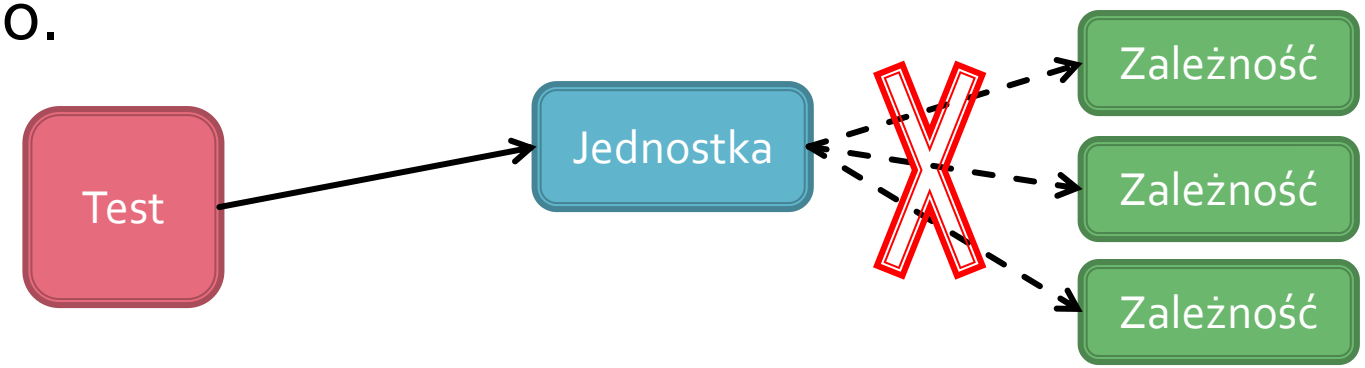
Testy jednostkowe

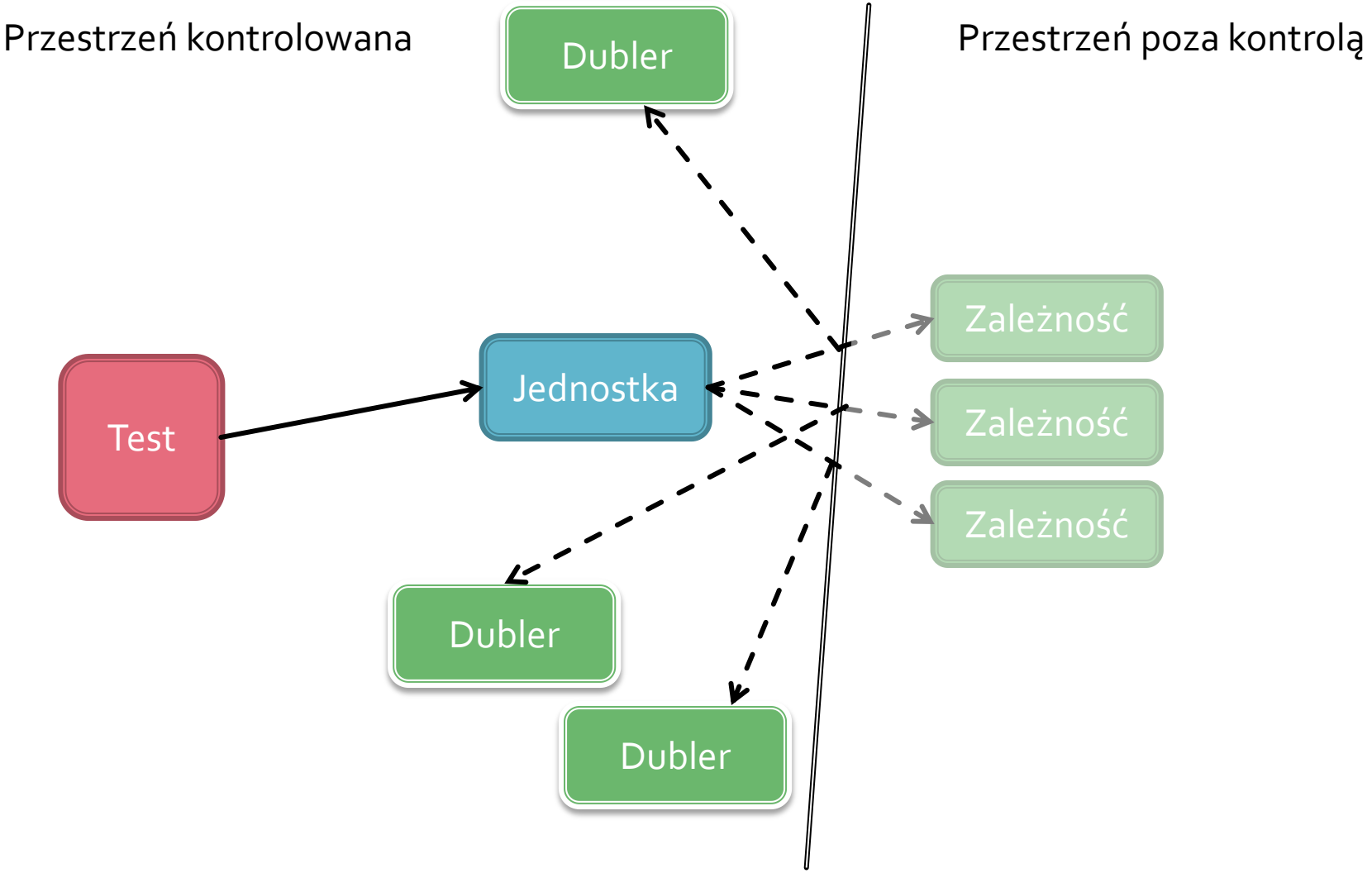
Izolacja

Zakres testu - przypomnienie

- Test jednostkowy
  - Testowanie zachowania poszczególnych elementów z których zbudowane jest oprogramowanie
  - Testowanie w izolacji
  - Sprawdzenie czy logika działa **dokładnie** tak jak założono.



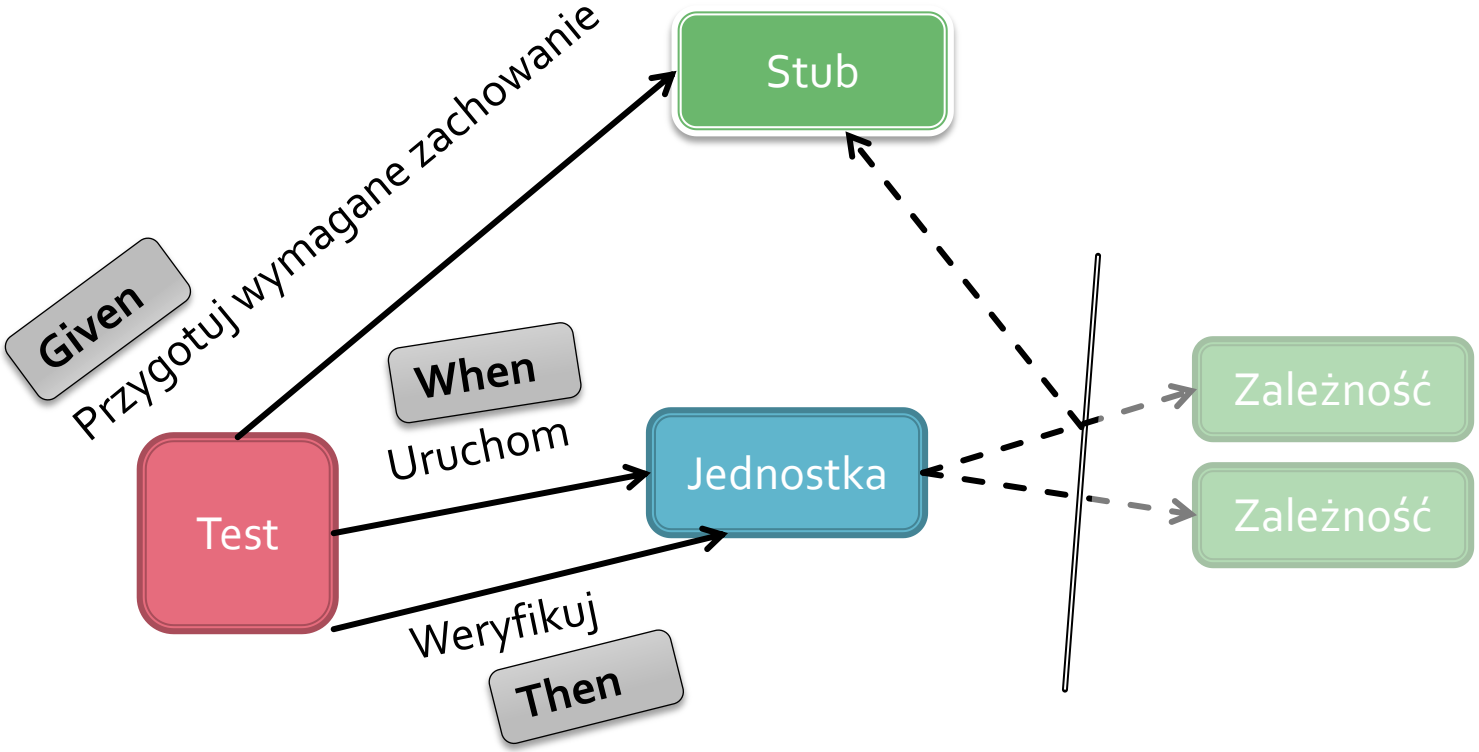
Izolacja testu – imitowanie rzeczywistych zależności



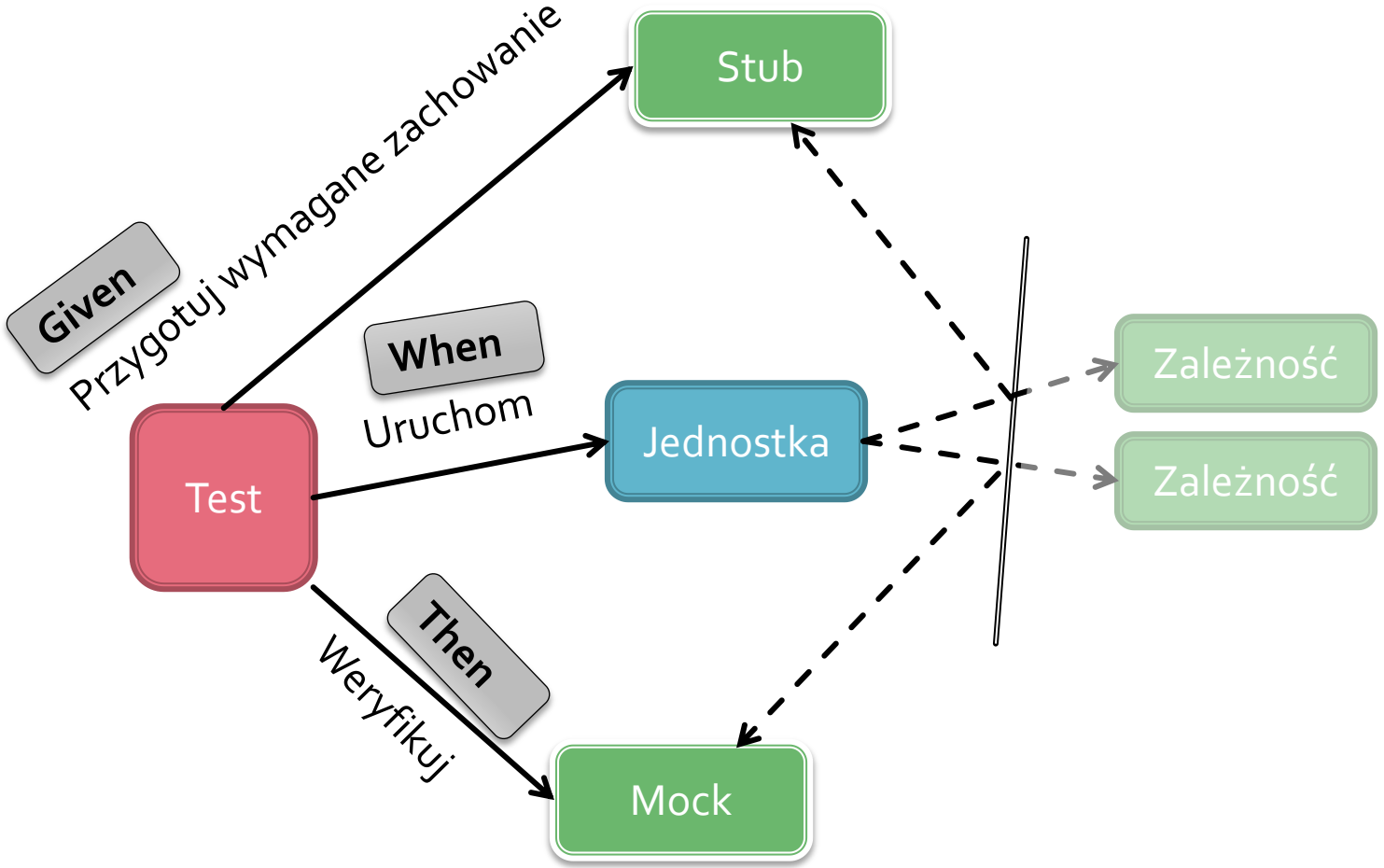
Typy dublerów

- **Dummy** - zaślepka
  - Obiekt nie wykorzystywany w ramach testu ale wymagany do uruchomienia testu
- **Fake** - imitacja
  - obiekt z uproszczoną (często nieprodukcyjną) implementacją
- **Stub** - pieniek
  - Obiekt zaimplementowany w zakresie potrzebnym dla testu – działający w ustalony sposób.
  - Może zapamiętywać pewne informacje nt. interakcji – nosi wówczas nazwę **Spy – szpieg**.
- **Mock** - pozorant
  - Obiekt z określonymi oczekiwaniami dotyczącymi zachowania testowanego kodu w kontekście zależności

Stub


Stub
Given
Przygotuj wymagane zachowanie
When
Uruchom
Test
Jednostka
Zależność
Zależność
Weryfikuj
Then

Mock


Stub
Given
Przygotuj wymagane zachowanie
When
Uruchom
Test
Jednostka
Zależność
Zależność
Then
Weryfikuj
Mock

Implementacja dublerów

- Dotyczy zazwyczaj typów Stub i Mock
- Dużo nadmiarowego kodu na którego tworzeniu i **pielęgnacji** musimy się skupiać
- Często prowadzi do problemów „architektury" testów, braku czytelności
  - Zajmujemy się nie tym co trzeba

Framework izolujący - isolation framework

- Zestaw mechanizmów API ułatwiających opracowywanie obiektów typu Stub i Mock. Zwalnia programistę z potrzeby pisania kodu symulującego interakcję obiektów.
- Standardowo wykorzystywany mechanizm w automatyzacji testów.

Wzorce działania frameworków izolujących w kontekście weryfikacji interakcji (typ: mock)

- Record-replay-verify (Expect-run-verify)
  - Wstępne nagrywanie akcji wykonywanych na dublerach, następnie ich odtwarzanie i weryfikacja
  - Mało intuicyjny, pierwotny model w starszych frameworkach.

```
// given
HistoryProvider historyProvider = createMock(HistoryProvider.class);
historyProvider.collect(anyObject(), anyObject());
expectLastCall().times(2);

replay(historyProvider);

// when do something with mocks

// then
verify(historyProvider);
```

Wzorce działania frameworków izolujących w kontekście weryfikacji interakcji (typ: mock)

- Arrange - Act - Assert
  - Utworzenie dublera, wykonanie testu, weryfikacja wykonanych operacji (opcjonalna)
  - Nie ma potrzeby definiowania zachowania przed uruchomieniem testu

```
// given
HistoryProvider historyProvider = mock(HistoryProvider.class);

// when do something with mock

// then
verify(historyProvider, times(2)).collect(anyObject(), anyObject());
```

Mockito – framework izolujący

- Arrange-act-assert
- Wykorzystuje framework hamcrest
- Termin mock wykorzystywany w tak w kontekście symulowania wymaganego zachowania zależności (stub) jak i weryfikacji poprawności interakcji z zależnością (mock).
- Dublowany może być interfejs, klasa, obiekt

Stub w Mockito

- Utworzenie pieńka - given

```
LinkedList mockedList = mock(LinkedList.class);
```

- Zdefiniowanie zachowania dla wybranych metod - given

```
when(mockedList.get(0)).thenReturn("first");
```

- Uruchomienie testu - when

```
String result = mockedList.get(0)
```

- Weryfikacja testu – then

```
assertThat(result, equalTo("first"));
```

Mock w Mockito

- Utworzenie pozoranta - given

```
List mockedList = mock(List.class);
```

- Uruchomienie testu - when

```
mockedList.add("one");
mockedList.clear();
```

- Weryfikacja testu – then

```
verify(mockedList).add("one");
verify(mockedList).clear();
```

Mockito – cechy

- Elastyczne dopasowywanie argumentów

```
when(mockedList.get(anyInt())).thenReturn("element");
```

- Różne zachowanie dla kolejnych wywołań

```
when(mock.someMethod("some arg")) .thenThrow(new
RuntimeException()) .thenReturn("foo");
```

- Weryfikacja liczby wywołań metody

```
verify(mockedObj, times(2)).add("twice");
verify(mockedObj, atLeast(2)).add("five times");
```

  - atMost(number), never()
- @Mock

Mock vs Spy

- Mock domyślnie zaślepia wszystkie metody dublowanej zależności
  - Dublowanie interfejsów
  - Można przekierować do oryginalnej metody
    - callRealMethod()
- Spy domyślnie wywołuje oryginalne metody dublowanej zależności
  - Dublowanie wybranych metod istniejących obiektów
  - Trzeba jawnie zaślepiać

Spy - szpieg

```
List list = new LinkedList();
List spy = spy(list);

when(spy.size()).thenReturn(100);

spy.add("one");
spy.add("two");
spy.get(0));
spy.size();

verify(spy).add("one");
verify(spy).add("two");
```