**Паттерны проектирования**

**ИСиТ,ПОИБМС-2**

**Лекция 1**

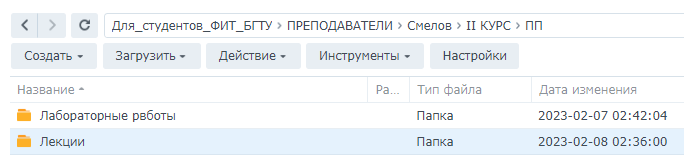
**Паттерны проектирования**

Лекции: 32 часа. Лабораторные работы: 32 часа. Зачет.

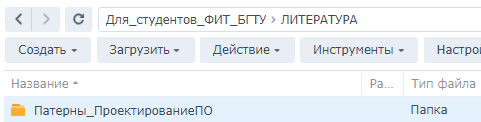
Содержание: 22 паттерна.

Инструменты: C#, JavaScript, https://www.draw.io.

Лекции, задания на лабораторные работы:

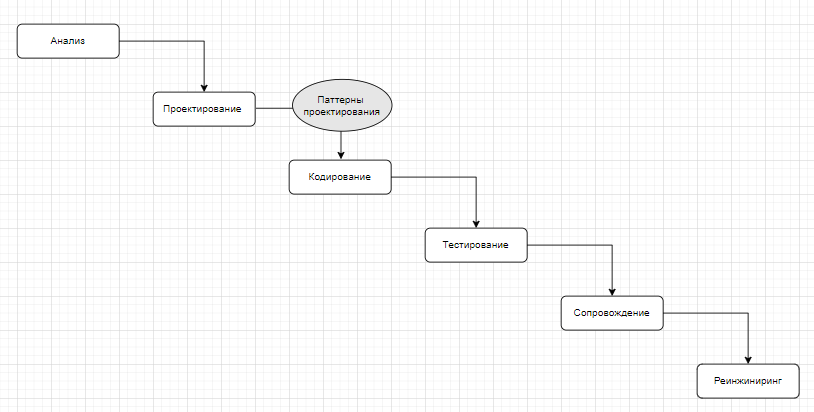


Литература:



**Введение**

1. **Паттерн:** pattern – узор, шаблон, схема.
2. **Паттерн проектирования:** схема решения часто встречаемой задачи, прием программирования, **computer science:** технология программирования, паттерн=шаблон, архитектор: применить паттерн, не библиотека, не фреймворк, не алгоритм, достоинства: проверенные решения, стандартный код, программистский словарь.
3. **Паттерн проектирования:** Кристофер Александер для проектирования человеческого окружения (городская архитектура, ландшафт, мебель…), Эрих Гамме, Ричард Хелм, Ральф Джонсон, Джон Влиссидес «Design Patterns: Elements of Reusable Object-Oriented Software», gang of four (GOF) – банда четырех.
4. **Решение о применении паттерна**



1. **Реализация паттерна**: разработка программного кода в соответствии с паттерном; 2 реализации одного и того же паттерна могут сильно отличаться.
2. **Классификация**: порождающие (гибкие методы создания объектов), структурные (связи между объектами), поведенческие (коммуникация между объектами).
3. **Классификация**: **идиомы** – низкоуровневые (зависят от языка программирования); **архитектурные** – самые высокоуровневые.
4. **Повторное использования кода**: реализованные в рамках системы программирования (классы, функции, библиотеки, контейнеры (коллекции, итераторы)), паттерны программирования, фреймворки (паттерн, поддерживаемый технологией, библиотекой, инструментарием).
5. **Описание паттерна:**

* описание проблемы;
* мотивация к применению;
* UML-диаграмма;
* структуры применяемых классов;
* пример на одном из языков программирования;
* особенности реализации;
* связи с другими паттернами.

1. **Фундамент паттернов проектирования:** объектно-ориентированное программирование. Наследование, инкапсуляцию и полиморфизм можно считать паттернами встроенными в систему ООП-программирования.

**Объектно-ориентированное программирование (повторение)**

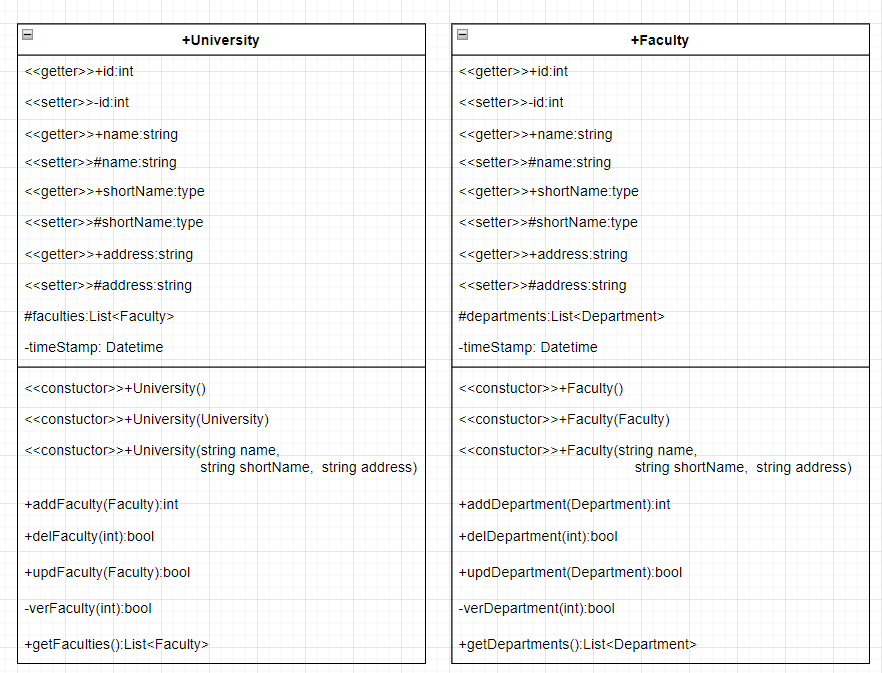
1. **ООП:** распространение методов системного анализа на процесс проектирования программного обеспечения.
2. **ООП**: упростить взаимодействие ***бизнес аналитик-проектировщик – разработчик***.
3. **ООП:** парадигмы: абстракция, инкапсуляция, полиморфизм.
4. **Объекты и классы**: класс: тип, структура объекта, объект: экземпляр класса, объект имеет тип. Класс – тип элементов системы (результат системного анализа); класс/объект: состояние, поведение

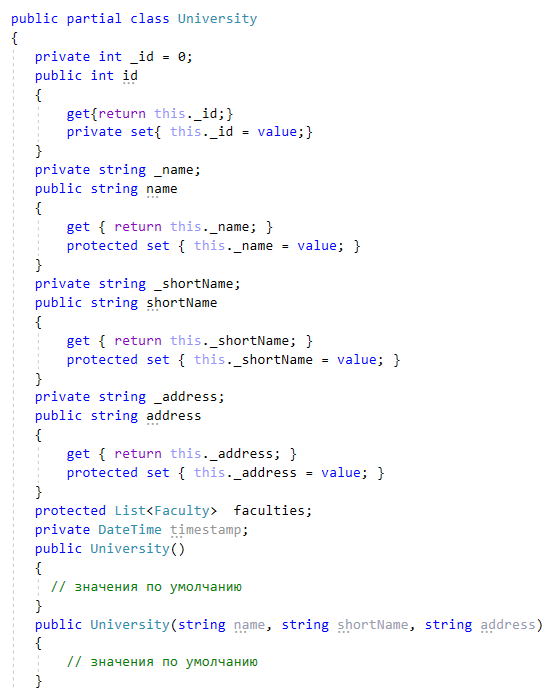
**Абстракция**

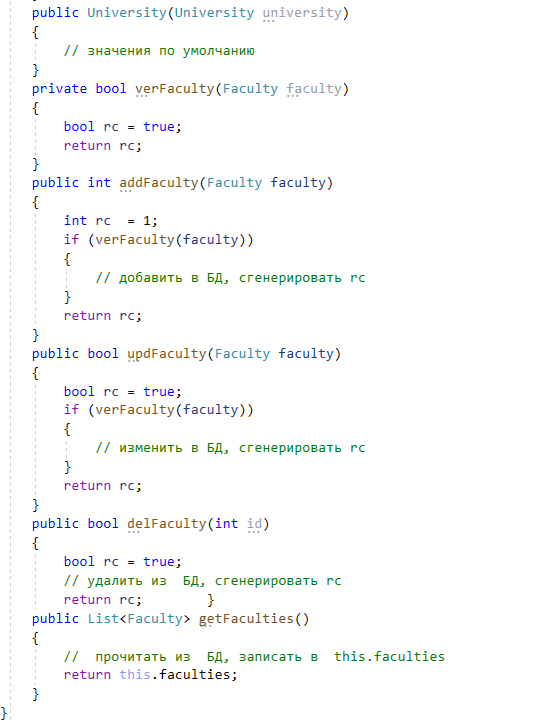
1. **Абстракция**: моделирование системы, классы – типы объектов, объекты – экземпляры типов, классы/объекты отражают реальный объект только с точки зрения наблюдателя.

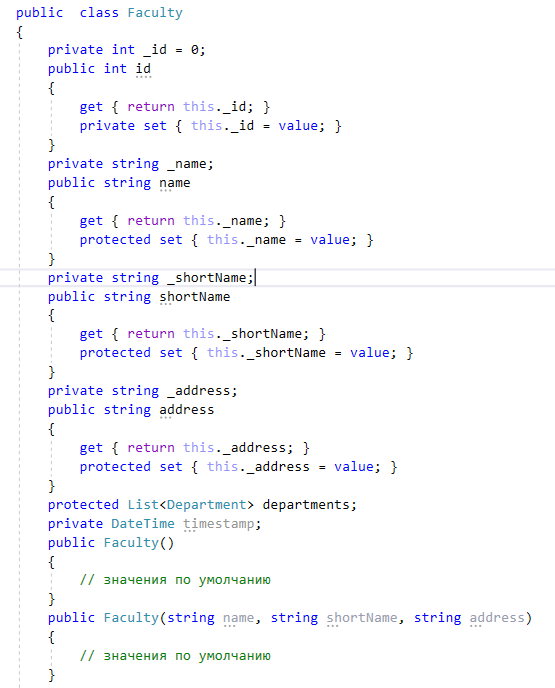
**Инкапсуляция**

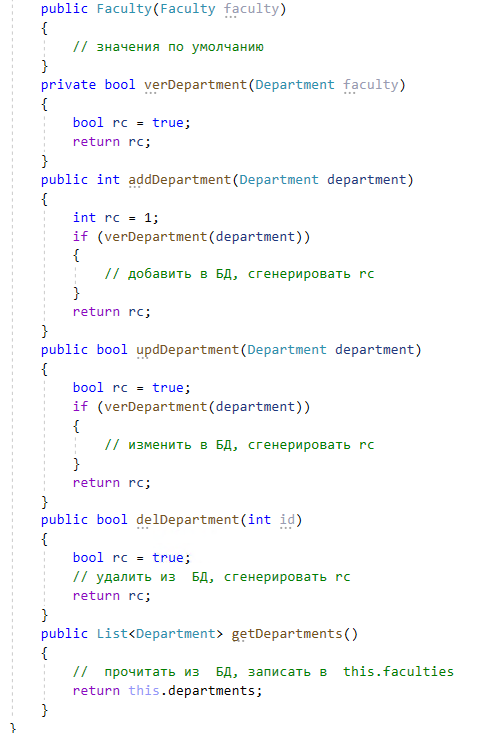
1. **Инкапсуляция:** способность объектов скрывать часть своего состояния и поведения.
2. **Инкапсуляция/проблема**: как сделать так, чтобы пользователь класса/объекта имел доступ только к тем свойствам, полям и методам, которые необходимы пользователям библиотеки классов?
3. **Инкапсуляция/мотивация**: контроль доступа к свойствам, состоянию и методам объекта.
4. **Инкапсуляция/UML**:



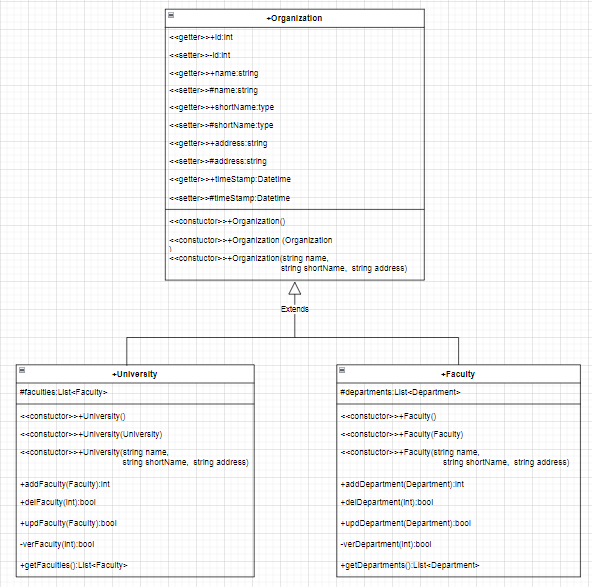
1. **Инкапсуляция/C#**



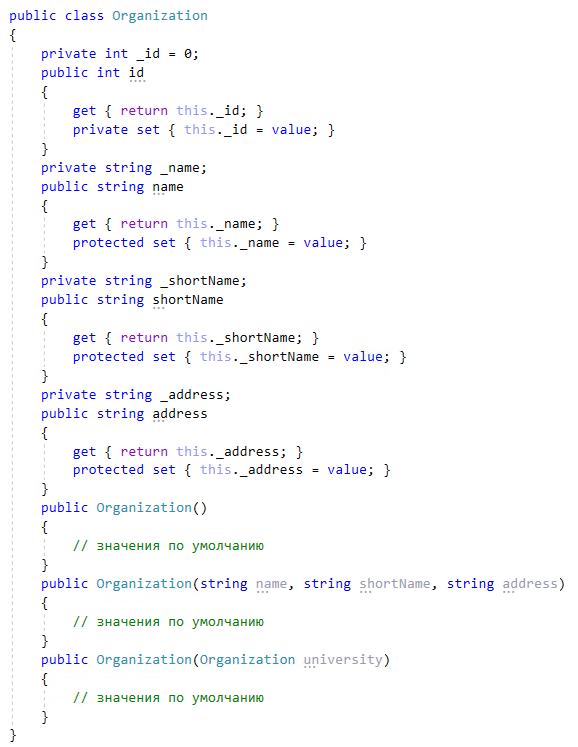




1. **Инкапсуляция/JS**:
2. **Инкапсуляция/особенности**: нет, это базовый паттерн
3. **Инкапсуляция/связь**: нет, это базовый паттерн
4. **Наследование:** повторное использование кода, позволяющее на основе одного класса, создать другой; принцип *ЕСТЬ*.
5. **Наследование/проблема**: много классов с одинаковыми свойствами и методами и между ними можно установить отношение *ЕСТЬ*.
6. **Наследование/мотивация**: сокращение кода.
7. **Наследование/UML**:

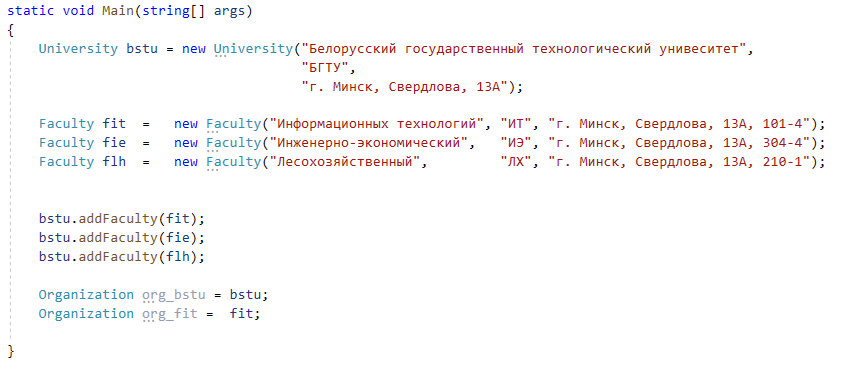


1. **Наследование/C#**









1. **Наследование/JS**:
2. **Наследование/особенности**: нет, это базовый паттерн
3. **Наследование/связь**: нет, это базовый паттерн
4. **Полиморфизм:** способность кода выбирать различные реализации в зависимости от контекста исполнения
5. **Полиморфизм/проблема**: один и тот же тип должен вести себя по разному в зависимости от контекста исполнения

1. **Полиморфизм/мотивация**: сокращение кода
2. **Полиморфизм/UML**:
3. **Полиморфизм/C#**
4. **Полиморфизм/JS**:
5. **Полиморфизм особенности**: нет, это базовый паттерн
6. **Полиморфизм связь**: нет, это базовый паттерн
7. Gg
8. Gg
9. Gg