Testy jednostkowe

rozmaitości

Problematyczne punkty testowania

- Od czego się izolować?
- Metody statyczne, konstruktory, itp.
- Logika wykorzystująca upływ czasu

Co jest zależnością w testowaniu jednostkowym?

- Radykalne podejście:
  - Przypadek testowy powinien przetestować jedną ścieżkę pojedynczej metody. Gdy wykonanie metody wywołuje metodę innego obiektu mamy do czynienia z zależnością.
- Rozsądne (realne) podejście:
  - Zależnością jest coś co wykracza poza logiczną jednostkę wykonania. Np. należy do innej warstwy architektonicznej.
  - Np. Struktura danych nie jest zależnością i nie należy jej dublować.

Testowanie a kod spadkowy

- Nowy kod można dostosować do potrzeb testowania
- Ale istniejący kod może: nie być dostosowany do automatyzacji testów; nie być dobrze zaprojektowany
  - Może zaistnieć potrzeba:
    - przetestowania metody statycznej, prywatnej, konstruktora
    - Wyciszenia niechcianego zachowania (jak w przypadku leków immunosupresyjnych)
  - Nie jest to zwykle domyślna przestrzeń pokrywana przez frameworki izolujące

PowerMock

- Rozszerzenie frameworków izolujących (np. Mockito) o obsługę „trudnych przypadków”
- Zaliczany do narzędzi „dla ekspertów”
- Ale dobrze o nim wiedzieć

Przykład metoda statyczna

https://github.com/mto-lab/testisolation

Upływ czasu a testy

- Testy jednostkowe powinny wykonywać się szybko.
- Logika upływu czasu powoduje, że wykonanie niektórych testów może trwać … (minuty, godziny, …)
- Zegar systemowy znajduje się w przestrzeni poza kontrolą
  - jest zewnętrzną zależnością
  - izolacja

Bieżący czas w Java

- `System.currentTimeMillis()`

- Java.util.Date

```
new Date() == new Date(System.currentTimeMillis())
```

- Biblioteka Joda Time

```
new DateTime() == new
DateTime(System.currentTimeMillis())
```

Fake system clock

```
public interface TimeSource {
  long currentTimeMillis();
}

public final class DefaultTimeSrc implements TimeSource {
       @Override
       public long currentTimeMillis() {
              return System.currentTimeMillis();
       }
}

public final class AdvancedTimeSrc implements TimeSource {
       private static final long ONE_DAY = 24*60*60*1000;
       @Override
       public long currentTimeMillis() {
              return System.currentTimeMillis() + ONE_DAY;
       }
}
```

Bieżący czas w Java

- Java.util.Date

```
new Date() ->
new Date(SysTime.getTimeSource().currentTimeMillis())
```

- Biblioteka Joda Time

```
new DateTime()->
new DateTime(SysTime.getTimeSource().currentTimeMillis())
```