Dependency Injection в JUnit5

Интерфейс ParameterResolver

- Входит в экспериментальный API.
- Обеспечивает динамическую привязку (*resolving*) параметров на этапе исполнения.
- Имплементации необходимо регистрировать.
- Позволяет инжектировать любое число параметров в любом требуемом порядке.

Интерфейс ParameterResolver

- Поддерживает *constructor injection*.
- Поддерживает *method injection* в методах со след. аннотациями:

  @Test

  @TestFactory

  @BeforeEach, **@AfterEach**, **@BeforeAll** или **@AfterAll**

Встроенные резолверы

- Зарегистрированы и активированы по умолчанию:

  TestInfoParameterResolver

  TestReporterParameterResolver

  RepetitionInfoParameterResolver

- Все прочие резолверы параметров надо активировать путем регистрации соответствующих расширений через **@ExtendWith**.

TestReporterParameterResolver

- Служит для инжекции объекта типа **TestReporter** в тот или иной метод.
- Объект **TestReporter** может применяться для публикации дополнительных сведений о текущем тесте.

TestReporter

- Рекомендован к использованию, когда надо передать информацию в каналы *stdout* или *stderr*.
- Эта информация поступает в метод **TestExecutionListener.reportingEntryPublished()** и может обрабатываться средствами IDEs или включаться в отчеты.
- Методы:

```
void publishEntry(Map<String, String> record)

void publishEntry(String key, String value)

void publishEntry(String key)
```

RepetitionInfoParameterResolver

- Инжектирует объект типа **RepetitionInfo** в методы, маркированные аннотациями **@RepeatedTest**, **@BeforeEach** или **@AfterEach**.
- Зарегистрирован по умолчанию, но работает только в контексте **@RepeatedTest**.

Когда объект **RepetitionInfo** используется в качества параметра **@BeforeEach**- или **@AfterEach**-метода, все тестовые методы должны иметь аннотацию **@RepeatedTest**.

RepetitionInfo

- Применяется для извлечения информации:

  о текущей тестовой итерации

  о числе итераций, заданных для соответствующего **@RepeatedTest**-метода

- Методы:

  ```
  int getCurrentRepetition()
  ```

  ```
  int getTotalRepetitions()
  ```

Класс DynamicTest

@TestFactory

- Эта аннотация служит для динамического создания тестов.
- Входит в экспериментальный API.
- Аннотированный метод должен:

  возвращать объект типа **Stream<DynamicTest>**, **Collection<DynamicTest>**, **Iterable<DynamicTest>** или **Iterator<DynamicTest>**.

  быть нестатическим.

@TestFactory

- Для генерации конкретного теста можно воспользоваться следующим методом:

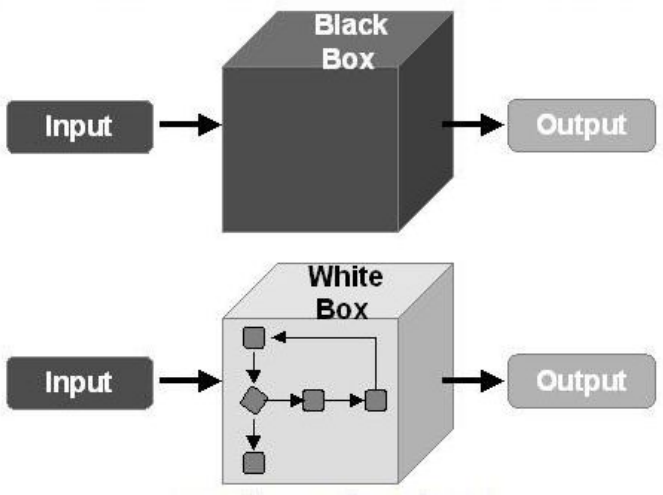
```
static DynamicTest dynamicTest(
    String displayName, Executable executable)
```

- displayName: индицируемое имя метода.
- executable: тестовая бизнес-логика.

BlackBox Testing

vs

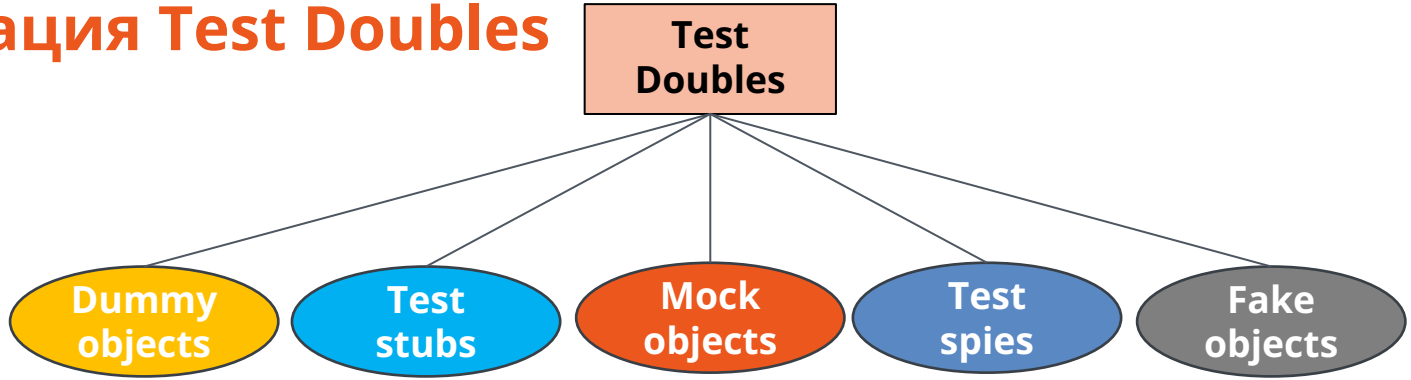
WhiteBox Testing

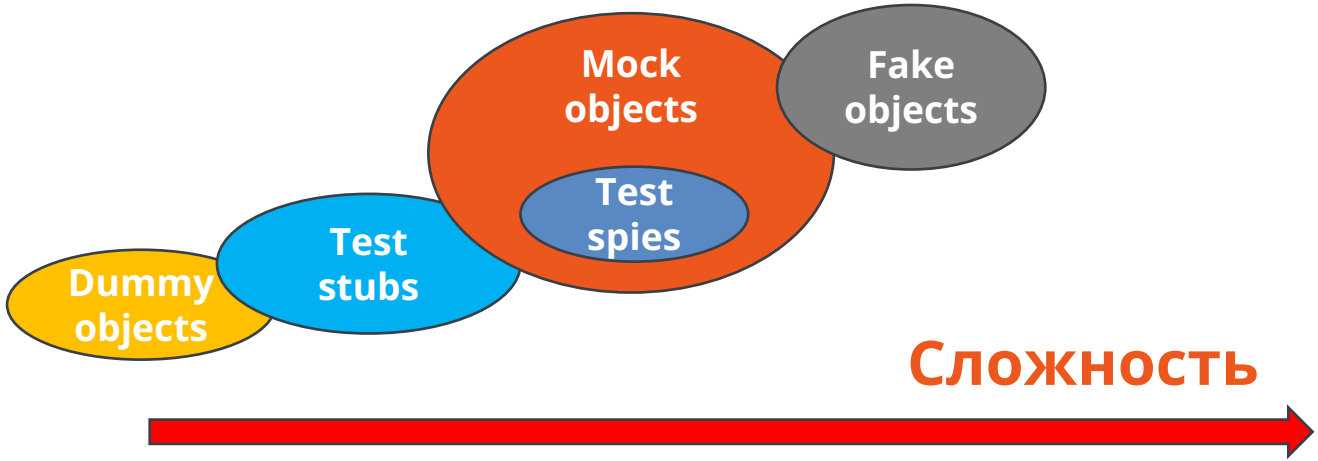


BlackBox-тестирование	WhiteBox-тестирование
• Основано на спецификации модуля • Охватывает заявленный контракт • Не выявляет ошибки имплементации	• Основано на коде • Охватывает закодированное поведение • Не выявляет пропущенные сценарии

Интеграция
с Mockito

Классификация Test Doubles





UML Sequence Diagram