - название). Формировать документ в Excel по бесплатным книгам (в каждой строке текст с информацией: название книги и ее описание). Формировать отчет в Word с информацией по всем книгам (шапка: первые столбец и строка). По столбцу идет заполнение значениями идентификаторов книг. В первой строке остальные заголовки: название, жанр, стоимость (если пусто, то строчкой писать «бесплатная»). Круговая диаграмма в Pdf, сколько бесплатных книг какого жанра.

Пользовательский интерфейс состоит из следующих форм:

1. Основная форма (Рисунок 1). На ней выводится информация по книгам в виде дерева в формате вложенности жанр - стоимость - идентификатор - название. Представлен следующий функционал:
   1. Добавление новой книги. Комбинация клавиш Ctrl+A или Действия->Добавить.
   2. Редактирование выбранной книги. Комбинация клавиш Ctrl+U или Действия->Изменить.
   3. Удаление выбранной книги. Комбинация клавиш Ctrl+D или Действия->Удалить.
   4. Создание документа Excel с текстом. Комбинация клавиш Ctrl+S или Документы ->Строчки в Excel.
   5. Создание документа Excel с таблицей. Комбинация клавиш Ctrl+T или Документы ->Таблица Word.
   6. Создание документа Excel с круговой диаграммой. Комбинация клавиш Ctrl+C или Документы ->Круговая диагр PDF.

Рисунок 1. Основная форма

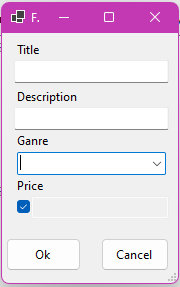
1. Форма добавления/редактирования книги (Рисунок 2). В форме происходит заполнение всех полей сущности за исключением идентификатора. Для заполнения жанра книги использовался визуальный компонент выбора из выпадающего списка. Книга может быть бесплатной, поэтому поле для заполнения цены - визуальный компонент ввода с возможностью оставления не заполненного значения. Для заполнения остальных полей использовались стандартные компоненты Windows Forms. При загрузке записи на редактирование все поля формы заполнены значениями из этой записи.

Рисунок 2. Форма добавления/редактирования записи

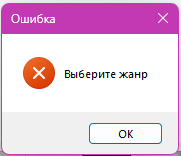
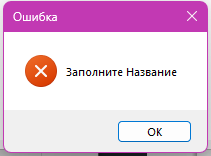
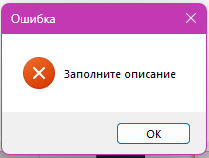
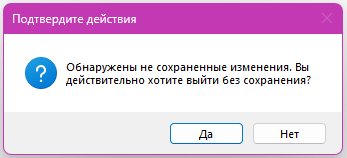
Присутствует валидация всех полей формы (Рисунок 3, 4, 5). Также присутствует фиксация факта изменения значений в полях (либо заполнение полей) и при попытке пользователя закрыть форму без сохранения выдается предупреждение о наличии не сохраненных изменений (Рисунок 6).

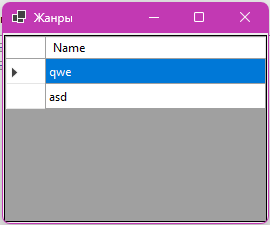
Рисунок 3. Валидация полей.

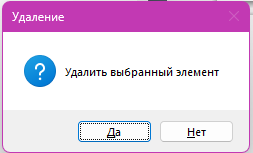
Рисунок 4. Валидация полей.

Рисунок 5. Валидация полей.

Рисунок 6. Предупреждение о наличии не сохраненных изменений.

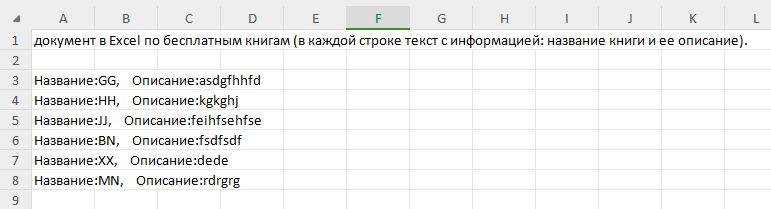
1. Форма работы со справочником жанров (Рисунок 7). На форме используется компонент DataGridView. Добавление нового жанра происходит по нажатию на клавишу Insert, после чего появляется строка в таблице, куда можно вписать название нового жанра (пустая строка не сохранится). Редактировать запись можно прямо в таблице (сохранение происходит либо по нажатию на клавишу Enter, либо при переходе к следующей записи). Удаление жанра (также можно удалить сразу несколько) происходит по нажатию на клавишу Delete. Удаление жанра обязательно необходимо подтвердить (Рисунок 8).

Рисунок 7. Справочник.

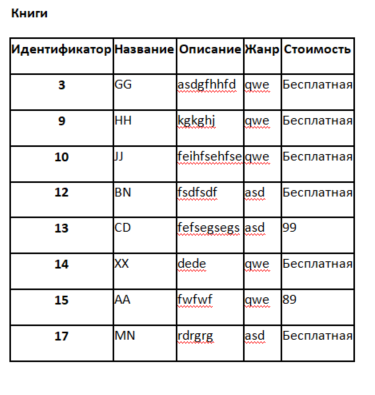
Рисунок 8. Подтверждение удаления жанра.

Создание документов:

1. Создание простого документа. Формирование документа Excel (Рисунок 9) по бесплатным книгам (в каждой строке текст с информацией: название книги и ее описание). У пользователя спрашивается путь до папки и имя файла и происходит сохранения документа по указанному пути.

Рисунок 9. Пример простого документа.

1. Создание документа с настраиваемой таблицей. Формирование отчета в Word с информацией по всем книгам (шапка: первые столбец и строка). По столбцу идет заполнение значениями идентификаторов книг. В первой строке остальные заголовки: название, жанр, стоимость. У пользователя спрашивается путь до папки и имя файла и происходит сохранения документа по указанному пути.

 Рисунок 10. Пример документа с настраиваемой таблицей.

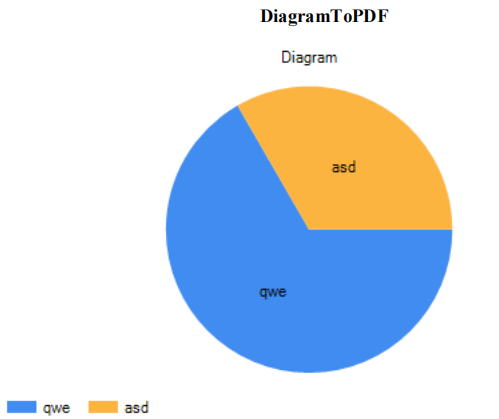
1. Создание документа с диаграммой. Формирование круговой диаграмма в Pdf, сколько бесплатных книг какого жанра. У пользователя спрашивается путь до папки и имя файла и происходит сохранения документа по указанному пути.

Рисунок 10. Пример документа с диаграммой.

Использованные сторонние компоненты:

|  |  |  |
| --- | --- | --- |
| Компонент | Преимущества | Недостатки |
| MyTree | 1. Предоставляет удобное отображение данных в виде древовидной структуры, что может быть полезным для иерархического представления информации. 2. Имеет возможность настройки отображаемых типов свойств и их значений, что позволяет гибко настроить отображение данных. 3. Позволяет настроить иерархию отображения данных. | Нет возможности настроить оформление. |
| controlInputNullableInt | 1. Есть возможность ввода пустого значения. 2. Позволяет вводить и редактировать данные с учетом определенных форматов или ограничений, обеспечивая удобство для пользователей. | Не выявлено |
| componentDocumentWithTableMultiHeaderWord | 1. Позволяет строить таблицу в документе Word на основе передаваемых данных. 2. Удобная настройка заголовков колонок и какие данные будут представлены в данной колонке. | Неудобная настройка ширины колонок |
| diagramTopdf | 1. Позволяет создавать красочные круговые диаграммы для визуализации информации. 2. Обладает возможностью настройки заголовка, расположения легенды | Текст только на английском |

**Лабораторная работа 4**

Описать механизм работы с плагинами, как нужно оформлять плагин, чтобы он "подтянулся" в вашу программу.

Для работы с плагинами был создан интерфейс IPluginConvention, включающий в себя набор методов и свойств, который должен реализовывать плагин. Это интерфейс-соглашение, по которому плагин «цепляется» к решению.

В главной форме создан словарь плагинов, добавление элементов в который осуществляется в методе LoadPlugins:

1. Получение всех файлов типа .dll из папки Plugins (все библиотеки загружаются в эту папку с помощью инструкции, которая выполняется после сборки приложения - copy /Y "$(TargetDir)\*.dll" "$(SolutionDir)WinFormsAppByPlugins\bin\Debug\net6.0-windows\Plugins\\*.dll")
2. Из каждой сборки (создается из dll файла) получаются типы данных, которые она содержит.
3. Для каждого типа выполняется проверка, которая исключает те типы, которые являются интерфейсами или к которым нельзя присвоить IPluginsConvention.
4. Через рефлексию создается объект, класс которого реализует IPluginsConvention, и добавляется в словарь плагинов.
5. Заполняется меню формы, через которое пользователь будет использовать функционал подключенных плагинов.

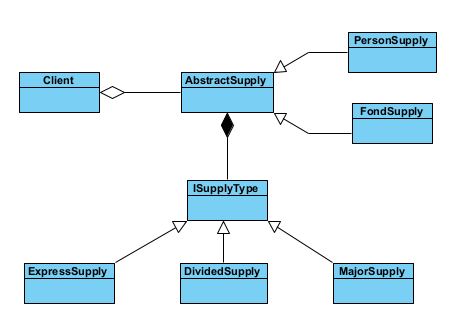
**Лабораторная работа 5**

Для каждого паттерна из варианта: дать описание паттерна, создать диаграмму классов, отражающую взаимодействие новых сущностей (а также используемый паттерн) с классами, созданными в рамках 3 лабораторной работы. Отдельно указать какие роли выполняют классы, являющиеся участниками паттерна.

**Bridge**

Мост (Bridge) позволяет отделить абстракцию от реализации таким образом, чтобы и абстракцию, и реализацию можно было изменять независимо друг от друга. Даже если отделить абстракцию от конкретных реализаций, то все равно все наследуемые классы будут жестко привязаны к интерфейсу, определяемому в базовом абстрактном классе. Для преодоления жестких связей и служит паттерн Мост.

Применяется:

1. Когда надо избежать постоянной привязки абстракции к реализации.
2. Когда наряду с реализацией надо изменять и абстракцию независимо друг от друга. То есть изменения в абстракции не должно привести к изменениям в реализации.

**Abstraction:** AbstractSupply. Абстрактный класс, представляющий функционал поставки книг в библиотеку.

**RefinedAbstraction:** PersonSupply и FondSupply. Представляет передачу (поставку) книг библиотеке от физического лица. Представляет закупку книг для библиотеки фондом.

**Implementor:** ISupplyType. Интерфейс, предоставляющий метод поставки.

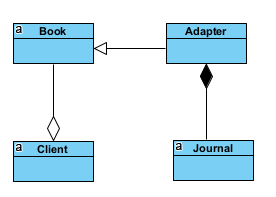
**ConcreteImplementorA и ConcreteImplementorB:** ExpressSupply, DividedSupply, MajorSupply. Быстрая поставка маленькой партией. Поставка несколькими партиями. Поставка одной большой партией.

**Adapter**

Паттерн Адаптер (Adapter) предназначен для преобразования интерфейса одного класса в интерфейс другого. Благодаря реализации данного паттерна можно использовать вместе классы с несовместимыми интерфейсами.

Применяется:

1. Когда необходимо использовать имеющийся класс, но его интерфейс не соответствует потребностям.
2. Когда надо использовать уже существующий класс совместно с другими классами, интерфейсы которых не совместимы.



**Target:** интерфейс Book, который реализует книгу.

**Adaptee:** интерфейс Journal, который реализует журнал.

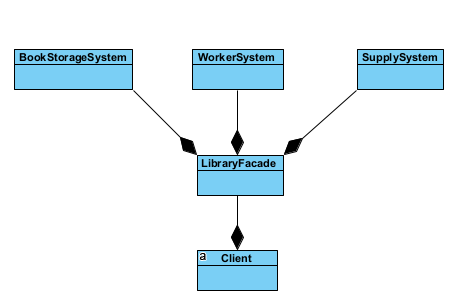
**Adapter:** класс, который позволяет работать журналами как с книгами.

**Facade**

Фасад (Facade) представляет шаблон проектирования, который позволяет скрыть сложность системы с помощью предоставления упрощенного интерфейса для взаимодействия с ней.

Применяется:

1. Когда имеется сложная система, и необходимо упростить с ней работу. Фасад позволит определить одну точку взаимодействия между клиентом и системой.
2. Когда надо уменьшить количество зависимостей между клиентом и сложной системой. Фасадные объекты позволяют отделить, изолировать компоненты системы от клиента и развивать и работать с ними независимо.
3. Когда нужно определить подсистемы компонентов в сложной системе. Создание фасадов для компонентов каждой отдельной подсистемы позволит упростить взаимодействие между ними и повысить их независимость друг от друга.

**SubsystemA, SubsystemB, SubsystemC:** BookStorageSystem, WorkerSystem и SupplySystem. Класс, описывающий функционал системы контроля за книгами. Класс, описывающий функционал системы работников. Класс, описывающий функционал пополнения библиотеки книгами.

**Facade:** LibraryFacade. Класс-фасад, описывающий библиотеку.

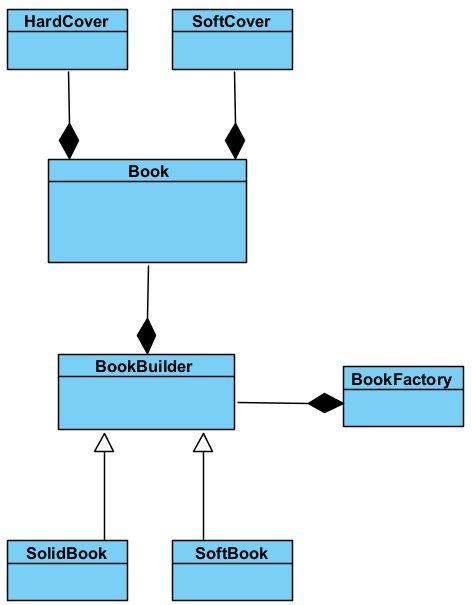
**Лабораторная работа 6**

Для каждого паттерна из варианта: дать описание паттерна, создать диаграмму классов, отражающую взаимодействие новых сущностей (а также используемый паттерн) с классами, созданными в рамках 3 лабораторной работы. Отдельно указать какие роли выполняют классы, являющиеся участниками паттерна.

**Builder**

Данный паттерн инкапсулирует создание объекта и позволяет разделить его на различные этапы.

Применяется:

1. Когда процесс создания нового объекта не должен зависеть от того, из каких частей этот объект состоит и как эти части связаны между собой.
2. Когда необходимо обеспечить получение различных вариаций объекта в процессе его создания.

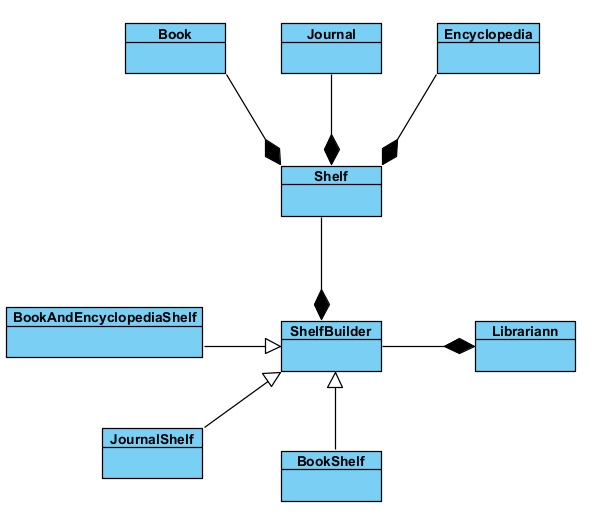
**Product:** Book. Класс, описывающий книгу.

**Components:** HardCover, SoftCover. Твердая обложка. Мягкая обложка.

**Builder:** BookBuilder. Класс-построитель книги.

**ConcreteBuilder:** SolidBook, SoftBook. Класс, создающий книгу с твердой обложкой. Класс, создающий книгу с мягкой обложкой.

**Director:** BookFactory. Класс, описывающий завод, производящий книги.

**Product:** Shelf. Класс, описывающий полку.

**Components:** Book, Journal, Encyclopedia. Книга. Журнал. Энциклопедия.

**Builder:** ShelfBuilder. Класс-построитель полки.

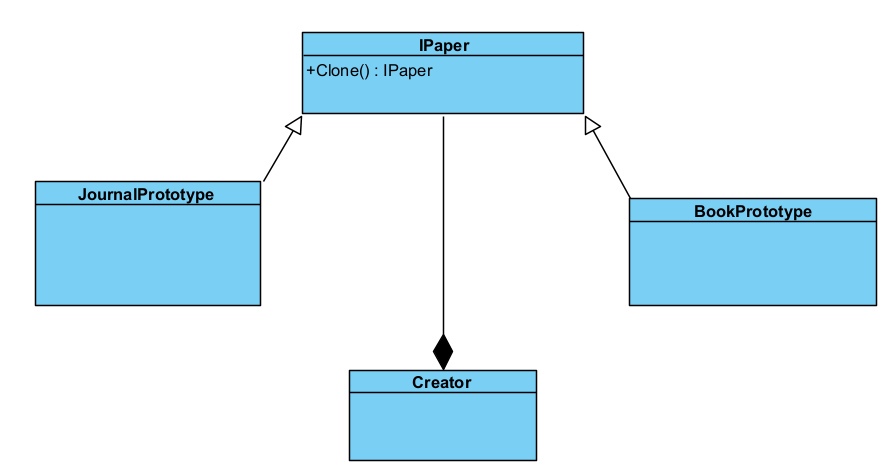
**ConcreteBuilder:** BookAndEncyclopediaShelf, JournalShelf, BookShelf. Класс, описывающий полку с книгами и энциклопедиями. Класс, описывающий полку с журналами. Класс, описывающий полку с книгами.

**Director:** Librarian. Класс, описывающий библиотекаря, который расставляет все.

**Prototype**

Паттерн Прототип (Prototype) позволяет создавать объекты на основе уже ранее созданных объектов-прототипов. То есть по сути данный паттерн предлагает технику клонирования объектов.

Применяется:

1. Когда конкретный тип создаваемого объекта должен определяться динамически во время выполнения.
2. Когда нежелательно создание отдельной иерархии классов фабрик для создания объектов-продуктов из параллельной иерархии классов (как это делается, например, при использовании паттерна Абстрактная фабрика).
3. Когда клонирование объекта является более предпочтительным вариантом нежели его создание и инициализация с помощью конструктора. Особенно когда известно, что объект может принимать небольшое ограниченное число возможных состояний.

**Prototype:** IPaper. Интерфейс, описывающий бумажный носитель информации.

**ConcretePrototypes:** BookPrototype, JournalPrototype. Класс, описывающий книгу. Класс, описывающий журнал.

**Client:** Creator. Класс, отвечающий за создание бумажных носителей информации.

**Лабораторная работа 7**

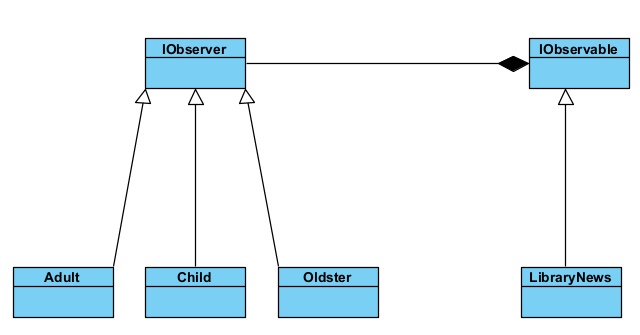
Для каждого паттерна из варианта: дать описание паттерна, создать диаграмму классов, отражающую взаимодействие новых сущностей (а также используемый паттерн) с классами, созданными в рамках 3 лабораторной работы. Отдельно указать какие роли выполняют классы, являющиеся участниками паттерна.

**Observer**

Паттерн Наблюдатель представляет поведенческий шаблон проектирования, который использует отношение «один ко многим». В этом отношении есть один наблюдаемый объект и множество наблюдателей. И при изменении наблюдаемого объекта автоматически происходит оповещение всех наблюдателей.

Применяется:

1. Когда система состоит из множества классов, объекты которых должны находиться в согласованных состояниях.
2. Когда общая схема взаимодействия объектов предполагает две стороны: одна рассылает сообщения, другая получает сообщения и реагирует на них. Отделение логики обеих сторон позволяет их рассматривать независимо и использовать отдельно друга от друга.
3. Когда существует один объект, рассылающий сообщения, и множество подписчиков, которые получают сообщения. При этом точное число подписчиков заранее неизвестно и процессе работы программы может изменяться.

**IObservable:** IObservable. Интерфейс, описывающий издателя.

**ConcreteObservable:** LibraryNews. Класс-реализация издателя в виде новостного паблика библиотеки.

**IObserver:** IObserver. Интерфейс, описывающий подписчика.

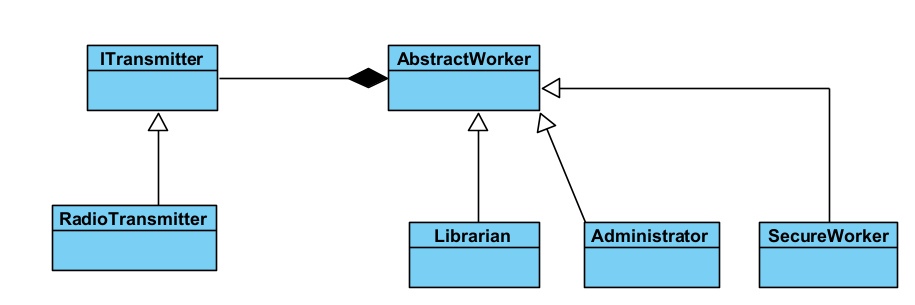
**ConcreteObserver:** Adult, Child, Oldster. Класс, описывающий взрослого читателя, которому интересны новости про поступление детективов. Класс, описывающий подростка читателя, которому интересны новости про поступление научной фантастики. Класс, описывающий пожилого читателя, которому интересны новости про поступление мемуаров известных личностей.

**Mediator**

Паттерн Посредник представляет такой шаблон проектирования, который обеспечивает взаимодействие множества объектов без необходимости ссылаться друг на друга. Тем самым достигается слабосвязанность взаимодействующих объектов.

Применяется:

1. Когда имеется множество взаимосвязанных объектов, связи между которыми сложны и запутаны.
2. Когда необходимо повторно использовать объект, однако повторное использование затруднено в силу сильных связей с другими объектами.



**Mediator:** ITransmitter. Интерфейс, описывающий передатчик.

**Colleague:** AbstractWorker. Класс, описывающий пользователя, который может отправлять и принимать сообщения.

**ConcreteColleague1 и ConcreteColleague2:** Librarian, Administrator, SecureWorker. Класс, описывающий библиотекаря. Класс, описывающий администратора. Класс, описывающий работника охраны.

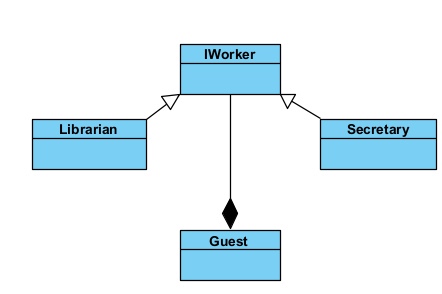
**ConcreteMediator:** RadioTransmitter. Класс-реализация для обмена информацией по рации.

**Chain of Responsibility**

Цепочка Обязанностей – поведенческий шаблон проектирования, который позволяет избежать жёсткой привязки отправителя запроса к получателю, позволяя нескольким объектам обработать запрос. Все возможные обработчики запроса образуют цепочку, а сам запрос перемещается по этой цепочке, пока один из её объектов не обработает запрос. Каждый объект при получении запроса выбирает, либо обработать запрос, либо передать выполнение запроса следующему по цепочке.

Применяется:

1. Когда имеется более одного объекта, который может обработать определённый запрос.
2. Когда надо передать запрос на выполнение одному из нескольких объект, точно не определяя, какому именно объекту.
3. Когда набор объектов задаётся динамически.



**Handler:** IWorker. Интерфейс, описывающий обработчик запроса.

**ConcreteHandler1 и ConcreteHandler2:** Secretary, Librarian. Класс, описывающий секретаря в библиотеки (смотрит, есть ли у клиента читательский билет, оформляет необходимые бумаги). Класс, описывающий библиотекаря (ищет и приносит нужную книгу).

**Client:** Guest. Класс, описывающий клиента библиотеки.