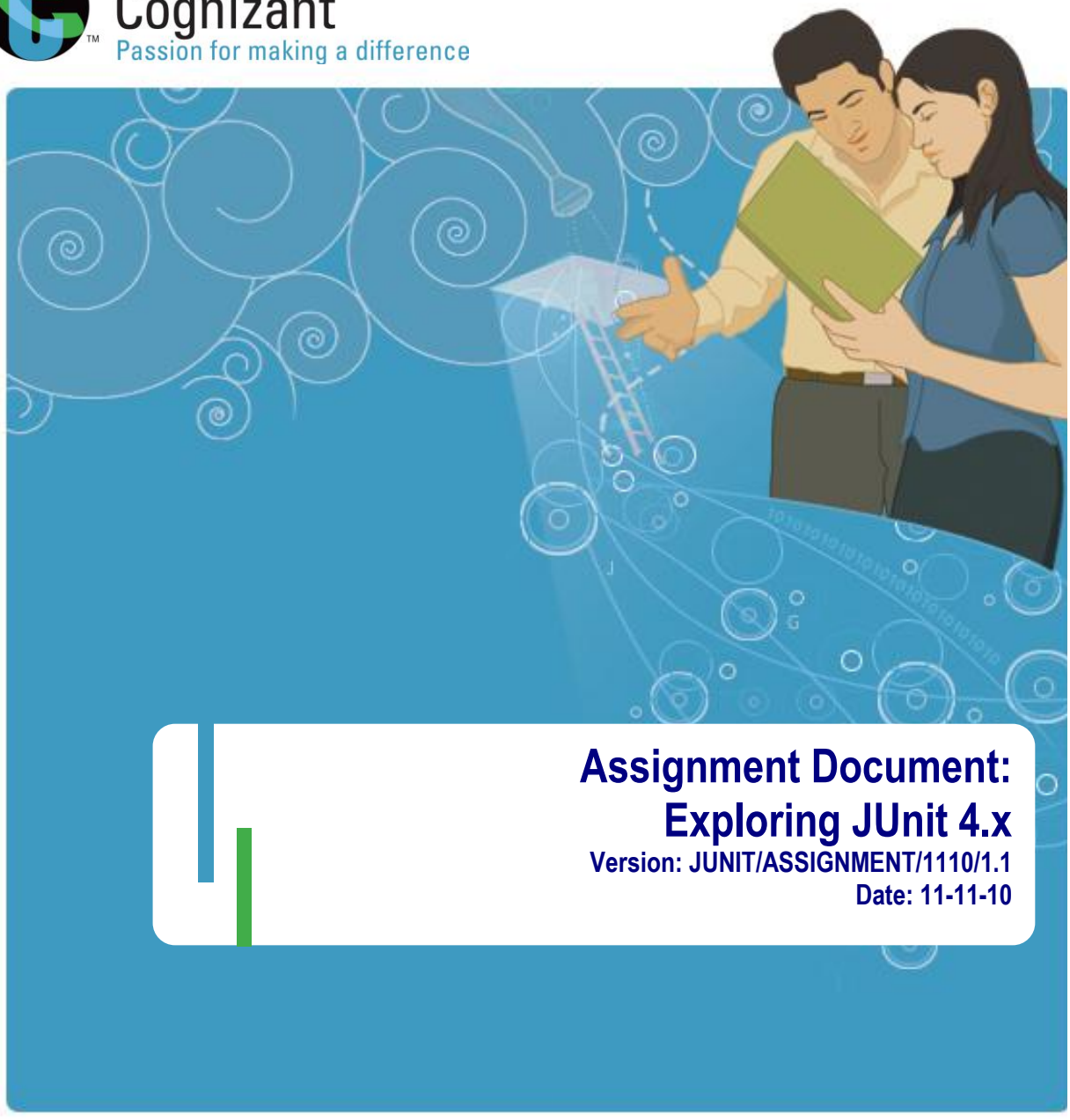
Cognizant
Passion for making a difference



Assignment Document: Exploring JUnit 4.x

Version: JUNIT/ASSIGNMENT/1110/1.1
Date: 11-11-10

Cognizant
500, Glen Pointe Center West,
Teaneck, NJ 07666.
Ph: 201-801-0233
www.cognizant.com

Contents

Topic: Exploring JUnit 4.x

Hands-On Exercises

Hands-On Exercise 1: Write business classes

Estimated Completion Time: 30 Minutes

(xx Marks)

Objective:

- Create classes using sound design principles
- Apply interfaces, inheritance and refactoring techniques

Complete the following assignment:

i. Create Scholar, ResearchScholar and StudentScholar in com.junit.scholar package

ii. Create an interface scholar with the following methods :

Members	**Purpose**
boolean register(String name, int age, String subject, String guideName)	Registers the details of the scholar
void rank(int marks)	Assigns a rank based on the marks

iii. Create a class ResearchScholar which implements Scholar

Members	**Purpose**
boolean register(String name, int age, String subject, String guideName)	Registers the details of the scholar if age >= 28 years
void rank(int marks)	If registration is successful, then the method assigns a rank • Marks <= 50; Rank = D • Marks > 50 and <=70; Rank = C • Marks > 70 and <=90; Rank = B • Marks > 90; Rank = A

** Create relevant data members to store scholar state with assessors and mutators.*

iv. Create a class StudentScholar which implements Scholar

Members	**Purpose**
boolean register(String name, int age, String subject, String guideName)	Registers the details of the scholar if age >= 18 years
void rank(int marks)	If registration is successful, then the

	method assigns a rank • Marks <= 60; Rank = D • Marks > 60 and <=75; Rank = C • Marks > 75 and <=90; Rank = B • Marks > 90; Rank = A

** Create relevant data members to store scholar state with assessors and mutators.*

Deliverables

- **Interface**:
  - com.junit.scholar.Scholar.java
- **Classes**:
  - com.junit.scholar.ResearchScholar
  - com.junit.scholar.StudentScholar

Hands-On Exercise 2: Jumpstart JUnit

Estimated Completion Time: 1 Hour and 30 Minutes

(xx Marks)

Objective:

- Write test programs using JUnit framework
- Write tests using *@Test* annotation
- Manage fixtures using *@Before, @After, @BeforeClass and @AfterClass* annotation
- Build test suite using *Suite.class* runner

Complete the following assignment:

i. Create two test cases – ResearchScholarTest, StudentScholarTest under com.junit.scholar package
ii. Identify the business methods and the scenarios on which it should be tested
iii. Create the test methods for each identified scenario in the respective test cases
iv. Manage the test fixtures shared across test methods by using *@Before, @After, @BeforeClass and @AfterClass* annotations
v. Create a test suite – AllScholarTests under com.junit.scholar package that executes all the *@Test* methods of ResearchScholarTest and StudentScholarTest

Deliverables

- **Classes**:
  - com.junit.scholar.AllScholarTests
  - com.junit.scholar.ResearchScholarTest
  - com.junit.scholar.StudentScholarTest

Hands-On Exercise 3: About @Test annotation

Estimated Completion Time: 60 Minutes

(xx Marks)

Objective:

- Check for exceptions thrown by tests
- Use timeouts to fail test that longer than required

Complete the following assignments:

vi. Modify the signature and behavior of the *register* methods as indicated below:

Interface/Class	Members	Purpose
Scholar	void register(…)	Registers the details of the scholar
ResearchScholar	void register(…)	If age < 28 years, then the method should throw RegistrationFailedException with the exception message "Research scholar *name* cannot be registered as the *age* is less than 28 years"
StudentScholar	void register(…)	If age < 18 years, then the method should throw RegistrationFailedException with the exception message "Student scholar *name* cannot be registered as the *age* is less than 18 years"

vii. Write new test methods to check for exceptions thrown by *register* method using *expected* parameter of *@Test* annotation and assert the exception message using *catch* block

viii. Write new test methods to fail if the *rank* method takes more than 2 seconds to complete using *timeout* parameter of *@Test* annotation

Deliverables

- **Interface**:
  - com.junit.scholar.Scholar.java
- **Classes**:
  - com.junit.scholar.ResearchScholar
  - com.junit.scholar.StudentScholar
  - com.junit.scholar.ResearchScholarTest
  - com.junit.scholar.StudentScholarTest

Hands-On Exercise 4: Hamcrest

Estimated Completion Time: 60 Minutes

(xx Marks)

Objective:

- Learn the new notation of **assertThat**
- Know the objective of hamcrest library
- Use hamcrest logical matchers
- Use hamcrest object matchers
- Use hamcrest number matchers
- Use hamcrest collection matchers

Complete the following assignment:

i. Modify the interface and classes with the following new methods

<table>
<tr><th>Interface/Class</th><th>Members</th><th>Purpose</th></tr>
<tr><td>Scholar</td><td>Object getGroup()</td><td>Returns the group of the scholar.</td></tr>
<tr><td>ResearchScholar</td><td>Object getGroup ()</td><td>Returns an Integer object based on the below rules:
<table>
<tr><th>Age</th><th>Group</th></tr>
<tr><td>28, 29</td><td>20</td></tr>
<tr><td>30, 31, 32</td><td>21</td></tr>
<tr><td>33, 34,35</td><td>22 ,23</td></tr>
<tr><td>36 and above</td><td>20, 21, 24</td></tr>
</table></td></tr>
<tr><td>StudentScholar</td><td>Object getGroup ()</td><td>Returns a Double object based on the below rules:
<table>
<tr><th>Age</th><th>Group</th></tr>
<tr><td>18, 19, 20</td><td>50.5</td></tr>
<tr><td>21, 22, 23</td><td>55.7</td></tr>
<tr><td>24 and above</td><td>60.3, 60.4</td></tr>
</table></td></tr>
</table>

ii. Write new test methods in ResearchScholarTest and StudentScholarTest for the getGroup method

   a. Should test for equality – *is, equalTo*
   b. Should test for null – *null, notNull*
   c. Should test for data type safety – *instanceOf, is*
   d. Should test for ordering – *greaterThan, lessThan, greaterThanOrEqualTo, lessThanOrEqualTo*
   e. Should test if group returned is acceptable – *isIn, notIsIn*

Deliverables

- **Interface**:
  - com.junit.scholar.Scholar.java
- **Classes**:
  - com.junit.scholar.ResearchScholar
  - com.junit.scholar.StudentScholar
  - com.junit.scholar.ResearchScholarTest
  - com.junit.scholar.StudentScholarTest

Hands-On Exercise 5: Parameterized Tests

Estimated Completion Time: 60 Minutes

(xx Marks)

Objective:

- Use *Parameterized.class* as the test runner
- Write a feeder method using *@Parameters* annotation
- Decorate generic tests with *@Test* annotation

Complete the following assignments:

i. Modify the interface Scholar with the following new methods

Members	**Purpose**
double awardPrizeAmount(int year)	Awards the prize amount based on the rank and year

ii. Modify the ResearchScholar class with the following new methods

Members	**Purpose**
double awardPrizeAmount(int year)	Awards the prize amount based on the below rules:

Rank	**Year**	**Amount**
A	2006	1000
B	2006	750
C	2006	500
D	2006	250
A	2007	1500
B	2007	1250
C	2007	1000
D	2007	750

iii. Modify the StudentScholar class with the following new methods

Members	**Purpose**

<table>
<tr>
<td>double awardPrizeAmount(int year)</td>
<td>Awards the prize amount based on the below rules:
<table>
<tr><th>Rank</th><th>Year</th><th>Amount</th></tr>
<tr><td>A</td><td>2006</td><td>Marks+75</td></tr>
<tr><td>B</td><td>2006</td><td>Marks+50</td></tr>
<tr><td>C</td><td>2006</td><td>Marks+25</td></tr>
<tr><td>D</td><td>2006</td><td>Marks+5</td></tr>
<tr><td>A</td><td>2007</td><td>Marks+175</td></tr>
<tr><td>B</td><td>2007</td><td>Marks+150</td></tr>
<tr><td>C</td><td>2007</td><td>Marks+125</td></tr>
<tr><td>D</td><td>2007</td><td>Marks +50</td></tr>
</table>
</td>
</tr>
</table>

iv. Create two parameterized test cases - ResearchScholarParameterizedTest and StudentScholarParameterizedTest under com.junit.scholar package
v. Write a feeder method to return the test data using *@Parameters* annotation
vi. Write an argument constructor to accept the parameters and set them to the class member variables
vii. Write a generic test that uses the class member variables and asserts the behavior of awardPrizeAmount method

Deliverables

- **Interfaces:**
  - com.junit.scholar.Scholar
- **Classes**:
  - com.junit.scholar.ResearchScholar
  - com.junit.scholar.StudentScholar
  - com.junit.scholar.ResearchScholarParameterizedTest
  - com.junit.scholar.StudentScholarParameterizedTest

Hands-On Exercise 6: Theories

Estimated Completion Time: 60 Minutes

(xx Marks)

Objective:

- Specify a set of data points using *@DataPoint* annotation
- Write generic test using *@Theory* annotation
- Use **assume*** methods to filter out data values in theory enabled test

Complete the following assignments:

i. Modify the interface Scholar with the following new methods

Members	Purpose

double awardBookAllowance(int year)	Awards the book allowance based on the rank and year

ii. Modify the ResearchScholar class with the following new methods

Members	**Purpose**
double awardBookAllowance(int year)	Awards the book allowance based on the below rules:

Rank	**Year**	**Amount**
A	2006	1000
B	2006	1000
C	2006	1000
D	2006	1000
A	2007	2000
B	2007	2000
C	2007	2000
D	2007	2000

iii. Modify the StudentScholar class with the following new methods

Members	**Purpose**
double awardBookAllowance(int year)	Awards the book amount based on the below rules:

Rank	**Year**	**Amount**
A	2006	750
B	2006	750
C	2006	500
D	2006	500
A	2007	2000
B	2007	1000
C	2007	1000
D	2007	1000

iv. Create two theory enabled test cases - ResearchScholarTheoryTest and StudentScholarTheoryTest under com.junit.scholar package
v. Indicate the rank (A,B,C,D) and the year (2006, 2007) as datapoints in the test case
vi. Write sufficient theories that express the general assertions across the number of valid datapoints

Deliverables

- **Interfaces:**
  - com.junit.scholar.Scholar

- **Classes**:
  - com.junit.scholar.ResearchScholar
  - com.junit.scholar.StudentScholar
  - com.junit.scholar.ResearchScholarTheoryTest
  - com.junit.scholar.StudentScholarTheoryTest