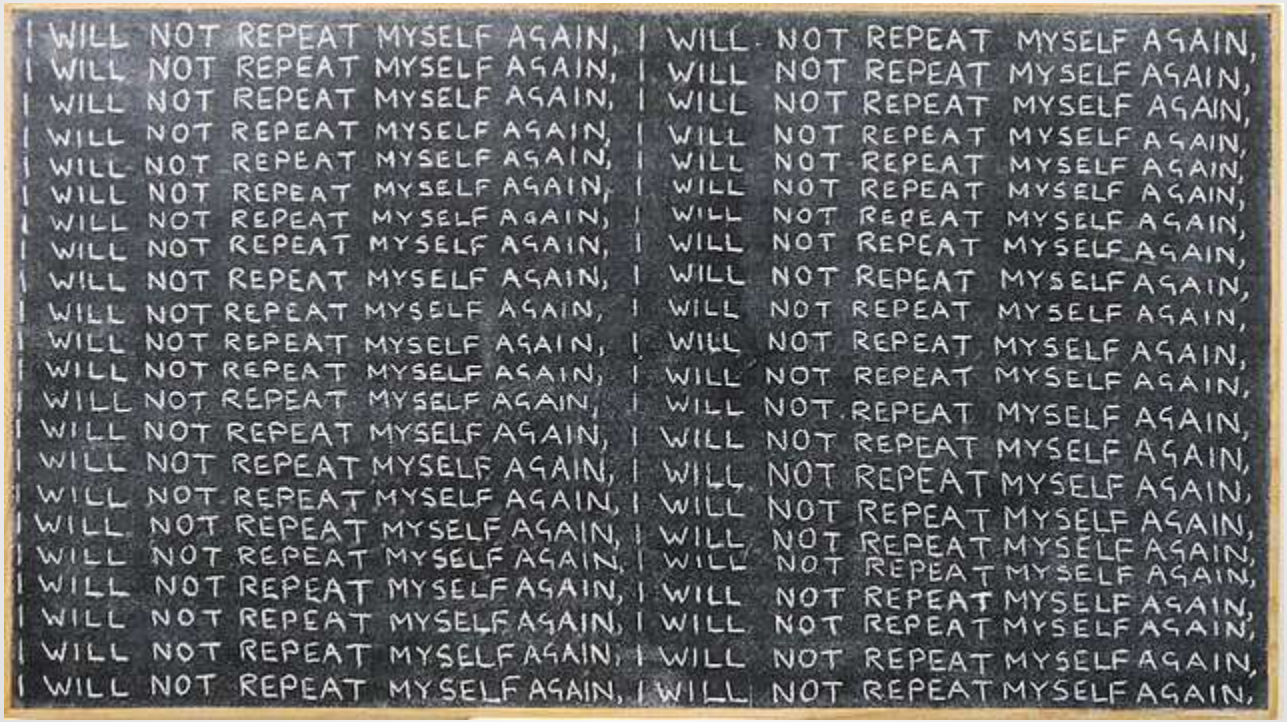
Сборка проекта

Промышленное программирование

Автор курса: Федор Лаврентьев
Лектор: Виктор Яковлев
МФТИ (с) 2016-2017

Использование готового кода



Утилиты

```
public String join(String sep, Object… tokens) {
    StringBuilder sb = new StringBuilder();
    for (Object token : tokens) {
        if (!sb.isEmpty()) {
            sb.append(sep);
        }
        sb.append(token);
    }
    return sb.toString();
}
```

Утилиты

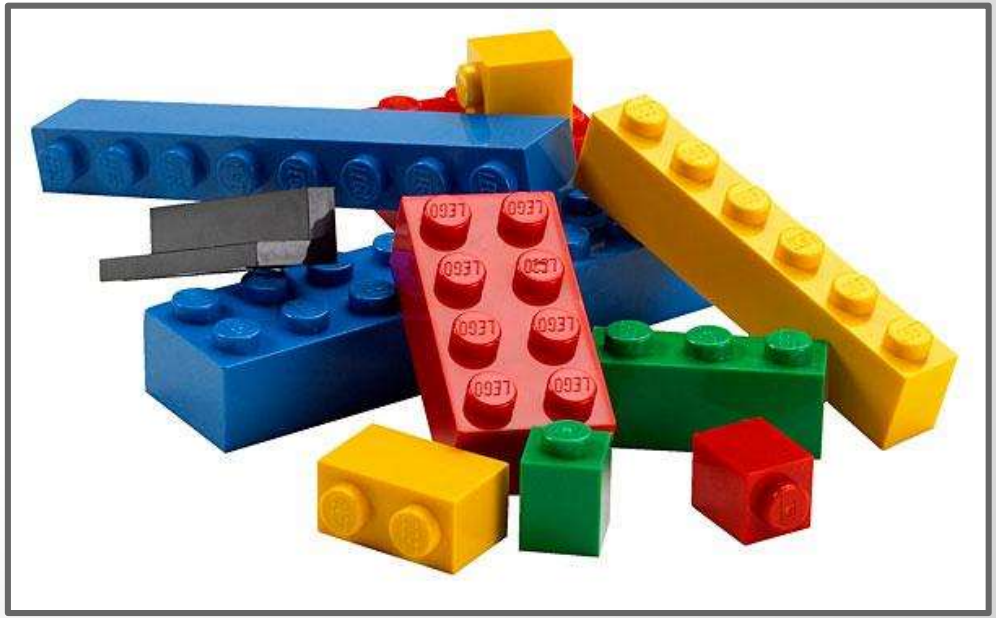
```
import com.google.guava....Joiner;

Joiner.on(“,”).join(anything);
```

Сборка и деплой

```
cd myProject
javac $(find src/ -name "*.java")
# Noo, what about classpath?
export CLASSPATH=lib/...
javac $(find src/ -name "*.java")
# Noo, what about cleaning?
rm -rf target/
javac $(find src/ -name "*.java")
# Noo, what about download some libraries?
wget http://repo1.maven.org/maven2/log4j/log4
1.2.17.jar
javac $(find src/ -name "*.java")
# Oh, sorry, no compiler for Java 8 =(
# FFFFFFFFFFUUUUUUUUUUUUU!!!!!1111
```

Сборка и деплой

```
mvn clean install
```



У Java очень большое сообщество

Все ваши задачи кто-то уже решил
К любому сервису кто-то уже написал API
К каждому багу кто-то уже настругал
~~костыль~~ workaround

Общие библиотеки

Подробно документированы
Обильно обсуждены на форумах
Почти стабильно работают
Доступны для скачивания в репозиториях

Google Guava

Обертки над объектами и исключениями
Новые коллекции, Immutable-коллекции
Функциональщина
Многопоточность
Кеширование
Работа со строками
Хеш-функции
Работа с потоками и файловой системой
Волшебные Reflection

Apache Commons

Расширения для java.lang
Новые коллекции
Потоки и файловая система
Алгоритмы кодирования и хеширования
Сжатие и распаковка
Connection Pooling
…

Логгинг

java.util.logging (JUL)
Apache commons-logging
Log4j
SLF4j
Logback

Тестирование

JUnit
TestNG
Unitils
Mockito
JMock
Spring Test *
…

Как подключить библиотеку?

Apache Maven

maven

Maven - “фреймворк для автоматизации сборки проектов...” (c) Wikipedia

Конфигурируется декларативно на XML-языке POM (Project Object Model)

Проект Maven ~= java-модуль

“Соглашение превыше конфигурации”

Maven репозиторий

Многие java-библиотеки публично опубликованы как проекты Maven

Maven Project ~= java-библиотека + POM-файл + ресурсы

Проекты доступны в центральном репозитории Maven http://search.maven.org/

Maven проекты

Каждый опубликованный maven-проект имеет уникальный идентификатор

Идентификатор состоит из имени группы, имени артефакта, версии.

Внутри проекта смежные артефакты различаются т.н. классификатором

Пример проекта

com.google.guava:guava-gwt:17.0

Группа - com.google.guava
Артефакт - guava-gwt
Версия - 17.0
Неявный классификатор - jar

Структура проекта

```
/lection01
  |-- src/
  |    |-- main/
  |    |    |-- java/
  |    |    |    \-- ru/fizteh/java/fediq/     # Структура пэкейджа
  |    |    |         \-- LectionMain.java
  |    |    \-- resources/
  |    |         \-- config.properties
  |    \-- test/
  |         \-- java/
  |              \-- ru/fizteh/java/fediq/
  |                   \-- LectionTest.java
  \-- pom.xml                                  # конфигурация проекта
```

Пример конфигурации Maven

```
<project>
    <modelVersion>4.0.0</modelVersion>          # Волшебство
    <groupId>ru.fizteh.java2.fediq</groupId>    # Аналог package
    <artifactId>lection-maven</artifactId>      # Имя проекта
    <version>1.0</version>                      # Версия
    <dependencies>
        <dependency>                            # Используемая библиотека
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>2.5.6</version>
        </dependency>
    </dependencies>
</project>
```

Конфигурация Maven

Жизненные циклы проекта Maven

- Сборка состоит из этапов - жизненных циклов (lifecycles)
- Циклы выполняются последовательно, с первого до целевого
- Краткий список жизненных циклов - verify, compile, test, install, deploy
- Сборочная задача может быть присвоена какому-то этапу или выполняться отдельно

Работа с Maven

```
$ mvn dependency:tree      # Показать дерево зависимостей
$ mvn dependency:copy-dependencies  # Скачать все зависимости

$ mvn compile              # Скомпилировать код
$ mvn test                 # Запустить юнит-тесты
$ mvn package              # Собрать (сжать и т.д.) артефакты
$ mvn install              # Загрузить артефакт в локальный репозиторий
$ mvn deploy               # Загрузить артефакт в удаленный репозиторий
$ mvn clean                # Убрать за собой всякое

$ mvn clean install        # Пересобрать проект

$ mvn install              # = mvn validate compile test package install
...
```

Зависимости

- Зависимости проекта перечисляются в блоке <dependencies>
- Maven умеет отслеживать транзитивные зависимости
- Во время сборки maven автоматически скачает все необходимые файлы
- Задача dependencies:copy-dependencies сложит зависимости в директорию target/

Scopes

Scope - ситуация или этап, в котором потребуется зависимость
provided - только для компиляции
runtime - только в рантайме
test - только в тесте
compile (default) - на стадии компиляции и в рантайме
system - предоставляется окружением

Пример указания зависимостей

```
<dependencies>
    <dependency>
        <groupId>com.jolbox</groupId>
        <artifactId>bonecp</artifactId>
        <version>0.8.0.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Плагины

Задачи в maven поставляются плагинами
Плагин содержит в себе несколько целей
Плагины содержатся в тех же репозиториях, что и другие проекты
Плагины также “достаются” автоматически
Плагин конфигурируется в pom-файле
Ряд плагинов подключен по умолчанию - clean, compile, install, surefire и др.

Пример конфигурации плагина

```
<build>
    ...
    <plugins>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>2.3.2</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
    </plugins>
</build>
```

JAR-файлы

По окончании сборки проект будет упакован в jar-файл

jar - это обычный zip-архив, содежащий скомпилированные классы и метаданные

При желании, в аналогичные jar-файлы можно сохранить исходники, ресурсы и пр

На последней стадии (деплое) jar-файлы заливаются в репозиторий

Наследование проектов

Для поддержки принципа DRY maven поддерживает наследование проектов

Основная конфигурация производится в “родительском” проекте

Дочерние проекты подключают его с помощью тега <parent>

Часто родительский проект является корнем “многомодульного” проекта

Многомодульные проекты

Группа близких проектов может быть объединена в многмодульный проект

Все модули перечисляются в корневом проекте в теге <modules/>

Также в корневом проекте указывается <packaging>pom</packaging>

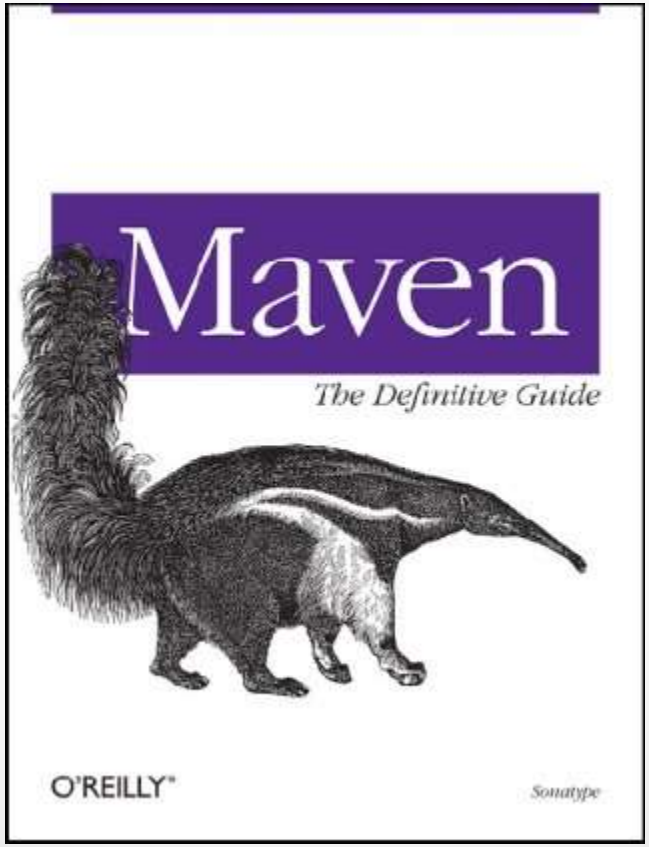
Пример - корневой проект

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>ru.fizteh.java2</groupId>
    <artifactId>parent-pom</artifactId>
    <packaging>pom</packaging>
    <version>1.0</version>
    <modules>
        <module>example-jdbc</module>
        <module>example-spring</module>
    </modules>
    <build/> ...
    <dependencies/> ...
</project>
```

Пример - дочерний проект

```
<project>
    <parent>
        <groupId>ru.fizteh.java2</groupId>
        <artifactId>parent-pom</artifactId>
        <version>1.0</version>
    </parent>

    <groupId/> <version/> … # Унаследованные параметры можно перезаписывать
    <artifactId>example-jdbc</artifactId>
    <packaging>jar</packaging>

    <dependencies/> ...
</project>
```

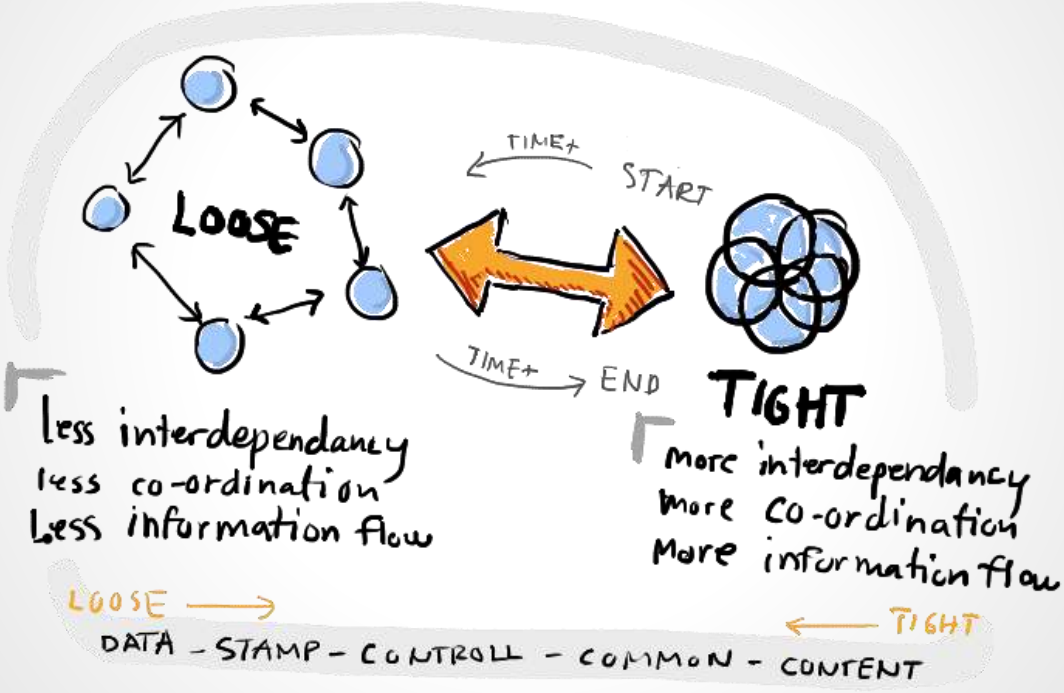
Рекомендуемая литература



Компоновка приложения

Промышленное программирование

Связанность кода



http://infomgmt.wordpress.com/

Больше кода - сложнее работать

Со временем число сущностей растет
Число связей растет на порядок быстрее
Если не предпринять мер, быстро наступает коллапс разработки

Логическая компонента

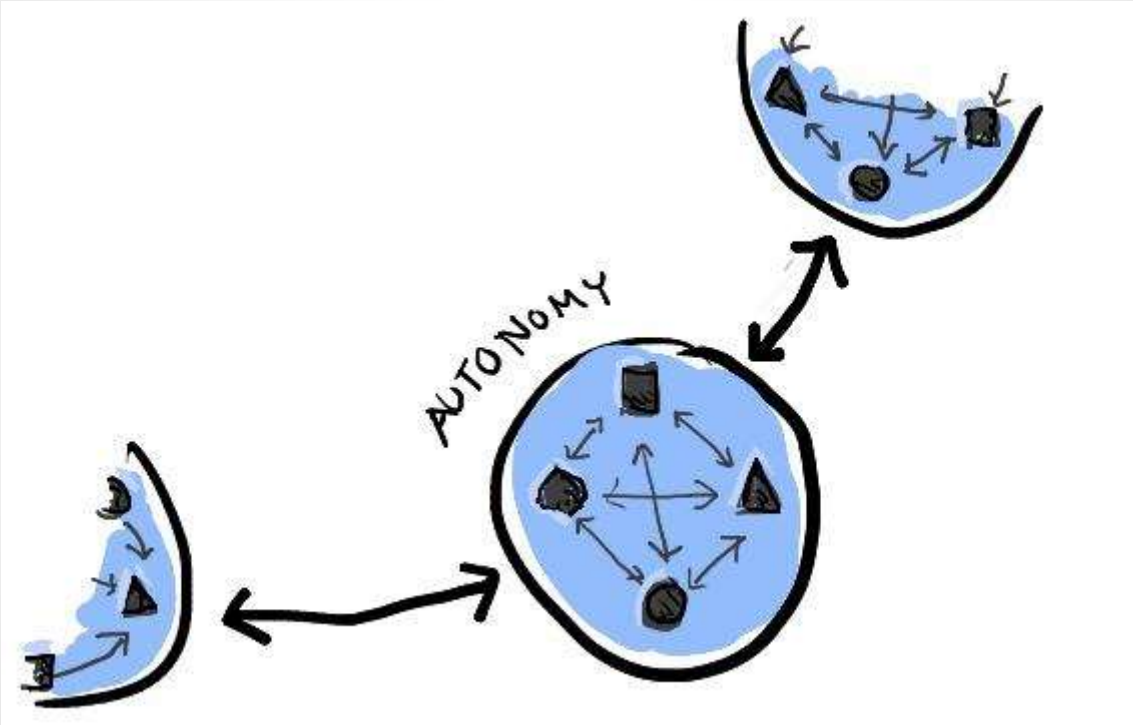
Компонента - автономная логическая единица кода

Внутри компоненты происходит контролируемое тесное взаимодействие

Наружу предоставляется API (контракт)

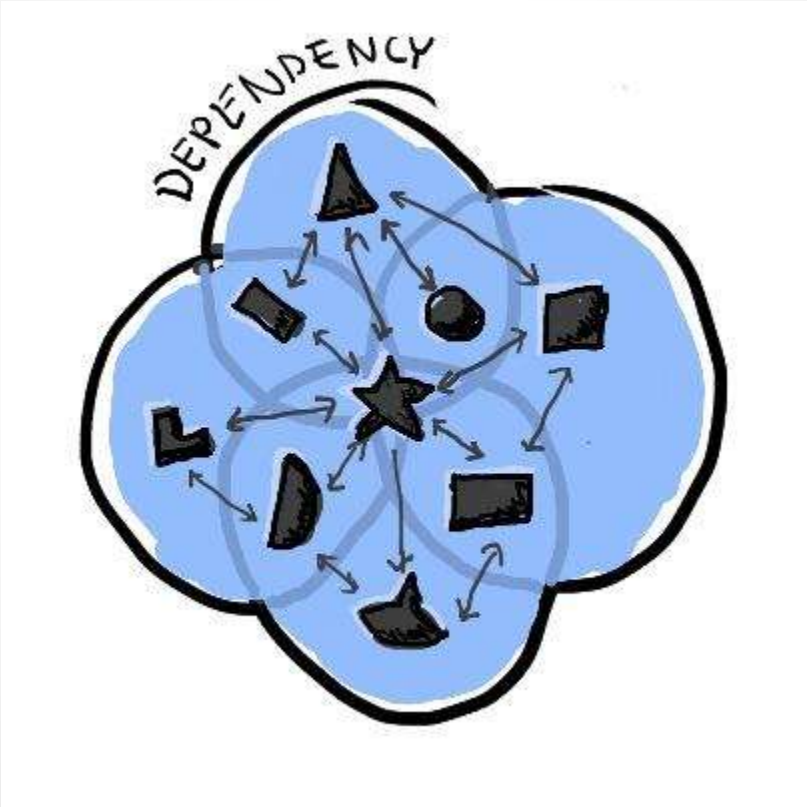
Компонента взаимодействует с соседями через их API

Слабая связанность



http://infomgmt.wordpress.com/

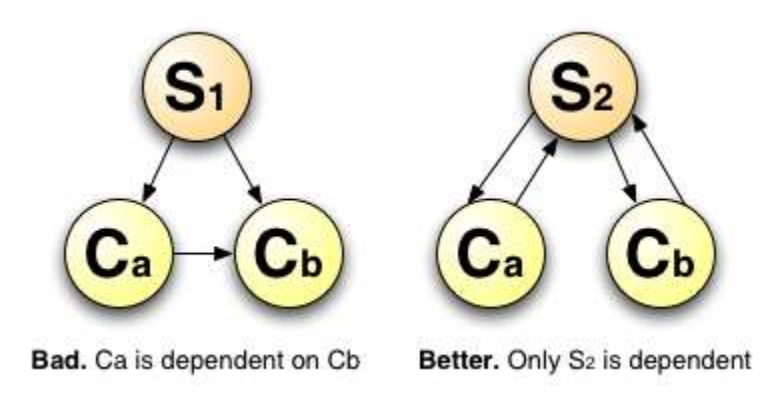
Сильная связанность



http://infomgmt.wordpress.com/

Закон Деметры

Если у А есть доступ к Б
и у Б есть доступ к В,
то А не нужен доступ к В



Уровни связанности

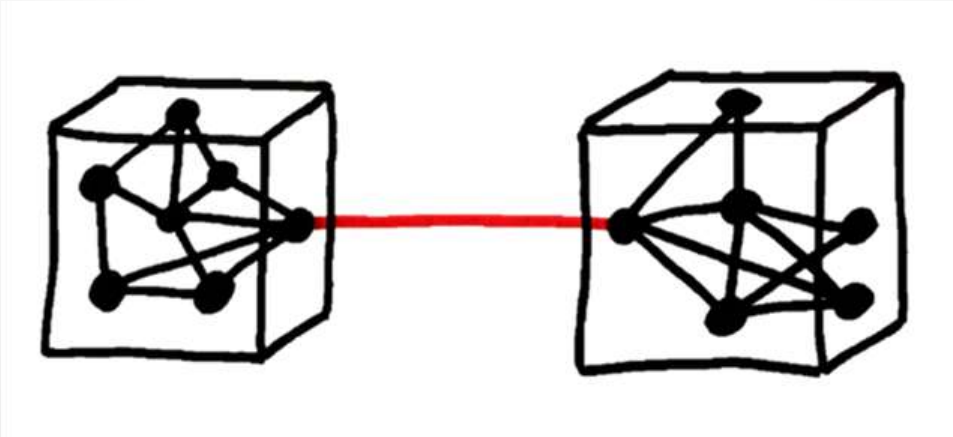
Содержимое
Общее состояние
Внешний контракт
Структура данных
Сообщения
Нет связности

Связность (сцепление) компоненты

Компонента должна быть осмысленна
Ее части должны иметь что-то общее
Связность - мера взаимосвязи между составными частями одной компоненты
Принцип наименьшего удивления опирается на связность

Основная идея проектирования

Меньше связанность - больше сцепление



Модульность

Модульность - принцип разработки ПО, согласно которому код разделяется на отдельные функционально законченные сущности - модули

Это позволяет переиспользовать код

Это упрощает проектирование

Это упрощает дистрибуцию (см. Maven)

Интерфейс и реализация

Как правило, API модуля не содержит сложных конструкций и зависимостей

Внешние интерфейсы (API) модуля можно выделить в отдельный легкий модуль

Реализация остается в отдельном тяжелом модуле, о котором пользователи могут не знать

Зависимости между модулями

Переизбыток зависимостей в системе вызывает непредсказуемые проблемы

Разделение реализации и интерфейса сильно сокращает объем знаний и зависимостей, необходимый для использования модуля