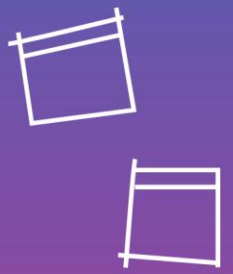
DBC
DIGITAL BUSINESS COMPANY®

VEM SER
DBC



Testes Unitários

Aula 2 – Mockito

Conteúdo da Aula

- Mockito
- Homework

Mockito

DBC
DIGITAL BUSINESS COMPANY®

VEM SER
DBC

Mockito

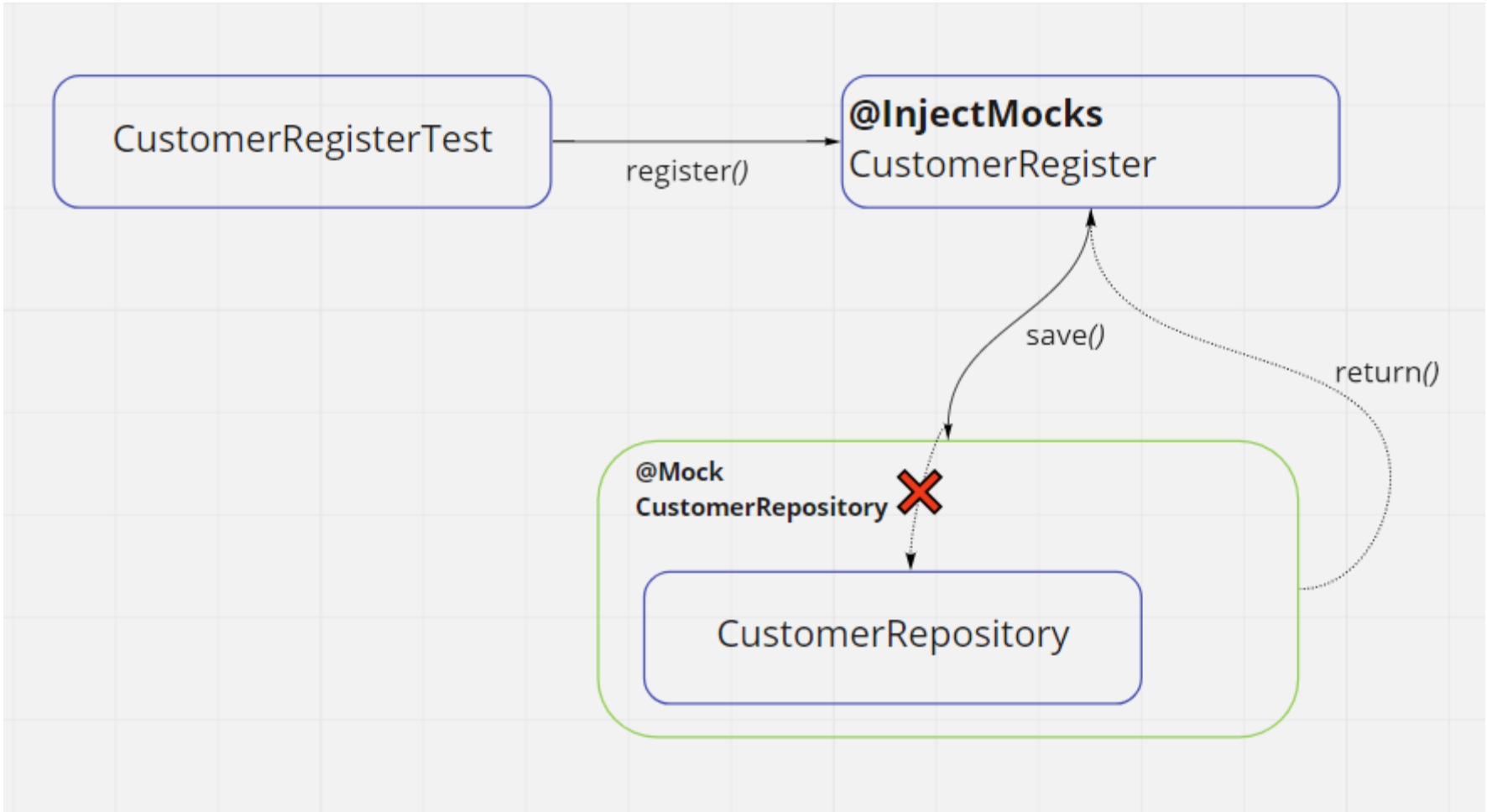
- É um framework de testes unitários e o seu principal objetivo é instanciar classes e controlar o comportamento dos métodos.

Mockito

- É um framework de testes unitários e o seu principal objetivo é instanciar classes e controlar o comportamento dos métodos.
- Isso é chamado de mock, na tradução livre quer dizer zombar

Mockito

- É um framework de testes unitários e o seu principal objetivo é instanciar classes e controlar o comportamento dos métodos.
- Isso é chamado de mock, na tradução livre quer dizer zombar
- Ao mockar a dependencia de uma classe, a classe que estou testando pensa estar invocando o metodo realmente, mas de fato não está.


CustomerRegisterTest
register()
@InjectMocks
CustomerRegister
save()
return()
@Mock
CustomerRepository
CustomerRepository

Dependências para projetos spring

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13.2</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
```

DBC
DIGITAL BUSINESS COMPANY®

show me your code;

VEM SER
DBC

Guia do Mockito

- **Mock**: cria uma instancia de uma classe, porém Mockada. Se você chamar um metodo ele não irá chamar o metodo real, a não ser que você queira.
- **Spy**: cria uma instancia de uma classe, que você pode mockar ou chamar os metodos reais. É uma alternativa ao *InjectMocks,* quando é preciso mockar metodos da propria classe que esta sendo testada.
- **InjectMocks**: criar uma intancia e injeta as dependências necessárias que estão anotadas com *@Mock.*
- **Verify**: verifica a quantidade de vezes e quais parametros utilizados para acessar um determinado metodo.
- **When**: Após um mock ser criado, você pode direcionar um retorno para um metodo dado um paremetro de entrada.
- **Given**: Mesmo propósito que o *when*, porém é utilizado para BDD. Fazendo parte do BDDMockito.

O que eu devo testar?

O que eu devo testar?

- Regras de negócio (services)
- Classes utilitárias
- Códigos com lógica e complexidade

O que eu não devo testar?

O que eu não devo testar?

- Conexão com banco de dados
- Conexão com API
- Get/Set
- Controllers...

Links úteis

https://site.mockito.org/

https://javadoc.io/doc/org.mockito/mockito-core/latest/org/mockito/Mockito.html

https://inside.contabilizei.com.br/conceitos-basicos-sobre-mockito-73b931ce0c2c

https://www.baeldung.com/mockito-series

https://medium.com/cwi-software/testando-seu-c%C3%B3digo-java-com-o-mockito-framework-8bea7287460a

#Homework

- Testar todos os métodos públicos das services do seu projeto final
- Commitar no repositório da equipe sendo:
  - Equipes com 3 pessoas: 3 services (uma service para cada pessoa)
  - Equipes com 2 pessoas: 2 services (uma service para cada pessoa)
  - Fazer a implementação e fazer um commit único para cada (se possível)
  - 100% de cobertura de testes nas services desenvolvidas