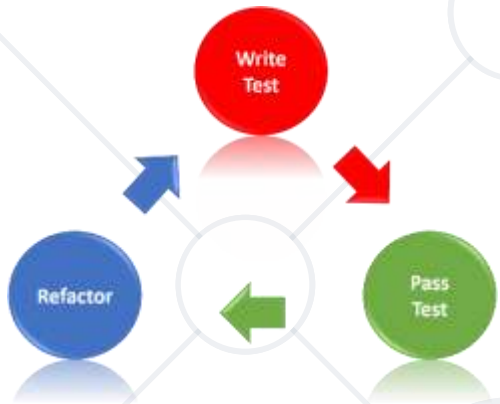


# Mocking and Test-Driven Development

Learn the "Test First" Approach to Coding



SoftUni Team  
Technical Trainers



**SoftUni**



Software University

<https://softuni.bg>

- Isolating Behaviors
- Mocking
- Test-Driven Development
  - Reasons to use TDD
  - Myths and Misconceptions

[sli.do](https://sli.do)

**#csharp-advanced**



# Isolating Behaviors

Dependencies

# Coupling and Testing (1)

- Consider testing the following code:
  - We want to test a **single behavior**

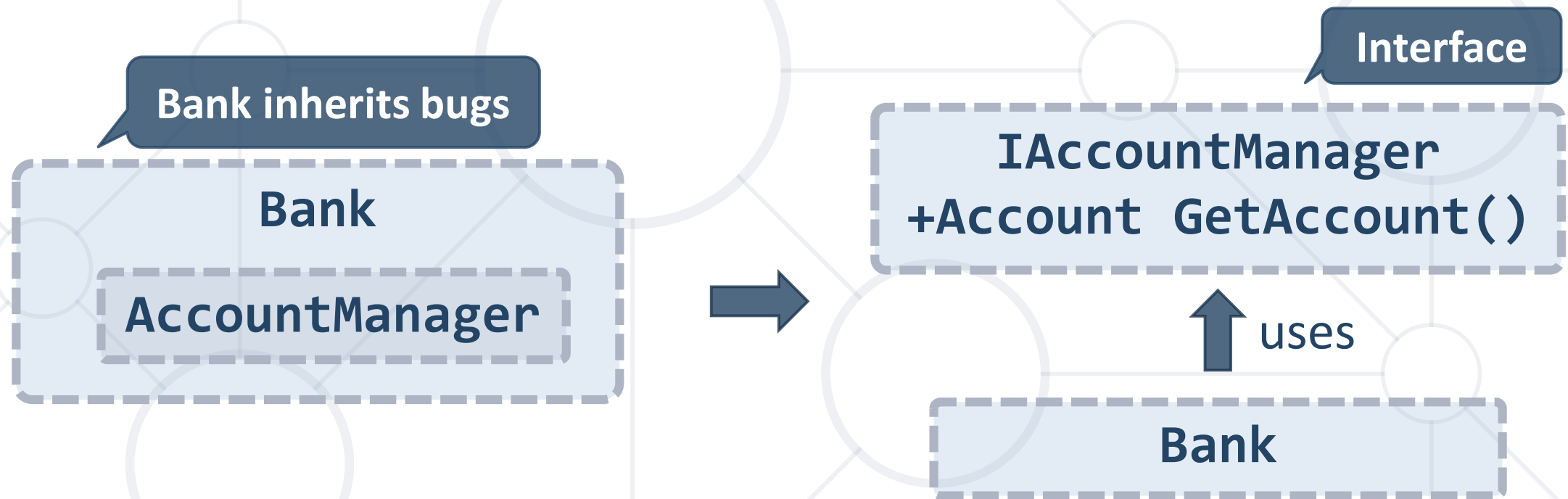
```
public class Bank {  
    private AccountManager accountManager;  
  
    public Bank()  
    {  
        this.AccountManager = new AccountManager();  
    }  
  
    public AccountInfo GetInfo(string id) { ... }  
}
```

Concrete  
Implementation

Bank **depends** on  
AccountManager

# Coupling and Testing (2)

- Need to find solution to **decouple classes**



- Decouples classes and makes code testable

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

Independent from  
Implementation

Injecting  
dependencies

- Not readable, cumbersome and has too much boilerplate

```
[Test]
public void TestRequiresFakeImplementationOfBigInterface() {
    // Arrange
    Database db = new BankDatabase()
    {
        // Too many methods...
    }

    AccountManager manager = new AccountManager(db);
    // Act...
    // Assert...
}
```

Not suitable for  
big interfaces



# Problem: Test GetGreeting

- Refactor **GetGreeting** to get the date time from outside
- Refactor **GetGreeting** to get the write location from outside
- Write tests to cover the **GetGreeting** method

```
public class GreetingProvider
{
    public string GetGreeting()
    {
        if (DateTime.Now.Hour < 12)
            Console.WriteLine("Good morning!");
        ...
    }
}
```



**Mocking**

- Mock objects **simulate behavior** of real objects
  - The object supplies non-deterministic results - **Time**
  - It has states that are difficult to create or reproduce - **Network**
  - It is slow - **Database**
  - It does not yet exist or may change behavior
  - It would have to include information and methods exclusively for testing purposes (and not for its actual task)

- Moq provides us with an easy way of **creating mock objects**
  - Simple to use
  - Strongly typed
  - Minimalistic

```
Mock<IContainer> mockContainer = new Mock<IContainer>();  
Mock<ICustomerView> mockView = new Mock<ICustomerView>();
```

```
Mock<ITarget> fakeTarget = new Mock<ITarget>();

fakeTarget
    .Setup(p => p.TakeAttack(It.IsAny<int>()))
    .Callback(() => hero.Weapon.DurabilityPoints -= 1);

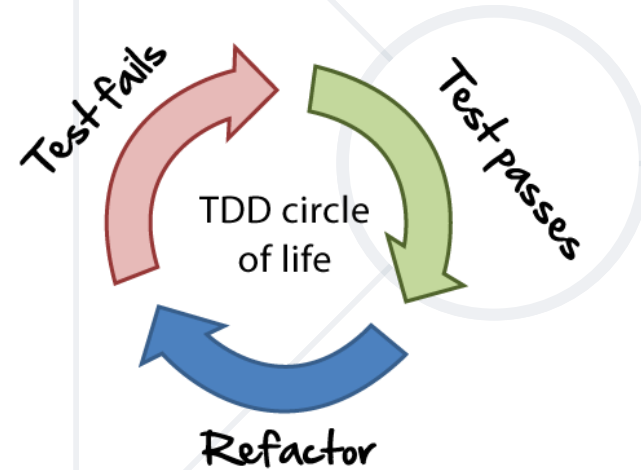
fakeTarget
    .Setup(p => p.Health)
    .Returns(0);
```



**Test-Driven Development**

# Unit Testing Approaches

- "Code First" (code and test) approach
  - Classical approach
- "Test First" approach
  - Test-driven development (TDD)



# The Code and Test Approach

**Write code**

**Write unit test**

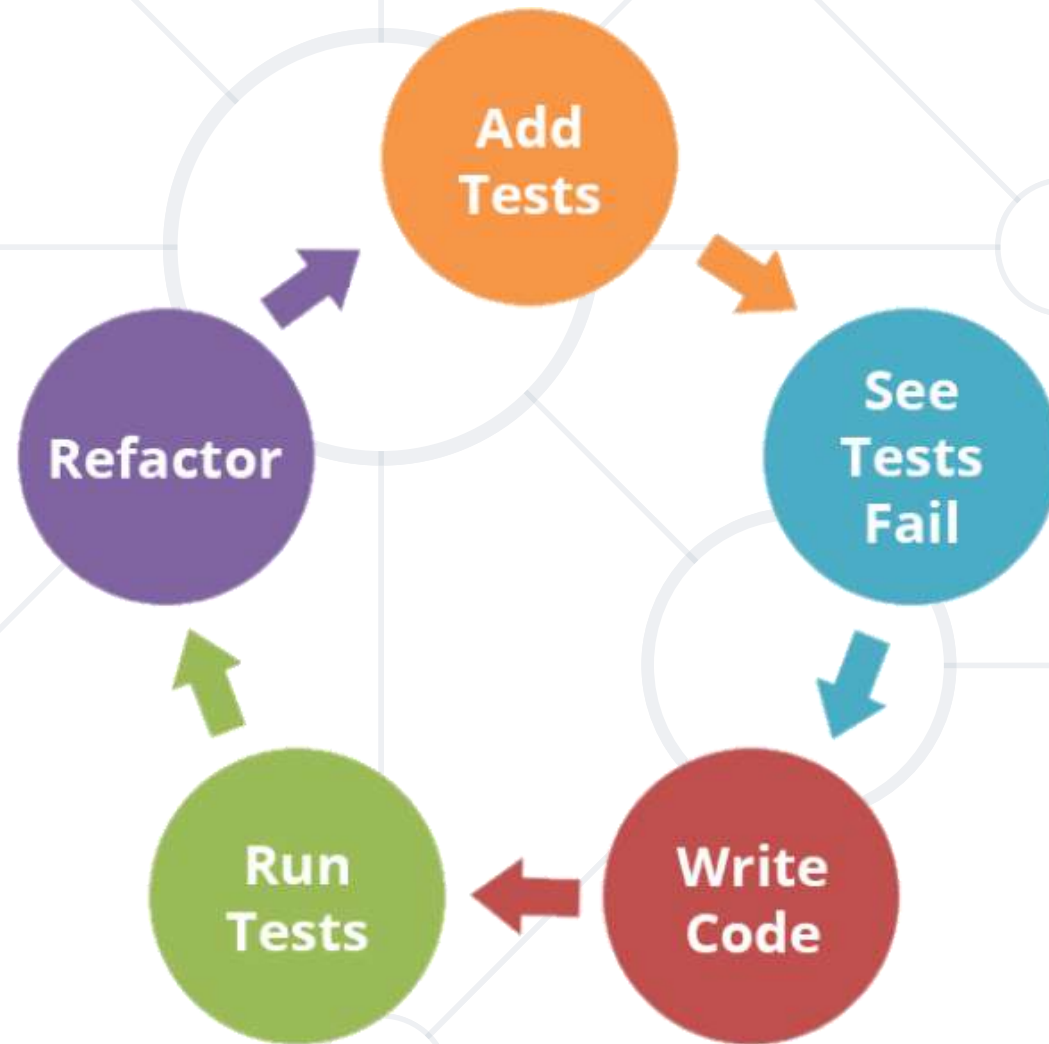
**Run and succeed**



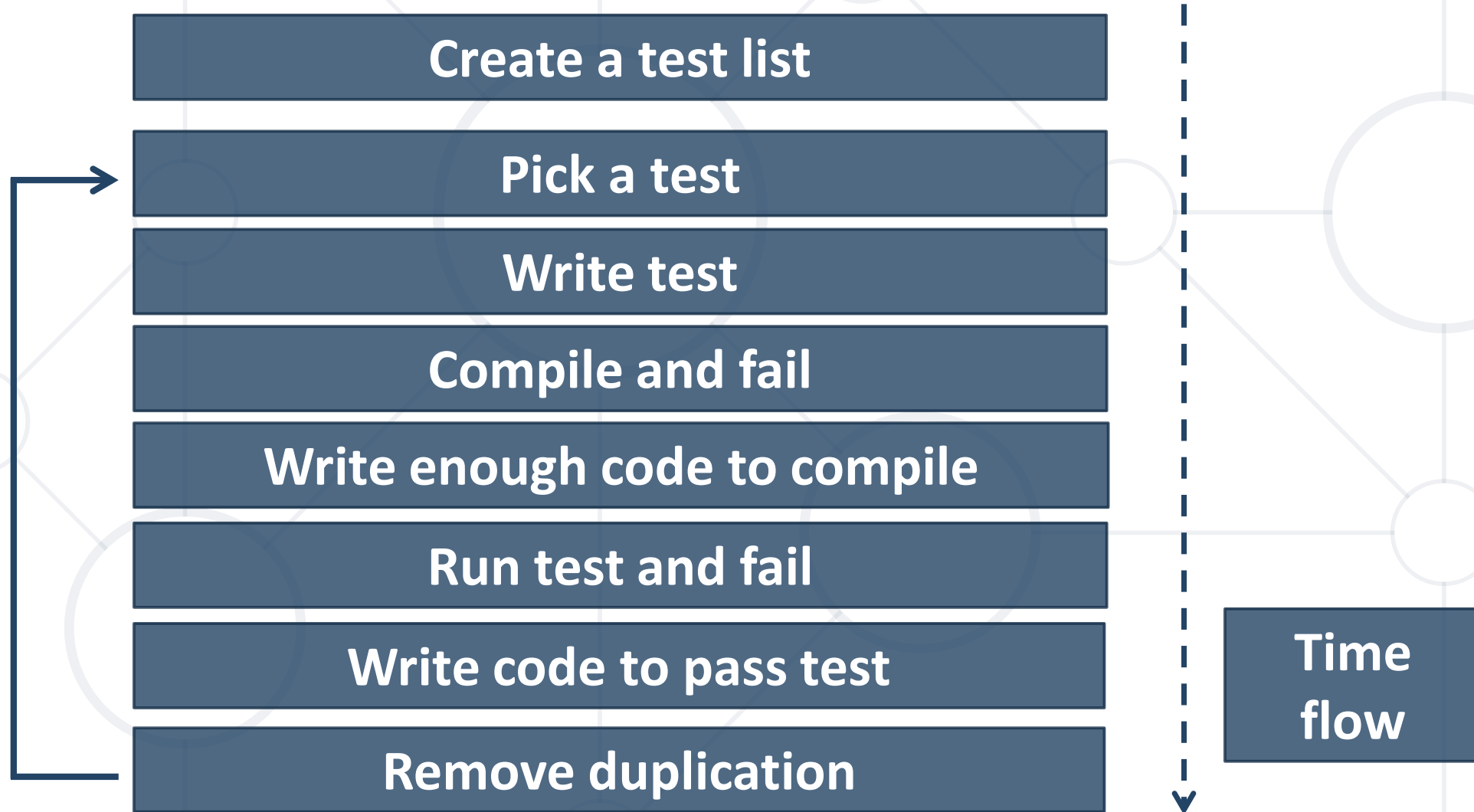
Time flow



# The Test-Driven Development Approach



# Test-Driven Development (TDD)



# Why TDD?

- **TDD** helps find design issues early
  - Avoids reworking
- Writing code to satisfy a test is a focused activity
  - **Less** chance of **error**
- **Tests** will be more **comprehensive** than if they are written after the code



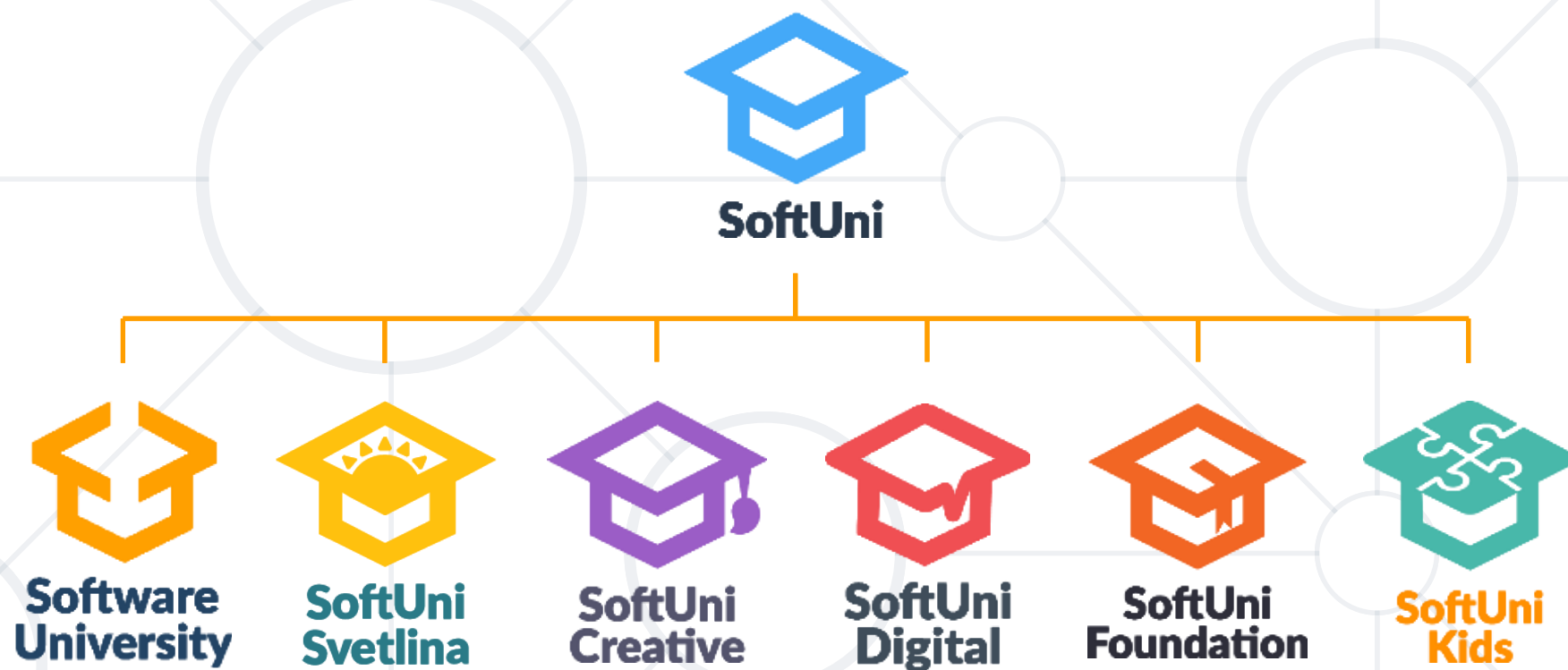
# Myths and Misconceptions

- You create a 100% regression test suite
- The unit tests form 100% of your design specification
- You only need to unit test
- TDD is sufficient for testing
- TDD doesn't scale (partially true)
  - Your test suite takes too long to run
  - Not all developers know how to test
  - Everyone might not be taking a TDD approach

- Code and Test
  - Write code, then test it
- Test-Driven Development
  - Write tests first
- Reasons to use TDD
  - Prevent some application design flaws
  - Manage complexity more easily



# Questions?



# SoftUni Diamond Partners



**SCHWARZ**



**SUPER  
HOSTING  
.BG**



**INDEAVR**  
Serving the high achievers

**Bosch.io**







- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](http://softuni.bg), [about.softuni.bg](http://about.softuni.bg)

- Software University Foundation

- [softuni.foundation](http://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)

- Software University Forums

- [forum.softuni.bg](http://forum.softuni.bg)



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

