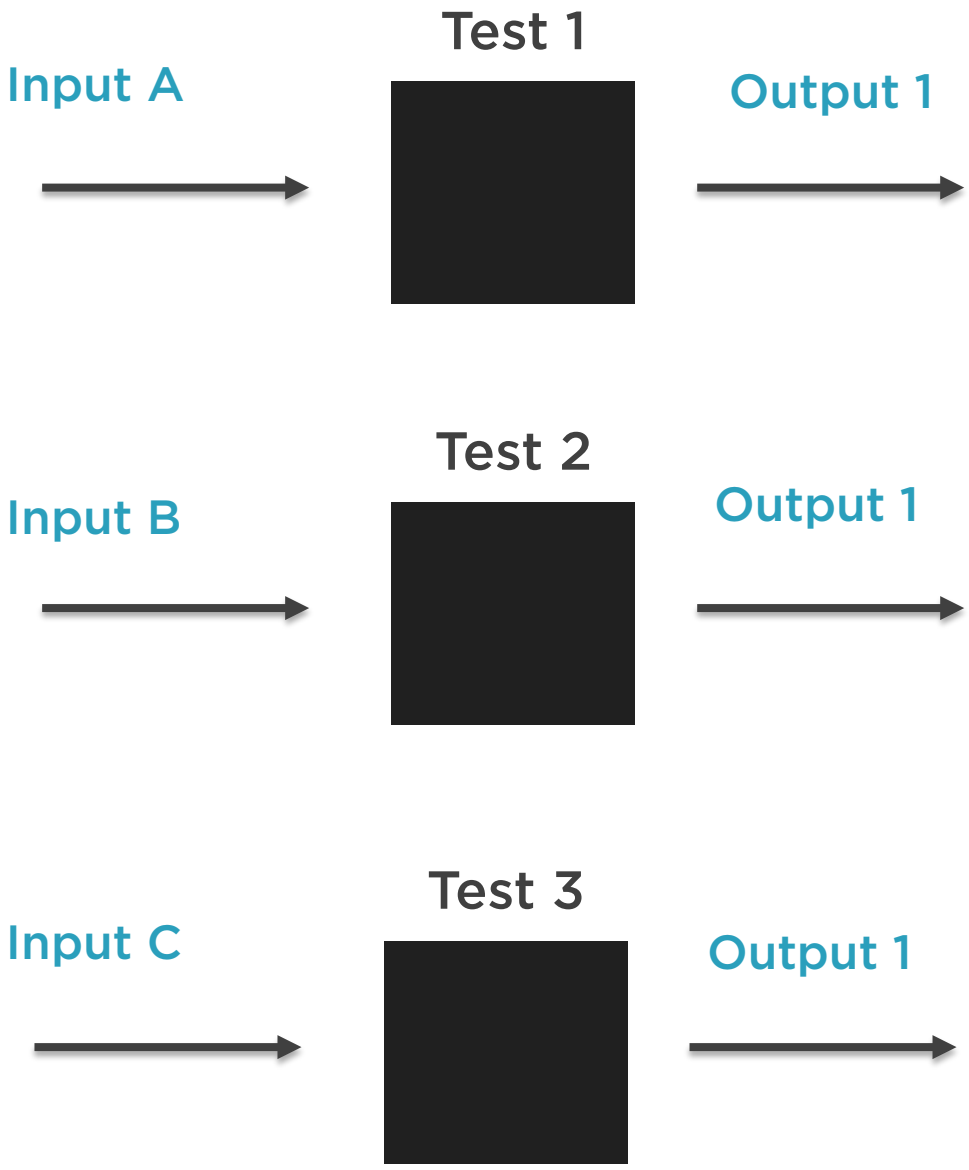
DataProviders

Andrejs Doronins
TEST AUTOMATION ENGINEER

public addUser(String email)




Test 1
Input A
Input B
Input C
Output 1

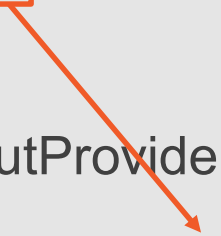
```
@Test
public void verifyX(){
        doStuff(value1);
}

@Test
public void verifyZ(){
        doStuff(value2);
}
```

```
@Test(dataProvider="myInputProvider")
public void verifyX(String param1){
        doStuff(param1);
}
```

```
@DataProvider
Object[][] myInputProvider() {
        return new Object[][] {
                {"val1"},
                {"val2"},
                {"val3"}};}

@Test(dataProvider="myInputProvider")
public void verifyX(String param1){
        doStuff(param1);
}
```



https://api.github.com/

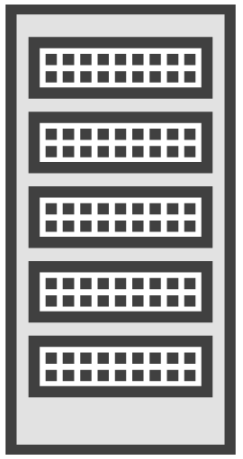
user

user/followers

notifications

HTTP GET Request

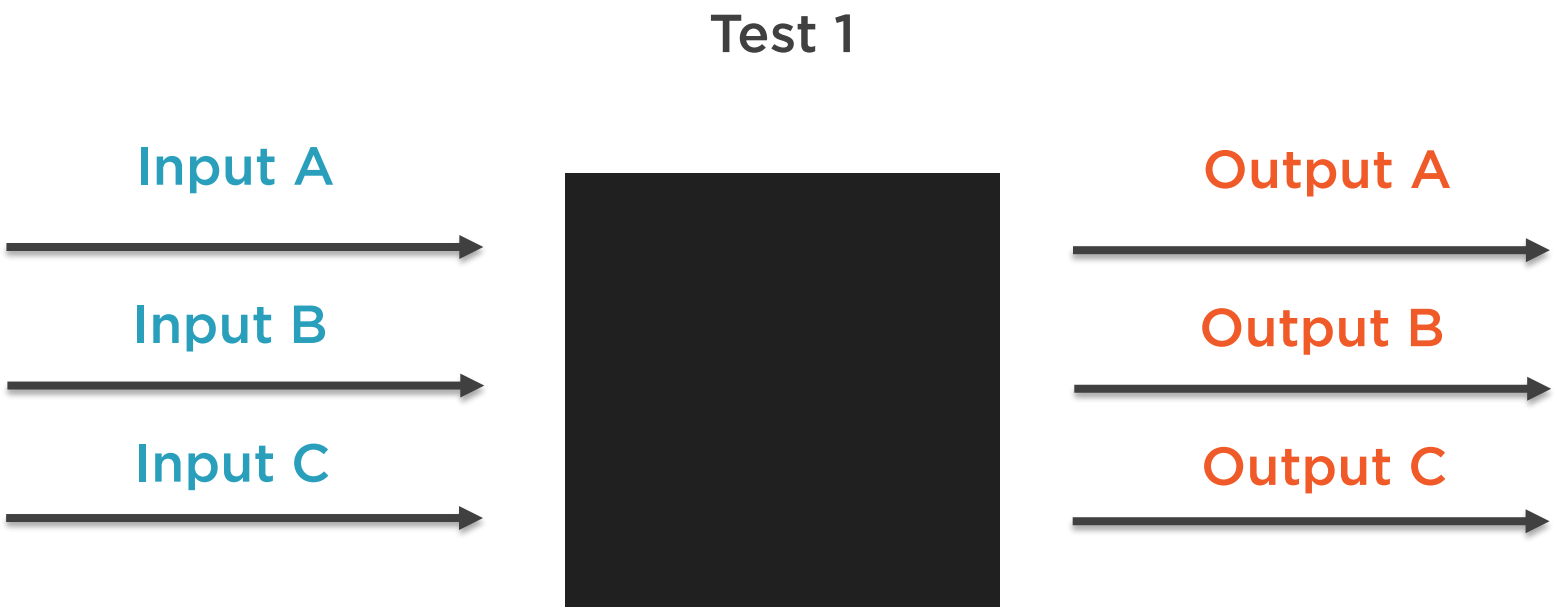
HTTP 401 Unauthorized



```
@DataProvider
Object[][] twoInputProvider() {
        return new Object[][] {
                {“email1“, “pwd1”},
                {“email2“, “pwd2”}
        };
}
```


Test 1
Input A
Input B
Input C
Output A
Output B
Output C

```
return new Object[][] {
                {“inputA“, “outputA”},
                {“inputB“, “outputB”},
                {“inputC“, “outputC”}
};}
```



```
@DataProvider
Object[][] twoInputProvider() {
        return new Object[][] {
                {"", "email must not be empty"}
                {"joe", "email must have an @ sign"}
                {"joe@email", "email must have one . after @ sign"}
        };
}
```

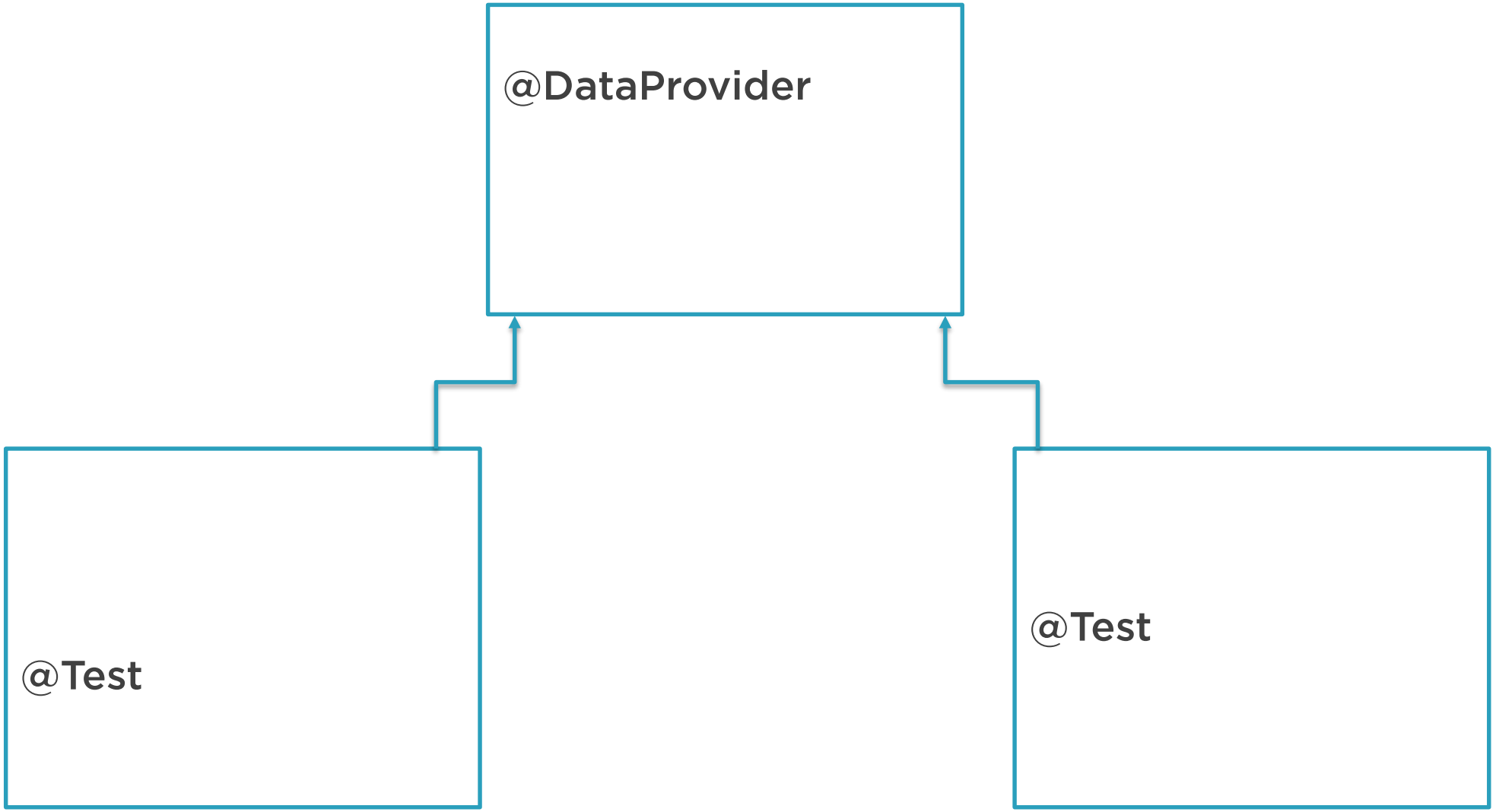
```
@Test(dataProvider=“myInputProvider”)
public void verifyX(String input, String output){
        if(input.equals(“something”){
                doOneThing();
        } else {
                doAnother();
        }
}
```

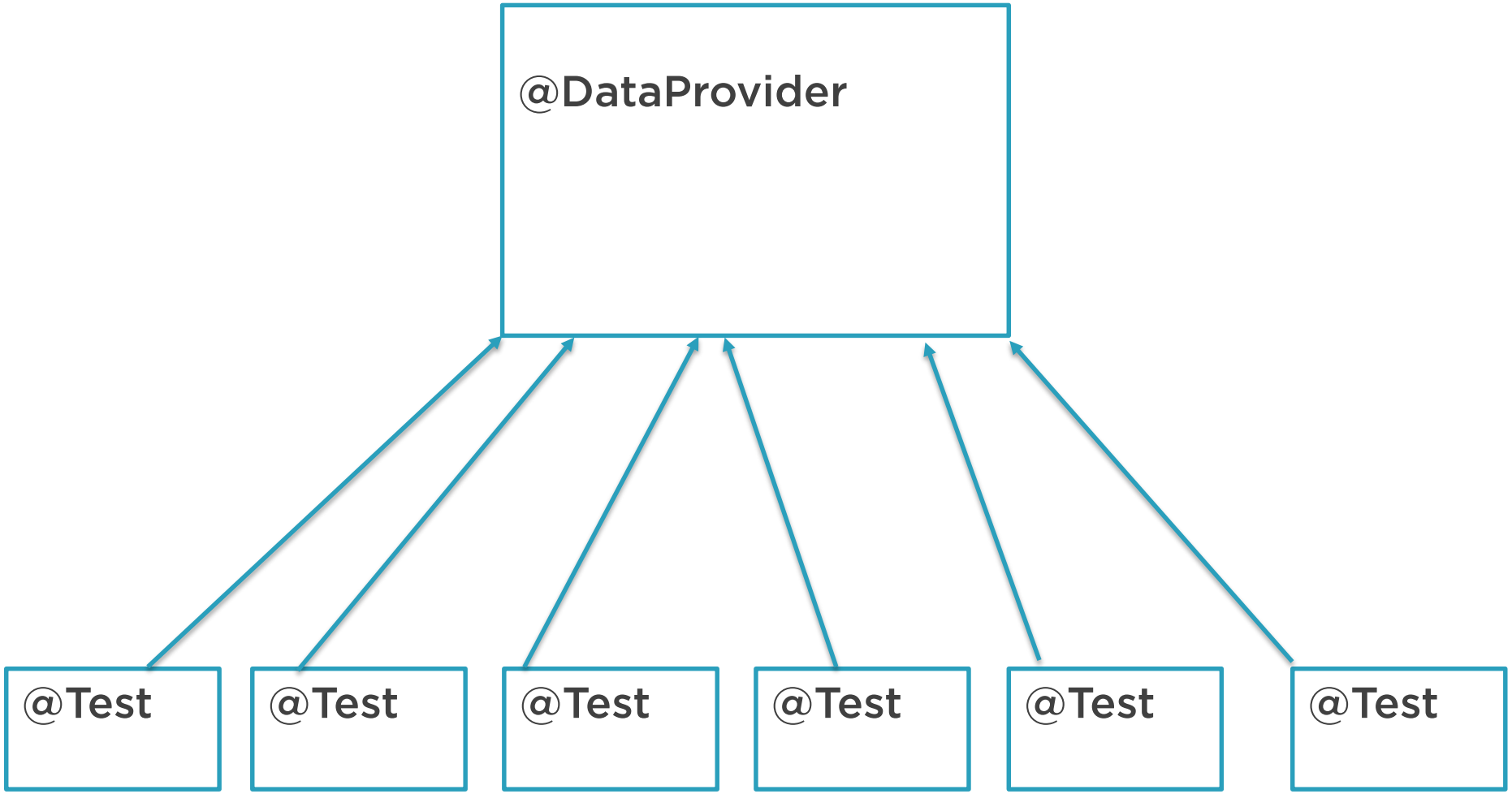
@Dataprovider

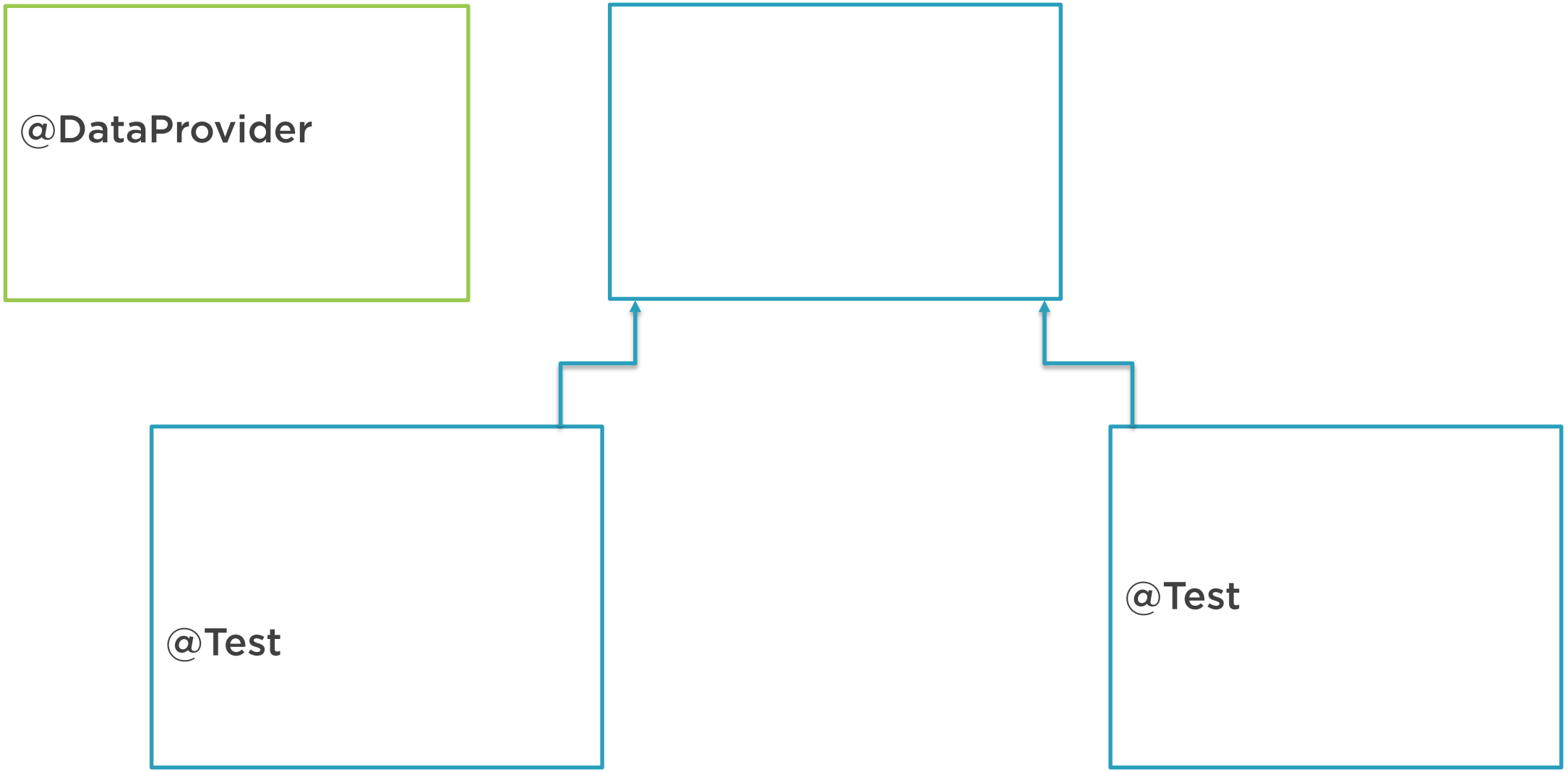
@Test

@DataProvider

@Test


@DataProvider
@Test
@Test


@DataProvider
@Test
@Test
@Test
@Test
@Test
@Test


@DataProvider
@Test
@Test

Summary

@DataProvider saves Lines of Code

@DataProvider Inheritance vs. Composition

@DataProvider Exceptions