Testy jednostkowe

Wzorce w projektowaniu testów

Inicjalizacja – przygotowywanie test fixture

- @Before, @BeforeClass
- Object Mother
- Test Builder

@Before, @BeforeClass

- Mechanizm frameworka JUnit pozwalający na wyciągnięcie „przed nawias” inicjalizacji danych dla potrzeb testów

```
public class BowlingGameTest {

    private Game g;

    @Before
    public void setUp() {
        g = new Game();
    }

    @Test
    public void gameWithNoPinsDown_shouldScore0() {

        rollMany(20, 0);

        assertEquals(0, g.score());
    }
```

- Ograniczone do jednej klasy testu

Wzorce projektowania testów

- Jednym z większych problemów w opracowaniu testów jest przygotowanie test fixture:
  - Nieczytelność, szczególnie w przypadku dużej liczby (struktur) danych
  - Duży potencjał naruszania zasady DRY

```
Invoice invoice = new Invoice(
    new Recipient("Sherlock Holmes",
        new Address("221b Baker Street",
                    "London",
                    new PostCode("NW1", "3RX"))),
    new InvoiceLines(
        new InvoiceLine("Deerstalker Hat",
            new PoundsShillingsPence(0, 3, 10)),
        new InvoiceLine("Tweed Cape",
            new PoundsShillingsPence(0, 4, 12))));
```

  - W kolejnym teście będzie tylko inny Address -> CTRL+C CTRL+V

ObjectMother

- Cechy
  - Dostarcza prefabrykowanych obiektów dla potrzeb testów (a'la fabryka)
  - Pozwala na dostosowywanie obiektów
  - Udostępnia mechanizmy aktualizujące i usuwające obiekty (testy integracyjne)
- Zastosowanie
  - Modyfikowalne struktury

ObjectMother

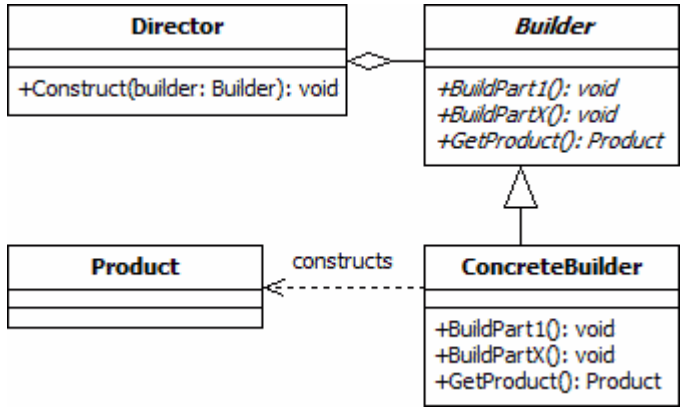
```
Invoice invoice = new Invoice(
    new Recipient("Sherlock Holmes",
        new Address("221b Baker Street",
                    "London",
                    new PostCode("NW1", "3RX"))),
    new InvoiceLines(
        new InvoiceLine("Deerstalker Hat",
            new PoundsShillingsPence(0, 3, 10)),
        new InvoiceLine("Tweed Cape",
            new PoundsShillingsPence(0, 4, 12))));
```

```
Invoice invoice = TestInvoices.newDeerstalkerAndCapeInvoice();
```

- Uważany za antywzorzec
  - Zachęca do budowania wielkich klas

Wzorzec Builder

- Pozwala na odseparowanie konstrukcji złożonego obiektu od jego reprezentacji. Dzięki temu ten sam proces budowy można wykorzystać do tworzenia różnych reprezentacji.



Skomplikowane konstruowanie obiektów (struktur danych)

- Opcje
  - Dużo konstruktorów wprowadzających opcjonalność parametrów (permutacje)
  - Prosty konstruktor wspierany przez settery (potencjalny niespójny stan)
  - Wykorzystanie wzorca Builder.
    - Klient tworzy obiekt builder i wywołuje metody ustawiające wybrane parametry (pozostałe przyjmują wartości domyślne). Ostatecznie wołana jest metoda konstruująca.

Test Builder

- Wykorzystanie wzorca Builder do przygotowania test fixture
- https://github.com/mto-lab/testBuilder
- https://github.com/sabram/BuilderPattern/