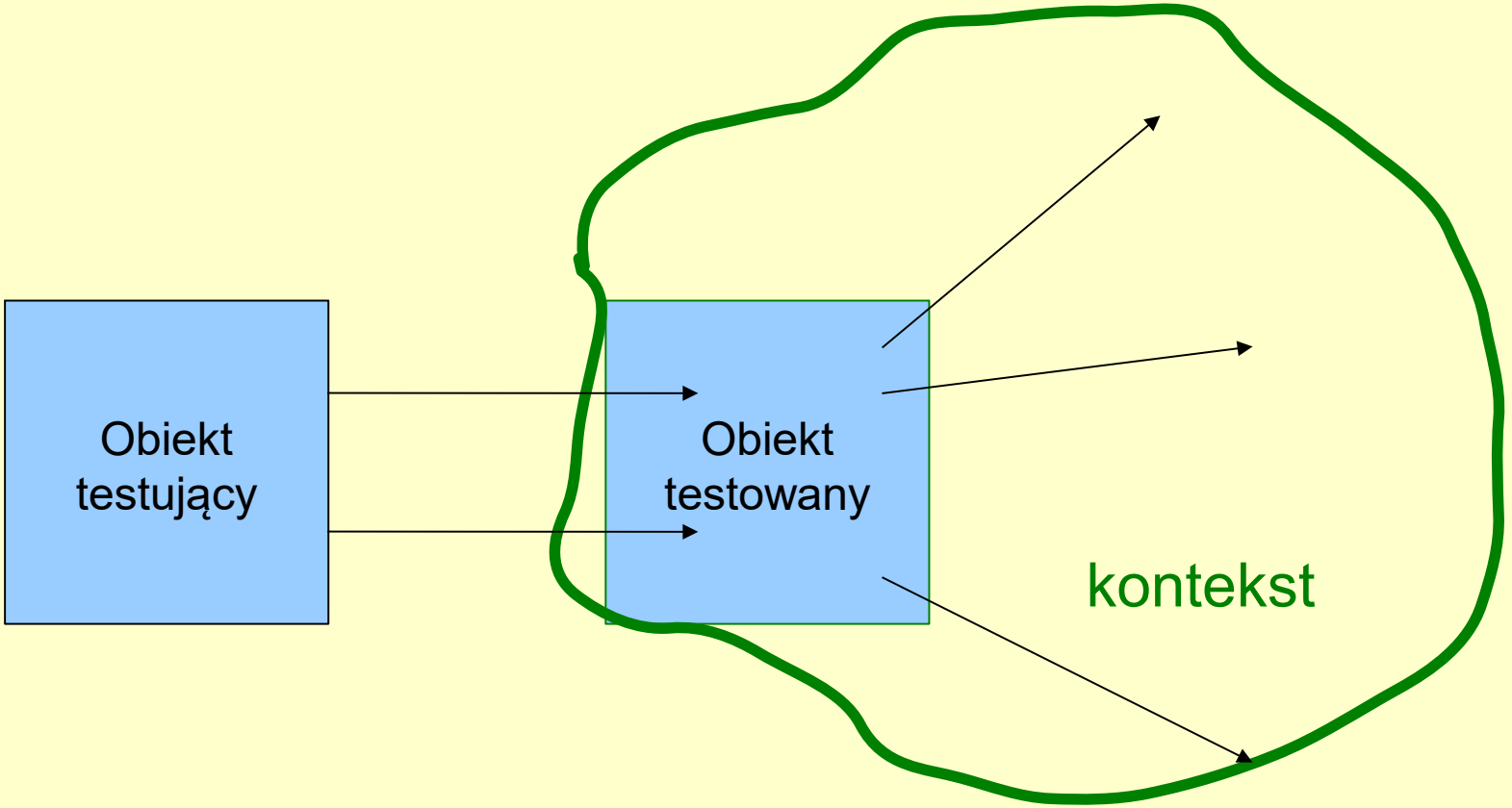
Testowanie

Obiekty zastępcze

Plan

- Przesłanki,
- Koncepcja,
- Mockito,
- unittest.mock.

Przesłanki



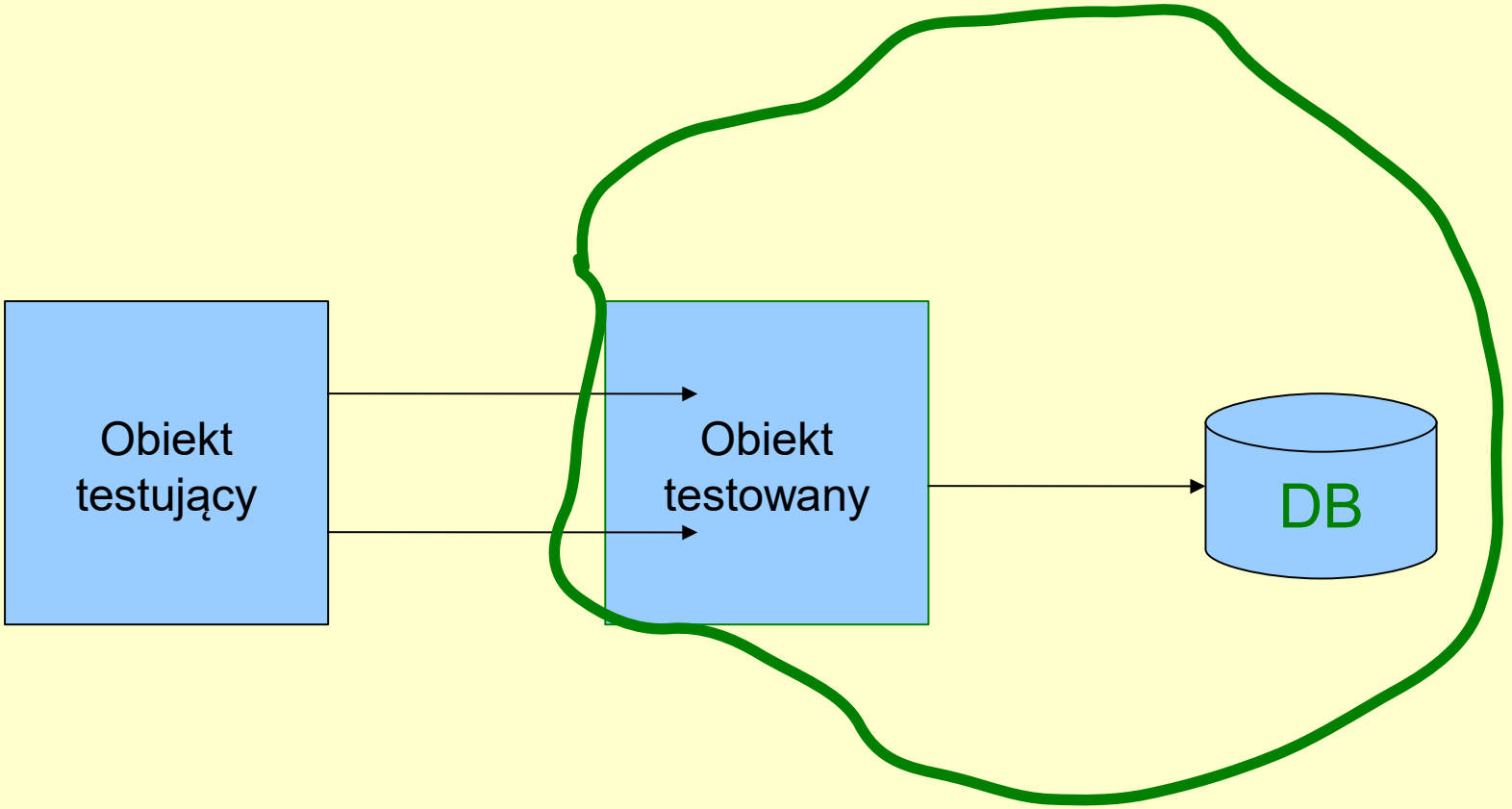
Przesłanki

- Klasa testowana istnieje w pewnym kontekście,
- Klasa testowana posiada zależności, bez których nie może istnieć lub nie może poprawnie funkcjonować,
- Testując klasę, testujemy też jej zależności.

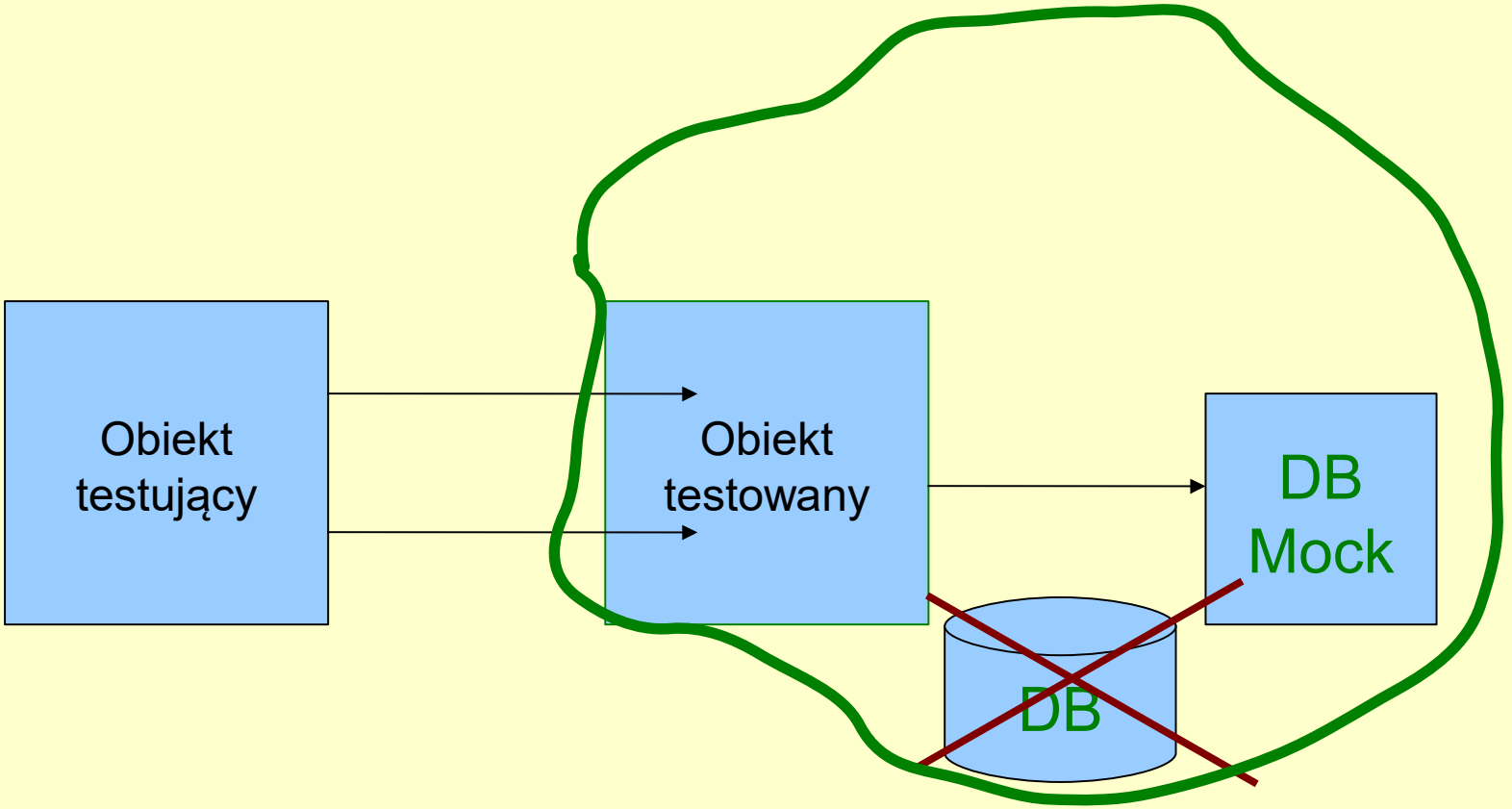
Przesłanki

- Pewne zależności mogą spowodować długie uruchamianie testów – np. wykorzystanie serwera aplikacji, połączenia z bazą danych,
- Pewne zależności mogą nie być dostępne na wczesnych etapach projektu (np. projektowana baza danych).

Koncepcja



Koncepcja



Koncepcja

- Rzeczywista zależność jest spełniana obiektem zastępczym (mock object),
- Obiekt zastępczy posiada interfejs obiektu zastępowanego i w najprostszej postaci zwraca z góry ustalone wartości.

Koncepcja

- Obiekt zastępczy może być konfigurowalny,
- Obiekt zastępczy może rejestrować interakcje z obiektem testowanym.

Mockito

- Popularna biblioteka do korzystania z obiektów zastępczych w języku Java.
- Biblioteka wspiera mockowanie klas i interfejsów,
- Obiekty zastępcze są tworzone dynamicznie,
- Mockito wspiera rejestrację interakcji z uwzględnieniem kolejności i liczności wywołań.

Mockito

```
// given

Person p = new Person("imie", "nazwisko", "pesel");// dane

DB dbMock = mock(DB.class);// tworzymy mocka

when(dbMock.getPerson("pesel")).thenReturn(p);// instruujemy

Catalog c = new Catalog(dbMock, null);// tworzymy obiekt testowany

// when

Person result = c.getPerson("pesel");

// then

Assert.assertEquals(p, result);// weryfikacja testu

verify(dbMock, atLeastOnce()).getPerson("pesel");// dodatkowa weryfikacja mocka
```

unittest.mock

- Biblioteka wspiera mockowanie klas,
- Obiekty zastępcze są tworzone dynamicznie,
- Biblioteka wspiera rejestrację interakcji z uwzględnieniem kolejności i liczności wywołań,
- Podstawowe klasy do tworzenia obiektów zastępczych: Mock, MagicMock, NonCallableMock, NonCallableMagicMock.

unittest.mock

```
# given

p = Person("imie", "nazwisko", "pesel"); # given

mock = DB();  # tworzymy obiekt, który będzie mockowany

mock.get_person = MagicMock(return_value=p);  # instruujemy

c = Catalog(mock, None);  # tworzymy obiekt testowany

# when

result = c.get_person("pesel");

# then

self.assertEqual(p, result);  # weryfikacja testu

mock.get_person.assert_called_with("pesel");  # dodatkowo weryfikacja mocka
```

Mockito/unittest.mock

- Obiekty zastępcze mogą być tworzone dla klas i interfejsów (Mockito).
- Liczność wywołań może być konfigurowana:
  - można określić dokładnie,
  - przynajmniej jedno,
  - dowolna liczba.

Mockito/unittest.mock

- Możliwe jest ustalanie oczekiwanej wartości zwracanej z metody,
- Obiekty zastępcze mogą być resetowane.

Podsumowanie

- Spełnianie zależności,
- Testowanie w izolacji,
- http://mockito.org/
- https://docs.python.org/3/library/unittest.mock.html

Q&A