Коллоквиум

Паттерны проектирования, ИСиТ-2

1. Перечислите основные парадигмы объектно-ориентированного программирования(ООП).

Инкапсуляция, наследование, полиморфизм, абстракция.

1. Поясните парадигму ООП «абстракция».

Абстракция подразумевает разделение и независимое рассмотрение интерфейса и реализации

абстракция - уровень описания/представления модели чего либо

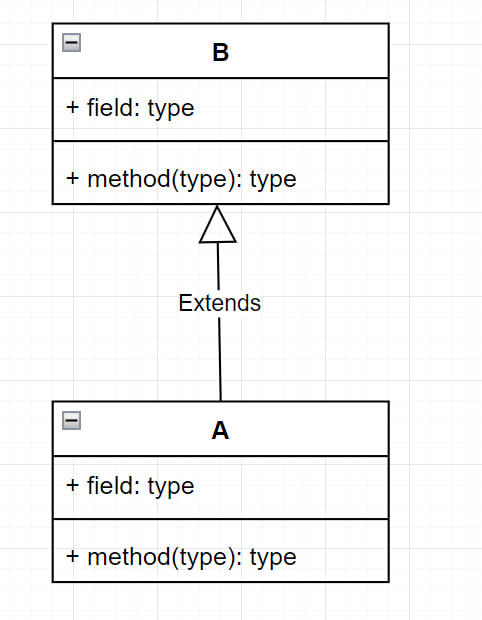
1. Поясните принцип «Бритва Оккама».

**Бри́тва** **О́ккама**  — методологический **принцип**, в кратком виде гласящий: «Не следует множить сущее без необходимости». Не создавайте ненужных сущностей без необходимости. Можно создавать несколько интерфейсов.

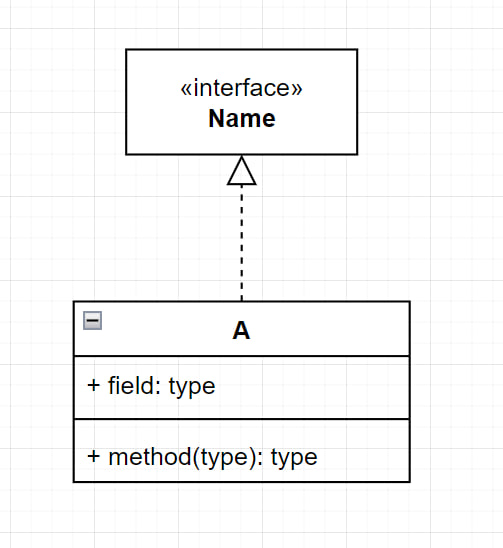
1. Поясните понятие ООП «интерфейс».

перечень сигнатур методов, предназначенных для реализации.

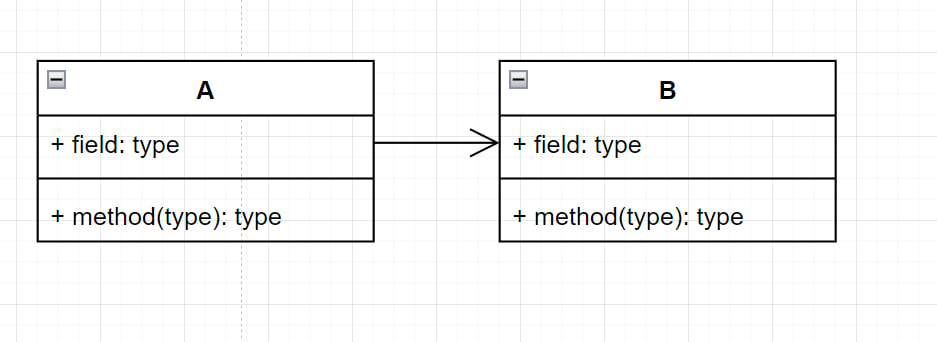
1. Нарисуйте UML-диаграмму: класс A является производным от класса В.



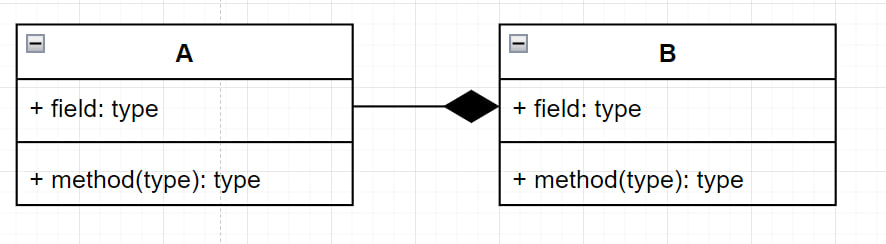
1. Нарисуйте UML-диаграмму: класс А реализует интерфейс I.



1. Нарисуйте UML-диаграмму: класс А использует тип класса B.



1. Нарисуйте UML-диаграмму: объект класса А содержит объект класса B.



1. Переведите с английского языка слово pattern.

Узор, шаблон, схема.

1. Дайте простое определение (максимум 4 слова) понятия паттерна.

схема решения часто встречаемой задачи, прием программирования.

1. Что такое «реализация паттерна»?

разработка программного кода в соответствии с паттерном;

1. Поясните понятие «идиома».

**идиомы** – низкоуровневые паттерны (зависят от языка программирования);

1. Поясните понятие «фреймворк».

Фреймворк - набор библиотек и инструментов, предназначенных для разработки ПО.

1. Поясните (расшифруйте) аббревиатуру SOLID.

SOLID - Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion.

1. Дайте простое определение (максимум 5 слов) SOLID.

SOLID - пять основных принципов проектирования в ООП.

1. Поясните кратко Single Responsibility Principe (максимум 7 слов).

Каждый класс должен иметь только одну зону ответственности.

1. Поясните кратко Open/Close Principe (максимум 7 слов).

Классы должны быть открыты для расширения, но закрыты для изменения.

1. Поясните кратко Liskov Substitution Principe (LSP) (максимум 7 слов).

При подстановке производных классов вместо базовых код должен работать

1. Поясните кратко 1 правило LSP.

типы параметров одноименных метода производного класса должны иметь такой же тип как у базового или более абстрактный;

1. Поясните кратко 2 правило LSP.

тип возвращаемого методом значения должен быть такой же как у базового или производным от возвращаемого типа методом базового класса;

1. Поясните кратко 3 правило LSP.

производный метод не должен выбрасывать исключения не свойственные базовому(можно выбрасывать производные);

1. Поясните кратко 4 правило LSP.

производный метод не должен ужесточать предусловия;

1. Поясните кратко 5 правило LSP.

метод не должен ослаблять постусловия;

1. Поясните кратко 6 правило LSP.

производный класс не изменяет инвариантов (определяющих свойств и методов), в идеале: производный класс только добавляет новые свойства или методы;

1. Поясните кратко 7 правило LSP.

производный класс не должен изменять приватные поля базового класса (рефлексия).

1. Поясните кратко Interface Segregation Principle (максимум 7 слов).

Не должно быть реализаций интерфейсов с пустой реализацией, они не должны быть слишком большими

1. Поясните кратко Dependency Inversion Principle максимум 7 слов).

модули верхних уровней не должны зависеть от модулей нижних уровней, а оба типа модулей должны зависеть от абстракций; сами абстракции не должны зависеть от деталей, а вот детали должны зависеть от абстракций.

1. Перечислите наименование 3 группы паттернов. Укажите количество паттернов в каждой группе.

Порождающие (5), структурные (7), поведенческие (11) паттерны.

1. Библиотека классов реализует Abstract Factory. Что библиотека предоставляет программисту-пользователю? Что разрабатывает программист-разработчик библиотеки.

программист-пользователь получает интерфейс для создания семейств связанных объектов, программист-разработчик реализует конкретные фабрики.

1. Библиотека классов реализует Facade. Что библиотека предоставляет программисту-пользователю? Что разрабатывает программист-разработчик библиотеки?

Facade: программист-пользователь получает простой интерфейс к сложной системе классов, библиотеке или фреймворку., программист-разработчик реализует упрощенный интерфейс, инкапсулируя сложную логику.

1. Библиотека классов реализует Decrator. Что библиотека предоставляет программисту-пользователю? Что разрабатывает программист-разработчик библиотеки?

Позволяет программисту-пользователю динамически добавлять объектам новую функциональность, оборачивая их в полезные «обёртки»

программист-разработчик реализует декораторы, расширяющие функциональность базовых объектов.

1. Библиотека классов реализует Observer. Что библиотека предоставляет программисту-пользователю? Что разрабатывает программист-разработчик библиотеки?

программисту-пользователь создаёт механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах.

программист-разработчик реализует объекты "наблюдателей" и "субъектов".