Jbehave Vs Cucumber

↳ completely diff^t although meant for the same purpose : Acceptance tests

jbehave → stories , (now support feature) as well.
Cucumber → features

Feature is a collection of stories.
↓
feature file

J behave
pure java f/w

→ Both has native Junit support.

→ Both support Gherkin standard & its rule.

↳ 'composite steps' feature is avl only in Jbehave not in Cucumber.
→ increases re-usability but in long run we may need to change dependent steps so that time maintenance will be painful.

Cucumber
based on Ruby.

↳ can be used with java via a maven plugin Cuke4Duke, Not very good from performa pt of view as internally it uses ruby.

→ Now There is a pure java implementh of cucumber called "Cucumber — JVM"

→ so let scenario & steps indept & decoupled as much as possible

BDD DSL → Gherkin → GWT format of scenario
↳ a DSL used by BDD practitioner

→ report can be uploaded to ALM / JIRA

Jbehave	Cucumber.
Both has same flexibility in passing parameters.	implement^{n} is straight-forward & less painful as c. to Jbehave.
→ default reporting is old fashioned no visual rep^{n} (Graph, charts etc)	→ more flexible & easy to generate.
⇒ Jbehave has more configurat^{n} to make.	online people/community are more Active on Cucumber as compared to Jbehave
⇒ Cucumber is simple & easy, if you want something more powerful u shud look at Jbehave	→ multiline input feature only in cucu.
→ Good external data support in Jbehave	→ In built support lamba expr^{n}

- Both supports below features
1. User stories / features
2. Scenario
3. Background → avl only in Cucumber
4. Scenario Outline
5. Examples
6. Given / when / Then / And / But
7. Tags (@automated, @ –) → jbehave uses Meta Keyword for that but Cucu Not
8. Tables

Contains those steps which needs to be executed before each scenario in the feature

- Composite Steps are not a feature of BDD / Ghorkin standard.
- If you need a BDD tool, ~~Reporting~~ Cucumber has almost all the power you need. So its safe to go with cucumber.

↓↓

- Common Issues Both tools will have

Notes

performance issues when No. of test Scenarios are increasing in your proj. So organise your BDD Scenario neatly.

⇒ Almost both are equally good.

Jbehave

17 Wednesday

February 2016
Week 08
Day 048 • 318
Date 17 • 02 • 2016

→ test method name should be sentences

→ a f/w for BDD

Features:

→ Pure java implement^{n}

→ user stories can be specified as classpath resources or external URL based resource

→ U.S. → can be executed concurrently, by specifying no. of concurrent threads

→ Annot^{n} based binding of textual steps to java methods,

with auto conversion of string argument to any parameter type (including generic type) via custom parameter convertors.

↳ Dependency Injection support (Spring)

↳ Report in HTML, XML, TXT, CONSOLE format

↳ Auto generat^{n} of pending steps

↳ can be run as Junit tests

Advance Jbehave

arguments → value
parameters → variable

M	T	W	T	F	S	S	MAR '16
	1	2	3	4	5	6	
7	8	9	10	11	12	13	
14	15	16	17	18	19	20	
21	22	23	24	25	26	27	
28	29	30	31				

(1.) Automatic Conversion of textual values to Java objects in the steps class.
→ or step defn class

Given a stock of symbol STK1 and a threshold of 10.0

```
(2) Given ("a stock of symbol $symbol — of $threshold")
public void aStock(String symbol, double threshold)
    {
        }
```

even if we have comma separated values it wud be automatically handled:

Given a stock of — STK1, STK2 and — 10.0, 20.0 is traded on 09/09/2009

```
(3) Given(" $symbol — $threshold — $tradedOn")
P. V. astock(String symbol, double threshol, Date tradedOn)
    {
    }
```

↓ should be list actually

19
Friday

February 2016
Week 08
Day 050 • 316
Date 19 • 02 • 2016

	M	T	W	T	F	S	S
5	1	2	3	4	5	6	7
6	8	9	10	11	12	13	14
7	15	16	17	18	19	20	21
8	22	23	24	25	26	27	28
9	29						

(2) Parameter Injection (Jbehave supports multiple mechanism for parameter Inj^n

(A) Ordered Parameters

This is default behavior.
arguments from steps are matched in natural order to the parameters in the annotated java method.

(B) Annotated named Parameter:

Advantage is that we can have method parameter appearing in any order.

Given a threshold value 2

@Given(" a threshold value $value")

p. v. giventhreshold (@Named("value") int value)
{ = 3

(3) Parameterized Scenario

Given a stock of <symbol> and a <threshold>

Examples: → Examples Keyword indicates scenario is parameterized & shud be repeated as many times as there are data rows in the example table.

pipe-symbol →

symbol	threshold
STK1	10.0
STK2	11.0

(4) Revsing Textual Stories - 'GivenStories'

GivenStories:

xyz / ABC.story

Scenario: verify enterprise name.

) the given one or more stories will be executd be the given scenario.

(5) Pending steps! steps that do not match any public Java method in steps classes.

» They are marked as Not Performed in steps failure

(6) Meta Info: To help users better manage their stories.

info is provided as name-value properties

can be used at both level {stories, scenario}

Notes

Meta:

(a) Sprint GRT2

(b) Type positive

(c) F 635

(d) author deepak

(7) custom parameter delimiter

Given a stock of <symbol> }
$symbol
[symbol] }

(8) Pattern Variants: → alternative of @Alias
are built by the
patternvariantBuilder:

e.g.
A{x|y|z}B ⇒ AxB, AyB, AzB

when the item cost is 10.0
when the price is 10.0
when the cost is 10.0

@When("the{item|}{price|cost} is $price")

#if we use @Alias, it is cumbersome ~~to~~ list all difft patterns when diff b/w

Notes them is small.

9 BDD is used to check both whether:

Code meets customer Expect^{n} → Sel^{n} → UI & Present^{n}

Code Meets Programmer expectations → Unit testing → Classess /models

1 DDD: Domain Driven Design

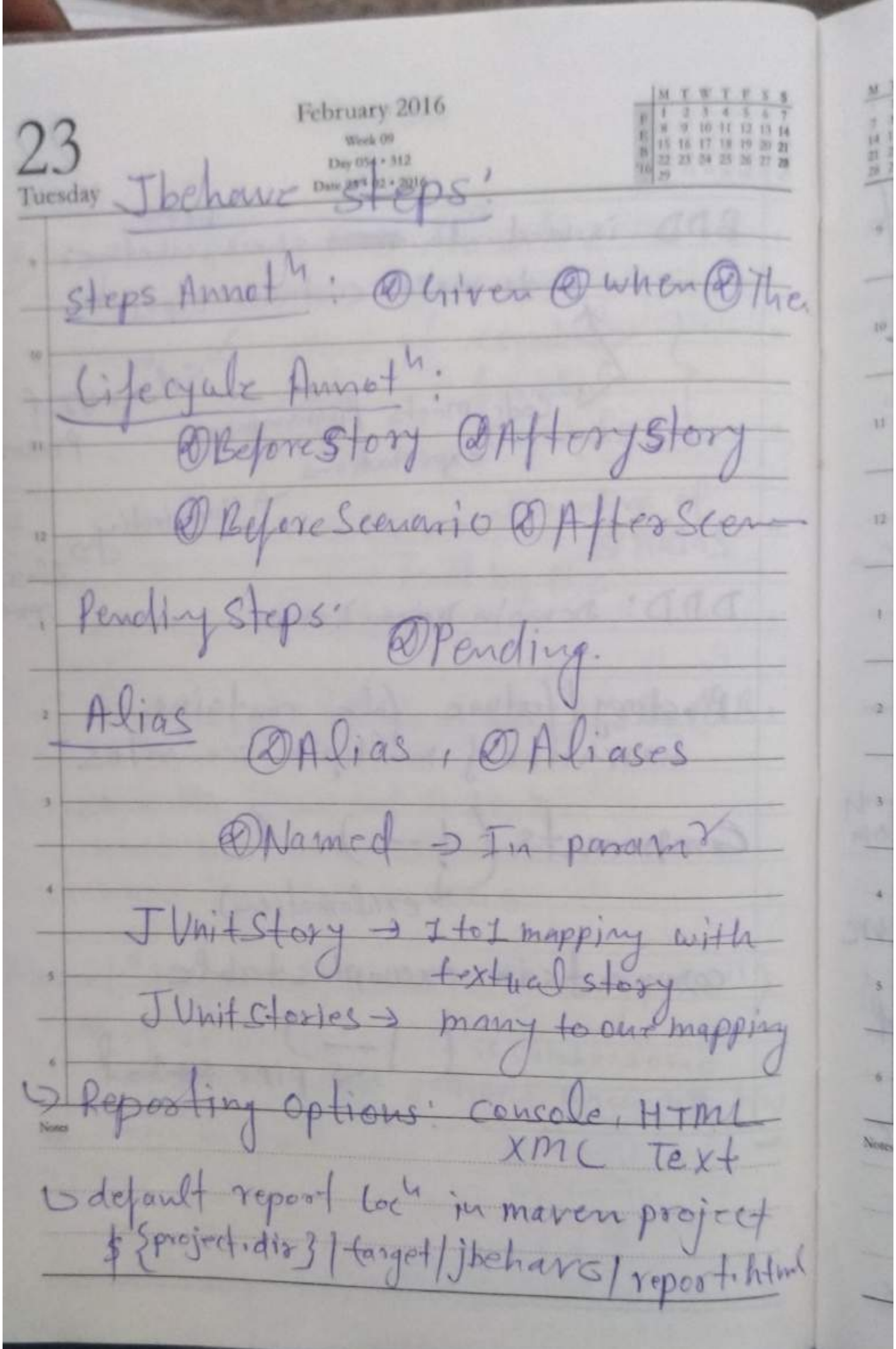
2 A story / feature file contains a narrative & multiple scenarios

3 Comments (! --) → exclamation)

5 comments in examples table

(| --) → pipe symbol

Notes



23
Tuesday

February 2016
Week 09
Day 054 • 312

Jbehave steps'

Steps Annot^{n}: @Given @When @The

Lifecycle Annot^{n}:
@BeforeStory @AfterStory
@BeforeScenario @AfterScen—

Pending Steps: @Pending.

Alias @Alias, @Aliases

@Named → In param^{r}

JUnitStory → 1 to 1 mapping with textual story
JUnitStories → many to one mapping

↳ Reporting options: console, HTML XML Text

↳ default report loc^{n} in maven project
${project.dir}/target/jbehave/report.html

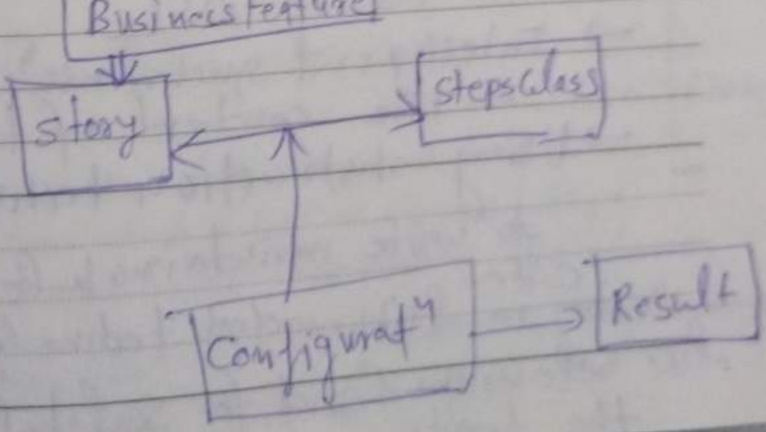
→ Test Integrⁿ : JUnit & TestNG

Jbehave EclipsePlugin

- Keyword highlighting
- jumping from stories to step class
- Auto complete.
  if no step match it will indicate Δ

Jbehave → Can be used web & API Tg both
Jbehave → UI Tg (Selenium) → POM
Jbehave → Services Tg (Spring library)

We shud keep assertions in steps mapping classes.



Jbehave have excellent support for spring f/w.

Major challenges in Autom^n^

Maintainability
Reusability
Usability
Scalability
Reliability

How to overcome challenges in Web Autom^n^

→ multitier Automation Architecture
→ write test in DSL
→ Intelligent synchronis^n^ based on the context of test
→ Using abstraction technique to write maintainable test code.
→ using multithreaded technology like selenium Grid for scalability of the tests.

→ Advantage of Cucumber/Jbehave

→ You can involve business Stakeholders
10 who can not code.
11 → End user experiance is priority
→ High Code Reuse

12

→ 2 files req. to run cucu test
1 Features & step def^{n}

→ Cucu tool supports BDD & is as cool as (cucumber:)

3 { Rspec is used for unit testg
{ Cucu → for BDD
4 ↓ Both Ruby based

5 ↳ Feature files also serve as DOD (Def^{n} of Done)

6

means when all scenario has been
Notes implementd & tested successfully.
We can mark the story as done.

feature file: contains common understanding of trio Dev-QA-BA

Serenity → Java based open source Test Automn F/W

→ built on top of selenium,

previously known as "Thucydides"

an open source library

produces a rich meaningful test reports "living document"

which not only report on test results but also what features hv been tested

↳ helps to write more maintainable automated acceptance criteria

Serenity with Junit

Serenity takes care of Driver life cycle

i.e. instantiating webd', opening the browser & shutting it down at the end.

@Steps → used in stepdefn file → How steps ⇓ assertion is used here.

BuyerSteps buyerSteps;

@Steps tells serenity that this variable is a step library.

→ In serenity, we use step libraries to add a layer of abstraction b/w what & How of the acceptance tests

→ Building a F/W from scratch is not always a good idea as it requires lot time & efforts.
We should utilize existing open source F/W.

Serenity :

↳ can be easily integrated with BDD tools like Cucumber & Jbehave

↳ provides lot of built-in functionalites

- * webDriver Management
- * managing states b/w steps
- * Screenshots
- * Running tests in parallel
- * JIRA Integration
- * Awesome reports.
  - ↓ can be used to testResults
  - ↓ can be used a living docⁿ for ur Appⁿ.

⇓ No need to write any code for above features

Notes

	M	T	W	T	F	S	S
[illegible]	1	2	3	4	5	6	7
[illegible]	8	9	10	11	12	13	14
[illegible]	15	16	17	18	19	20	21
[illegible]	22	23	24	25	26	27	28
[illegible]	29						

Step library This class is just an ordinary java class with methods annotated with @Step annot^n

```
Public class BuyerSteps {

  HomePage homePage;
@Step ⟶ indicate the method will appear as a step in the report
  p. v. searches_for_items_containing(String keyw)
   {
       homepage. searchesFor(Keywords);
   }
@Step
  p. v. should_see_items_related_to(String key)
   {
     List <String> result = searchPage.getTitle().
   result.stream().forEach(title ->
   assertThat(title.contains(Keyword)));
```

~~To search an ele in chrome~~

Identify locator in chrome:

$x(" ") ⟶ Xpath ← console

$(" ") ⟶ CSS

Page Objects (Page class) → encapsulates how a test → interacts with a webpage. They hide webdriver implementatⁿ details.

Like steps page objects are reusable compo't that makes tests easier to understand & maintain.

→ Serenity automatically instantiates page obj for you & injects the current webd^r instance

```
@ DefaultUrl ("https://——")
public class HomePage extends PageObject
  {
      @ FindBy (css = "—")
      WebElement searchButton;

      p. v  searchFor (String Keywords)
      {
           $("#search-query").sendKeys(Keum);
                searchbutton.click();
      }
```

Notes

↳ we can use $ method to access elements directly using csss or Xpath

2 Wednesday

March 2016
Week 10
Day 062 • 304
Date 02 • 03 • 2016

	M	T	W	T	F	S	S
M		1	2	3	4	5	6
A	7	8	9	10	11	12	13
R	14	15	16	17	18	19	20
'16	21	22	23	24	25	26	27
	28	29	30	31			

```
@FindBy (css = ".listing-card")
List<WebElement> listingCards;

public List<String> getRTitles()
{
    return listingCards.stream()
                .map(element -> ele.getText())
                .collect(Collectors.toList());
}
```

Maven Dependency:

To use Serenity with JUnit we should include serenity-core & serenity-junit in pom.xml

<artifactId>

→ To get aggregated report:

<artifactId>serenity-maven-plugin</artifactId>

→ Serenity report is gen'd at:
target/site/serenity/index.html

@Test (attr1 = value1, attr2 = value2, ...)

↳ To integrate Serenity with Jbehave

<artifactId> serenity-jbehave </aId>

↳ Serenity supports integr^n with REST- assured. we need dep in pom

<artifactId> serenity-rest-assured </aId>

↳ Serenity support Integr^n with JIRA
pom dependency for this

<aId> serenity-jira-req^{ts}-provider </aId>

public class whenAuthenticating — TestNG or Junit test class

```
{
   @Steps
   private DecoUser user;
                    → junit annot^n
   @Test
   p. v.  shouldBeAbleToLoginAsAdmin()
   {                                      → This will be
       user.isOnTheHomePage();              annotated with
       User.logsInAsAdmin();                @step
       user.shouldBeOnLandingPage();
   }
}
```

@Step("authenticates as an Admin User")

↓ By default the name of thes step inthe report is derived from methodName if want somethiy more readable pass that as String parameter ~~to~~ @step annot^n.

jQuery selector:

$() method with an Xpath or css expression can be used.

→ Behind the scene uses "findBy"() ← pass css or xpath

```
→ $("#username").Click();
→ return $(".notice").getText();
→ $("*[role='menu']" li").find(By.linkText("))•cli
```

Notes