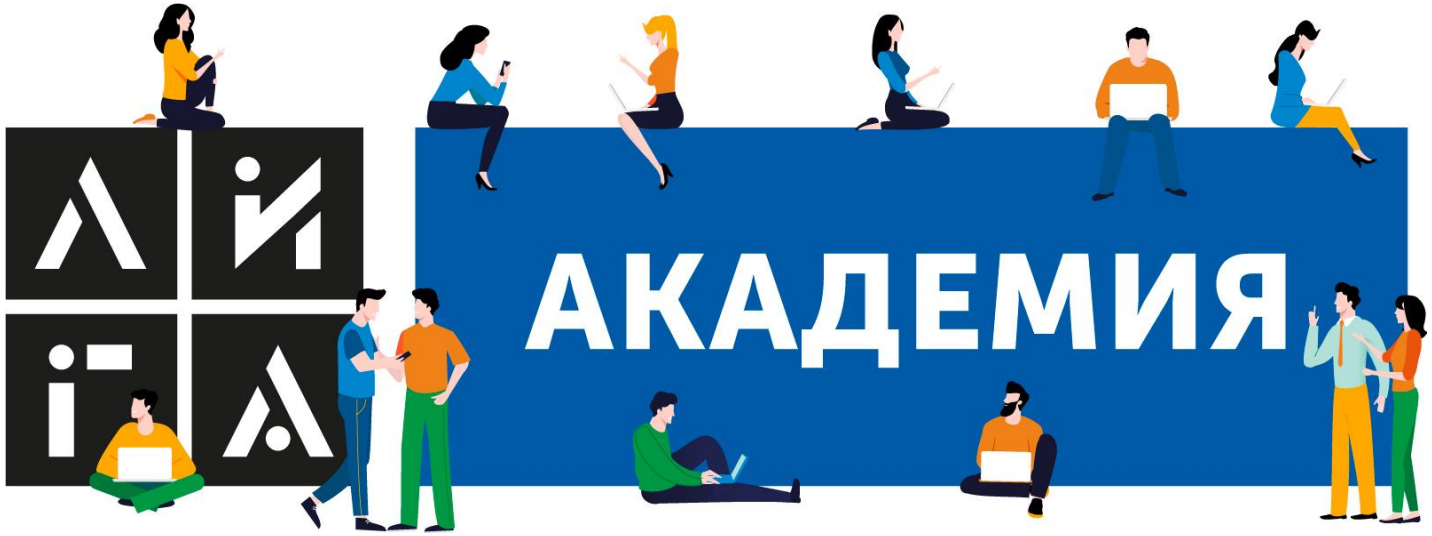

АКАДЕМИЯ

SOLID, DRY, KISS
и другие страшные слова

Зачем нужен CD/CD, Maven и Jenkins, Docker

01.Чистый код

Как эстетика помогает в работе

02.DRY, KISS, YAGNI

Главные базовые принципы

03.SOLID

Маркетинг от дядюшки Боба

04.CI/CD

Непрерывная поставка и развертывание

05.DOCKER

АКАДЕМИЯ

Чистый код

Как эстетика помогает в работе

Главный критерий качества кода

> Главным критерием качества кода следует признать его способность зарабатывать деньги

Кто-то из великих

Критерии качества кода

1. Код решает поставленную задачу
2. Код может быть легко модифицирован
3. Код работает быстро
4. Код понятен не только его автору
5. Предлагайте варианты

Главные книги по теме

Чистый код Роберта Мартина

- Книга начального уровня
- Много и справедливо критикуется
- Низкий порог входа, рекомендуется новичкам как вводная

Рефакторинг Мартина Фаулера

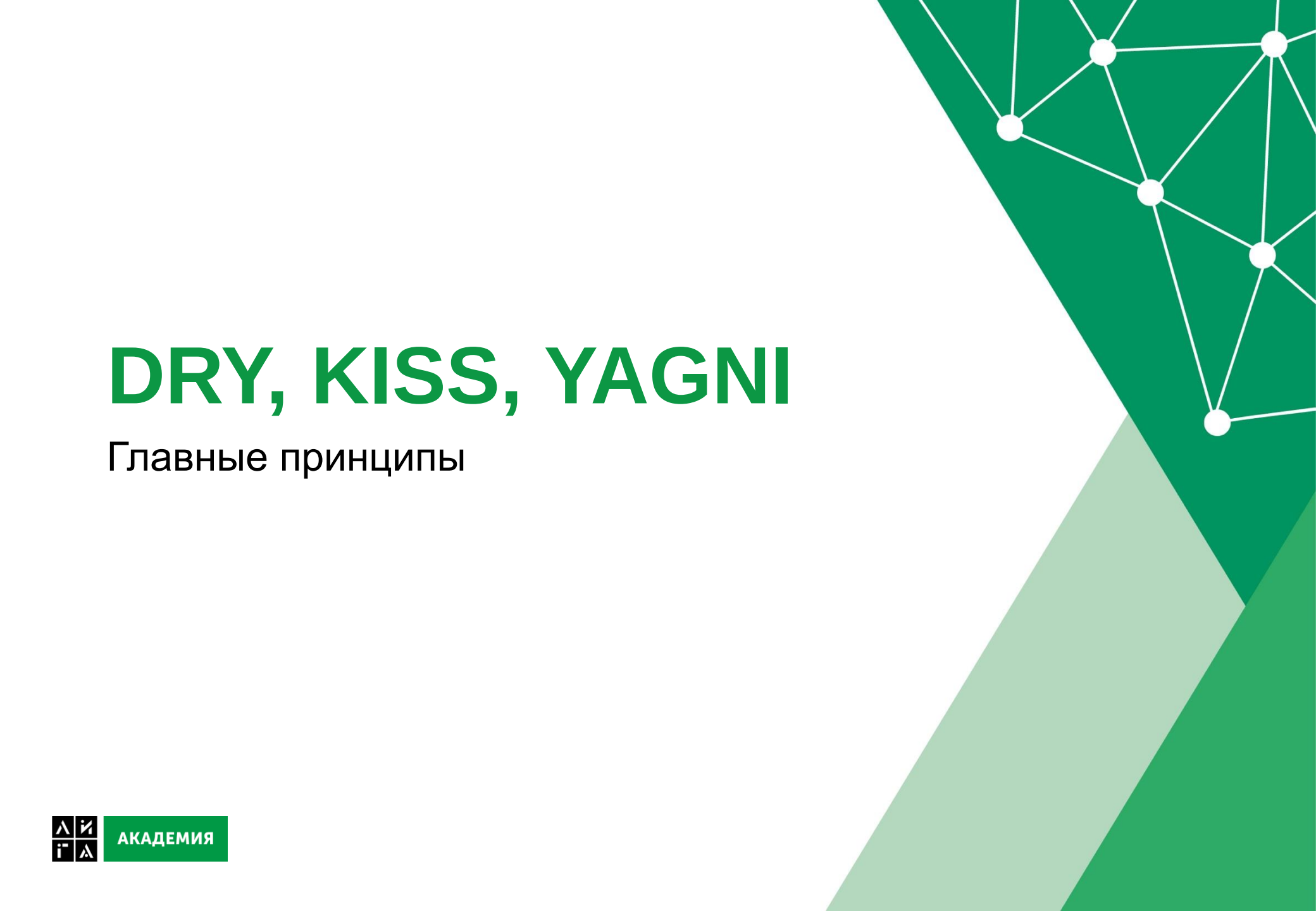
- Настольная книга каждого практикующего разработчика
- Указывает на конкретные недостатки кода, содержит рекомендации по улучшению или предотвращению некачественной разработки

Совершенный код Стивена Макконнелла

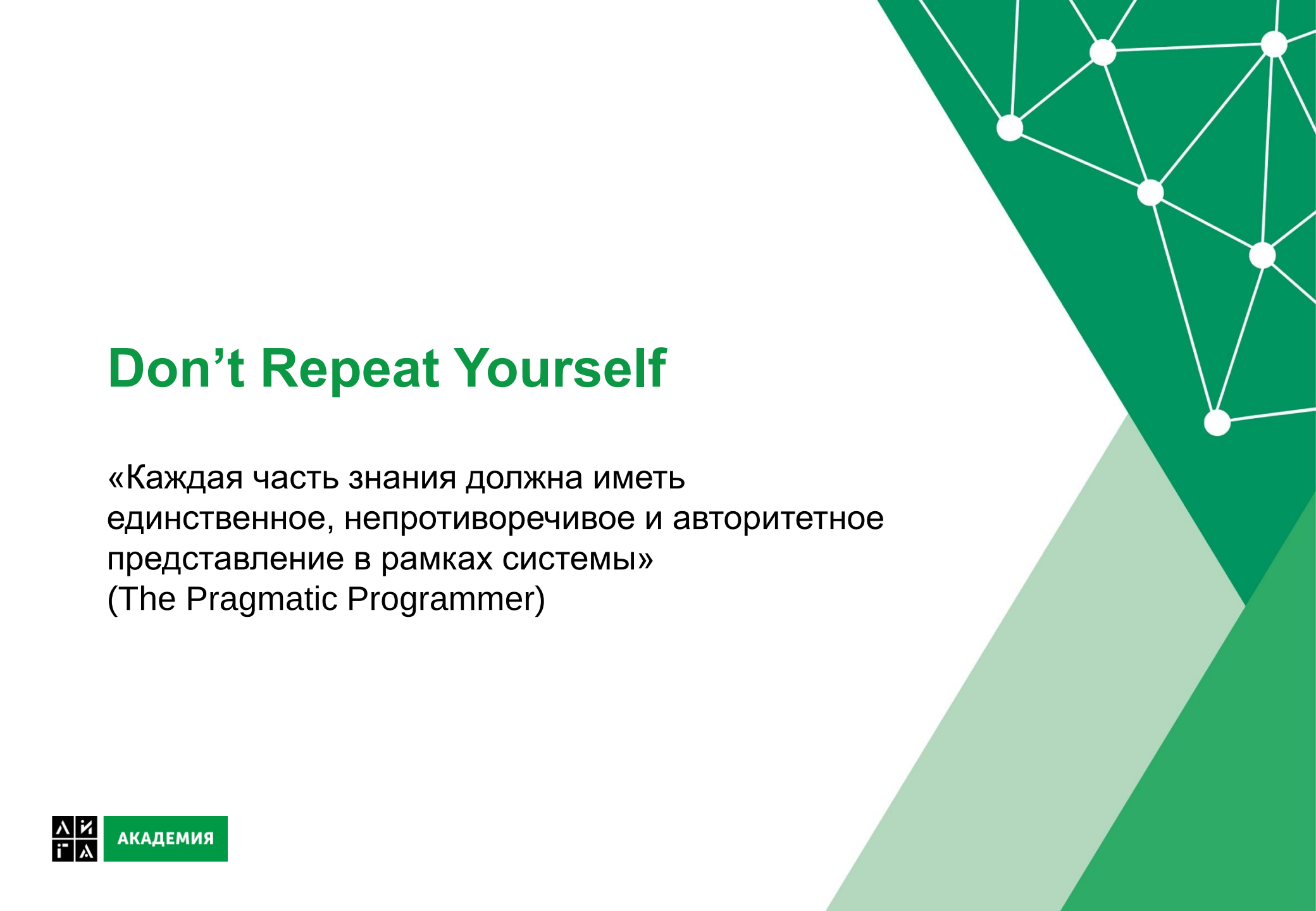
- Большая, детальная, подробная книга с указанием исследований в области разработки программного кода
- Требует определенного опыта, усидчивости и регулярного повторения

DRY, KISS, YAGNI

Главные принципы



АКАДЕМИЯ



Don’t Repeat Yourself

«Каждая часть знания должна иметь единственное, непротиворечивое и авторитетное представление в рамках системы»
(The Pragmatic Programmer)

АКАДЕМИЯ

Не повторяй одно и то же в разных местах

Если какой то алгоритм или кусок программного кода повторяется с незначительными изменениями во многих местах, его следует выделить в класс или метод

Keep It Simple Stupid

Не усложняй без необходимости

You aren’t gonna need it

Не пиши лишнего кода, в котором нет необходимости здесь и сейчас

SOLID

Маркетинг от дядюшки Боба

Single responsibility

Для каждого класса должно быть определено единственное назначение. Все ресурсы, необходимые для его осуществления, должны быть инкапсулированы в этот класс и подчинены только этой задаче.

Open–closed principle

Программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения.

Пример в IDEA

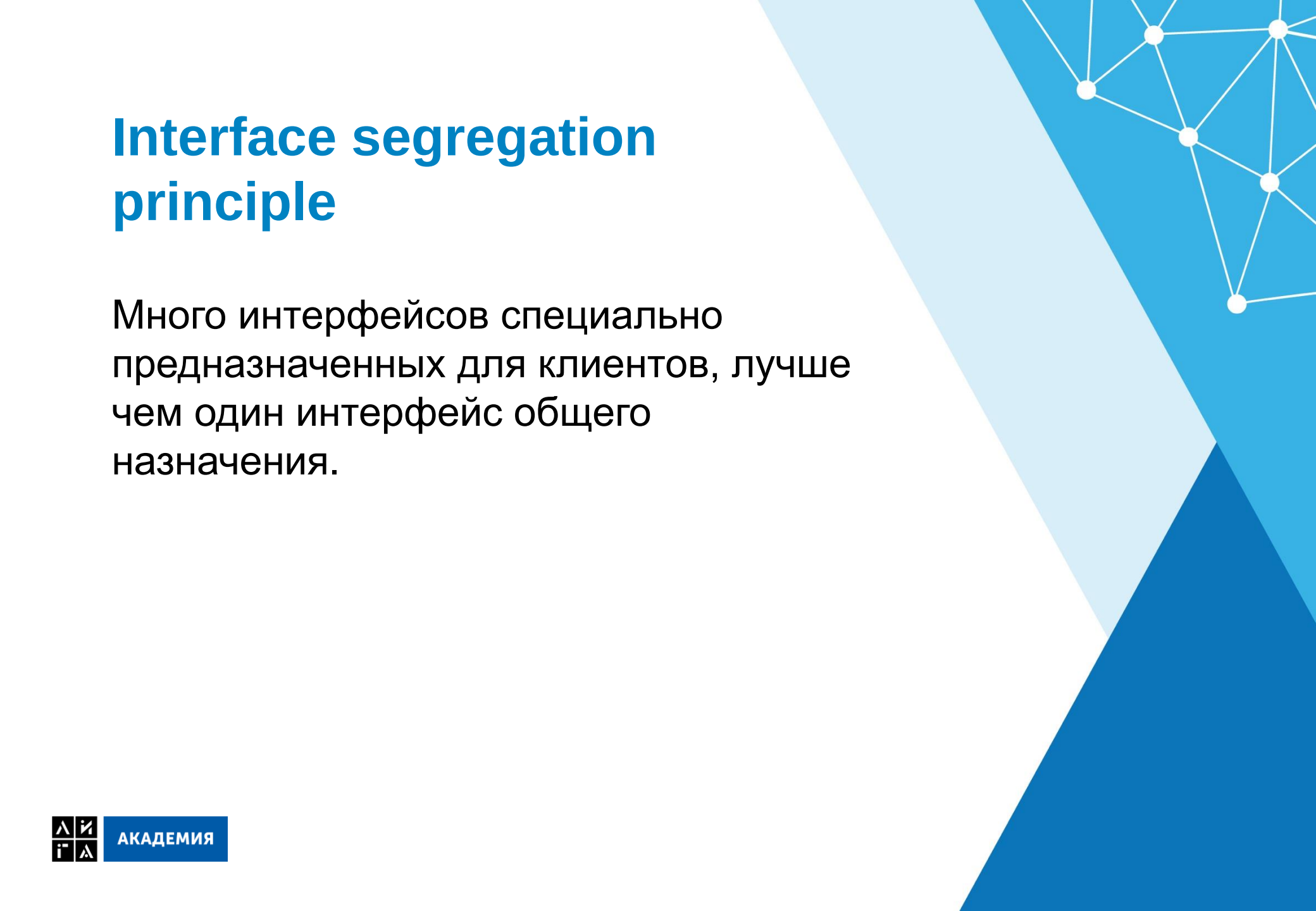
Liskov substitution principle

Функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа, не зная об этом.

Пример в IDEA

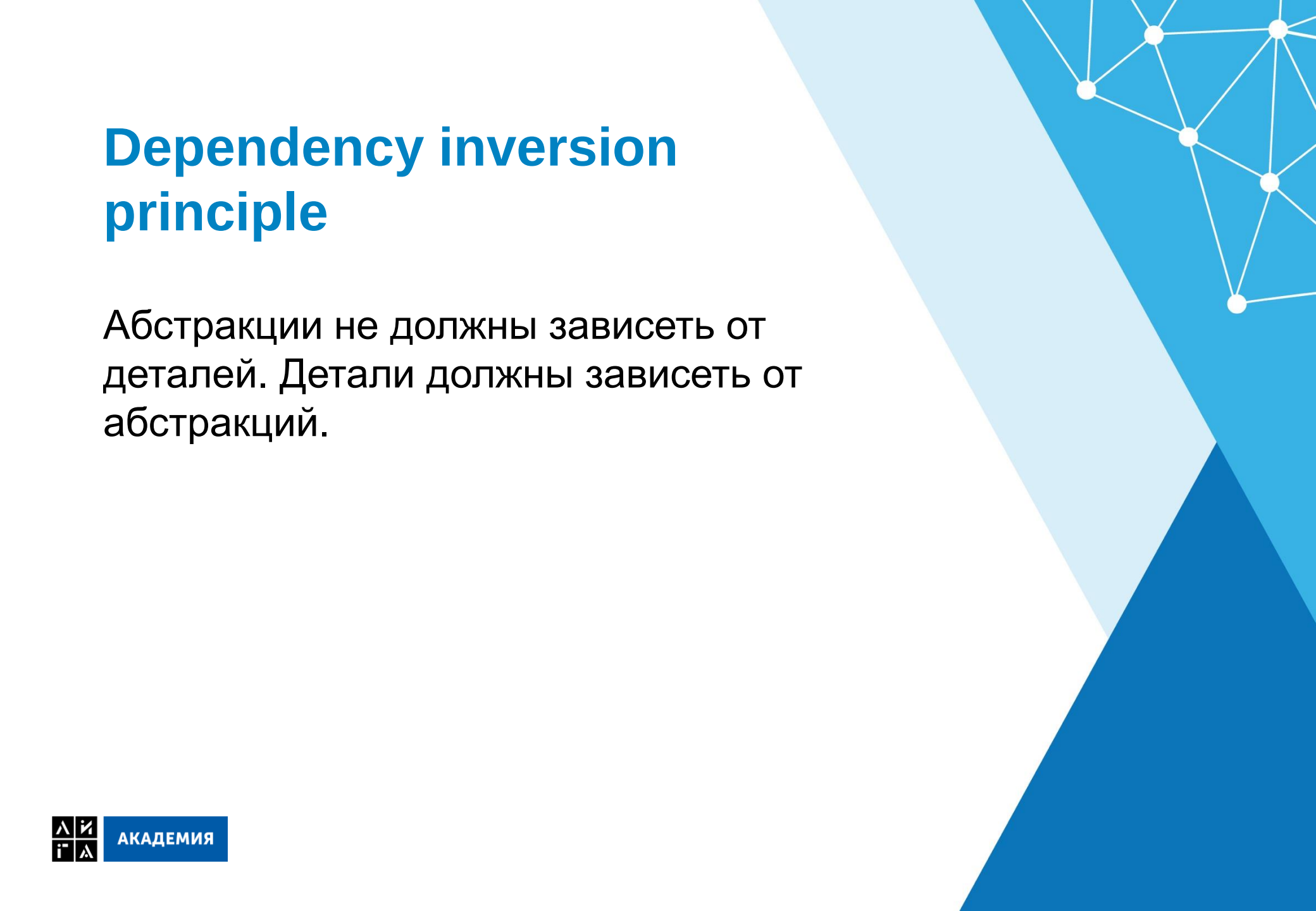
Interface segregation principle

Много интерфейсов специально предназначенных для клиентов, лучше чем один интерфейс общего назначения.

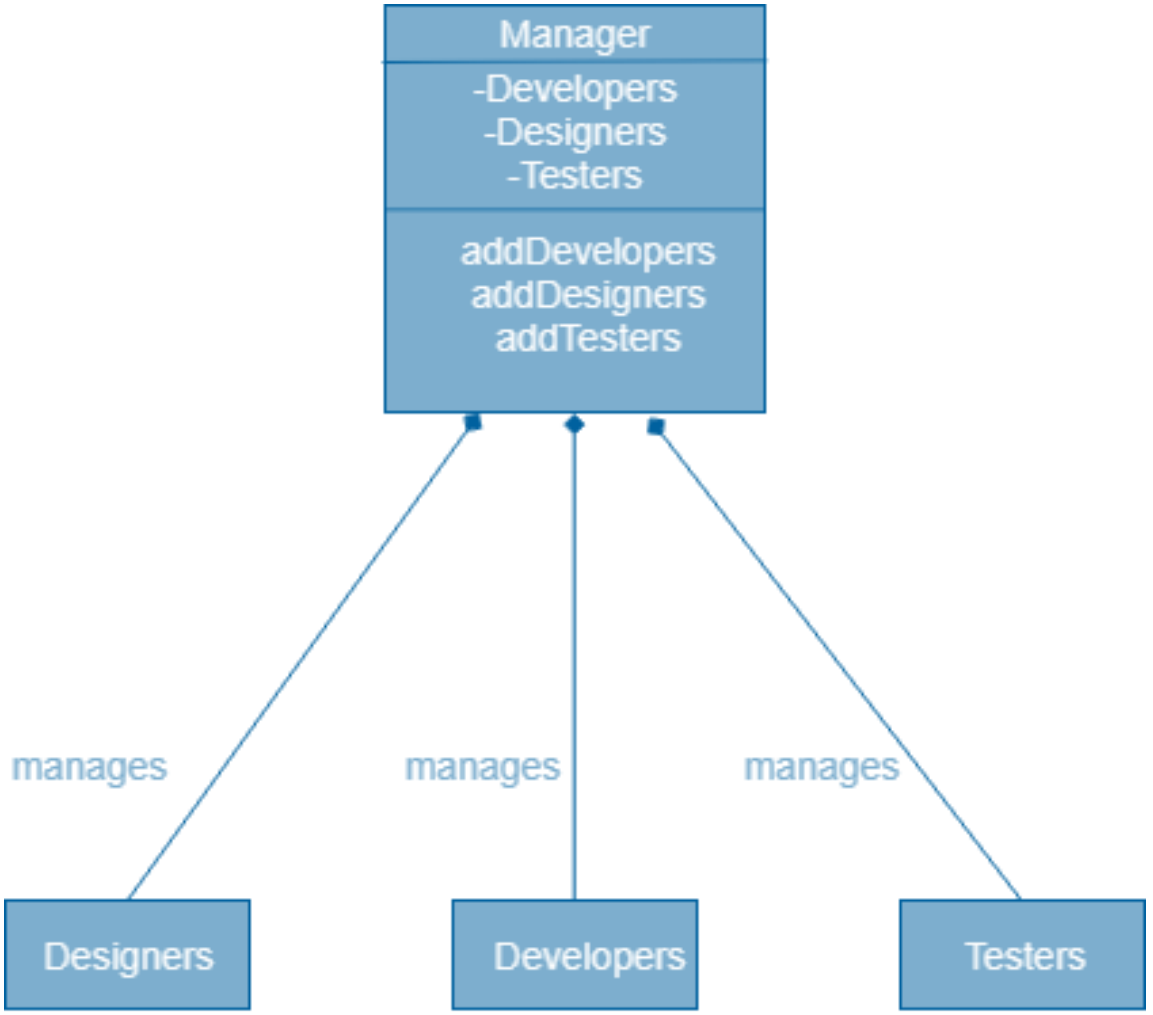


Dependency inversion principle

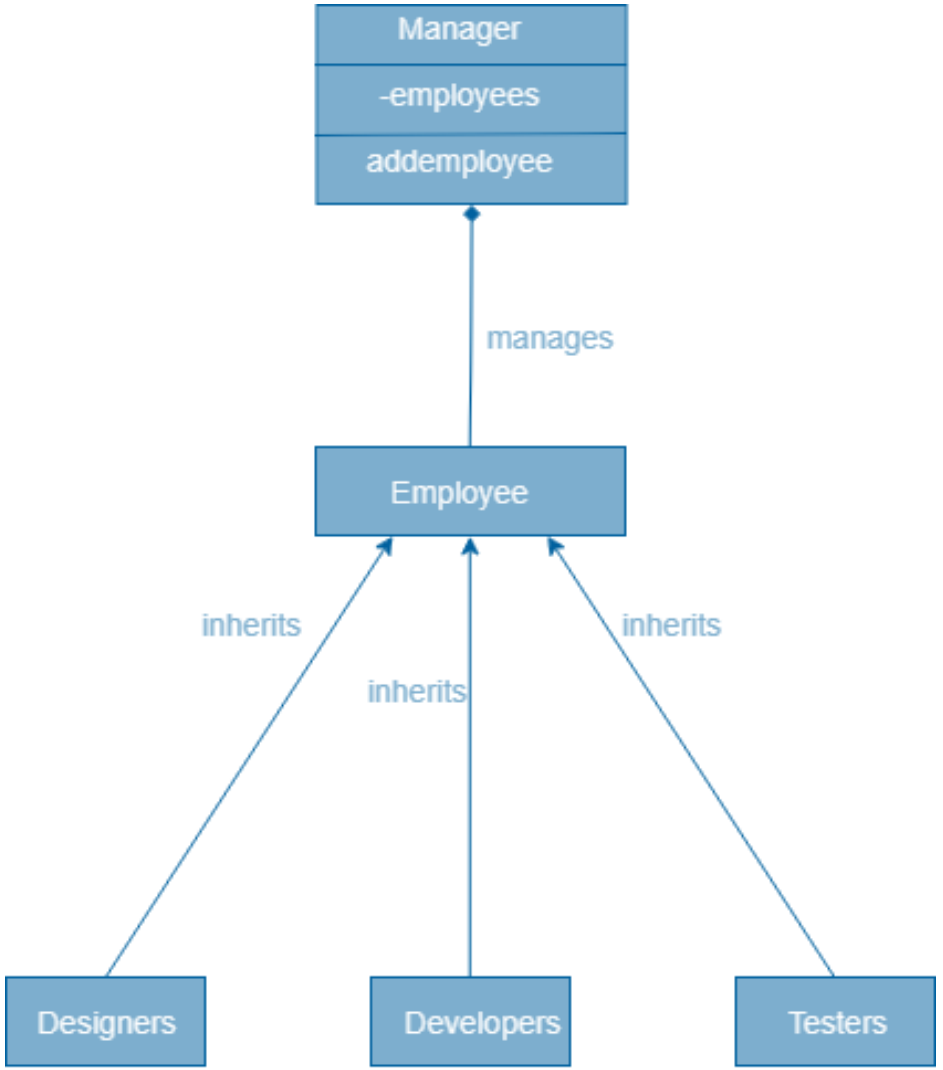
Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.



Инверсия зависимости (без использования принципа)



Инверсия зависимости



CI/CD

Непрерывная интеграция и развертывание

АКАДЕМИЯ

Continuous Integration

Непрерывная интеграция – это процесс и методика как можно более частой синхронизации программного кода между разработчиками в рамках проекта

Преимущества использования CI

1. Минимизация конфликтов в коде при слиянии
2. Быстрый доступ к новым функциям
3. Быстрая обратная связь при code review
4. Возможность использовать CD

Continuous Deployment

Непрерывное развертывание – это методика автоматизированного развертывания программной системы с минимизацией времени между написанием кода и запуском его в работу

Преимущества использования CD

1. Автоматизация развертывания
2. Облегчение отладки сложных систем
3. Быстрый доступ к тестируемым функциям
4. Возможность автоматизации тестирования

Основные инструменты

1

Intellij IDEA

Разработчик JetBrains

Среда разработки

2

Gitlab CI

Разработчик Gitlab

Открытая система

3

Jenkins

Разработчики Oracle, Косуке Кавагути и

сообщество

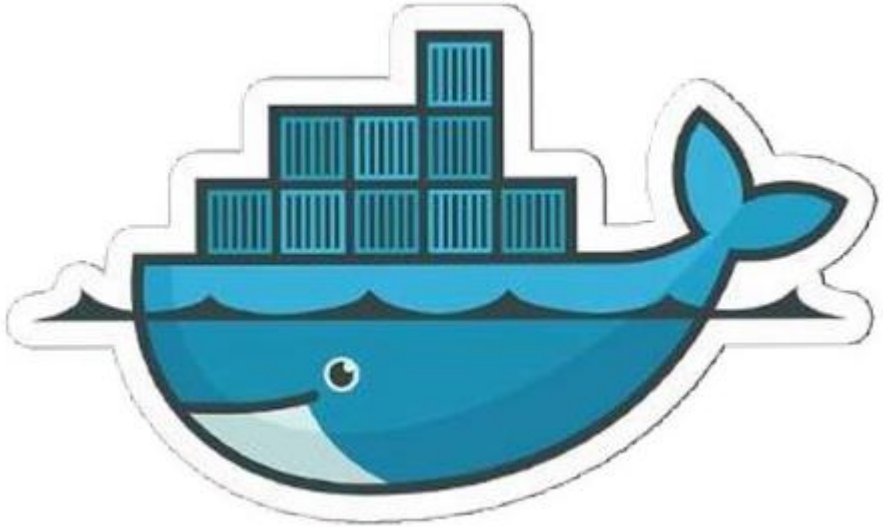
Открытая система

Использование Jenkins

1. Настройка получения программного кода
2. Настройка сборки программных артефактов
3. Настройка доставки и развертывания
4. Настройка запуска системы
5. Настройка тестирования
6. Настройка отчетов

DOCKER

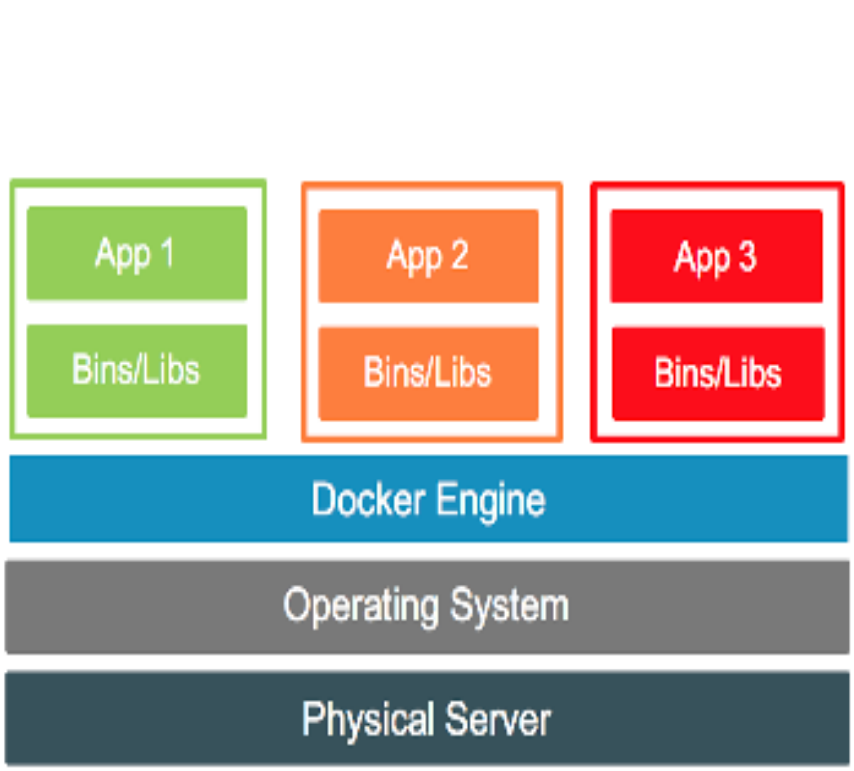
Что это и зачем он нужен разработчику?

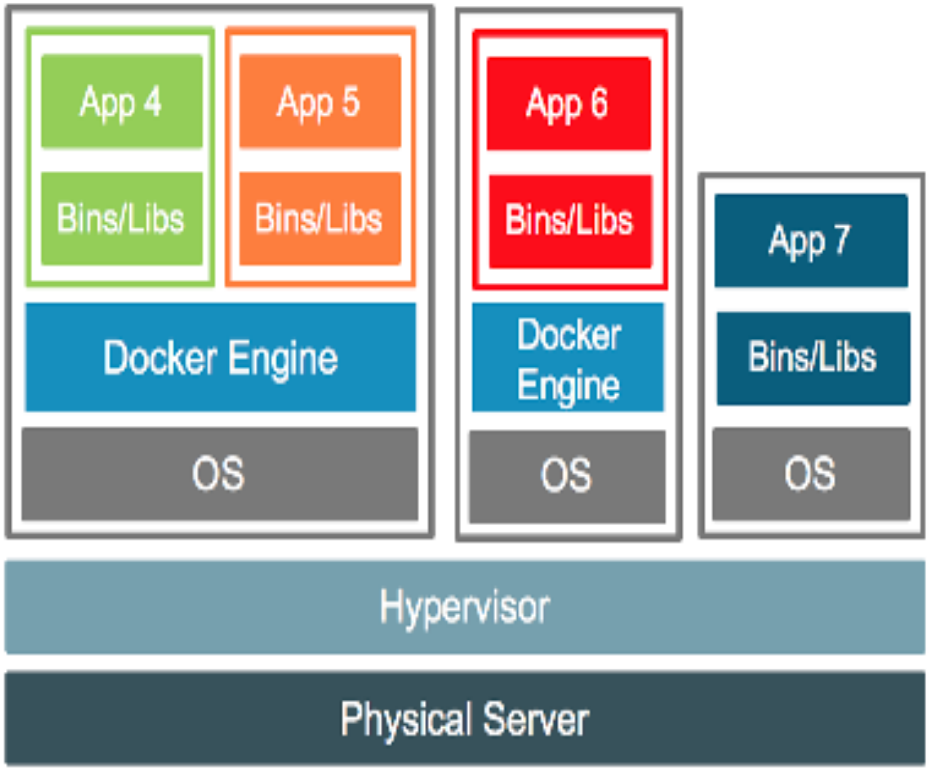


Для чего нужен (или плюсы):

1. Изолированный запуск приложений в контейнерах.
2. Упрощение разработки, тестирования и деплоя приложений.
3. Отсутствие необходимости конфигурировать среду для запуска — она поставляется вместе с приложением — в контейнере.
4. Упрощает масштабируемость приложений и управление их работой с помощью систем оркестрации контейнеров.

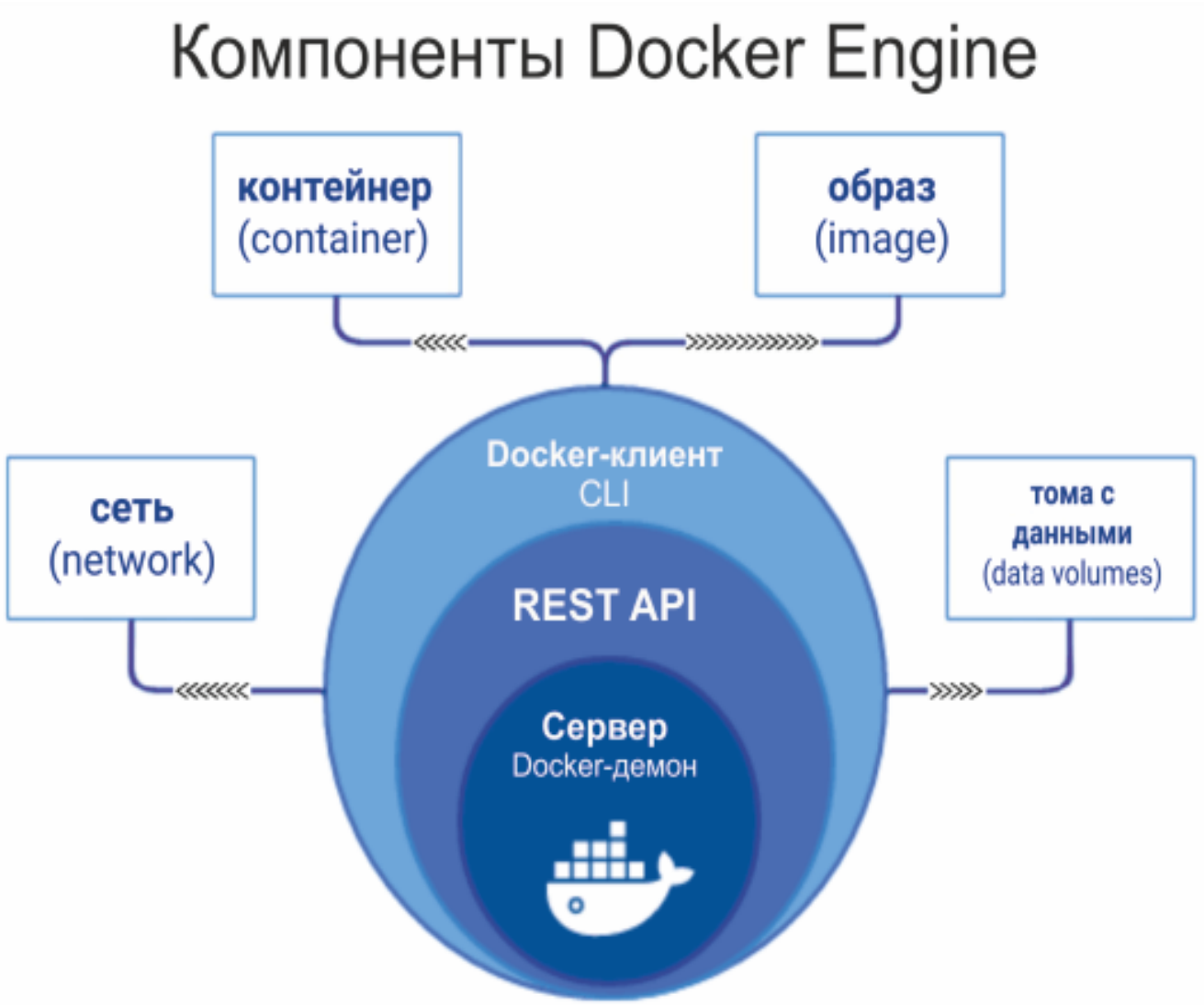
Чем отличается от Виртуальных Машин?





Из чего состоит?

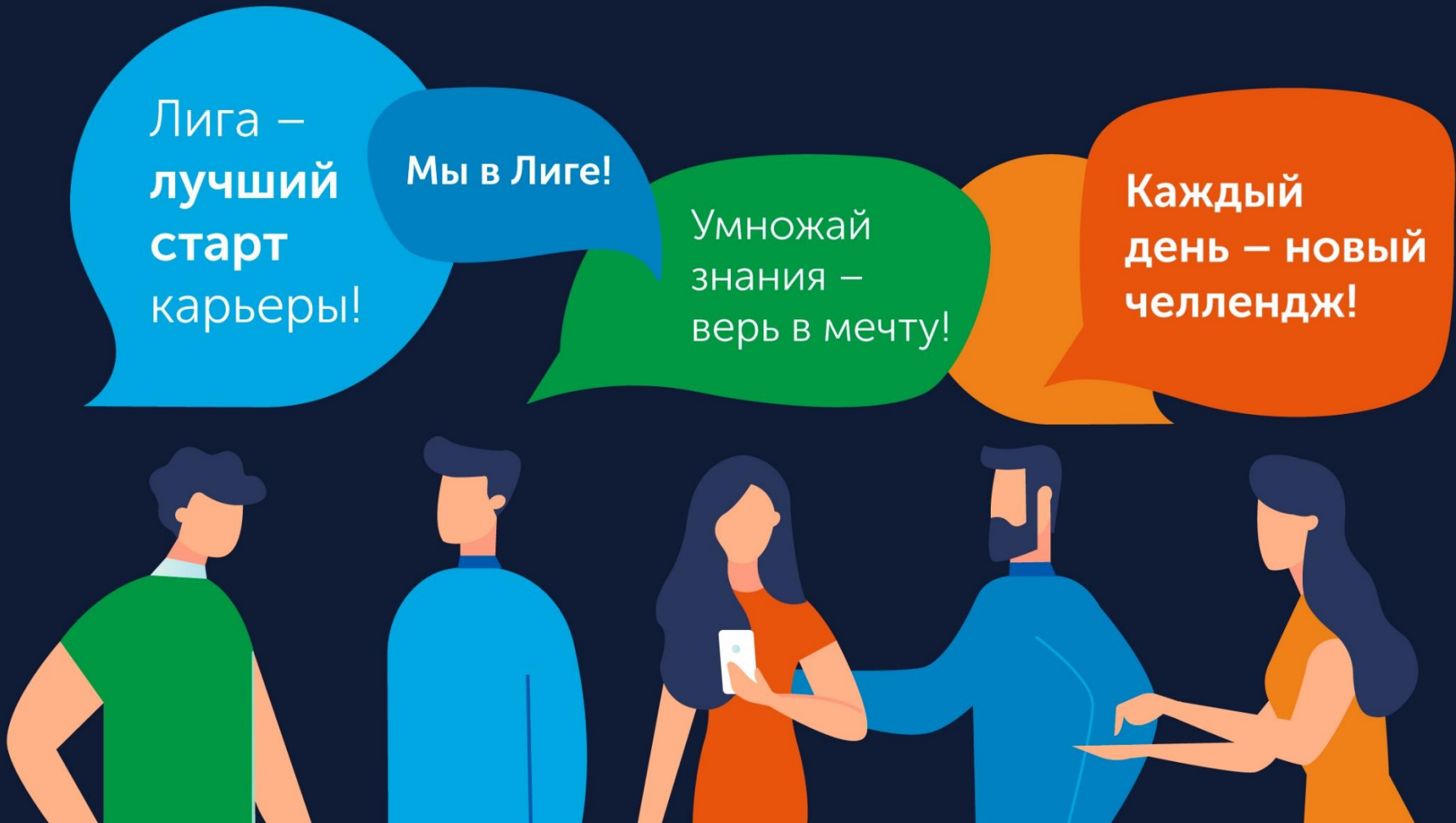
1. Образы
2. Контейнеры
3. Volumes
4. Networks



Документация

https://docs.docker.com/reference/ - Описание Dockerfile, docker-compose.yml, синтаксиса команд docker и docker-compose

Всем спасибо!



Сергей Кузнецов

Старший разработчик

АКАДЕМИЯ