Верификация ПО

Использование заглушек для тестирования

К.А.Кулаков

Петрозаводск — 2017

Причины использования

- Зависимые модули не реализованы
- Увеличение скорости работы
- "Выключение" зависимостей (например, сторонние ресурсы)
- Имитация редких ошибок
- Управление работой сторонних модулей
-

Заглушка

- Заглушка — управляемая замена существующей зависимости (или компаньона) в системе
- Функция-заглушка — функция, не выполняющая никакого осмысленного действия, возвращающая пустой результат или входные данные в неизменном виде.

- Преимущества
  - наглядность при проектировании структуры классов приложения
  - простота реализации
  - ограничение доступа к свойствам класса.

Использование заглушки

- Определение интерфейса вызова (прототипа)
  - модификация кода для выделения интерфейса
- Определение необходимого результата
  - возвращаемое значение
  - воздействие на систему
- Реализация заглушки
- Запуск теста для объекта с заглушкой

Как запустить заглушку?

- Необходима возможность подмены куска кода (зазор)
- Реализация взаимодействия через интерфейс
  - получить интерфейс в конструкторе и сохранить его в поле для последующего использования;
  - получить интерфейс в свойстве и сохранить его в поле для последующего использования;
  - получить интерфейс непосредственно перед вызовом в тестируемом методе одним из следующих способов:
    - в виде параметра метода (внедрение через параметр);
    - с помощью фабричного класса;
    - с помощью локального фабричного метода;
    - варианты вышеупомянутых способов.

Пример (работа с БД)

- Обычный код

```
public Component(String databaseURL)  {
   try  {
      databaseConnection = DriverManager.getConnection(databaseURL);
      ...
   } catch (SQLException e) {...}
}

public String get(String key)  {
   try {
      Statement stmt = databaseConnection.createStatement();
      ResultSet rs = stmt.executeQuery(

"SELECT value FROM Table1 WHERE key=" + key);
      ...
   } catch (SQLException e) {...}
}
```

Пример (работа с БД)

- Реализация интерфейса

```
interface KeyValuePairs {
    String get(String key);
    void put(String key, String value);
}
```

- Рефакторинг кода

```
class DatabaseAdapter implements KeyValuePairs {
…
}
```

Пример (работа с БД)

- Реализация заглушки

```
class FakeDatabase implements KeyValuePairs   {
        Hashtable table = new Hashtable();
        public String get(String key) {
              return (String) table.get(key);
         }
        public void put(String key, String value) {
              table.put(key, value);
         }
    }
```

Mock-объект

- Mock-объект (подделка, подставка) — тип объектов, реализующих заданные аспекты моделируемого программного окружения.
- Mock-объект представляет собой конкретную фиктивную реализацию интерфейса, предназначенную исключительно для тестирования взаимодействия и относительно которого высказывается утверждение.
- Отличие mock-объекта от заглушки — проверка состояния mock-объекта и его использования (взаимодействия) в ходе теста

Базовые функции mock-объектов

- Ведение лога использования объекта
- Реализация заглушек для методов
- Проверка наличия вызовов методов
- Переопределение методов
- "Слежение" за экземплярами классов

Изолирующие каркасы

- Программные средства для создания заглушек/подделок
- Автоматическая генерация заглушек на основе интерфейса во время выполнения теста
- Использование правил для отслеживания изменений
- Указание возвращаемых значений
- Указание значений свойств

Пример (Java, Mockito)

```
import static org.mockito.Mockito.*;

//вот он - mock-объект (List.class - это интерфейс)
List mockedList = mock(List.class);

//используем его
mockedList.add("one");
mockedList.clear();

//проверяем, были ли вызваны методы add с параметром "one" и clear
verify(mockedList).add("one");
verify(mockedList).clear();
```

Пример (Java, Mockito)

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

//stub'инг
when(mockedList.get(0)).thenReturn("first");
when(mockedList.get(1)).thenThrow(new RuntimeException());

//получим "first"
System.out.println(mockedList.get(0));

//получим RuntimeException
System.out.println(mockedList.get(1));

//получим "null" ибо get(999) не был определен
System.out.println(mockedList.get(999));
```

Симуляция объектов

- Имитация деятельности объекта/модуля
- Использование для сторонних объектов или связей
  - сторонние библиотеки
  - системные библиотеки ОС
  - legacy-код
- Необходимо наличие интерфейса взаимодействия (API)

Примеры

- Реализация системных утилит
- Реализация драйверов виртуальных объектов
- Реализация эмуляторов
- Реализация веб сервисов

Недостатки

- Написание дополнительного кода
- Заглушки могут быть дорогостоящими
- Необходимость рефакторинга для использования заглушек
- Проблема соответствия заглушки и реального объекта
- Проблема "интеллектуализации" заглушки
- Проблема определения результата работы заглушки
- Заглушки могут скрывать ошибки
- Сложно использовать заглушку в других тестах

Рекомендации

- Использование "правильного" кода
- Не более одного mock-объекта в тесте
- Простые заглушки
- Не более одной проверки в тесте
-