



Национална програма
"Обучение за ИТ умения и кариера"
<https://it-kariera.mon.bg>

Министерството на
образованието и науката
<https://www.mon.bg>



Инжектиране на ЗАВИСИМОСТИ

Dependency Injection

Свързване и тестване [1/2]

- Да разгледаме следния код
 - Искаме да тестваме **единично поведение**

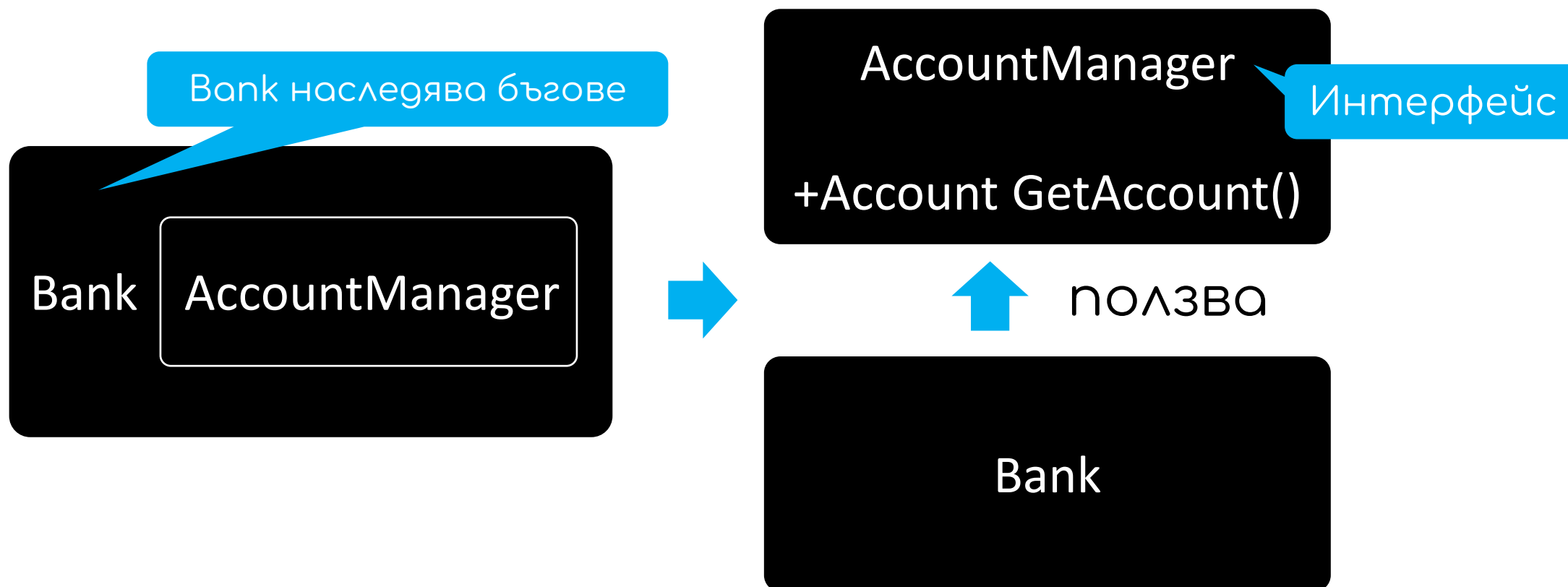
```
public class Bank {  
    private AccountManager accountManager;  
  
    public Bank() {  
        this.accountManager = new AccountManager();  
    }  
  
    public AccountInfo getInfo(String id) { ... }  
}
```

Конкретна
имплементация

Bank зависи от
AccountManager

Свързване и тестване [1/2]

- Трябва да намерим решение да **разграничим класовете**



Инжектиране на зависимост

Разграничава класовете и прави кода удобен за тестване

```
public interface IAccountManager {  
    Account Account { get; }  
}
```

Използайки интерфейс

```
public class Bank {  
    private IAccountManager accountManager;  
  
    public Bank(IAccountManager accountManager)  
    {  
        this.accountManager = accountManager;  
    }  
}
```

Независим от имплементацията

Инжектиране на зависимост

Цел: Изолиране на поведението за тестване

С групи думи, да **застопорим** всички **мърдащи се части**

```
[Test]
public void TestGetInfoById()
{
    AccountManager manager = new AccountManager()
    {
        public Account Account(String id) { ... }
    }

    Bank bank = new Bank(manager);
    AccountInfo info = bank.getInfo(id);

    // Assert...
}
```

Анонимен клас

Фалшива
имплементация на
интерфейс с
фиксирано поведение

Задача: Фалшиви Ахе и Dummy

- Тествайте дали герой **получава ХР** когато **мишената умре**
- За да направите това, първо:
 - Направете **Hero** класа **тестваем** (чрез **инжекция на зависимост**)
 - Направете **интерфейси** за Ахе и Dummy
 - Интерфейс IWeapon
 - Интерфейс ITarget
 - Създайте тест използвайки **фалшив Weapop** и **фалшив Dummy**

Решение: Фалшиви Axe и Dummy [1/4]

```
public interface IWeapon {  
    void Attack(Target target);  
    int AttackPoints { get; }  
    int DurabilityPoints { get; }  
}
```

```
public interface ITarget {  
    void TakeAttack(int attackPoints);  
    int Health { get; }  
    int GiveExperience();  
    bool IsDead();  
}
```

Решение: Фалшиви Ахе и Dummy [2/4]

```
// Hero: Инжекция на зависимост чрез конструктора
public Hero(String name, IWeapon weapon)
{
    this.name = name;
    this.experience = 0;
    this.weapon = weapon;
}
```

```
public class Axe : IWeapon {
    public void attack(ITarget target) { ... }
}
```


Решение: Фалшиви Axe и Dummy [3/4]

```
public class FakeTarget : ITarget
{
    public void TakeAttack(int attackPoints)
    {
    }
    public int Health => 0;
    public int GiveExperience()
    { return 20; }
    public bool IsDead()
    { return true; }
}
// TODO: Имплементирайте FakeWeapon
```

Решение: Фалшиви Axe и Dummy [4/4]

```
private const string HeroName = "Pesho"  
[Test]  
public void HeroGainsExperienceAfterAttackIfTargetDies()  
{  
    ITarget fakeTarget = new FakeTarget()  
    IWeapon fakeWeapon = new FakeWeapon()  
  
    Hero hero = new Hero(HeroName, fakeWeapon);  
    hero.attack(fakeTarget);  
  
    // Assert ...  
}
```

Фалшиви имплементации (Mocks)

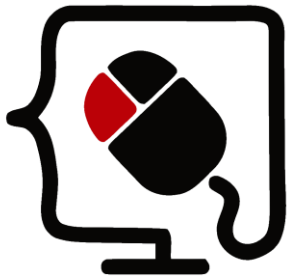
Не е четимо, трмаво и стереотипно

```
[Test]
public void TestRequiresFakeImplementationOfBigInterface() {
    // Arrange
    Database db = new BankDatabase() {
        // Твърде много методи...
    }

    AccountManager manager = new AccountManager(db);

    // Act...
    // Assert...
}
```

Не е удачно при големи интерфейси



Национална програма
"Обучение за ИТ умения и кариера"
<https://it-kariera.mon.bg>

Министерството на
образованието и науката
<https://www.mon.bg>



Документът е разработен за нуждите на Национална програма "Обучение за ИТ умения и кариера" на Министерството на образованието и науката (МОН) и се разпространява под свободен лиценз CC-BY-NC-SA (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).