RuleScape ODM Test Accelerator

The **RuleScape Test Accelerator** is an Eclipse plugin that generates the scaffolding (test projects, libraries, skeleton source code, etc.) which enables you to effectively use the **RuleScape Test Framework** library.

RuleScape Test Framework

What is it the RuleScape Test Framework?

The Rule Test Framework is a Java library which allows a technical developer to easily and quickly develop JUnit test cases that test Operational Decision Manager (ODM) decision services.

IBM's ODM is its flagship Business Rule Management System (BRMS) and is the best of breed among BRMS products. ODM enables a business to respond to real-time data with intelligent, automated decisions. In a decision service, complex decisions are automated by chaining together simple business rules in a business-oriented manner.

The importance of developer testing of any software artifact cannot be overstated. This is more so true for decision services which are meant to handle change and evolve rapidly. Continual, extensive testing is crucial for robust decision services.

ODM is unique in that business rules are created and managed by both business users and developers, using different ODM tooling. Developers use Rule Designer, while business-oriented rule authors use Decision Center. *Technical Releases*, handled by developers, are needed whenever the underlying execution object model (XOM) changes or when there is a change that is categorized by the project as a high-risk change. *Business Releases* can be handled by business rule authors to make more frequent, but less risky, changes.

The JUnit based developer test cases built using the RuleScape Test Framework are very important during initial rule project development and subsequent technical refinements of a decision service, such as a change to the underlying Java XOM, refactoring and other changes that require developer involvement.

These unit tests can be incorporated into the deployment DevOps pipeline as well. This implies that there should be a sufficient number of automated unit test cases to ensure quality and detect any regressions.

Why is developer testing important?

Some customers ask why developer testing is important when each rule is ideally simple and easily understood. Here are the reasons why:

- Even though individual rules may be simple, they work together to make complex decisions. Developers create the rule project structure and orchestrate the rules that make the decision using one or more ruleflows. After development of the rule foundation and before handoff to business rule authors, developers must create sample rules and create unit test cases to ensure that the XOM, business verbalization and orchestration is correct.

- Even during initial development, you sometimes encounter nasty surprises in the form of hard-to-debug, mysterious exceptions. These are easier to pinpoint if tests are created and run after every change. Test-driven development prevents hard-to-debug big-bang errors and exceptions. It is highly recommended that when a rule is added, one or more test cases be added to test the rule in conjunction with the other rules.
- The reality is that “Change Happens”, and all software artifacts need to keep up with the changes. Having a test suite that can keep up with refactoring of the XOM and/or rule orchestration enables developers to have confidence in the changes even after major refactoring.
- Integration of the test cases with the DevOps pipeline ensure quality of the deployed decision services.

What about ODM built-in testing?

Aren’t Decision Runner and Run Config execution enough? The ODM tutorials do a great job outlining how to do preliminary testing in Rule Designer using Run Configurations. While this is a handy tool to run small smoke tests, it is by no means a general mechanism to run an automated suite of complex tests.

Also, business users can test using Excel, which is perfect for business releases. However, for technical releases it falls short simply because it lacks support for refactoring, does not provide the level of logging that is typically needed for a developer to debug, and is harder to create complex edge test cases.

What are the salient features of this framework? How does the framework help?

As a developer, have you ever asked yourself any of the following questions:

1. I’ve refactored my XOM. Will there be any unintended consequences?
2. Can I use JUnit to test the decision service?
3. Can I get a fine-grained view of the rules and tasks that are executed for a test case?
4. Can I view the complete request and response from the decision service in JSON and/or XML?
5. Apart from Java, can I use XML and/or JSON as input data for my tests?
6. Which rules are yet to be tested? Do I have enough test coverage?
7. I’ve refined the rule foundation. How is the decision service performance impacted?

If you’ve said to any of these questions, then RuleScape Test Framework will help you. At a high level, it provides the following features:

- Ease creation of JUnit test cases that invoke your decision service
- Provide ability to define decision service payload data in Java, XML or JSON
- Provide handy methods that provide ability to test if a specific rule has fired
- Provide detailed logs of the request data, response data and fine-grained list of rule tasks and rules that fired. The data may be logged in XML or JSON. These logs can be persisted to the file system. These historical logs can prove to be useful when you encounter an unexpected result after a change.
- Test run reports that:
    - Create a rules histogram and identify if any rules are left untested

    - Report performance characteristics of the decision service
    - Create a reverse lookup report which list tests that have fired a specific rule
- Allows custom reporting by using a custom listener class

What do the test cases look like?

Here is a quick taste of what a test case looks like:

```
     @Before
     public void initRequest() {
       // create request here
          borrower = Builder.buildBorrower();
          loan = Builder.buildLoan();
    }

     /**
     * Check interest rate
     */
    @Test
    public void interestRate1() {
       // change request here as necessary
       loan.setNumberOfMonthlyPayments(12);
       loan.setLoanToValue(0.5);

       // execute rules
       RuleExecutionResponse ruleResp = executeRuleset(borrower, loan);

       assertNotNull(ruleResp);

       //fetch output params
       Report reportValue = (Report) ruleResp.getOutputParam("report");
       Integer scoreValue = (Integer) ruleResp.getOutputParam("score");
       String gradeValue = (String) ruleResp.getOutputParam("grade");

       // run tests
       assertNotNull(reportValue);
       assertNotNull(scoreValue);

       assertEquals(0.05, reportValue.getYearlyInterestRate(), 0.001);
    }
```

Where are the API Docs for the Test Framework?

The goal of the RuleScape Test Framework is to provide a simple, easy-to-use API. This API is described here: https://rulescape.github.io/rtf-apidocs/

What are the pre-requisites for using the Test Framework?

You need to have a full installation of the Rule Designer (i.e. not 'ODM for Cloud', which does not have the execution files). This framework is tested for Rule Designer v8.8 and above.

The intended audience for the Test Accelerator and Test Framework is an ODM developer familiar with the Rule Designer.

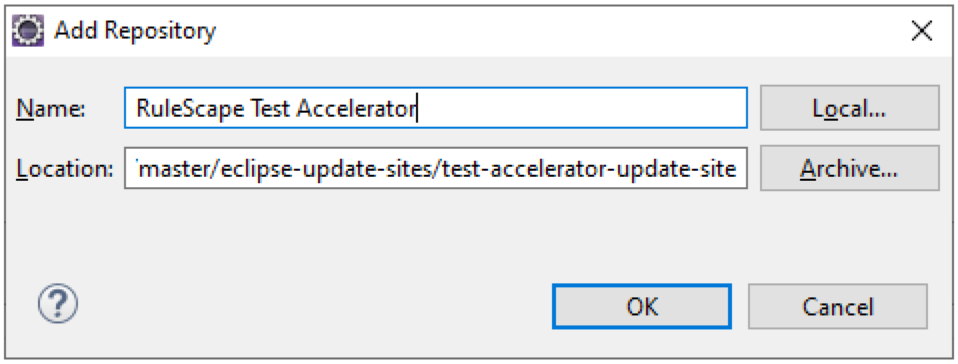
What are the limitations of the Test Framework?

The known limitations are:

1. Works only with a Java XOM (which is the best practice, in any case)
2. Expects all projects to be in the workspace folder (which is the most common Eclipse setup)

Installation

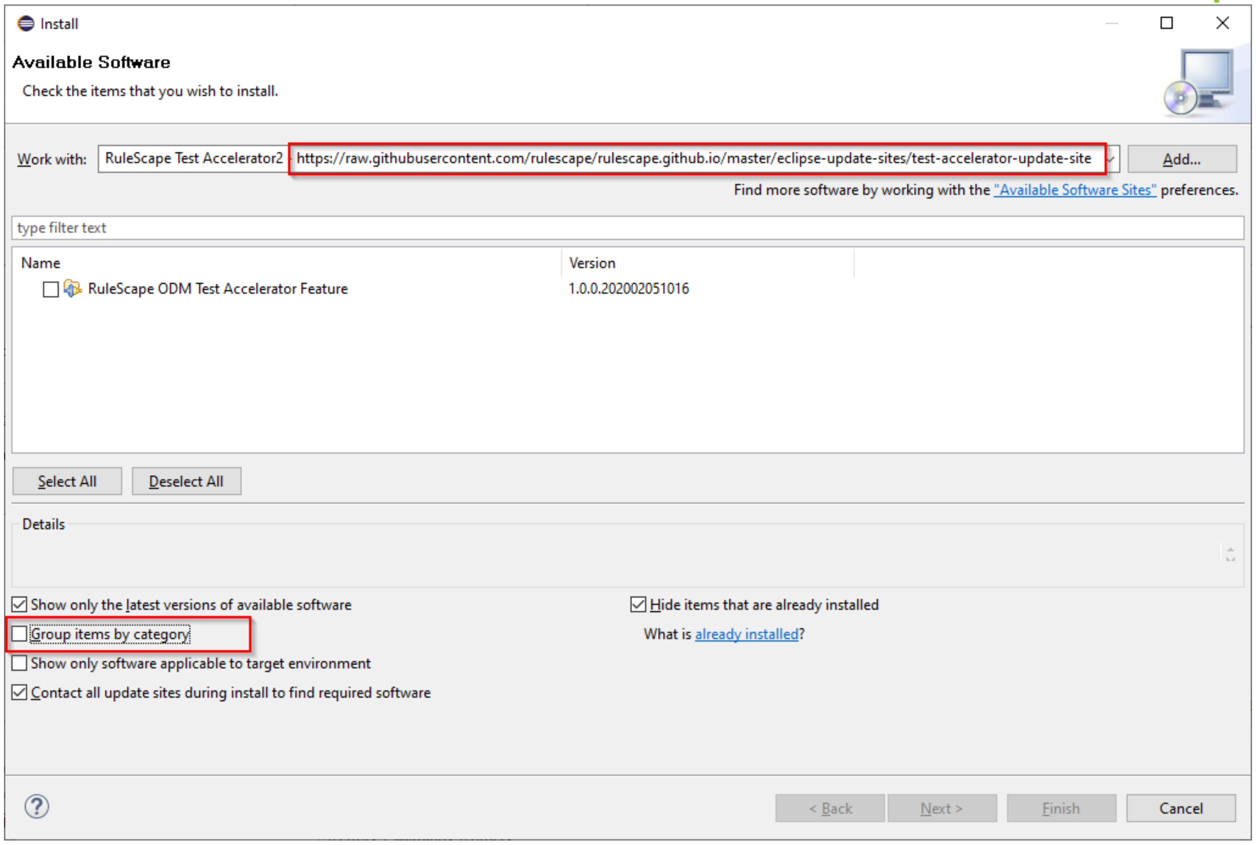
1. In Rule Designer, go to Help->Install New Software
2. Click ‘Add…’ button
3. Enter Name and repository location as shown below:
   a. Name: RuleScape Test Accelerator
   b. Location: https://raw.githubusercontent.com/rulescape/rulescape.github.io/master/eclipse-update-sites/test-accelerator-update-site


Add Repository

Name: RuleScape Test Accelerator

Location: master/eclipse-update-sites/test-accelerator-update-site

4. Select RuleScape ODM Test Accelerator feature as shown below:


Install

Available Software

Check the items that you wish to install.

Work with: RuleScape Test Accelerator2 - https://raw.githubusercontent.com/rulescape/rulescape.github.io/master/eclipse-update-sites/test-accelerator-update-site

Find more software by working with the "Available Software Sites" preferences.

Name	Version
☐ RuleScape ODM Test Accelerator Feature	1.0.0.202002051016

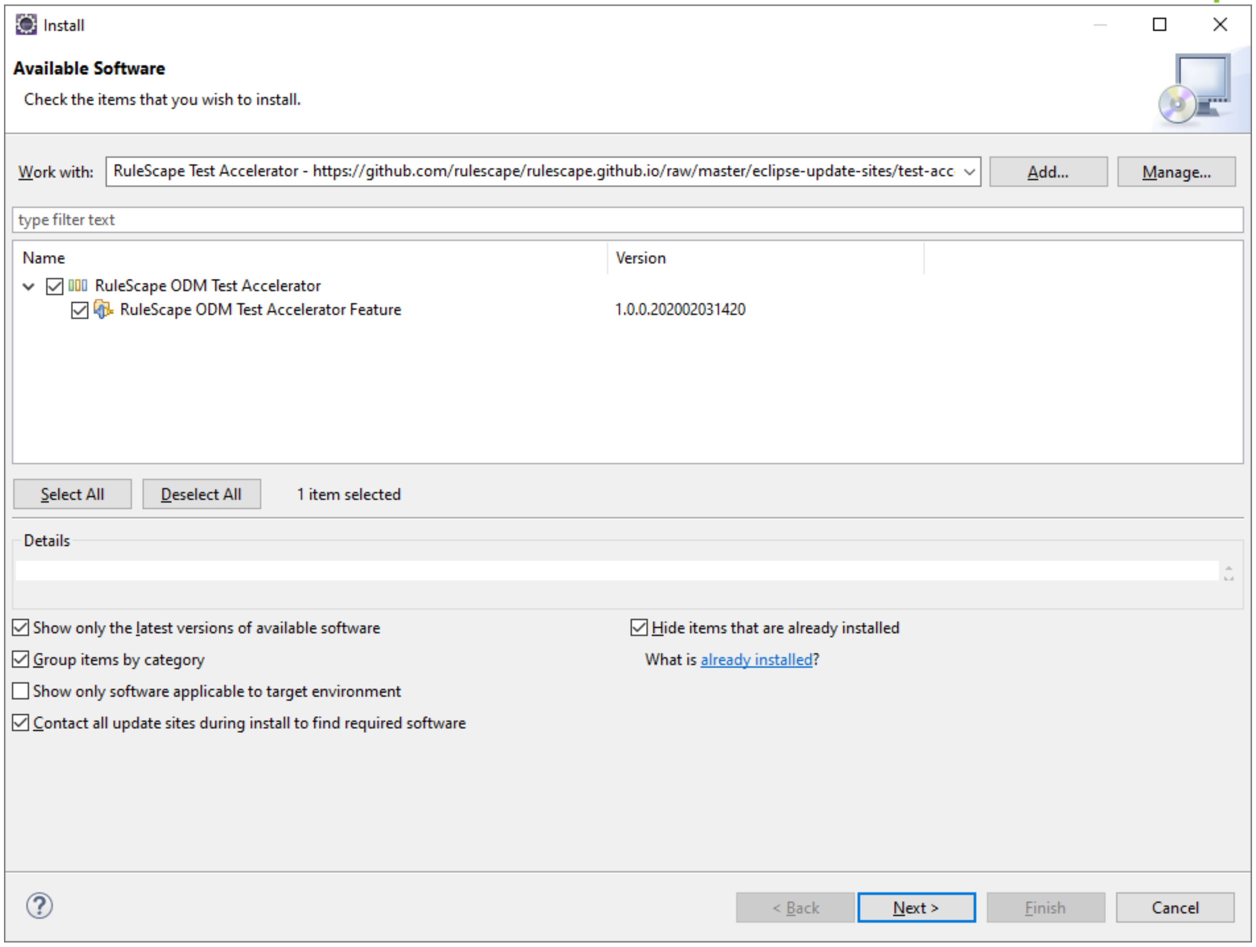
Details

- [✓] Show only the latest versions of available software
- [ ] Group items by category
- [ ] Show only software applicable to target environment
- [✓] Contact all update sites during install to find required software
- [✓] Hide items that are already installed

What is already installed?

Note: On Juno, you may need to uncheck “Group Items by category”.

Select the Test Accelerator feature.


Install

Available Software

Check the items that you wish to install.

Work with: RuleScape Test Accelerator - https://github.com/rulescape/rulescape.github.io/raw/master/eclipse-update-sites/test-acc

Name	Version
☑ RuleScape ODM Test Accelerator	
☑ RuleScape ODM Test Accelerator Feature	1.0.0.202002031420

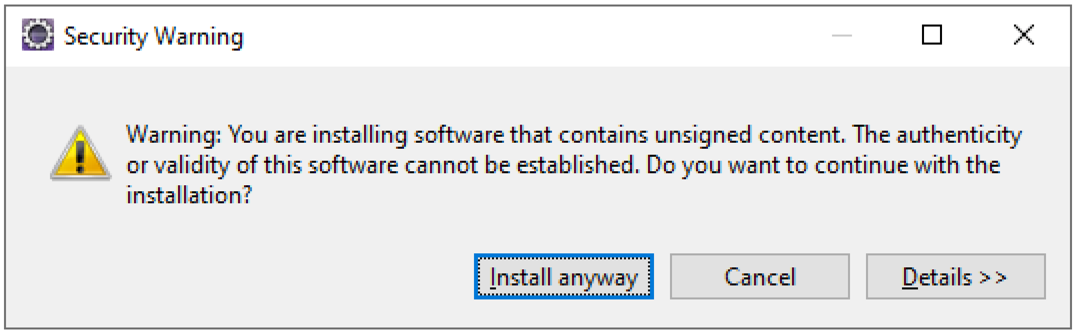
1 item selected

Details

- ☑ Show only the latest versions of available software
- ☑ Group items by category
- ☐ Show only software applicable to target environment
- ☑ Contact all update sites during install to find required software
- ☑ Hide items that are already installed

What is already installed?

Ignore the security warning.


Security Warning

Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?

Restart Eclipse when prompted.

Potential issues

If you get an “Unable to read repository” error, try adjusting the network setting as described here: https://stackoverflow.com/questions/45319531/eclipse-luna-shows-error-unable-to-read-repository-at/51359313#51359313

Tutorial

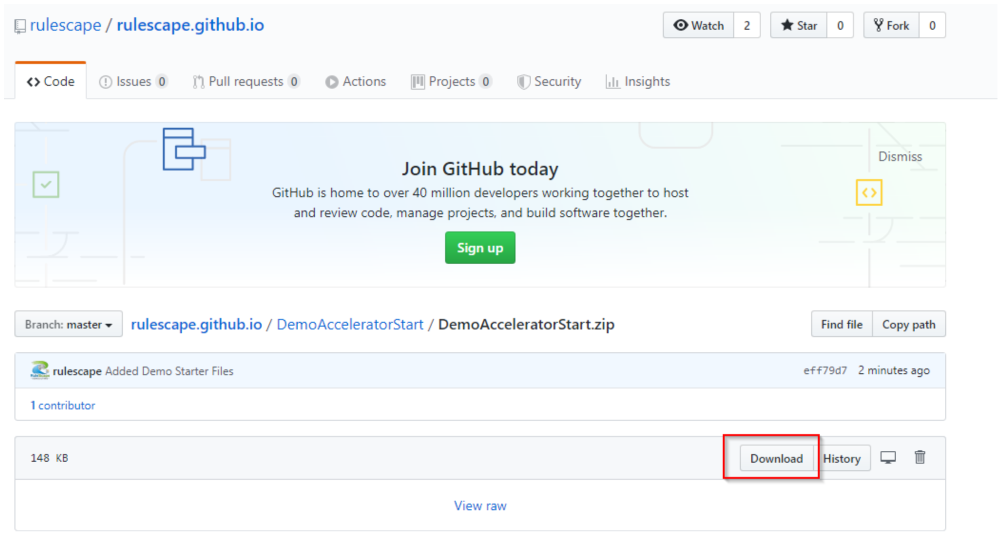
Introduction

The tutorial shows the basic steps in using the RuleScape Test Accelerator.

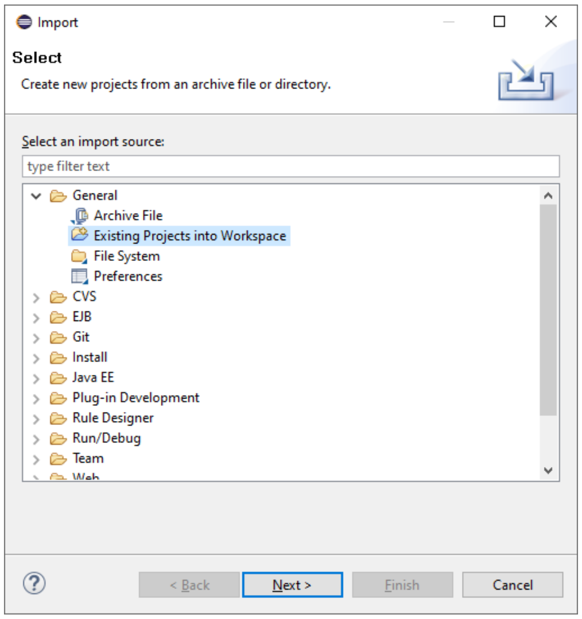
It is based on the ODM Mini Loan Tutorial. The tutorial has been expanded to include a few more rules to showcase testing.

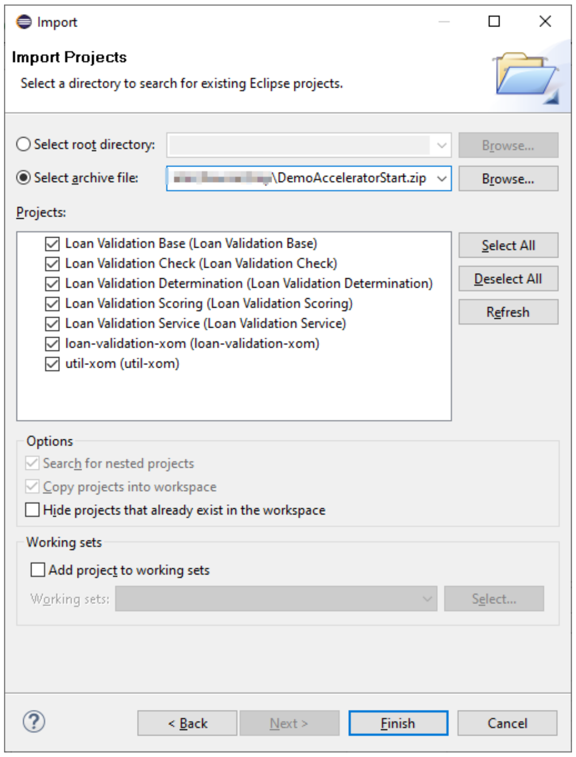
Download

Download the starting workspace from:
https://github.com/rulescape/rulescape.github.io/blob/master/test-accelerator-tutorial/DemoAcceleratorStart.zip


rulescape / rulescape.github.io
Watch 2
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Actions
Projects 0
Security
Insights
Join GitHub today
GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.
Sign up
Dismiss
Branch: master
rulescape.github.io / DemoAcceleratorStart / DemoAcceleratorStart.zip
Find file
Copy path
rulescape Added Demo Starter Files
eff79d7 2 minutes ago
1 contributor
148 KB
Download
History
View raw

Save it to a temporary folder (not the workspace). Do not unzip. Import this archive into Rule Designer (into a new workspace) using File -> Import….

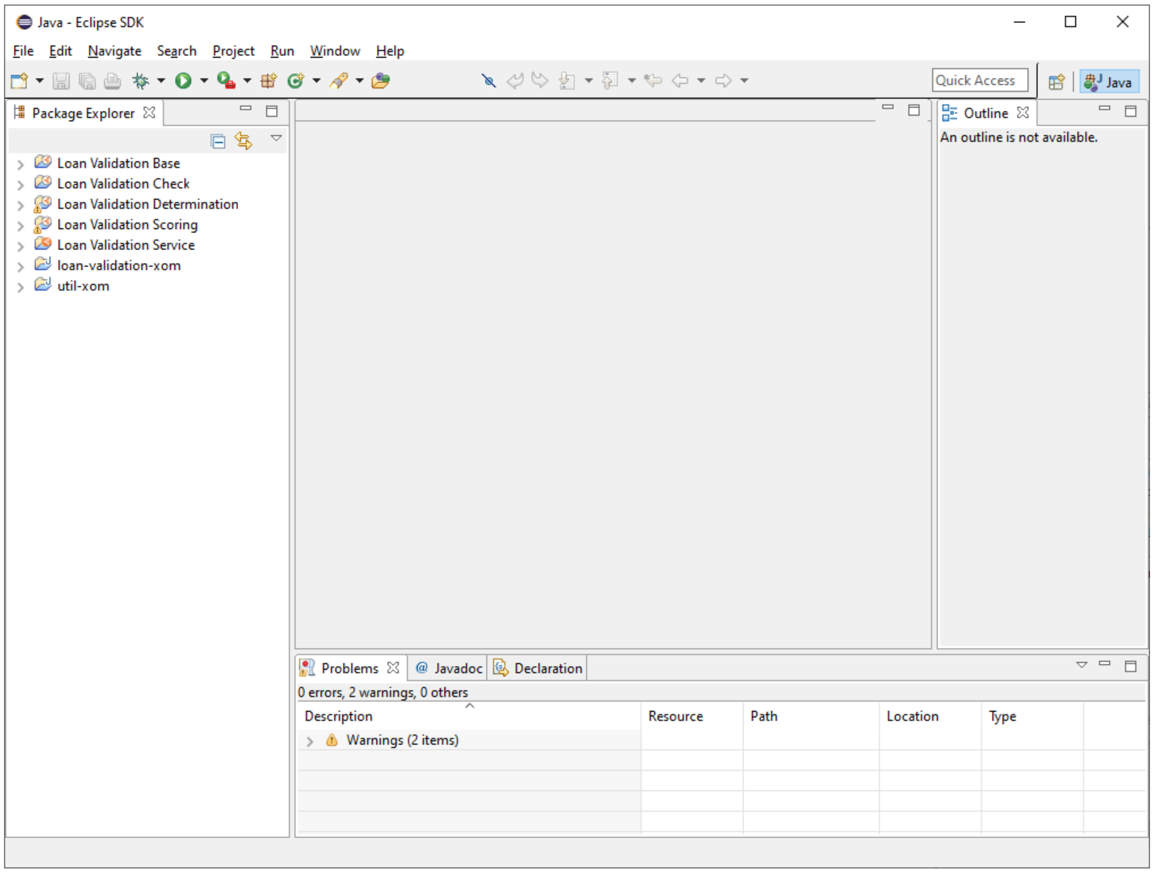




- [✓] Loan Validation Base (Loan Validation Base)
- [✓] Loan Validation Check (Loan Validation Check)
- [✓] Loan Validation Determination (Loan Validation Determination)
- [✓] Loan Validation Scoring (Loan Validation Scoring)
- [✓] Loan Validation Service (Loan Validation Service)
- [✓] loan-validation-xom (loan-validation-xom)
- [✓] util-xom (util-xom)

Options:
- [✓] Search for nested projects
- [✓] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

There should be no errors at this stage and your workspace should have the projects shown below.



Loan Validation Base
Loan Validation Check
Loan Validation Determination
Loan Validation Scoring
Loan Validation Service
loan-validation-xom
util-xom

An outline is not available.

Problems | Javadoc | Declaration

0 errors, 2 warnings, 0 others

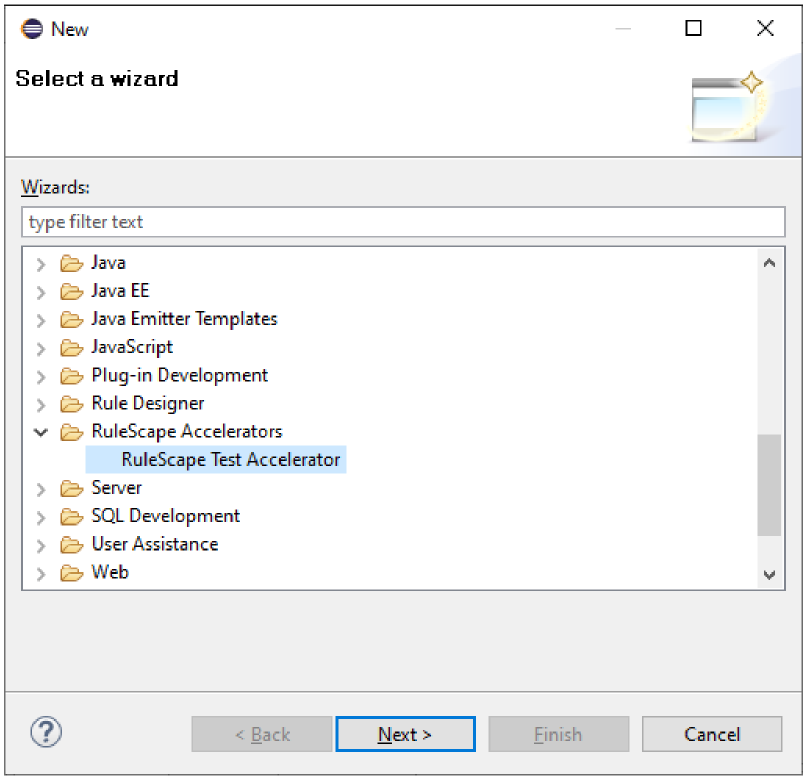
Description	Resource	Path	Location	Type
Warnings (2 items)				

Explore the rule projects. The ‘Loan Validation Service’ is the main decision service project.

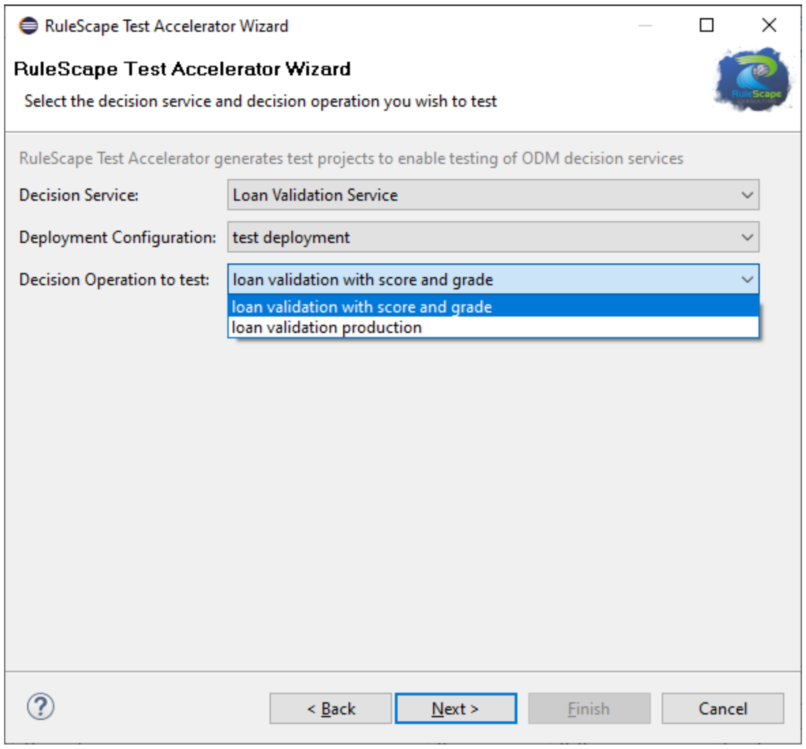
Run Test Accelerator

Now you are ready to run the Test Accelerator to create the test projects. Click on File->New->Other… and select RuleScape Test Accelerator.

Tip: You may search for ‘RuleScape’ in the wizards search box.



The Test Accelerator examines the workspace and identifies the decision service, rule parameters, etc. From the dropdown options, select ‘Loan Validation Service’ as the decision service; ‘test deployment’ as the deployment configuration and ‘loan validation with score and grade’ as the decision operation to test.


RuleScape Test Accelerator Wizard

RuleScape Test Accelerator Wizard

Select the decision service and decision operation you wish to test

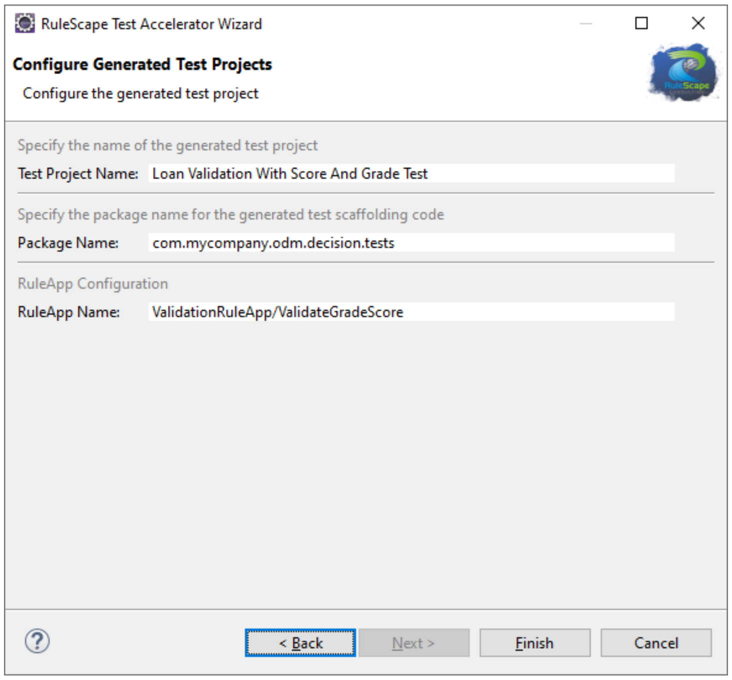
RuleScape Test Accelerator generates test projects to enable testing of ODM decision services

Decision Service:	Loan Validation Service
Deployment Configuration:	test deployment
Decision Operation to test:	loan validation with score and grade

loan validation with score and grade
loan validation production
< Back
Next >
Finish
Cancel

Next, you are presented with the screen to configure your test classes.

Important Tip: For the purposes of this tutorial, make sure you pick the defaults so you can use the downloaded files without having to refactor.


RuleScape Test Accelerator Wizard

Configure Generated Test Projects

Configure the generated test project

Specify the name of the generated test project

Test Project Name: Loan Validation With Score And Grade Test

Specify the package name for the generated test scaffolding code

Package Name: com.mycompany.odm.decision.tests

RuleApp Configuration

RuleApp Name: ValidationRuleApp/ValidateGradeScore

Potential Issue

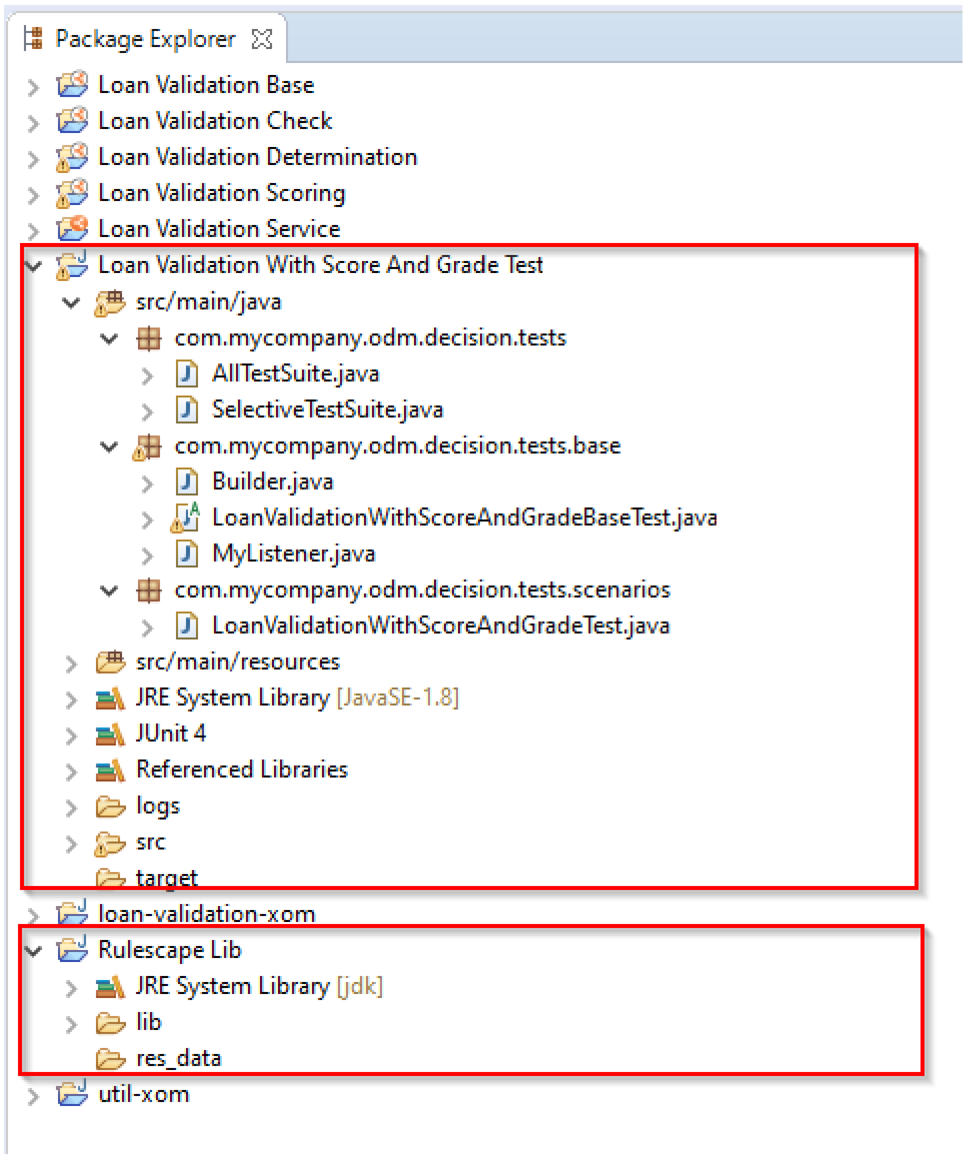
If you do not have the full Rule Designer installed, you do not have an Execution Server (i.e. Rule Designer plugin for Cloud installed). You may see this error message:

Unbound classpath variable: 'ILOG_EXECUTION_SERVER_HOME/lib/j2ee_connector-1_5-fr.jar' in project 'Loan Validation With Score And Grade Test'

Review of Generated Projects

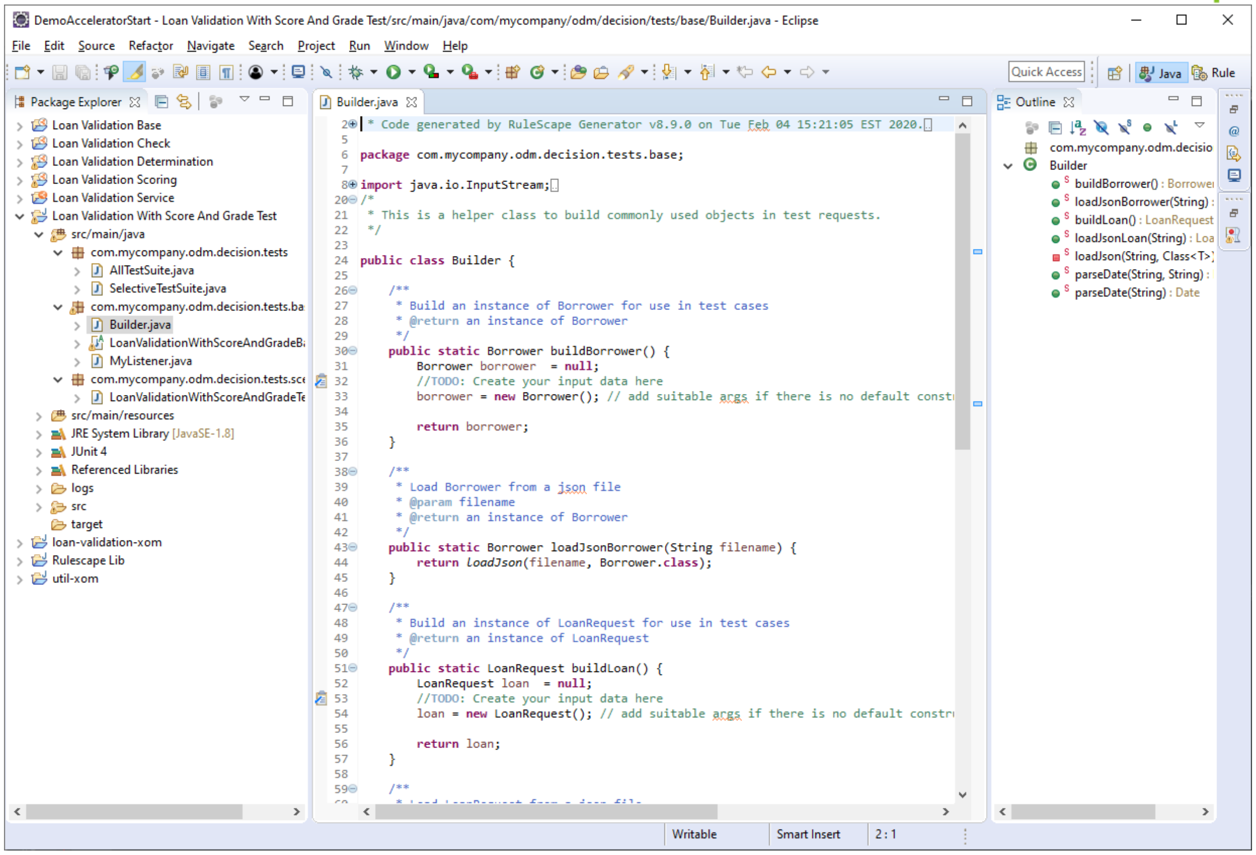
The Test Accelerator generates two Java projects: one to hold the Test Framework jars ('Rulescape Lib') and the other to hold the scaffolding files to test your decision service ('Loan Validation with Score and Grade Test'). Explore the files in these 2 projects. You will notice three java packages:

Package	**Contents**
com.mycompany.odm.decision.tests	Contains the test suite classes. AllTestSuite runs all the tests that are annotated with '@RunWith (ODMRunner.class)' or defined as subclasses of the base test class '`LoanValidationWithScoreAndGradeBaseTest`'
com.mycompany.odm.decision.tests.base	Contains an abstract base test class from which all your test classes for this decision operation should inherit. This base class identifies the rule app path and ruleset parameters. This is automatically generated by the Test Accelerator based on the configuration parameters. This package also contains a skeletal 'Builder.java' which contains utility methods for instantiating test data. Finally, the 'MyListener.java' class enables more advanced users to add their own listeners in addition to the ones provided by the framework.
com.mycompany.odm.decision.tests.scenarios	This package holds the test classes that contain test methods representing business scenarios. The plugin creates one sample test class. You should add other classes to test the decision operation.


Package Explorer

- Loan Validation Base
- Loan Validation Check
- Loan Validation Determination
- Loan Validation Scoring
- Loan Validation Service
- Loan Validation With Score And Grade Test
  - src/main/java
    - com.mycompany.odm.decision.tests
      - AllTestSuite.java
      - SelectiveTestSuite.java
    - com.mycompany.odm.decision.tests.base
      - Builder.java
      - LoanValidationWithScoreAndGradeBaseTest.java
      - MyListener.java
    - com.mycompany.odm.decision.tests.scenarios
      - LoanValidationWithScoreAndGradeTest.java
  - src/main/resources
  - JRE System Library [JavaSE-1.8]
  - JUnit 4
  - Referenced Libraries
  - logs
  - src
  - target
- loan-validation-xom
- Rulescape Lib
  - JRE System Library [jdk]
  - lib
  - res_data
- util-xom

For the purposes of this tutorial, we will update the Builder and test classes. The generated Builder.java contains boilerplate methods to instantiate your ruleset parameters. You are expected to enhance these methods to suit your test cases.


DemoAcceleratorStart - Loan Validation With Score And Grade Test/src/main/java/com/mycompany/odm/decision/tests/base/Builder.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Rule
Package Explorer
Loan Validation Base
Loan Validation Check
Loan Validation Determination
Loan Validation Scoring
Loan Validation Service
Loan Validation With Score And Grade Test
src/main/java
com.mycompany.odm.decision.tests
AllTestSuite.java
SelectiveTestSuite.java
com.mycompany.odm.decision.tests.ba
Builder.java
LoanValidationWithScoreAndGradeBi
MyListener.java
com.mycompany.odm.decision.tests.sce
LoanValidationWithScoreAndGradeTe
src/main/resources
JRE System Library [JavaSE-1.8]
JUnit 4
Referenced Libraries
logs
src
target
loan-validation-xom
Rulescape Lib
util-xom
Builder.java
Outline
com.mycompany.odm.decisio
Builder
buildBorrower() : Borrower
loadJsonBorrower(String) :
buildLoan() : LoanRequest
loadJsonLoan(String) : Loa
loadJson(String, Class<T>)
parseDate(String, String) :
parseDate(String) : Date
Writable
Smart Insert
2 : 1

```
 * Code generated by RuleScape Generator v8.9.0 on Tue Feb 04 15:21:05 EST 2020.

package com.mycompany.odm.decision.tests.base;

import java.io.InputStream;
/*
 * This is a helper class to build commonly used objects in test requests.
 */

public class Builder {

    /**
     * Build an instance of Borrower for use in test cases
     * @return an instance of Borrower
     */
    public static Borrower buildBorrower() {
        Borrower borrower  = null;
        //TODO: Create your input data here
        borrower = new Borrower(); // add suitable args if there is no default const

        return borrower;
    }

    /**
     * Load Borrower from a json file
     * @param filename
     * @return an instance of Borrower
     */
    public static Borrower loadJsonBorrower(String filename) {
        return loadJson(filename, Borrower.class);
    }

    /**
     * Build an instance of LoanRequest for use in test cases
     * @return an instance of LoanRequest
     */
    public static LoanRequest buildLoan() {
        LoanRequest loan  = null;
        //TODO: Create your input data here
        loan = new LoanRequest(); // add suitable args if there is no default constr

        return loan;
    }

    /**
```

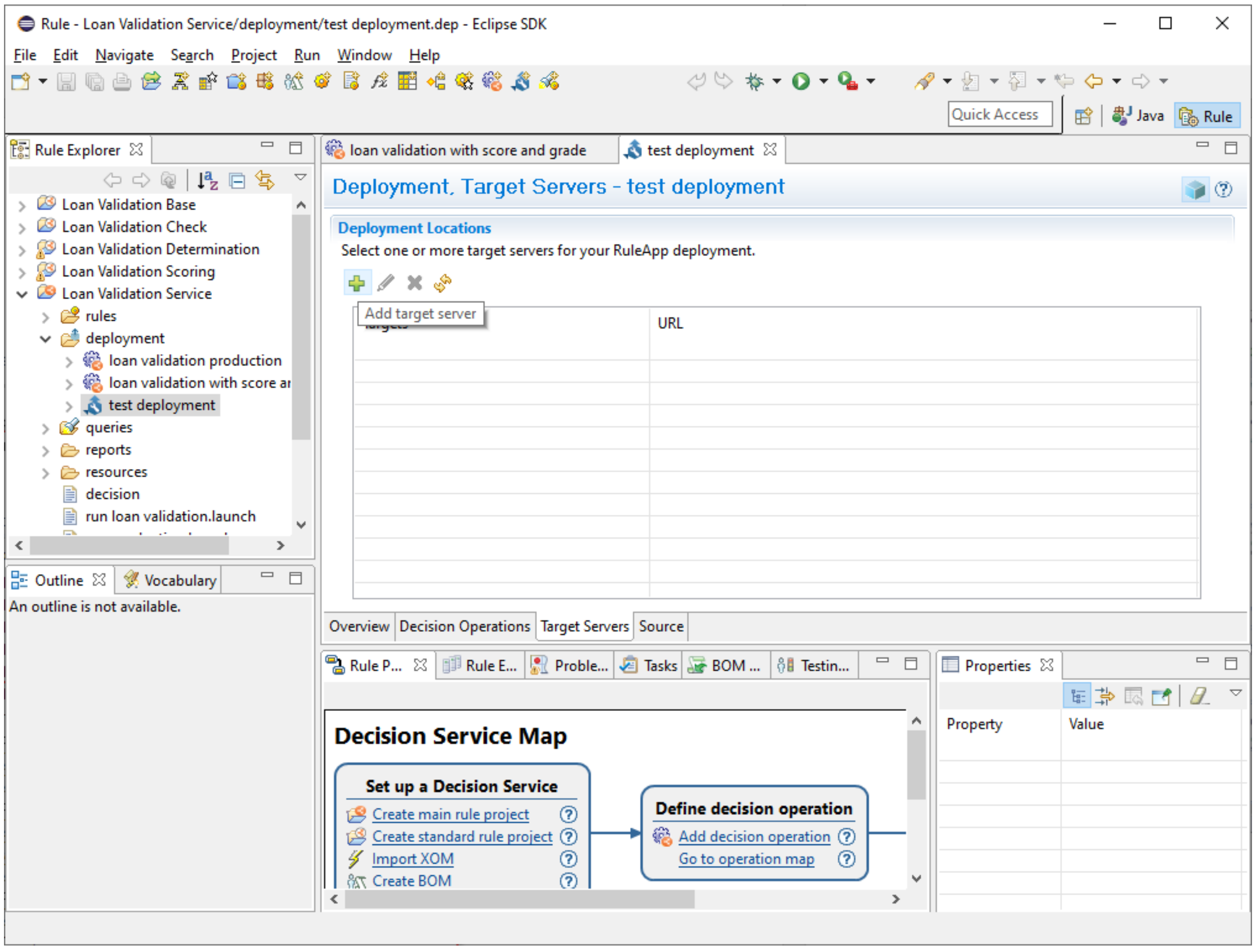
Update Source Code

Update builder methods (in Builder.java) by replacing the generated methods with the ones below. These methods construct test requests that can then be tweaked as part of each test case.

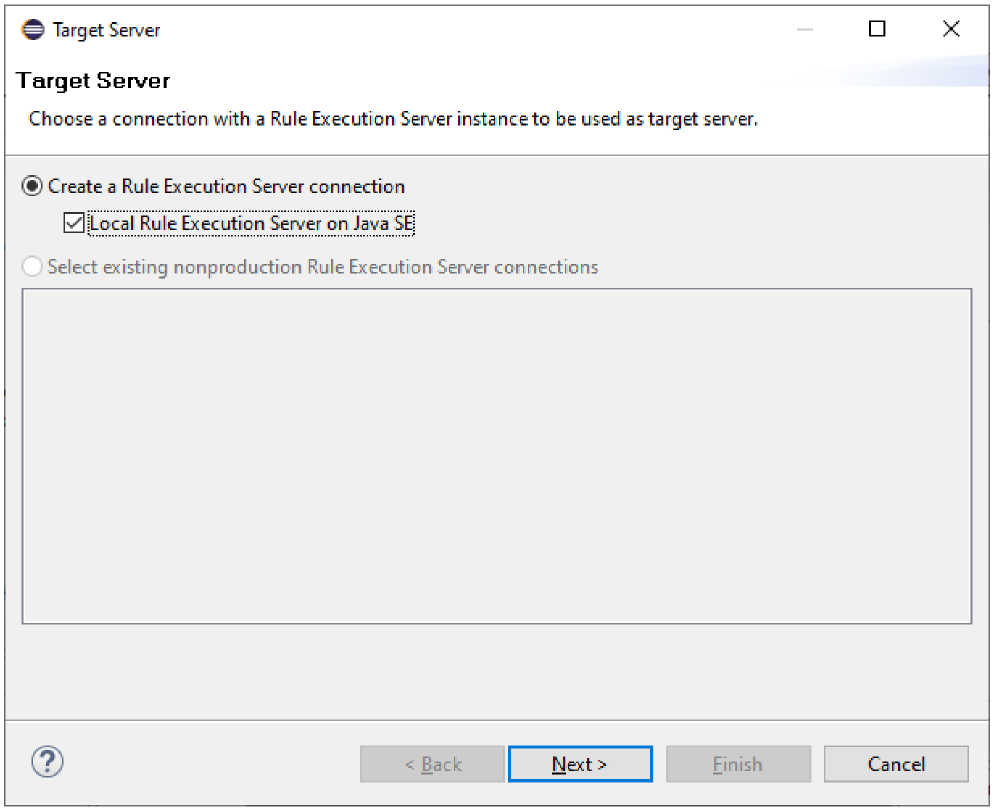
```
/**
 * Build an instance of Borrower for use in test cases
 * @return an instance of Borrower
 */
public static Borrower buildBorrower() {
   Borrower borrower  = null;
   borrower =
      new Borrower("Peter", "Parker", parseDate("01/01/1980"), "111-22-3333");
   borrower.setZipCode("90102");
   borrower.setCreditScore(699);
   borrower.setYearlyIncome(200000);
   return borrower;
}


/**
 * Build an instance of LoanRequest for use in test cases
 * @return an instance of LoanRequest
 */
public static LoanRequest buildLoan() {
   LoanRequest loan  = null;
   loan = new LoanRequest(new Date(), 360, 420000, 0.5d);
   return loan;
}
```

Update Deployment Configuration and Deploy

Switch to rule perspective. Add Target server to the ‘test deployment’ deployment configuration:


Rule - Loan Validation Service/deployment/test deployment.dep - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Quick Access
Java
Rule
Rule Explorer
Loan Validation Base
Loan Validation Check
Loan Validation Determination
Loan Validation Scoring
Loan Validation Service
rules
deployment
loan validation production
loan validation with score ar
test deployment
queries
reports
resources
decision
run loan validation.launch
loan validation with score and grade
test deployment
Deployment, Target Servers - test deployment
Deployment Locations
Select one or more target servers for your RuleApp deployment.
Add target server
URL
Overview
Decision Operations
Target Servers
Source
Outline
Vocabulary
An outline is not available.
Rule P...
Rule E...
Proble...
Tasks
BOM ...
Testin...
Decision Service Map
Set up a Decision Service
Create main rule project
Create standard rule project
Import XOM
Create BOM
Define decision operation
Add decision operation
Go to operation map
Properties
Property
Value

Select ‘Local Rule Execution Server on Java SE’.


Target Server

Target Server

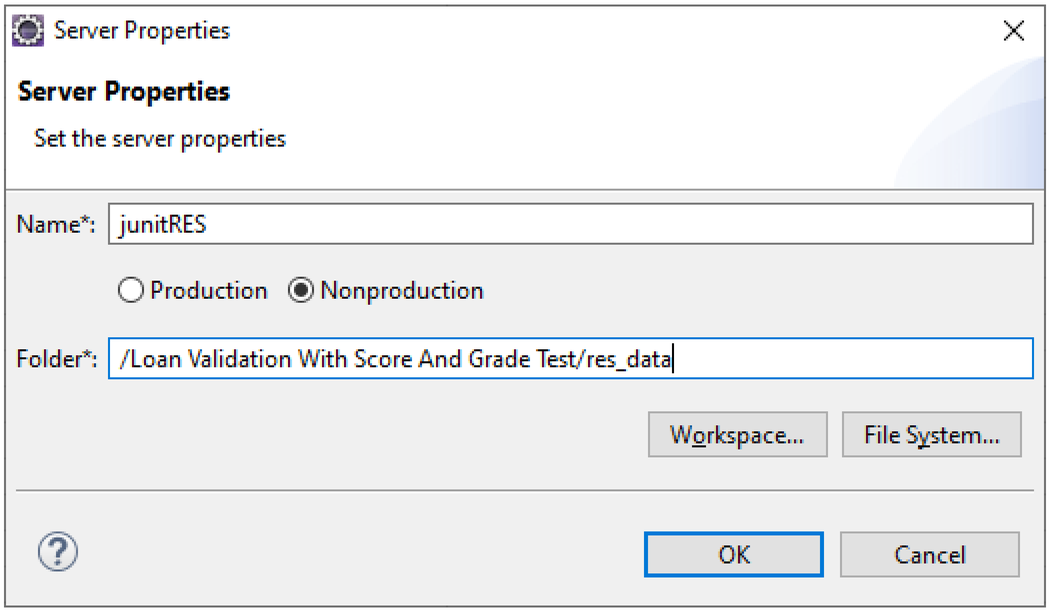
Choose a connection with a Rule Execution Server instance to be used as target server.

● Create a Rule Execution Server connection

☑ Local Rule Execution Server on Java SE

○ Select existing nonproduction Rule Execution Server connections

Pick 'Workspace..' and enter '/Loan Validation With Score And Grade Test/res_data' as shown below.


Server Properties

Server Properties

Set the server properties

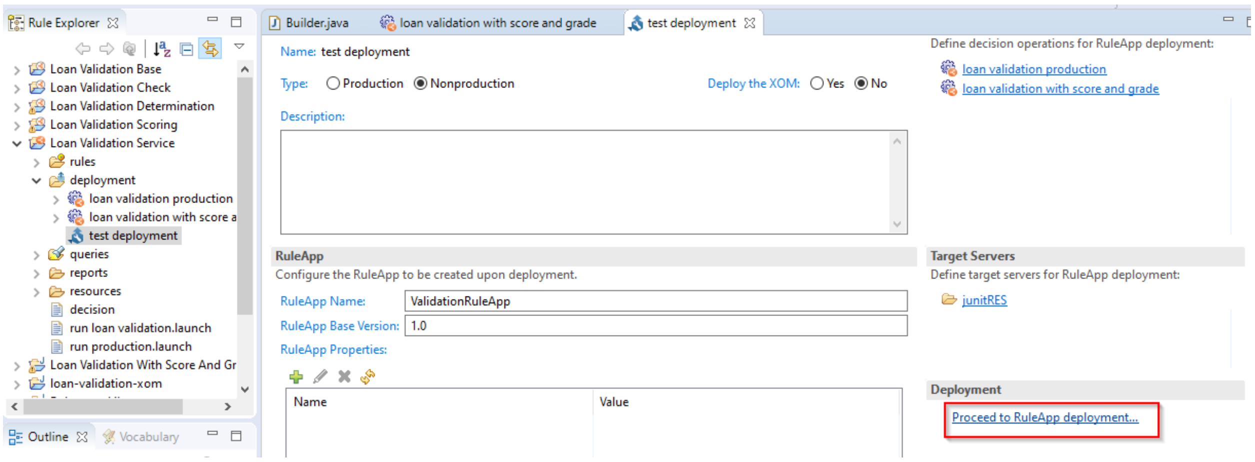
Name*: junitRES

○ Production ● Nonproduction

Folder*: /Loan Validation With Score And Grade Test/res_data

Deploy RuleApp

Once the deployment configuration is ready, you may proceed to RuleApp deployment.



Name: test deployment

Type: Production [✓ Nonproduction]

Deploy the XOM: Yes [✓ No]

Description:

RuleApp

Configure the RuleApp to be created upon deployment.

RuleApp Name: ValidationRuleApp

RuleApp Base Version: 1.0

RuleApp Properties:

Name	Value

Define decision operations for RuleApp deployment:

- loan validation production
- loan validation with score and grade

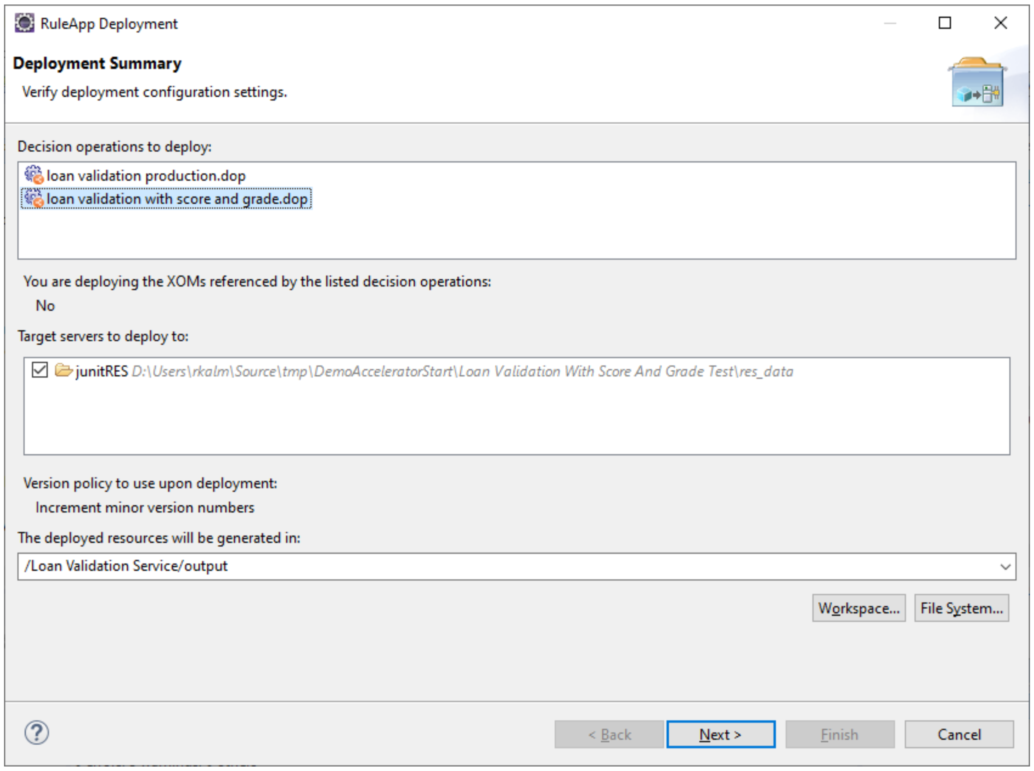
Target Servers

Define target servers for RuleApp deployment:

- junitRES

Deployment

Proceed to RuleApp deployment...



Deployment Summary

Verify deployment configuration settings.

Decision operations to deploy:

- loan validation production.dop
- loan validation with score and grade.dop

You are deploying the XOMs referenced by the listed decision operations:

No

Target servers to deploy to:

☑ junitRES *D:\Users\rkalm\Source\tmp\DemoAcceleratorStart\Loan Validation With Score And Grade Test\res_data*

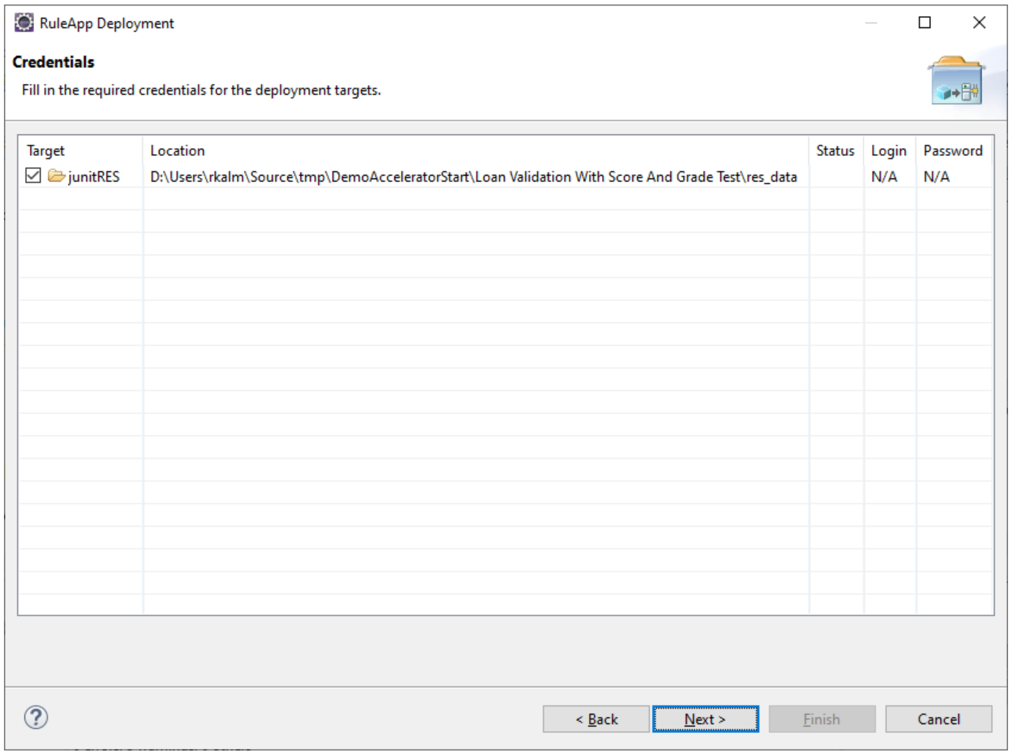
Version policy to use upon deployment:

Increment minor version numbers

The deployed resources will be generated in:

/Loan Validation Service/output

Click ‘Next>’.


RuleApp Deployment

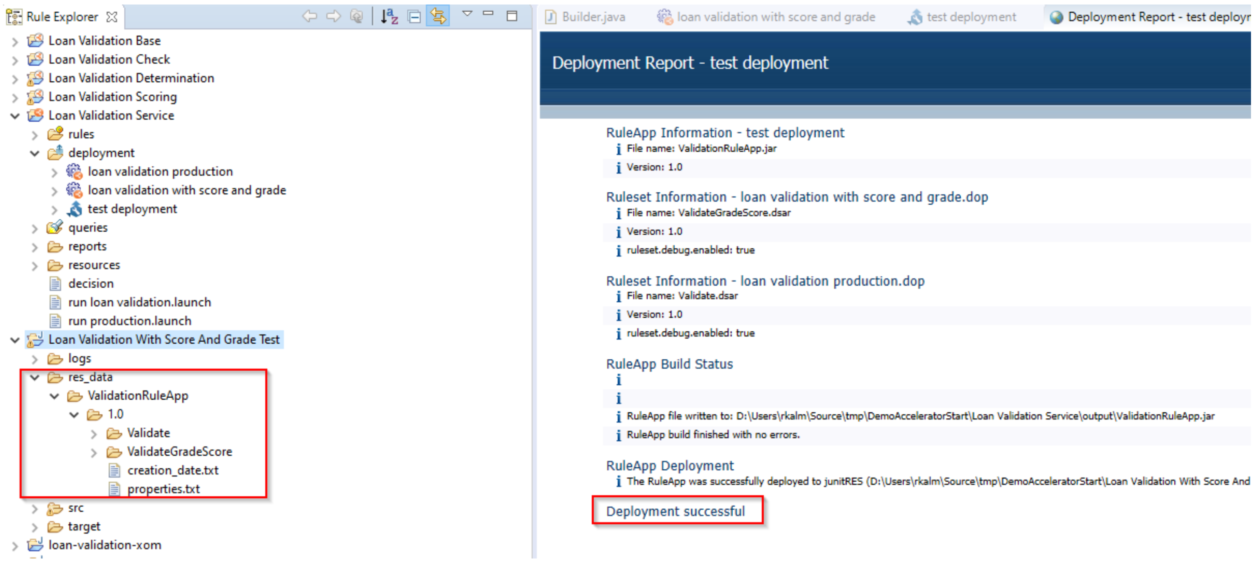
Credentials

Fill in the required credentials for the deployment targets.

Target	Location	Status	Login	Password
☑ junitRES	D:\Users\rkalm\Source\tmp\DemoAcceleratorStart\Loan Validation With Score And Grade Test\res_data		N/A	N/A

Click ‘Next >’ and then ‘Finish’.

The deployment report should display a “Deployment successful” message. The generated ruleapp can be found in the res_data directory:



Deployment Report - test deployment

RuleApp Information - test deployment

```
i File name: ValidationRuleApp.jar
i Version: 1.0
```

Ruleset Information - loan validation with score and grade.dop

```
i File name: ValidateGradeScore.dsar
i Version: 1.0
i ruleset.debug.enabled: true
```

Ruleset Information - loan validation production.dop

```
i File name: Validate.dsar
i Version: 1.0
i ruleset.debug.enabled: true
```

RuleApp Build Status

```
i
i
i RuleApp file written to: D:\Users\rkalm\Source\tmp\DemoAcceleratorStart\Loan Validation Service\output\ValidationRuleApp.jar
i RuleApp build finished with no errors.
```

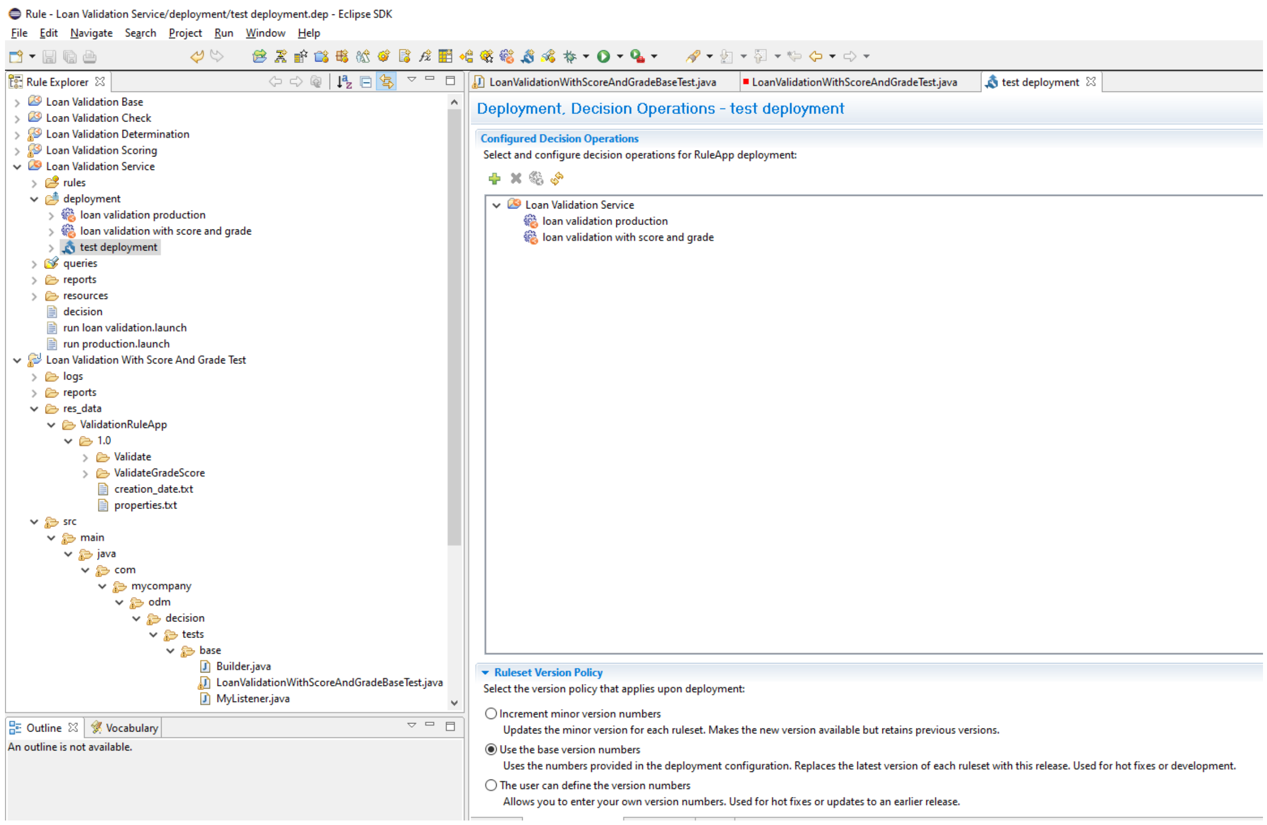
RuleApp Deployment

```
i The RuleApp was successfully deployed to junitRES (D:\Users\rkalm\Source\tmp\DemoAcceleratorStart\Loan Validation With Score And
```

Deployment successful

Tip: if you encounter a RuntimeException indicating 'Invalid path', make sure you select the 'Use the base version numbers' as shown in the diagram below.

```
 java.lang.RuntimeException: Invalid ruleapp path. Original exception: Invalid path
"ValidationRuleApp/ValidateGradeScore".

   at com.rulescape.odmtest.api.engine.RuleExecutionRequest.<init>(Unknown Source)
```


Rule - Loan Validation Service/deployment/test deployment.dep - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Rule Explorer
LoanValidationWithScoreAndGradeBaseTest.java
LoanValidationWithScoreAndGradeTest.java
test deployment
Outline
Vocabulary

Rule Explorer:

```
> Loan Validation Base
> Loan Validation Check
> Loan Validation Determination
> Loan Validation Scoring
v Loan Validation Service
  > rules
  v deployment
    > loan validation production
    > loan validation with score and grade
    > test deployment
  > queries
  > reports
  > resources
    decision
    run loan validation.launch
    run production.launch
v Loan Validation With Score And Grade Test
  > logs
  > reports
  v res_data
    v ValidationRuleApp
      v 1.0
        > Validate
        > ValidateGradeScore
          creation_date.txt
          properties.txt
  v src
    v main
      v java
        v com
          v mycompany
            v odm
              v decision
                v tests
                  v base
                      Builder.java
                      LoanValidationWithScoreAndGradeBaseTest.java
                      MyListener.java
```

An outline is not available.

Deployment, Decision Operations - test deployment

Configured Decision Operations

Select and configure decision operations for RuleApp deployment:

```
v Loan Validation Service
    loan validation production
    loan validation with score and grade
```

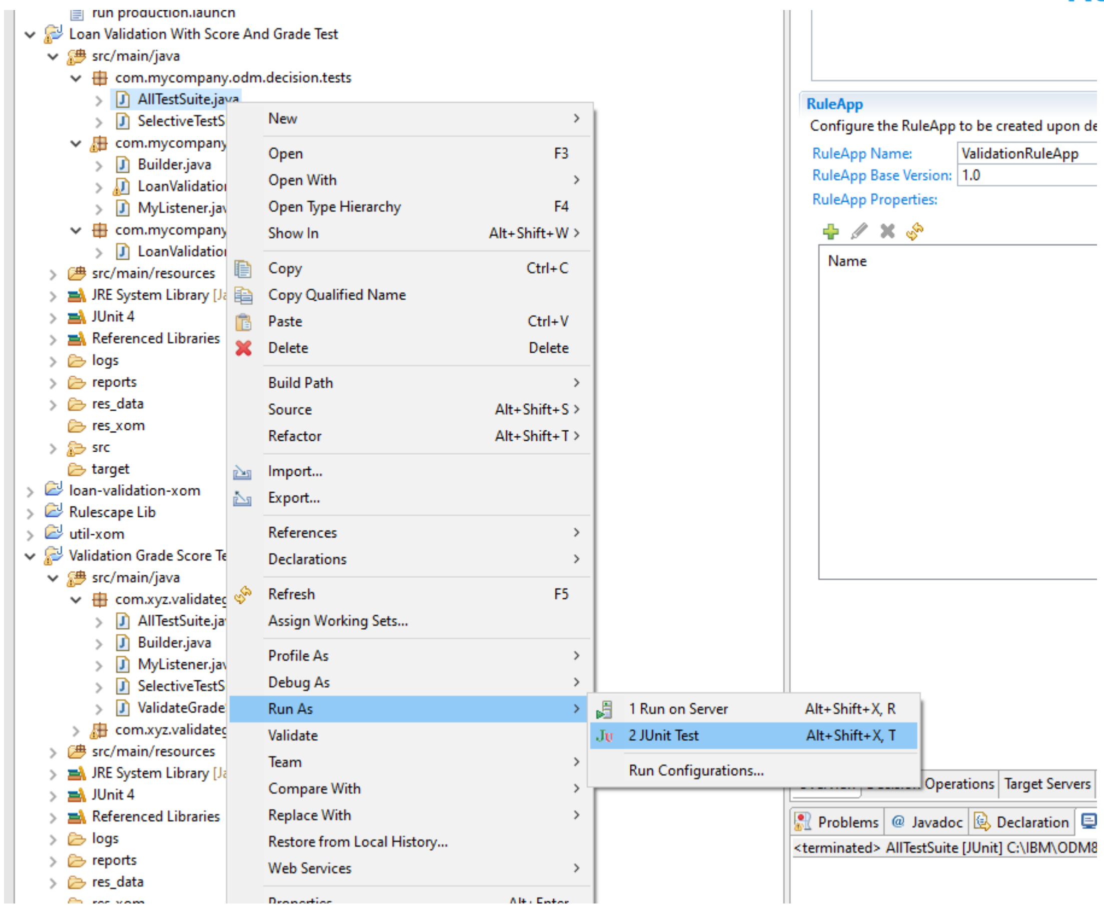
Ruleset Version Policy

Select the version policy that applies upon deployment:

- ○ Increment minor version numbers
  Updates the minor version for each ruleset. Makes the new version available but retains previous versions.
- ● Use the base version numbers
  Uses the numbers provided in the deployment configuration. Replaces the latest version of each ruleset with this release. Used for hot fixes or development.
- ○ The user can define the version numbers
  Allows you to enter your own version numbers. Used for hot fixes or updates to an earlier release.

Run the tests

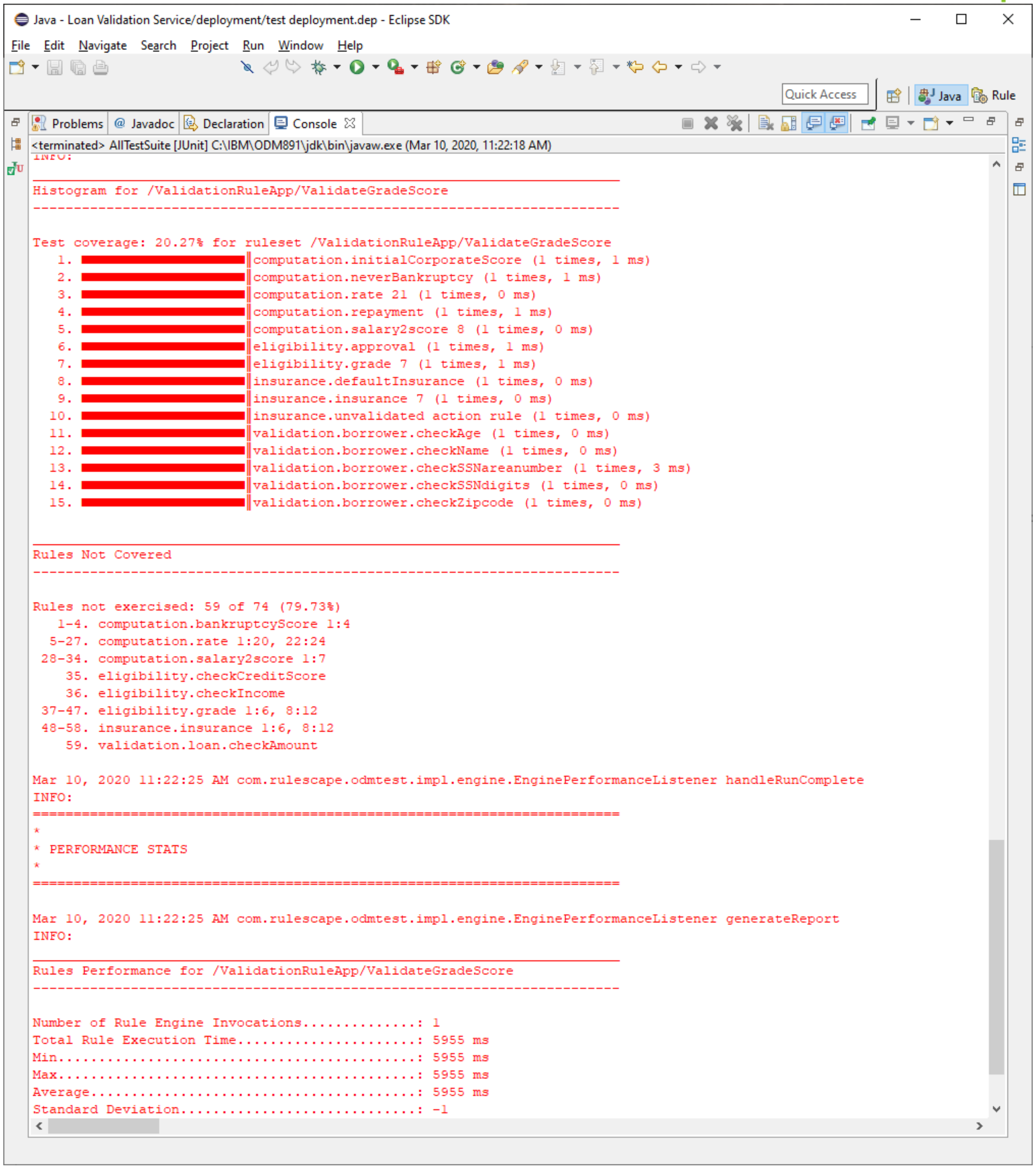
Switch to the Java perspective. Right click on 'AllTestSuite.java' and run as JUnit test.



This runs all the tests. At this stage, there is only one test class (LoanValidationWithScoreAndGradeTest) with one test method. In the console, you can see the following:

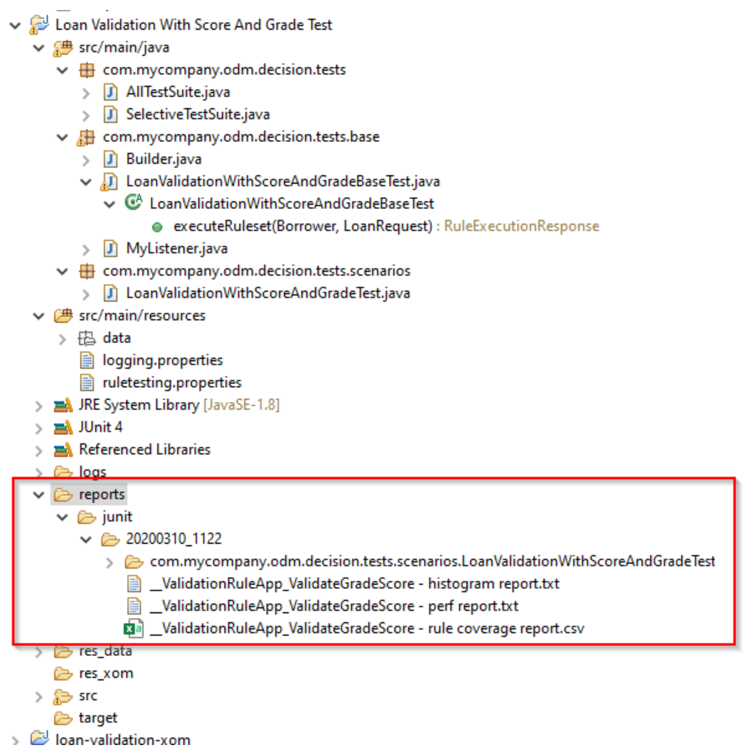
1. For each test:
   a. The request and response data in JSON
   b. A rule trace displaying the rule tasks and rules that were executed for this test
2. A histogram of all the rules that fired for all the tests
3. A listing of rules that were not executed by any of the tests
4. A performance report

The screenshot below shows this partially.


Java - Loan Validation Service/deployment/test deployment.dep - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Quick Access
Java
Rule
Problems
Javadoc
Declaration
Console

```
<terminated> AllTestSuite [JUnit] C:\IBM\ODM891\jdk\bin\javaw.exe (Mar 10, 2020, 11:22:18 AM)
INFO:
________________________________________________________________________
Histogram for /ValidationRuleApp/ValidateGradeScore
------------------------------------------------------------------------

Test coverage: 20.27% for ruleset /ValidationRuleApp/ValidateGradeScore
   1. ████████████████████|computation.initialCorporateScore (1 times, 1 ms)
   2. ████████████████████|computation.neverBankruptcy (1 times, 1 ms)
   3. ████████████████████|computation.rate 21 (1 times, 0 ms)
   4. ████████████████████|computation.repayment (1 times, 1 ms)
   5. ████████████████████|computation.salary2score 8 (1 times, 0 ms)
   6. ████████████████████|eligibility.approval (1 times, 1 ms)
   7. ████████████████████|eligibility.grade 7 (1 times, 1 ms)
   8. ████████████████████|insurance.defaultInsurance (1 times, 0 ms)
   9. ████████████████████|insurance.insurance 7 (1 times, 0 ms)
  10. ████████████████████|insurance.unvalidated action rule (1 times, 0 ms)
  11. ████████████████████|validation.borrower.checkAge (1 times, 0 ms)
  12. ████████████████████|validation.borrower.checkName (1 times, 0 ms)
  13. ████████████████████|validation.borrower.checkSSNareanumber (1 times, 3 ms)
  14. ████████████████████|validation.borrower.checkSSNdigits (1 times, 0 ms)
  15. ████████████████████|validation.borrower.checkZipcode (1 times, 0 ms)

________________________________________________________________________
Rules Not Covered
------------------------------------------------------------------------

Rules not exercised: 59 of 74 (79.73%)
   1-4. computation.bankruptcyScore 1:4
  5-27. computation.rate 1:20, 22:24
 28-34. computation.salary2score 1:7
    35. eligibility.checkCreditScore
    36. eligibility.checkIncome
 37-47. eligibility.grade 1:6, 8:12
 48-58. insurance.insurance 1:6, 8:12
    59. validation.loan.checkAmount

Mar 10, 2020 11:22:25 AM com.rulescape.odmtest.impl.engine.EnginePerformanceListener handleRunComplete
INFO:
========================================================================
*
* PERFORMANCE STATS
*
========================================================================

Mar 10, 2020 11:22:25 AM com.rulescape.odmtest.impl.engine.EnginePerformanceListener generateReport
INFO:
________________________________________________________________________
Rules Performance for /ValidationRuleApp/ValidateGradeScore
------------------------------------------------------------------------

Number of Rule Engine Invocations..............: 1
Total Rule Execution Time......................: 5955 ms
Min............................................: 5955 ms
Max............................................: 5955 ms
Average........................................: 5955 ms
Standard Deviation.............................: -1
```

The output can be controlled by configuration parameters in ruletesting.properties that is present in src/main/resources of the test project. Since 'write.trace.files' is set to true, these reports are written out to the filesystem under reports/junit (also configurable).



```
Loan Validation With Score And Grade Test
  src/main/java
    com.mycompany.odm.decision.tests
      AllTestSuite.java
      SelectiveTestSuite.java
    com.mycompany.odm.decision.tests.base
      Builder.java
      LoanValidationWithScoreAndGradeBaseTest.java
        LoanValidationWithScoreAndGradeBaseTest
          executeRuleset(Borrower, LoanRequest) : RuleExecutionResponse
      MyListener.java
    com.mycompany.odm.decision.tests.scenarios
      LoanValidationWithScoreAndGradeTest.java
  src/main/resources
    data
    logging.properties
    ruletesting.properties
  JRE System Library [JavaSE-1.8]
  JUnit 4
  Referenced Libraries
  logs
  reports
    junit
      20200310_1122
        com.mycompany.odm.decision.tests.scenarios.LoanValidationWithScoreAndGradeTest
        __ValidationRuleApp_ValidateGradeScore - histogram report.txt
        __ValidationRuleApp_ValidateGradeScore - perf report.txt
        __ValidationRuleApp_ValidateGradeScore - rule coverage report.csv
  res_data
  res_xom
  src
  target
loan-validation-xom
```

Tip: you will need to refresh the project to view the newly created files.

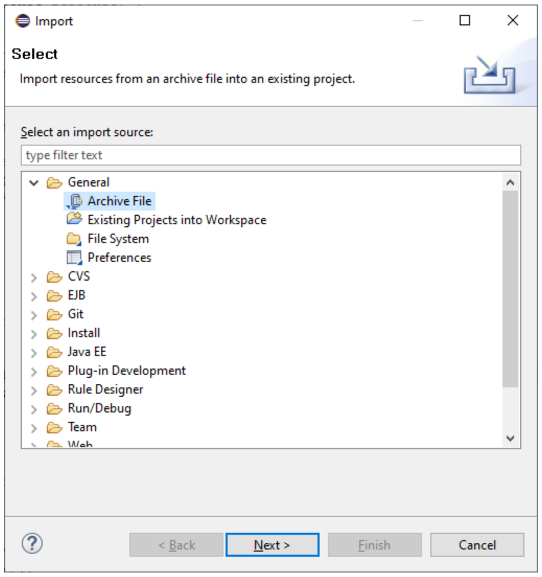
You will notice that there is an additional report '__ValidationRuleApp_ValidateGradeScore - rule coverage report.csv' that identifies the tests that executed a rule. This report is useful when the number of rules and tests grow to a sizable number.

Create additional tests

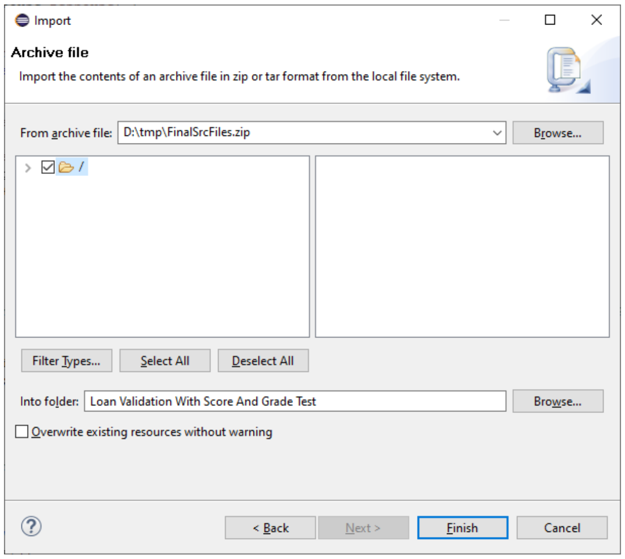
In this part of the tutorial, you will get a feel for how additional test cases are added. You will also see how a customer listener can be added.

Each logical grouping of test scenarios is placed in a test class, with each scenario being a test method. To add more tests, we will add more test classes. Instead of hand-coding them, you can download the files and import them.

1. Download file from https://github.com/rulescape/rulescape.github.io/blob/master/test-accelerator-tutorial/FinalSrcFiles.zip and save it to a local directory (not the workspace)
2. Import: right click on the test project and select Import… -> General -> Archive File



Select the downloaded archive file:



From archive file: D:\tmp\FinalSrcFiles.zip

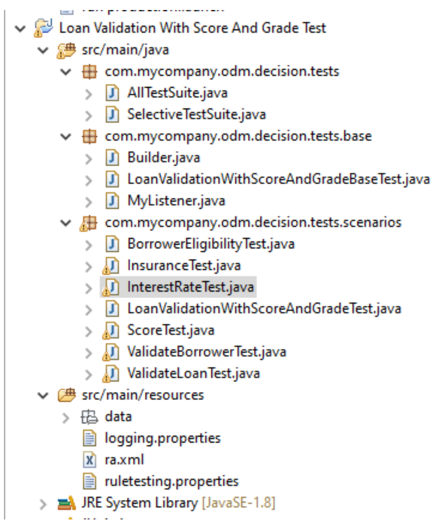
☑ /

Into folder: Loan Validation With Score And Grade Test

☐ Overwrite existing resources without warning

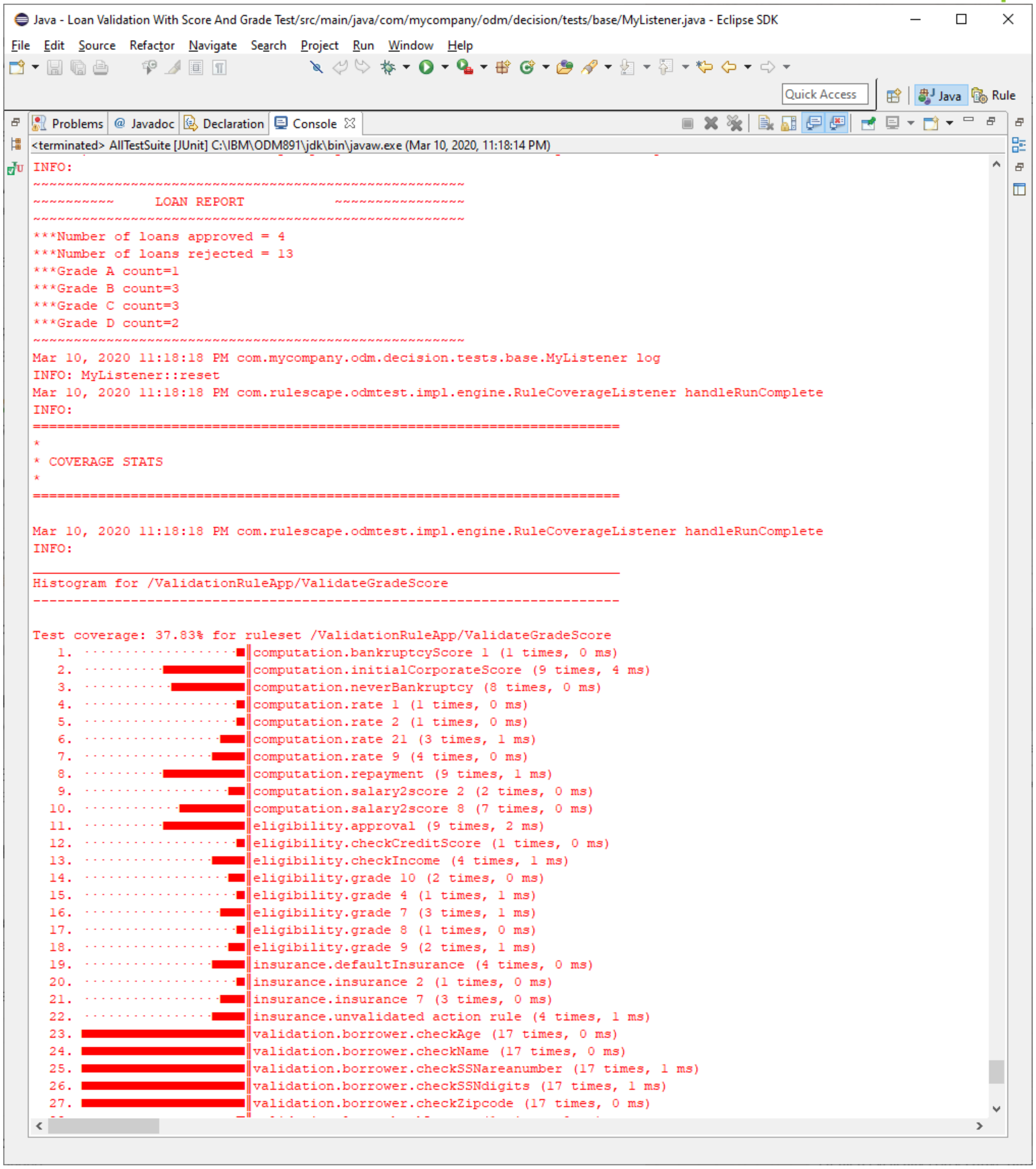
Click ‘Yes to All’ when prompted to if you want to overwrite.

You will now see several new test classes in the scenarios package.


```
Loan Validation With Score And Grade Test
  src/main/java
    com.mycompany.odm.decision.tests
      AllTestSuite.java
      SelectiveTestSuite.java
    com.mycompany.odm.decision.tests.base
      Builder.java
      LoanValidationWithScoreAndGradeBaseTest.java
      MyListener.java
    com.mycompany.odm.decision.tests.scenarios
      BorrowerEligibilityTest.java
      InsuranceTest.java
      InterestRateTest.java
      LoanValidationWithScoreAndGradeTest.java
      ScoreTest.java
      ValidateBorrowerTest.java
      ValidateLoanTest.java
  src/main/resources
    data
    logging.properties
    ra.xml
    ruletesting.properties
  JRE System Library [JavaSE-1.8]
```

Browse *InterestRateTest.java* to see how test cases are defined relating to interest rate. Also, browse *MyListener.java* to see how you can create custom reports -- in this case to get a histogram of credit ratings and loan approval counts for all the tests.

To run, as before execute AllTestSuite.java as a JUnit test. A partial screenshot of the results is shown below:


Java - Loan Validation With Score And Grade Test/src/main/java/com/mycompany/odm/decision/tests/base/MyListener.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Rule
Problems
Javadoc
Declaration
Console

```
<terminated> AllTestSuite [JUnit] C:\IBM\ODM891\jdk\bin\javaw.exe (Mar 10, 2020, 11:18:14 PM)
INFO:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~     LOAN REPORT           ~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
***Number of loans approved = 4
***Number of loans rejected = 13
***Grade A count=1
***Grade B count=3
***Grade C count=3
***Grade D count=2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Mar 10, 2020 11:18:18 PM com.mycompany.odm.decision.tests.base.MyListener log
INFO: MyListener::reset
Mar 10, 2020 11:18:18 PM com.rulescape.odmtest.impl.engine.RuleCoverageListener handleRunComplete
INFO:
========================================================================
*
* COVERAGE STATS
*
========================================================================

Mar 10, 2020 11:18:18 PM com.rulescape.odmtest.impl.engine.RuleCoverageListener handleRunComplete
INFO:
________________________________________________________________________
Histogram for /ValidationRuleApp/ValidateGradeScore
------------------------------------------------------------------------

Test coverage: 37.83% for ruleset /ValidationRuleApp/ValidateGradeScore
   1. ···················█|computation.bankruptcyScore 1 (1 times, 0 ms)
   2. ··········██████████|computation.initialCorporateScore (9 times, 4 ms)
   3. ···········█████████|computation.neverBankruptcy (8 times, 0 ms)
   4. ···················█|computation.rate 1 (1 times, 0 ms)
   5. ···················█|computation.rate 2 (1 times, 0 ms)
   6. ·················███|computation.rate 21 (3 times, 1 ms)
   7. ················████|computation.rate 9 (4 times, 0 ms)
   8. ··········██████████|computation.repayment (9 times, 1 ms)
   9. ··················██|computation.salary2score 2 (2 times, 0 ms)
  10. ············████████|computation.salary2score 8 (7 times, 0 ms)
  11. ··········██████████|eligibility.approval (9 times, 2 ms)
  12. ···················█|eligibility.checkCreditScore (1 times, 0 ms)
  13. ················████|eligibility.checkIncome (4 times, 1 ms)
  14. ··················██|eligibility.grade 10 (2 times, 0 ms)
  15. ···················█|eligibility.grade 4 (1 times, 1 ms)
  16. ·················███|eligibility.grade 7 (3 times, 1 ms)
  17. ···················█|eligibility.grade 8 (1 times, 0 ms)
  18. ··················██|eligibility.grade 9 (2 times, 1 ms)
  19. ················████|insurance.defaultInsurance (4 times, 0 ms)
  20. ···················█|insurance.insurance 2 (1 times, 0 ms)
  21. ·················███|insurance.insurance 7 (3 times, 0 ms)
  22. ················████|insurance.unvalidated action rule (4 times, 1 ms)
  23. ████████████████████|validation.borrower.checkAge (17 times, 0 ms)
  24. ████████████████████|validation.borrower.checkName (17 times, 0 ms)
  25. ████████████████████|validation.borrower.checkSSNareanumber (17 times, 1 ms)
  26. ████████████████████|validation.borrower.checkSSNdigits (17 times, 1 ms)
  27. ████████████████████|validation.borrower.checkZipcode (17 times, 0 ms)
```