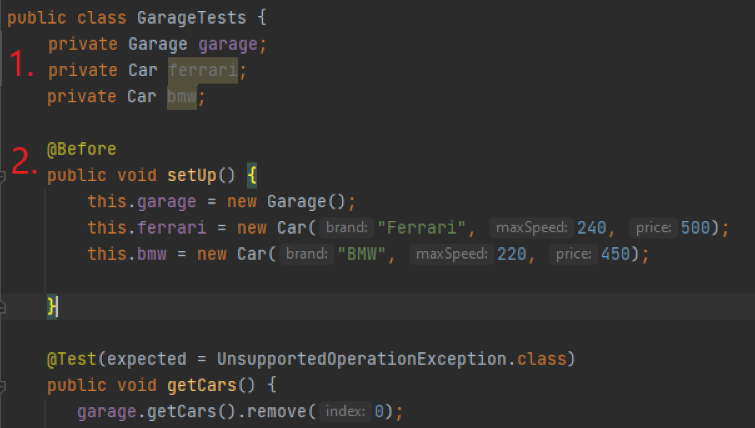
UNIT TESTING

# \* Подготовка входни данни



## А. Private Fields

Най-важното е да инициализираме класа, който ще тестваме. В случая това е **Gararage garage.** Ако инстанцията не приема параметри, можем да си създадем и обекти (“ferrari” и “bmw”), с които ще тестваме методите на инстнацията.

## B. @Before/publicv void setUp()

В setUp() най-важното е да вдигнем **инстанция на класа**, който тестваме. Така имаме достъп до методите на класа, който тестваме. Тук подобно на конструктор си инициализираме инстанцията. Ако се наложи си инициализираме и обекти, с които да тестваме.

# \* Тестове

# 1. TestCreateInstance

Aко класът, който ще тестваме,**е едновременно клас и обект**, който ще тестваме, можем да валидираме възможните варианти при създаването му. На практика тестваме **set()-ърите** на параметрите едновременно с тестването на setUp(). Това включва:

## А. Object **is NOT NULL** (Assert.asserNotNull(valid object))

## B. Object param one is **VALID** (“**name**”)

## C. Object param one is **NULL** (“name is **not null**”)

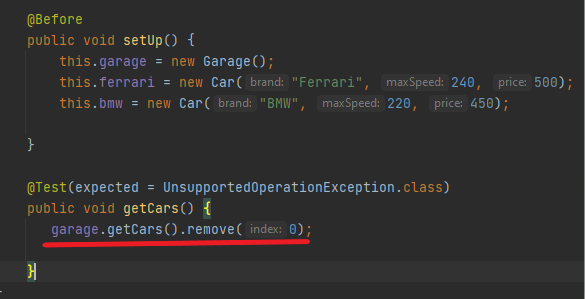
## D. Object param two is **VALID** (“health is **above 0**”)

## E. Objects param two is **INVALID** (health is **below 0)**

public class PlayerTests {  
 private Player playerTest;  
 private Player playerOne;  
 private Player playerTwo;  
  
 @Before  
 public void SetUp() {  
 playerTest = new Player("Pesho", 100);  
 playerOne = new Player("PlayerOne", 36);  
 playerTwo = new Player("PlayerTwo", 42);  
 }  
  
 //1. Object is not null  
 @Test  
 public void testSetUpCreateInstance(){  
 Assert.*assertNotNull*(playerOne);  
 }  
   
 //2. Object name is VALID  
 @Test  
 public void testPlayerUsername() {  
 String actualName = playerTest.getUsername();  
 String expectedName = "Pesho";  
 Assert.*assertEquals*(expectedName, actualName);  
 }  
  
 //3. Object name is NULL  
 @Test(expected = NullPointerException.class)  
 public void testPlayerNameNotNull() {  
 playerTest = new Player(null, 20);  
 }  
   
 //4.Object health is NOT valid  
 @Test(expected = IllegalArgumentException.class)  
 public void testPlayerHealthIsValid() {  
 new Player("Pesho", -1);  
 }  
  
 //5. Object health is VALID  
 @Test  
 public void testGetHealth() {  
 int expectedHealth = 100;  
 int actualHealth = playerTest.getHealth();  
 Assert.*assertEquals*(expectedHealth, actualHealth);  
 }

# 2. Test if unmodifiable list

Извикваме инстанцията **„garage“**, на която извикваме **метода getCars()**, който знаем, че връща **unmodifiable list**. Затова се опитваме да **.remove()** по индекс, при което получаваме желания **exception.**



# 3. Test addMethod()

## A. Test if we add null element

public class ComputerManagerTests {  
 private ComputerManager computerManager;  
 private Computer computer;  
 private Computer computer2;  
  
 @Before  
 public void setUp() {  
 this.computerManager = new ComputerManager();  
 computer = new Computer("DELL", "A4532", 300);  
 computer2 = new Computer("ASUS", "ROG", 500);  
 }

@Test(expected = IllegalArgumentException.class)  
 public void addNullComputer() {  
 this.computerManager.addComputer(null);  
}

## B.Test if we add existing element

public class ComputerManagerTests {  
 private ComputerManager computerManager;  
 private Computer computer;  
 private Computer computer2;  
  
 @Before  
 public void setUp() {  
 this.computerManager = new ComputerManager();  
 computer = new Computer("DELL", "A4532", 300);  
 computer2 = new Computer("ASUS", "ROG", 500);  
 }

@Test(expected = IllegalArgumentException.class)  
public void addSecondComputer() {  
 this.computerManager.addComputer(computer);  
 this.computerManager.addComputer(computer);  
}

## C. add() method works correct

Добавяме елемент (в случая computer) в колекцията и проверяваме с **this.computerManager().getCount()** дали **size() на компютрите** е колкото очакваме – а именно **единица**.

public class ComputerManagerTests {  
 private ComputerManager computerManager;  
 private Computer computer;  
 private Computer computer2;  
  
 @Before  
 public void setUp() {  
 this.computerManager = new ComputerManager();  
 computer = new Computer("DELL", "A4532", 300);  
 computer2 = new Computer("ASUS", "ROG", 500);  
 }

@Test  
public void testAdd() {  
 this.computerManager.addComputer(computer);  
 Assert.*assertEquals*(1, this.computerManager.getCount());  
}

# 4. Test remove()

## A. Test boolean remove()

*Example:*

public boolean removeGun(Gun gun) {  
 return this.guns.remove(gun);  
}

Добавяме си оръжие през инстанцията. После създаваме boolean променлива от резултата на removeGun(gun) и asser-тваме. Ако assertTrue върне (true) значи сме remove-нали успешно обекта.

@Test  
public void testRemoveGun() {  
 Gun gun = new Gun("Rifle", 13);  
 playerTest.addGun(gun);  
 boolean actualResult = playerTest.removeGun(gun);  
 Assert.*assertTrue*(actualResult);  
}

# 4. Test getters()

## A. Try get() with “null” params

Проверяваме параметрите, които ни се подават отвън. Проверяваме ги пооотделно, с два теста, тъй като са два параметра. Ако единият е null, то хвърялме exception.

public class ComputerManagerTests {  
 private ComputerManager computerManager;  
 private Computer computer;  
 private Computer computer2;  
  
 @Before  
 public void setUp() {  
 this.computerManager = new ComputerManager();  
 computer = new Computer("DELL", "A4532", 300);  
 computer2 = new Computer("ASUS", "ROG", 500);  
 }

@Test(expected = IllegalArgumentException.class)  
 public void testIfNullManufactureParam() {  
 this.computerManager.getComputer(null, "test");  
}  
  
 @Test(expected = IllegalArgumentException.class)  
 public void testIfNullModelParam() {  
 this.computerManager.getComputer("test", null);  
}

## B. Try Get() null object

През инстанцията се опитваме да достъпим до несъществуващ компютър, което ще върне exception. This.computerManager.get(“null,null”).computer на празна инстанция new ComputerManager().

public class ComputerManagerTests {  
 private ComputerManager computerManager;  
 private Computer computer;  
 private Computer computer2;  
  
 @Before  
 public void setUp() {  
 this.computerManager = new ComputerManager();  
 computer = new Computer("DELL", "A4532", 300);  
 computer2 = new Computer("ASUS", "ROG", 500);  
 }

@Test(expected = IllegalArgumentException.class)  
 public void TestNoneExistingComputer() {  
 this.computerManager.getComputer(computer.getManufacturer(), computer.getModel());  
}

## C. GetReturned() works correct

1. **Добавяме** си компютри чрез инстанцията и add() метода й. (this.computerManager.addComputer(computer1)).

2. Създаваме си returnedComputer по полетата на computer1.

3. Правим две отделни проверки дали полетата съвпадат.

public class ComputerManagerTests {  
 private ComputerManager computerManager;  
 private Computer computer;  
 private Computer computer2;  
  
 @Before  
 public void setUp() {  
 this.computerManager = new ComputerManager();  
 computer = new Computer("DELL", "A4532", 300);  
 computer2 = new Computer("ASUS", "ROG", 500);  
 }

@Test  
 public void testGetReturnCorrectly() {  
 this.computerManager.addComputer(computer);  
 this.computerManager.addComputer(computer2);

//Actual - > Тест Get() метод през инстанция.  
 Computer returnedComputer = this.computerManager.getComputer  
 (this.computer.getManufacturer(), this.computer.getModel());

Assert.*assertNotNull*(returnedComputer);

Assert.*assertEquals*(computer.getManufacturer(), returnedComputer.getManufacturer());  
 Assert.*assertEquals*(computer.getModel(), returnedComputer.getModel());  
}

## D. Try get(Object) => return object

Проверяваме дали като Try get() по obejct (в случая gun), успяваме да го върнем като резултат.

Example:

public Gun getGun(String name) {  
 Gun gun = this.guns.stream().filter(g -> g.getName().equals(name)).findFirst().orElse(null);  
 return gun;

На практика просто добавяме оръжието към инстнацията “**playerTest**” и после го достъпваме чрез getGun() по инстнацията**. Едновременно тестваме add() и get() методите**, които реферират към “**playerTest**”.

@Test  
 public void testGetValidGun() {  
 Gun gun = new Gun("Rifle", 123);  
 playerTest.addGun(gun);  
 Gun expectedGun = gun;  
 Gun actualGun = playerTest.getGun(gun.getName());  
 Assert.*assertEquals*(gun, actualGun);  
 }  
}

# 5. Test boolean methods

Винаги тестваме boolean methods с Assert.assertTrue()

В случая искаме да тестваме :

public boolean removeGun(Gun gun) {  
 return this.guns.remove(gun);  
}

Правим го :

@Test  
public void testGetGun() {  
 Gun gun = new Gun("Rifle", 13);  
 player.addGun(gun);  
 boolean actualResult = player.removeGun(gun);  
 Assert.*assertTrue*(actualResult);  
}

# 5. List<> test

## A. Проверка дали съдържаме елемент в List по параметър.

public class ComputerManagerTests {  
 private ComputerManager computerManager;  
 private Computer computer;  
 private Computer computer2;  
  
 @Before  
 public void setUp() {  
 this.computerManager = new ComputerManager();  
 computer = new Computer("DELL", "A4532", 300);  
 computer2 = new Computer("ASUS", "ROG", 500);  
 }

@Test  
 public void testGetByManufacture() {  
 this.computerManager.addComputer(computer);  
 this.computerManager.addComputer(computer2);

List<Computer> list =  
 //getComputersByManufacturer връща List<Computer> this.computerManager.getComputersByManufacturer(computer.getManufacturer());  
  
 Assert.*assertNotNull*(list);

Assert.*assertEquals*(list.get(0).getManufacturer(), computer.getManufacturer());  
}

# 6. Test methods

## A. TakeDamage

public void takeDamage(int damage) {  
 if (this.health <= 0) {  
 throw new IllegalStateException("Player is dead!");  
 }  
  
 if (this.health - damage < 0) {  
 this.health = 0;  
 } else {  
 this.health -= damage;  
 }  
}

Отново трябва да минем през инстанцията на класа (playerTest). Упражняваме методът takeDamage() върхуя playerTest. После assert-ваме колко с колко кръв е останал playerTest.

@Test  
public void testTakeDamageWorkCorrect() {  
 playerTest.takeDamage(50);  
 int actualHealth = playerTest.getHealth();  
 int expectedHealth = 50;  
 Assert.*assertEquals*(expectedHealth, actualHealth);  
}

7. Test BuyProduct()

Example:

public double buyProduct(String name, int quantity) {  
 Product product = this  
 .products  
 .stream()  
 .filter(c -> c.getName().equals(name))  
 .findFirst()  
 .orElse(null);  
  
 if (product == null) {  
 throw new IllegalArgumentException("There is no such product.");  
 }  
  
 if (product.getQuantity() < quantity) {  
 throw new IllegalArgumentException("There is not enough quantity of this product.");  
 }  
  
 int productQuantity = product.getQuantity();  
 double finalPrice = product.getPrice() \* quantity;  
 product.setQuantity(productQuantity - quantity);  
 return finalPrice;  
}