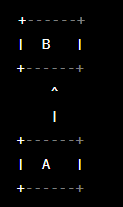
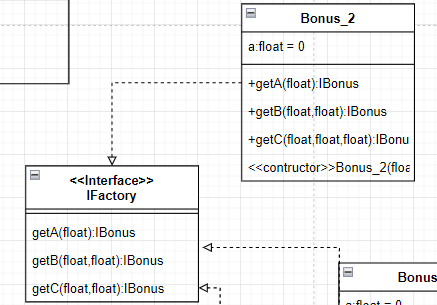
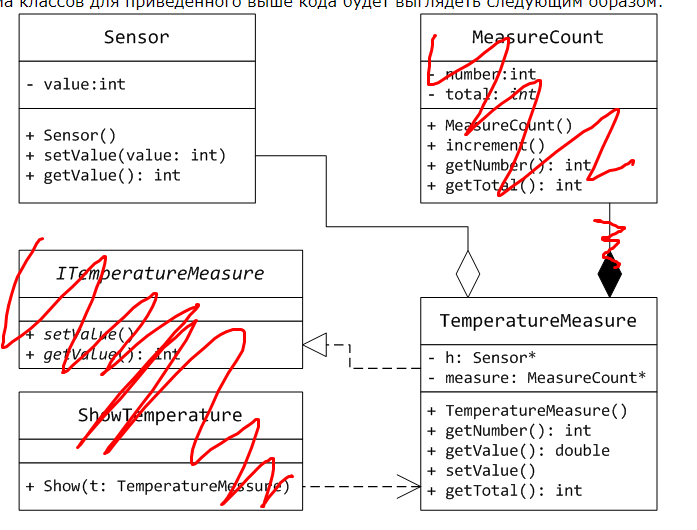
1. Основные парадигмы объектно-ориентированного программирования (ООП) включают инкапсуляцию, наследование и полиморфизм.
2. Парадигма ООП "абстракция" означает выделение существенных характеристик объекта и скрытие всех остальных деталей реализации.
3. Принцип "Бритва Оккама" гласит, что если есть несколько возможных объяснений явления, то следует выбрать самое простое, которое требует наименьшего числа допущений.
4. Интерфейс в ООП - это абстрактный тип данных, который определяет набор методов или свойств, которые должны быть реализованы в классе, который использует данный интерфейс.
5. Нарисуйте UML-диаграмму: класс A является производным от класса В.



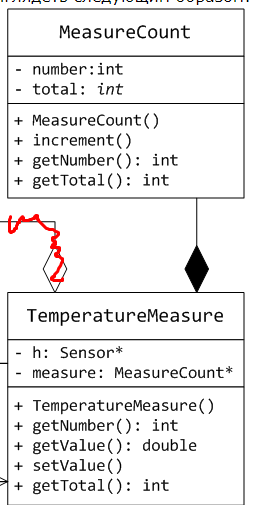
1. Нарисуйте UML-диаграмму: класс А реализует интерфейс I.



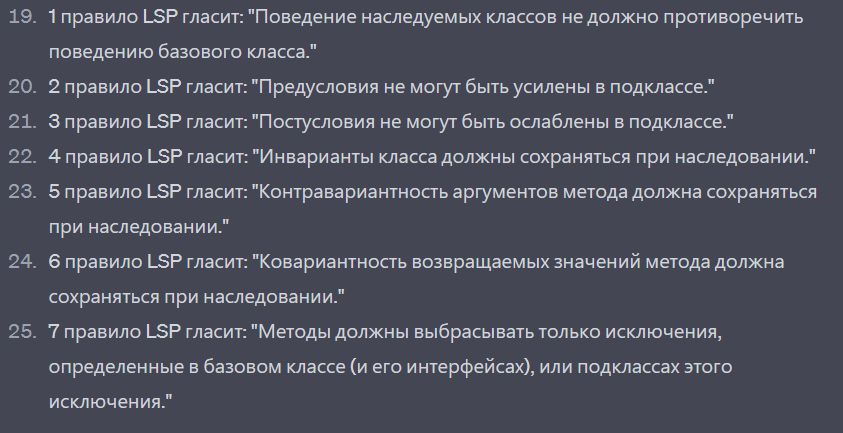
1. Нарисуйте UML-диаграмму: класс А использует тип класса B.



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



1. Перевод слова pattern с английского означает "шаблон".
2. Паттерн - это повторяемое решение проблемы, которое может быть применено в разных ситуациях и контекстах.
3. Реализация паттерна - это конкретная реализация шаблона в программном коде.
4. Идиома - это типичный способ использования языка программирования, который используется программистами для решения определенных задач.
5. Фреймворк - это набор соглашений, правил и инструментов, который помогает разработчикам создавать приложения более эффективно и быстро.
6. Аббревиатура SOLID расшифровывается как "пять основных принципов объектно-ориентированного проектирования".
7. пять основных принципов объектно-ориентированного проектирования. Это набор принципов дизайна ПО.
8. Каждый класс должен иметь одну ответственность. Класс должен иметь только одну причину для изменения.
9. Программные сущности должны быть открыты для расширения, но закрыты для модификации.
10. Объекты должны быть заменяемыми на экземпляры их подтипов.
11. Подтипы не должны изменять поведение базового типа.
12. Предусловия не могут быть усилены подтипом.
13. Постусловия не могут быть ослаблены подтипом.
14. Инварианты типа должны быть сохранены подтипом.
15. Методы должны вызывать те же исключения или не выбрасывать их вовсе, что и их базовые реализации.
16. Не должно быть никаких новых исключений, объявленных методом подтипа.
17. Контракты типа должны быть соблюдены подтипом.



1. Клиенты не должны зависеть от методов, которые они не используют.
2. Зависимости должны быть на абстракциях, а не на конкретных реализациях.
3. Порождающие (5), структурные (7), поведенческие (11).
4. Предоставляет абстрактные фабрики. Разрабатывает фабрики конкретных объектов. Паттерн Abstract Factory (Абстрактная Фабрика) является порождающим паттерном проектирования, который позволяет создавать семейства объектов связанных между собой или взаимозависимых объектов, без указания их конкретных классов.

Abstract Factory предоставляет интерфейс для создания объектов, который инкапсулирует создание объектов конкретных классов.

Клиентский код использует этот интерфейс для создания семейств объектов, не привязываясь к конкретным классам объектов.

Таким образом, паттерн позволяет создавать объекты семейства, соответствующие определенному интерфейсу, но не зависящие от конкретных классов, что обеспечивает более гибкую архитектуру и упрощает замену семейства объектов в программе.

1. Facade: Предоставляет унифицированный интерфейс для доступа к подсистеме. Разрабатывает интерфейс к сложной системе.
2. Decrator: Предоставляет возможность добавлять функциональность объекту динамически. Разрабатывает классы-обертки, которые реализуют дополнительную функциональность.
3. Observer: Предоставляет возможность объектам реагировать на изменения в других объектах. Разрабатывает наблюдателей (observer), которые реагируют на изменения в субъекте (subject).