Japanese Using JBehave with Groovy

JBehave is a Behaviour Driven Development (BDD) framework for Java.

The sections below illustrate using JBehave for the examples from Using Testing Frameworks with Groovy.

The Stack Example

Here is how you might use JBehave to test the Stack Example:

```
// require(url:'http://jbehave.org/', jar='jbehave-1.0.1.jar')
import org.jbehave.core.Run
import org.jbehave.core.behaviour.Behaviours

class AlmostEmptyFixedStackBehavior {
    private stack

    void setUp() {
        stack = new FixedStack()
        stack.push 'anything'
        assert !stack.isEmpty()
    }

    void shouldRemainNotEmptyAfterPeek() {
        stack.peek()
        assert !stack.isEmpty()
    }

    void shouldBecomeEmptyAfterPop() {
        stack.pop()
        assert stack.isEmpty()
    }
}

class AlmostFullFixedStackBehavior {
    private stack

    void setUp() {
        stack = new FixedStack()
        (1..<FixedStack.MAXSIZE).each{ x -> stack.push x }
        assert !stack.isFull()
    }

    void shouldBecomeFullAfterPush() {
        stack.push 'anything'
        assert stack.isFull()
    }
}

class EmptyFixedStackBehavior extends GroovyTestCase {
    private stack = new FixedStack()

    void shouldInitiallyBeEmpty() {
        assert stack.isEmpty()
    }

    void shouldNoLongerBeEmptyAfterPush() {
```

```
        stack.push 'anything'
        assert !stack.isEmpty()
    }

    void shouldComplainWhenSentPeek() {
        shouldFail(StackUnderflowException) {
            stack.peek()
        }
    }

    void shouldComplainWhenSentPop() {
        shouldFail(StackUnderflowException) {
            stack.pop()
        }
    }
}

class FullFixedStackBehavior extends GroovyTestCase {
    private stack

    void setUp() {
        stack = new FixedStack()
        (1..FixedStack.MAXSIZE).each{ x -> stack.push x }
        assert stack.isFull()
    }

    void shouldRemainFullAfterPeek() {
        stack.peek()
        assert stack.isFull()
    }

    void shouldNoLongerBeFullAfterPop() {
        stack.pop()
        assert !stack.isFull()
    }

    void shouldComplainOnPush() {
        shouldFail(StackOverflowException) {
            stack.push 'anything'
        }
    }
}

class NonEmptyFixedStackBehavior {
    private stack

    void setUp() {
        stack = new FixedStack()
        ('a'..'c').each{ x -> stack.push x }
        assert !stack.isEmpty()
    }

    void shouldAddToTheTopWhenSentPush() {
        stack.push 'd'
        assert stack.peek() == 'd'
    }

    void shouldBeUnchangedWhenSentPushThenPop() {
        stack.push 'anything'
```

```
        stack.pop()
        assert stack.peek() == 'c'
    }

    void shouldReturnTheTopItemWhenSentPeek() {
        assert stack.peek() == 'c'
    }

    void shouldNotRemoveTheTopItemWhenSentPeek() {
        assert stack.peek() == 'c'
        assert stack.peek() == 'c'
    }

    void shouldReturnTheTopItemWhenSentPop() {
        assert stack.pop() == 'c'
    }

    void shouldRemoveTheTopItemWhenSentPop() {
        assert stack.pop() == 'c'
        assert stack.pop() == 'b'
    }
}

class AllBehaviours implements Behaviours {
    Class[] getBehaviours() {
        return [
            AlmostEmptyFixedStackBehavior,
            AlmostFullFixedStackBehavior,
            EmptyFixedStackBehavior,
            FullFixedStackBehavior,
            NonEmptyFixedStackBehavior
        ]
    }
}
```

```
Run.main('AllBehaviours')
```

The Item Storer Example

Here is how you might use JBehave to test the Item Storer Example:

```
// require(url:'http://jbehave.org/', jar='jbehave-1.0.1.jar')
import org.jbehave.core.Run

class JBehaveStorerBehavior {
    def storer = new Storer()

    def static checkPersistAndReverse(cut, value, reverseValue) {
        cut.put(value)
        assert value == cut.get()
        assert reverseValue == cut.getReverse()
    }

    void shouldReverseStrings() {
        checkPersistAndReverse storer, 'hello', 'olleh'
    }

    void shouldReverseNumbers() {
        checkPersistAndReverse storer, 123.456, -123.456
    }

    void shouldReverseLists() {
        checkPersistAndReverse storer, [1, 3, 5], [5, 3, 1]
    }
}

Run.main('JBehaveStorerBehavior')
```