

C++

Урок 26



Сложное ПО



Что такое сложное ПО?

Сложное ПО

Комплексность

- внутреннее устройство из множества составных частей;

Сложное ПО

Комплексность

- внутреннее устройство из множества составных частей;

Сопровождение

- способность архитектуры быть устойчивыми к изменениям (кода);

Сложное ПО

Комплексность

- внутреннее устройство из множества составных частей;

Сопровождение

- способность архитектуры быть устойчивыми к изменениям (кода);

Рефакторинг — это контролируемый процесс улучшения кода, без написания новой функциональности.

Результат рефакторинга — это чистый код и простой дизайн.

Сложное ПО

Комплексность

- внутреннее устройство из множества составных частей;

Сопровождение

- способность архитектуры быть устойчивыми к изменениям (кода);

Рефакторинг — это контролируемый процесс улучшения кода, без написания новой функциональности.
Результат рефакторинга — это чистый код и простой дизайн.

Работа в команде

- невозможность разработать и поддерживать сложное ПО в одиночку за приемлемое время;

Сложное ПО

Комплексность

- внутреннее устройство из множества составных частей;

Сопровождение

- способность архитектуры быть устойчивыми к изменениям (кода);

Рефакторинг — это контролируемый процесс улучшения кода, без написания новой функциональности.
Результат рефакторинга — это чистый код и простой дизайн.

Работа в команде

- невозможность разработать и поддерживать сложное ПО в одиночку за приемлемое время;

Распределенные вычисления

- обеспечение производительности вычислений.

A hand-drawn blue oval frame with a slightly irregular, sketchy border, centered on the page. It contains the text 'Проблемы и решение'.

Проблемы и решение

Проблемы

При **разработке** сложного продукта появляются некоторые **проблемы**

- Нечеткие или меняющиеся требования;
- Большое количество взаимодействующих плохо разделяемых и/или нечётких понятий предметной области;
- Необходимость периодического внесения изменений в исходный код ПО ;
- Большое количество разработчиков.

Решение

Для того, чтобы все требования заказчика и исполнителя были выполнены проект должен учитывать:

- Написание **адаптивной** архитектуры, **адаптивного** кода.
- Использовать **методологии**, позволяющие повысить качество определения и контроля требований на всем жизненном цикле разработки (Enterprise: MSF, RUP и др.) или сокращающие время итерации жизненного цикла, привлекая заказчика к проектированию и верификации (Agile: SCRUM, XP)
- Применение **закона Конвея** и применение обратного закона Конвея.
- Автоматизация коллективной разработки (**CI/CD**).
- Использование инструментов, поддерживающие и объединяющие эти и другие решения (автоматические тесты, автоматическое документирование и др.).

Решение

Для того, чтобы все требования заказчика и исполнителя были выполнены проект должен учитывать:

- Написание **адаптивной** архитектуры, **адаптивного** кода.
- Использовать **методологии**, позволяющие повысить качество определения и контроля требований на всем жизненном цикле разработки (Enterprise: MSF, RUP и др.) или сокращающие время итерации жизненного цикла, привлекая заказчика к проектированию и верификации (Agile: SCRUM, XP)
- Применение **закона Конвея** и применение обратного закона Конвея.
- Автоматизация коллективной разработки (**CI/CD**).
- Использование инструментов, поддерживающие и объединяющие эти и другие решения (автоматические тесты, автоматическое документирование и др.).

Все эти решения можно реализовать на языке C++.



Черное и белое

Язык C++ имеет много сложных конструкций, особенно с C++11.

“Если разработчик хочет выстрелить себе в ногу, ЯП не будет ему мешать”



Тёмная сторона	Светлая сторона
Указатели	Интеллектуальные указатели STL C++11: std::unique_ptr , std::shared_ptr , std::weak_ptr ; передача по значению, ссылки
new / delete	Интеллектуальные указатели STL C++11, методы std::make_unique и std::make_shared
Ссылки	Интеллектуальные указатели STL C++11 (частично), передача по значению, семантика перемещения, ссылки
Массивы	Контейнерные классы STL C++11, ссылки
Оператор goto	Не используем.
Глобальные переменные	Объекты, передаваемые в параметрах, паттерн Одиночка
Макросы	Используем спецификаторы const и constexpr при объявлении переменной
Макрос NULL	Не используем. Ключевое слово nullptr в C++11



ООП



Что такое класс?



Что такое объект?



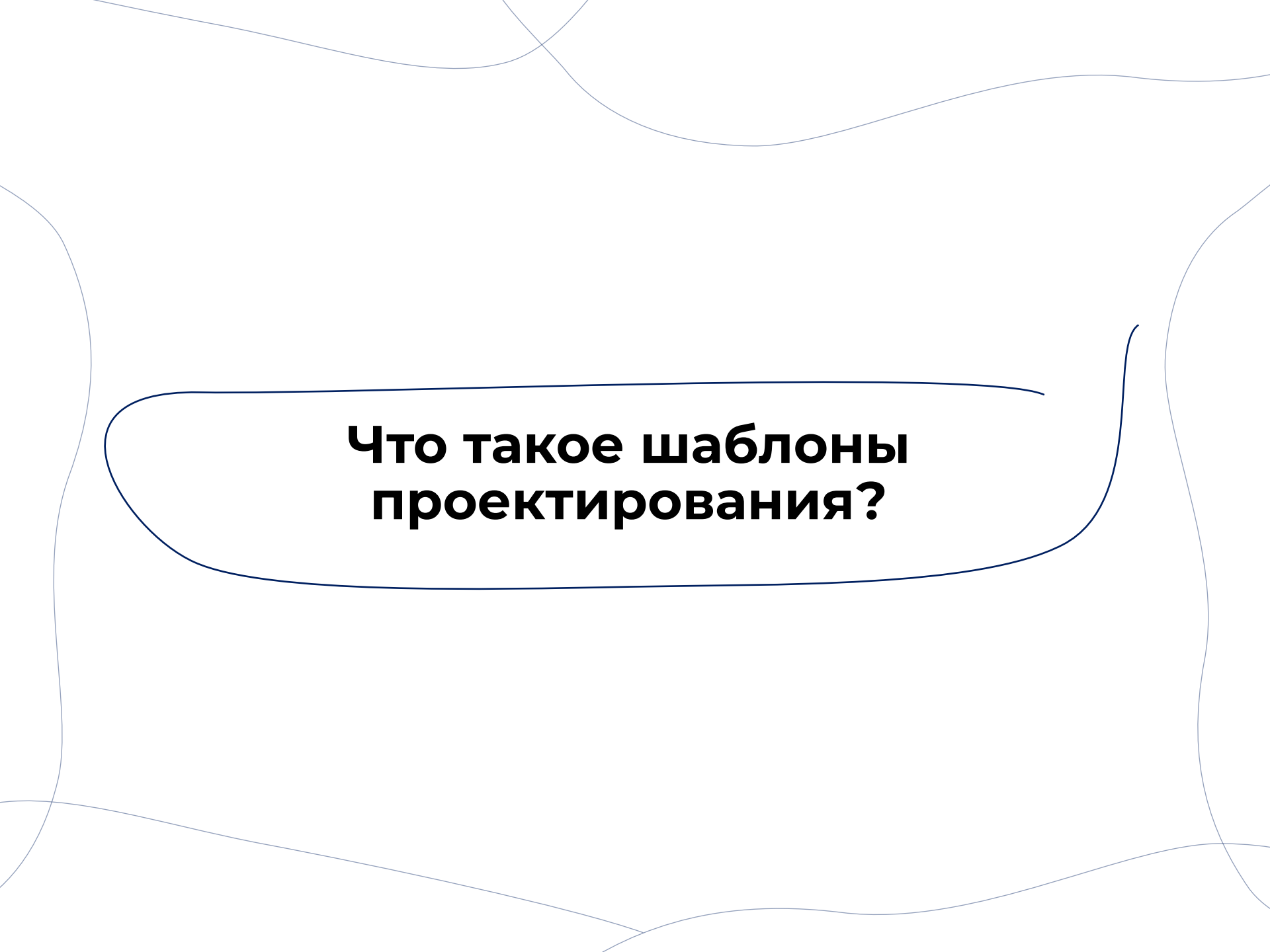
Что такое отношение?

ООП

Основные понятия



Шаблоны проектирования



**Что такое шаблоны
проектирования?**



Где они применяются?

**Их можно использовать только
на C++?**

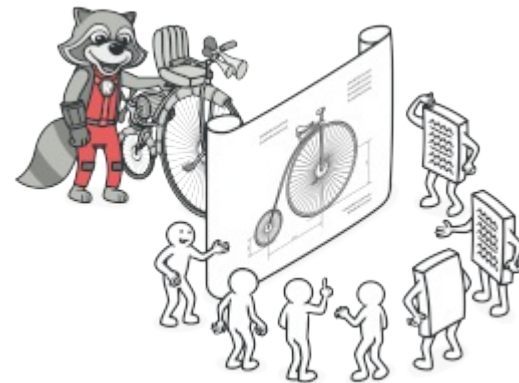
Шаблоны Проектирования

Шаблон проектирования или паттерн в разработке программного обеспечения — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Используются при проектировании систем, а значит **не привязаны** к языку программирования.



Шаблоны Проектирования



Другими словами: разработка ПО идет уже не первый год и путем проб и ошибок выявлены некие конструкции, которые подходят для той или иной проблемы.

В данной ситуации возникает вопрос. “Зачем создавать велосипед заново?”.

Паттерны помогают быстро решить задачу, подогнав ее под один из паттернов проектирования, тем самым экономится время и силы разработчика.

Шаблоны Проектирования

Виды ШП:

Низкоуровневые - идиомы программирования.

Шаблоны проектирования - уровень взаимодействия объектов программы

Архитектурные шаблоны - охватывают архитектуру ПО

Шаблоны Проектирования

Важно понимать:

- ШП не являются пазлами — потребуется адаптация для того, чтобы применить их в свой проект.
- ШП не решения на все случаи — могут существовать более простые / оптимальные структуры

Шаблоны Проектирования

Классификация паттернов:

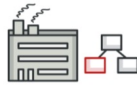
Порождающие паттерны беспокоятся о гибком создании объектов без внесения в программу лишних зависимостей.

Структурные паттерны показывают различные способы построения связей между объектами.

Поведенческие паттерны заботятся об эффективной коммуникации между объектами.

Шаблоны Проектирования

Порождающие паттерны беспокоятся о гибком создании объектов без внесения в программу лишних зависимостей.



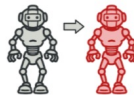
Фабричный метод
Factory Method



Абстрактная фабрика
Abstract Factory



Строитель
Builder



Прототип
Prototype



Одиночка
Singleton




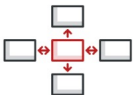
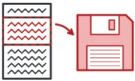
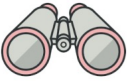




Шаблоны Проектирования

Структурные паттерны показывают различные способы построения связей между объектами.



Шаблоны Проектирования

Поведенческие паттерны заботятся об эффективной коммуникации между объектами.

 <p>Цепочка обязанностей Chain of Responsibility</p>	 <p>Команда Command</p>	 <p>Итератор Iterator</p>	 <p>Посредник Mediator</p>
 <p>Снимок Memento</p>	 <p>Наблюдатель Observer</p>	 <p>Состояние State</p>	 <p>Стратегия Strategy</p>
 <p>Шаблонный метод Template Method</p>	 <p>Посетитель Visitor</p>		