דוגמא

נרצה לכתוב מחלקה המייצגת כלב. הנה המחלקה:

```
public class Dog {
	//TODO
	public Dog(int age) {
		//TODO
	}

	public void notTakenForAWalk(int hours) {
		//TODO
	}

	public void hoursCleaningFloors(int hours) {
		//TODO
	}

	public String houseCondition() {
		//TODO
	}
}
```

אך, לפי מתודולוגיית הפיתוח, תחילה אנו נכתוב את בדיקת היחידה למחלקה ורק לאחר מכן את המחלקה עצמה.

לפי המתודולוגיה בה אנו עובדים, הבדיקות מורכבות משני חלקים – סיפור וקידוד בדיקה.

הנה סיפור לדוגמא:

```
Given a Dog of age 6
When the dog is not taken out for a walk, and the number of
hours is 5   (זהו משפט אחד)
Then the house condition is clean
```

נקודד מחלקת בדיקה:

```
public class DogStoryTest {
      protected Dog dog;
      @Given("a Dog of age &age")
      public void aDog(Integer age) {
            dog = new Dog(age);
      }

      @When("the dog is not taken out for a walk, and the
number of hours is &hours")
      public void dogNotTakenForAWalk(Integer hours) {
            dog.notTakenForAWalk(hours);
      }

      @Then("the house condition is &condition")
      public void theHouseCondition(String condition) {
      Assert.assertEquals(condition, dog.houseCondition());
      }
}
```

ניתן לראות שכל אנוטציה מקבלת מחרוזת. בסוף כל מחרוזת המילה האחרונה מייצגת פרמטר (אותו פרמטר שהגדרנו עבורו ערך במשפט בסיפור). פרמטר מסומן על ידי התו & בראשיתו. שם הפרמטר כשם המשתנה שהמתודה מקבלת. כל מחרוזת כזו תותאם למשפט חוקי בסיפור.

עבור כל סיפור, יש להריץ לפני כל רצף של When גיבוי לאובייקט של מחלקת הבדיקה. כלומר, אם הסיפור היה כתוב כך:

```
Given a Dog of age 6
When the dog is not taken out for a walk, and the number of
hours is 5   (זהו משפט אחד)
Then the house condition is clean

When the dog is not taken out for a walk, and the number of
hours is 22   (זהו משפט אחד)
Then the house condition is smelly
```

סדר הפעולות שנעשה יהיה כך:

- Create object TEST of class DogStoryTest
- aDog (6)
- BackUp object TEST
- dogNotTakenForAWalk(5)
- theHouseCondition(clean)
- if *assertEquals fails and throws* `ComparisonFailure` restore object TEST

- BackUp object TEST
- dogNotTakenForAWalk(22)
- theHouseCondition(smelly)
- if *assertEquals fails and thorws* `ComparisonFailure` restore object TEST
- if any Then sentence has failed throw StoryTestException

הרחבת הדוגמא:

ניקח מחלקת בדיקות יותר מסובכת:

```
public class DogStoryDerivedTest extends DogStoryTest {
	@When("the house is cleaned, and the number of hours is
&hours")
	private void theHouseCleaned(Integer hours) {
		dog.hoursCleaningFloors(hours);
	}

	public class InnerClass extends DogStoryTest {
		@Given("a Dog that his age is &age")
		public void aDog(Integer age) {
			dog = new Dog(age);
		}
	}
}
```

נקח את הסיפור:

```
"Given a Dog of age 6\n"
"When the dog is not taken out for a walk, and the number of
hours is 15\n"
"When the house is cleaned, and the number of hours is 11\n"
"Then the house condition is clean"
```

ונריץ את הסיפור באמצעות המתודה **testOnNestedClasses** עם מחלקת הטסט
DogStoryDerivedTest, סדר הפעולות שנעשה יהיה כך:

- Create object TEST of class DogStoryDerivedTest
- DogStoryTest::aDog (6)
- BackUp object TEST
- DogStoryTest::dogNotTakenForAWalk(15)
- DogStoryDerivedTest::theHouseCleaned(11)
- DogStoryTest::theHouseCondition(clean)
- if *assertEquals fails and throws* ComparisonFailure restore object TEST
- if any Then sentence has failed throw StoryTestException

או אם ניקח את הסיפור:

```
"Given a Dog that his age is 6\n"
"When the dog is not taken out for a walk, and the number of
hours is 5\n"
"Then the house condition is clean"
```

ונריץ את הסיפור באמצעות המתודה **testOnNestedClasses** עם מחלקת הטסט
DogStoryDerivedTest, סדר הפעולות שנעשה יהיה כך:

- Create object TEST of class DogStoryDerivedTest::InnerClass
- DogStoryDerivedTest::InnerClass::aDog (6)
- BackUp object TEST
- DogStoryTest::dogNotTakenForAWalk(5)
- DogStoryTest::theHouseCondition(clean)
- if *assertEquals fails and thorws* ComparisonFailure restore object TEST
- if any Then sentence has failed throw StoryTestException

עכשיו ניתן לממש את המחלקה Dog:

```
public class Dog {
	private int age,hoursCleaningFloors;
	private boolean peeOnFloor;

	public Dog(int age) {
		this.age = age;
		this.hoursCleaningFloors=0;
		this.peeOnFloor = false;
	}

	public void notTakenForAWalk(int hours) {
		if(hours>8&& this.age > 5){
			this.peeOnFloor=true;
			hoursCleaningFloors=0;
		}
	}

	public void hoursCleaningFloors(int hours) {
		if(this.peeOnFloor){

	this.hoursCleaningFloors=this.hoursCleaningFloors+hours;
		}
		if(this.hoursCleaningFloors>10){
			this.peeOnFloor=false;
			this.hoursCleaningFloors=0;
		}
	}

	public String houseCondition() {
		if(this.peeOnFloor){
			return "smelly";
		}
		return "clean";
	}
}
```

ולהריץ עליו את הבדיקות. תעשו זאת בעזרת הטסטים שניתנו לכם.