УДК 004.432.2

МОДУЛЬНОЕ ТЕСТИРОВАНИЕ КАК АЛЬТЕРНАТИВНЫЙ ИНСТРУМЕНТ ДЛЯ ПРОВЕРКИ ТИПОВ

К.И. ЛАПКОВСКИЙ
(Представлено: Д.В. ПЯТКИН)

В статье рассматривается использование модульного тестирования для проверки интерфейсов объекта или класса. Сначала познакомимся с проблемой проверки интерфейсов, которая стоит перед языками с динамической типизацией. Реализуем проверку интерфейса для произвольного объекта на javascript. Проанализируем плюсы и минусы использования такого подхода, сравним со статической типизацей.

Введение. Динамическая типизация - приём, широко используемый в языках программирования и языках спецификации, при котором переменная связывается с типом в момент присваивания значения, а не в момент объявления переменной. Таким образом, в различных участках программы одна и та же переменная может принимать значения разных типов. Примеры языков с динамической типизацией – Smalltalk, Python, Objective-C, Ruby, PHP, Perl, JavaScript, Lisp, xBase, Erlang, Visual Basic [1].

Основная проблема языков с динамической типизации - их ненадежность. В таких языках нельзя описать абстрактный метод или интерфейс. Хорошее решение в такой ситуации - выбрасывать ошибку при обращении к методу базового класса, однако это не заставит программиста переопределить всех потомков абстрактный метод, скорее запретит вызов метода базового класса. Сам факт вызова такого метода может обнаружиться только во время выполнения программы, поэтому ошибка может долго оставаться незамеченной.

Основной раздел. Реализуем проверку типов на языке программирования JavaScript. Для написания модульных тестов используются библиотеки mocha и chai.

Проверка наличия свойств объекта

В листинге 1 создается функция, которая на вход принимает объект. В этой функции с помощью утилит describe и it соответственно описывается и создается модульный тест. В модульном тесте производится проверка наличия свойства в объекте.

Создавая такую функции для каждого интерфейса, который необходимо проверить, можно существенно сократить повторяемость кода, по сравнению с подходом, где для каждой реализации интерфейса создаются свои тесты.

```
const expect = require('chai').expect

const interfaceChecker = instance => {
 describe('#propA', () =>{
       it('should has propA', () => {
             expect(instance).to.have.property('propA')
       })
 })
}

interfaceChecker({
      propA:1
})
```

Листинг 1. – Проверка наличия свойств объекта

Проверка наличия методов объекта

Методы соответствуют определенному интерфейсу если сигнатуры этих методов совпадают. Сигнатура метод – это совокупность названия метода и количества принимаемых аргументов. Предыдущие два утверждения справедливы только для языков с динамической типизацией.

```
const expect = require('chai').expect

const interfaceChecker = instance => {
 describe('#methodA', () =>{
       it('should has methodA', () => {
```

```
            expect(instance).respondTo('methodA')
        })

        it('methodA should have 2 arguments', () =>{
            expect(instance.methodA.length).equals(2)
        })
  })
}


interfaceChecker({
     methodA(a,b){}
})
```

Листинг 2. – Проверка наличия метода объекта

В листинге 2 проверяется наличие метода methodA у объекта, переданного в функцию interfaceChecker.

Основные плюсы использования тестов:

– выявление ошибок реализации интерфейса во время выполнения модульных тестов

– возможность реализовать проверку типов вне зависимости от языка программирования

минусы использования тестов:

– затраты на написание тестов

– необходимость наличия этапа выполнения модульных тестов во время сборки или деплоя программы.

Заключение. Использование модульных тестов для проверки типов отлично подходит для повышения надежности критически важных частей программы, не влияя на часто изменяющиеся части программы не имеющие стабильного интерфейса. Однако модульные тесты необходимо поддерживать, поэтому если программа объемна, то лучше задуматься о использовании языка программирования со статической типизацией.

ЛИТЕРАТУРА

1. Wikipedia [Электронный ресурс] Динамическая типизация. Режим доступа: https://ru.wikipedia.org/wiki/ Динамическая_типизация. Дата доступа: 26.09.2019.