

# ВВЕДЕНИЕ В ОБЪЕКТНО ОРИЕНТИРОВАННЫЙ ДИЗАЙН С JAVA

---

Принципы и шаблоны объектно-ориентированного проектирования

18+

ТИМУР МАШНИН

Тимур Машнин

**Введение в объектно-  
ориентированный дизайн с Java**

«Автор»

2022

**Машнин Т.**

Введение в объектно-ориентированный дизайн с Java /  
Т. Машнин — «Автор», 2022

Эта книга ориентирована на тех, кто уже знаком с языком программирования Java и хотел бы углубить свои знания и изучить объектно-ориентированный анализ и проектирование программного обеспечения. Вы познакомитесь с основными принципами и паттернами объектно-ориентированного дизайна, используемыми при разработке программных систем Java. Вы научитесь моделировать системы Java с помощью UML диаграмм, познакомитесь с основными понятиями и принципами объектно-ориентированного подхода, изучите порождающие, структурные и поведенческие шаблоны проектирования. Вы узнаете, как создавать модульное, гибкое и многоразовое программное обеспечение, применяя объектно-ориентированные принципы и шаблоны проектирования.

# Содержание

Введение	6
Вопросы	14
Основные понятия	16
Принципы ООД (Объектно-ориентированного дизайна)	28
Принцип Абстракции в UML	39
Принцип Инкапсуляции в UML	43
Принцип Декомпозиции в UML	46
Принцип Обобщения в UML	52
Вопросы	59
Связанность и когезия	63
Разделение ответственостей	67
Скрытие информации	70
Концептуальная целостность	72
Моделирование поведения. UML диаграммы последовательности	74
Задание	78
UML диаграмма состояний	80
Задание	83
Вопросы	84
Паттерны проектирования	87
Factory Method Pattern	92
Abstract Factory Pattern	100
Singleton Pattern	104
Prototype Pattern	107
Builder Design Pattern	109
Structural design patterns. Adapter Pattern	111
Bridge Pattern	115
Composite Pattern	118
Decorator Pattern	126
Facade Pattern	135
Flyweight Pattern	141
Proxy Pattern	144
Задание	148
Задание	150
Вопросы	155
Поведенческие шаблоны проектирования. Chain Of Responsibility Pattern	158
Command Pattern	162
Interpreter Pattern	168
Iterator Pattern	170
Mediator Pattern	172
Memento Pattern	174
Observer Pattern	176
State Pattern	181
Strategy Pattern	188
Template Pattern	190
Visitor Pattern	195

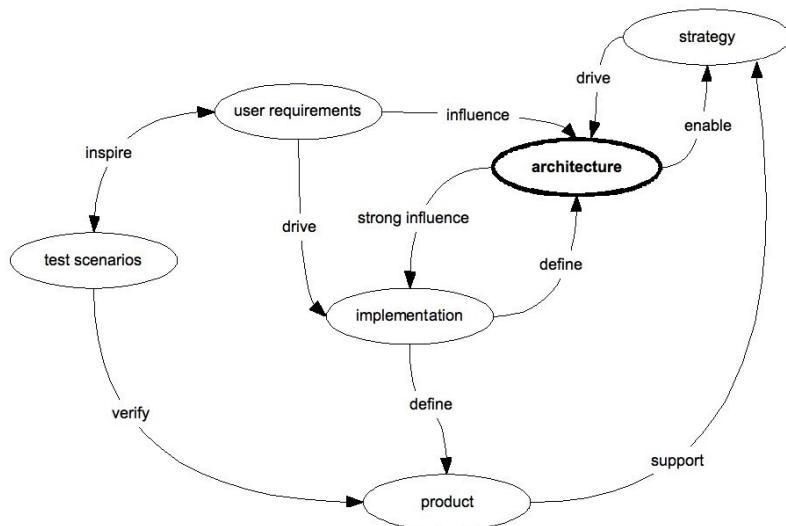
Задание	197
Вопросы	199
MVC Pattern	204
Задание	209
Принципы проектирования. Принцип подстановки Лисков	210
Открыто-закрытый принцип	214
Инверсии зависимостей	217
Принцип композиции объектов	222
Разделение интерфейса	225
Принцип наименьшего знания	227
Анти-паттерны	232
Вопросы	243

# Тимур Машнин

## Введение в объектно-ориентированный дизайн с Java

### Введение

Что такое дизайн и архитектура программного обеспечения?



И как это может улучшить программные продукты?

Давайте рассмотрим сценарий.

Предположим, вы присоединяетесь к проекту, который находится уже в разработке некоторое время.

Вы смотрите на код проекта, и вы не можете понять, для чего предназначены эти куски кода, так как он плохо организован, и проектной документации не существует.

Вы даже не знаете, с чего начать.

Это все признаки того, что проект не был хорошо разработан с самого начала.

Или, допустим, вы сейчас работаете над персональным проектом.

Когда вы начинали, вы не были уверены, какая конкретно функциональность должна быть реализована, но тем не менее вы начали кодирование.

Для вас не имело значения, что код будет неорганизованным, потому что вы были единственным, кто работал над проектом.

И предположим, вы придумали замечательную новую функцию для своего продукта, но при ее реализации вы нарушили программу в других местах. И теперь вы должны все исправлять во многих местах своего кода.

Чего не произошло бы, если бы вы правильно и хорошо с самого начала спроектировали бы свой продукт.

И такие сценарии довольно часто встречаются в индустрии программного обеспечения, что показывает, почему дизайн и архитектура программного обеспечения так полезны.

В этом разделе вы узнаете, как применять принципы и паттерны дизайна и архитектуры для создания многоразовых и гибких программных систем. Вы узнаете, как задокументировать дизайн и архитектуру программного продукта визуально.

Итак, в чем разница между дизайном программного обеспечения и архитектурой программного обеспечения?

Роль дизайнера программного обеспечения или архитектора программного обеспечения может сильно отличаться от компании к компании.

На это влияют такие характеристики, как размер компании, объем проекта, опыт команды разработчиков, организационная структура и возраст компании.

В некоторых компаниях могут работать отдельные дизайнеры или архитекторы.

В других компаниях эта работа может выполняться членом или членами команды разработчиков.

И как правило, дизайнер программного обеспечения отвечает за определение программного решения для конкретной проблемы путем проектирования деталей отдельных компонентов и их обязанностей.

Дизайнер программного обеспечения отвечает за просмотр всей системы и выбор подходящих фреймворков, систем хранения данных, за решения и определения взаимодействий компонентов друг с другом.

И это подводит нас к основному различию между дизайном программного обеспечения и архитектором программного обеспечения.

Дизайнер программного обеспечения смотрит на аспекты системы более низкого уровня, тогда как архитектор программного обеспечения работает с более крупной картиной – с более высокими аспектами системы.

Подумайте об этом, как о проектировании здания.

Архитектор сосредотачивается на основных структурах и службах, в то время как дизайнер интерьера фокусируется на меньших пространствах внутри здания.

Дизайн программного обеспечения – это процесс превращения пожеланий и требований заказчика в рабочий код, который является стабильным и поддерживаемым в долгосрочной перспективе, и может быть развит и стать частью более крупной системы.

Архитектура программного обеспечения в первую очередь начинается с понимания того, в чем состоит бизнес-задача, которую должен решить клиент.

И основная задача заключается в том, чтобы выяснить, чего хочет клиент, тогда можно двигаться дальше.

Потому что, если вы понимаете задачу, вы можете начать думать о возможных решениях, а затем вы начинаете понимать, как будет выглядеть общее решение.

И архитектура важна, потому что, если вы ошибетесь, ваш проект не удастся.

Все просто.

Мы знаем это в области строительства, и мы это знаем в области программного обеспечения.

Архитектура – это понимание взаимосвязи между требованиями пользователя и способностью создавать систему, которая будет обеспечивать эти требования.

При этом самая большая проблема, с которой мы сталкиваемся, – это понимание проблемы клиента.

Что он действительно хочет сделать?

И во многих случаях клиент фактически не знает, что он хочет делать. Он приходит лишь с частичным пониманием, смутным чувством, что он может сделать что-то лучше.

И одна из первых задач состоит в том, чтобы помочь ему лучше понять его проблему.

Задача архитекторов программного обеспечения – это взаимодействие между продуктом, клиентом и инженерными командами.

Архитектор программного обеспечения похож на архитектора здания. И он отвечает за общую концептуальную целостность проекта.

Возможно, вы слышали термин «объектно-ориентированное моделирование».

Что это?

При решении задачи, объектно-ориентированное моделирование включает в себя практику представления ключевых понятий через объекты в вашем программном обеспечении.

И в зависимости от задачи многие концепции становятся отдельными объектами в программном обеспечении.

Подумайте об объектах.

Вокруг нас все объекты.

Почему вы должны использовать объекты для представления вещей в вашем коде?

Это способ держать ваш код организованным, гибким и многоразовым.

Объектный подход создает организованный код, содержа связанные детали и конкретные функции в разных, легко доступных местах.

Это создает гибкость, поскольку вы можете легко изменять детали модульным способом, не затрагивая остальную часть кода. Также вы можете повторно использовать разные части кода.

Давайте рассмотрим, как может выглядеть объектно-ориентированное моделирование.

Рассмотрим, например, помещение для семинаров.

Первый объект, который мы идентифицируем, является сама комната.

В комнате есть такие детали, как номер комнаты и места для сидения.

Также мы можем идентифицировать объекты, которые содержатся в этой комнате.

Существует множество физических объектов, такие как стул, стол, проектор и белая доска.

Каждый из этих физических объектов может быть представлен объектами в программном обеспечении.

И существуют конкретные детали, связанные с каждым объектом.

Проектор имеет характеристики, связанные с его производительностью, такие как разрешение и яркость.

И объекты также могут иметь индивидуальные обязанности или поведение.

Например, проектор принимает видеопоток и отображает изображение.

Вы можете думать о разработке программного обеспечения как о процессе, который берет задачу и создает решение с помощью программного обеспечения.

И как правило, это итеративный процесс, при этом каждая итерация берет набор требований для реализации и тестирования и в конечном итоге создается полное решение.

Многие разработчики стремятся сразу кодировать, несмотря на то, что не полностью понимают, что программируют в первую очередь.

И погружение прямо в работу по реализации является основной причиной отказа проекта.

Если вы не хотите, чтобы ваши проекты потерпели неудачу, найдите время, чтобы сформировать требования и создать дизайн.

Вы не можете сделать их идеальными, но их важность для эффективного создания хорошего программного обеспечения не следует упускать из виду.

Выявление требований требует активного изучения видения клиента, задавая вопросы о проблемах, которые клиент, возможно, не рассмотрел.

Помимо выявления конкретных потребностей, нужно спрашивать о возможных компромиссах, которые клиент может принять в решении.

С четким представлением о том, что вы пытаетесь выполнить, далее вы можете обратиться к шаблонам дизайна и диаграммам.