# Упражнения: Компонентно тестване

Можете да проверите решенията си в **Judge системата**: <https://judge.softuni.org/Contests/4181/19-Unit-Testing-Advanced>

**Използвайте** предоставения **скелет** за всяко от упражненията. За всяка задача трябва да **добавите** правилната **проектна референция**.

## Подаръци

Ще получите скелет с клас **Present** и **Bag**. Класът **Bag** има методи, полета и конструктур, които работи правилно. Не е позволено да променяте класовете. Вашата задача е да изтествате класа **Book** с компонетни тестове.

### Насоки

1. В метода SetUp() създайте инстанция на класовете Bag и Present.
2. Създайте методи за тестване на метода Create(Present present) на Bag:
   1. Проверете дали методът хвърля ArgumentNullException() при подаване на null като параметър.
   2. Проверете дали методът хвърля InvalidOperationException() при добавяне на вече съществуващ подарък.
   3. Проверете дали успешното **добавяне** връща правилното **съобщение** и увеличава броя на **подаръците**.
3. Създайте методи за тестване на метода Remove(Present present) на Bag:
   1. Проверете дали методът връща true при успешно **премахване** и **променя** **броя** на **подаръците**.
   2. Проверете дали методът връща false при опит за **премахване** на **несъществуващ** **подарък**.
4. Тези насоки предоставят **примери** за **тестване** на **някои** от **методите** на класа Bag. Можете да продължите със същия подход за тестване на **другите методи**.

## Гараж

Ще получите скелет с класовете **Car** и **Garage**. Класът **Garage** има методи, полета и конструктур, които работи правилно. Класът **Car** има три свойства и конструктор. Не е позволено да променяте класовете. Вашата задача е да изтествате класа **Garage** с компонетни тестове.

### Насоки

1. Създайте методи за тестване на метода АddCar(Car car) на Garage:
   1. Добавете **кола** към **гараж** и проверете дали **броят на колите** се **увеличава**.
   2. Извикайте метода при наличие на **максимален брой коли** и проверете дали хвърля InvalidOperationException().
2. Създайте методи за тестване на метода FixCar(string carModel)на Garage:
   1. Създайте **гараж** и добавете **кола без проблеми** към **него**. Извикайте метода с **името** на **колата** и проверете дали **броят** на **проблемите** на **колата** е **нула**.
   2. Извикайте метода с **невалидно** **име** и проверете дали **хвърля** InvalidOperationException().
3. Създайте методи за тестване на метода RemoveFixedCar() на Garage:
   1. Проверете дали методът **премахва** **всички** **фиксирани** **коли** и дали връща **правилния брой**.
4. Създайте методи за тестване на метода Report() на Garage:
   1. Проверете дали методът връща **правилния отчет** за **не ремонтираните коли**.
5. Тези насоки предоставят **примери** за **тестване** на **някои** от **методите** на класа Bag. Можете да продължите със същия подход за тестване на **другите методи**.

## Магазин

Ще получите скелет с класовете **Shop** и **Smartphone**. Класът **Shop** има методи, полета и конструктур, които работи правилно. Класът **Smartphone** има три свойства и конструктор. Не е позволено да променяте класовете. Вашата задача е да изтествате класа **Shop** с компонетни тестове.

## Книги

Ще получите скелет с клас **Book**. Класът **Book** има методи, полета и конструктур, които работи правилно. Не е позволено да променяте класовете. Вашата задача е да изтествате класа **Book** с компонетни тестове.