Ответы на Коллоквиум

Паттерны проектирования, ИСиТ-2

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

Абстракция, Инкапсуляция, Наследование, Полиморфизм.

1. **Поясните парадигму ООП «абстракция».**

Выделение важных характеристик объекта и игнорирование прочих.

1. **Поясните принцип «Бритва Оккама».**

Не стоит множить сущее без необходимости.

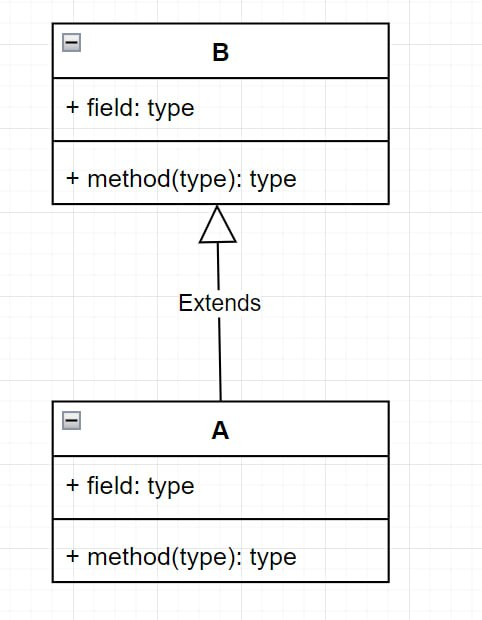
1. **Поясните понятие ООП «интерфейс».**

Перечень сигнатур методов, предназначенных для реализации;

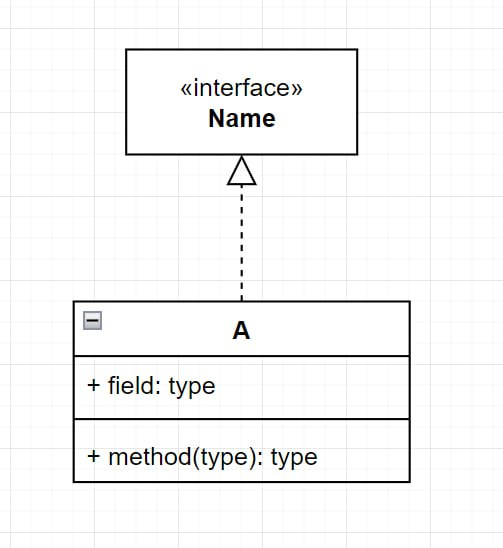
Контракт с фреймворком;

Спецификация фреймворка.

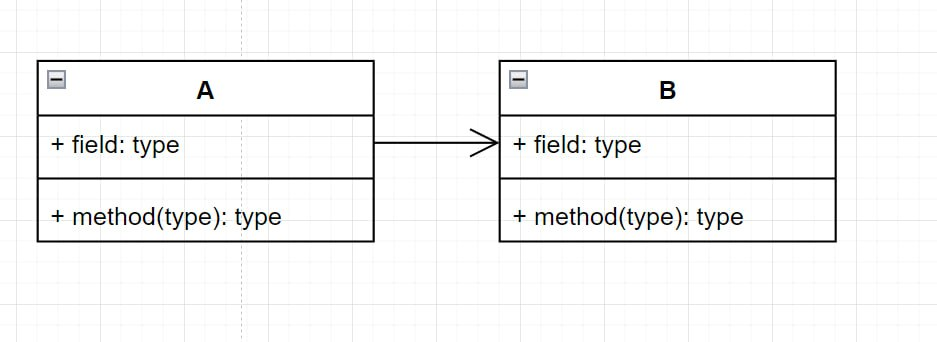
1. **Нарисуйте UML-диаграмму: класс A является производным от класса В.**



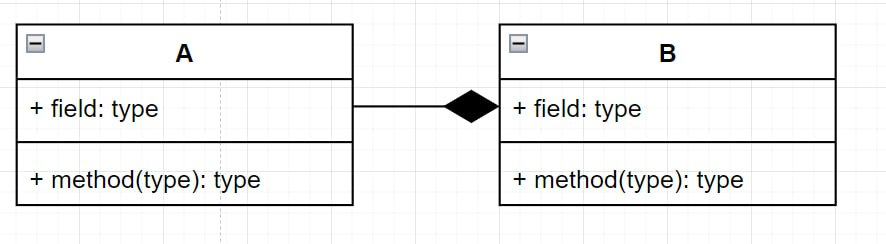
1. **Нарисуйте UML-диаграмму: класс А реализует интерфейс I.**



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.**

Шаблон, узор, схема.

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

Схема решения часто встречаемой задачи.

1. **Что такое «реализация паттерна»?**

Разработка программного кода в соответствии с паттерном.

1. **Поясните понятие «идиома».**

Специфичный для языка программирования паттерн.

1. **Поясните понятие «фреймворк».**

Совокупность библиотек и инструментов для разработки приложений.

1. **Поясните (расшифруйте) аббревиатуру SOLID.**

Single Responsibility Principle,

Open/Closed Principle,

Liskov Substitution Principle,

Interface Segregation Principle,

Dependency Inversion Principle,

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

Основные принципы объектно-ориентированного проектирования и программирования.

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

Класс должен решать одну задачу.

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

Классы должны быть открыты к расширению, но закрыты к изменению.

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

Подклассы могут заменять базовые классы.

1. **Поясните кратко 1 правило LSP.**

Типы параметров должны иметь такой же тип или более абстрактный.

1. **Поясните кратко 2 правило LSP.**

Тип возвращаемого значения должен быть таким же или производным.

1. **Поясните кратко 3 правило LSP.**

Новые исключения в подклассах должны быть подмножеством исключений базового класса.

1. **Поясните кратко 4 правило LSP.**

Производный метод не должен ужесточать предусловия.

1. **Поясните кратко 5 правило LSP.**

Производный метод не должен ослаблять постусловия.

1. **Поясните кратко 6 правило LSP.**

Подклассы должны расширять, а не заменять свойства базового класса.

1. **Поясните кратко 7 правило LSP.**

Производный класс не должен изменять приватные поля базового класса.

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

Множество небольших специализированных интерфейсов.

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

Зависимость от абстракций, а не от конкретных реализаций.

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

Порождающие (5), Структурные (7), Поведенческие (10).

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?**

Программист-пользователь получает упрощенный интерфейс для работы со сложной подсистемой, программист-разработчик реализует упрощенный интерфейс, инкапсулируя сложную логику.

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

Программист-пользователь получает возможность динамически добавлять новые обязанности к объектам, программист-разработчик реализует декораторы, расширяющие функциональность базовых объектов.

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

Программист-пользователь получает возможность создания системы с механизмом подписки на события, программист-разработчик реализует объекты "наблюдателей" и "субъектов".