



**Cognizant | Academy**  
Passion for making a difference

# Exploring JUnit 4.x

Targeted at: Entry Level Trainees



## Session 9: Hamcrest

# About the Author

<b>Created By:</b>	B Sai Prasad, (105582)
<b>Credential Information:</b>	Sun Certified Java Programmer, Microsoft Certified Technology Specialist, PMI-certified Project Management Professional
<b>Version and Date:</b>	JUnit/PPT/1110/1.0

## Cognizant Certified Official Curriculum



# Icons Used



**Questions**



**Tools**



**Hands-on  
Exercise**



**Coding  
Standards**



**Test Your  
Understanding**



**Reference**



**Try it Out**



**A Welcome  
Break**



**Contacts**



# Session 09: Hamcrest overview

## ■ Introduction:

- » When writing tests it is sometimes difficult to get the balance right between *over specifying* the test and *not specifying* enough
- » For writing “just right” tests requires a tool that allows you to pick out precisely the aspect under test and describe the values it should have
- » In this chapter, associates would learn the role of hamcrest and how it can be used for assertions



# Session 09- Hamcrest: Objective

## ▪ **Objective:**

After completing this chapter, associates will be able to:

- » 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



# Using assertThat

```
import static org.junit.Assert.*;
...
assertEquals(expectedTax, calculatedTax);
```

- The above statement do not read well:  
“Assert that are equal expectedTax and calculatedTax”
- A new notation for assert statements is designed to make the intentions of the developer clearer and easier to read

```
import static org.hamcrest.Matchers.*;
...
assertThat(calculatedTax, is(expectedTax));
```

- The above statement reads:  
“Assert that calculated tax is [the same as] expected tax.”



# About Hamcrest

- Hamcrest was born from the popular feature of jMock called *constraints*
- Hamcrest is a framework for writing *matcher objects* allowing 'match' rules to be defined declaratively
- Hamcrest has been designed from the outset to integrate with different frameworks
- Hamcrest comes with a library of useful matchers:
  - » Core
  - » Logical
  - » Object
  - » Numbers
  - » Collections



# Hamcrest Matchers - Logical

- Assert that ... *is*

```
String color = "red";  
assertThat(color, is("red"));
```

- Assert that ... *not*

```
String color = "red";  
assertThat(color, not("blue"));
```

- Assert that ... *is one of*

```
String color = "red";  
assertThat(color, isOneOf("red", "blue"));
```





# Hamcrest Matchers - Object

- Assert that ... *is null*

```
String color = null;  
assertThat(color, is(nullValue()));
```

- Assert that ... *is not null*

```
String color = "red";  
assertThat(color, is(notNullValue()));
```

- Assert that ... *is the same instance*

```
String color1 = new String("red");  
String color2 = color1;  
assertThat(color2,  
            is(sameInstance(color1)));
```



# Hamcrest Matchers - Number

- Assert that ... *closeTo*

```
double value = 15.5;  
assertThat(value, close(16, 1.5));
```

- Assert that ... *lessThan*

```
int value = 15;  
assertThat(value, lessThan(20));
```

- Assert that ... *greaterThanOrEqualTo*

```
int value = 25;  
assertThat(value, greaterThanOrEqualTo(20));
```



# Hamcrest Matchers - Collections

- Assert that ... *has item*

```
List<String> colors = new ArrayList<String>();  
colors.add("red");  
colors.add("green");  
colors.add("yellow");  
assertThat(colors, hasItem("red"));
```

- Assert that ... *not has item less than*

```
List<Integer> ages = new ArrayList<Integer>();  
ages.add(20);  
ages.add(30);  
ages.add(40);  
assertThat(ages, not(hasItem(lessThan(18))));
```





# Demonstration

- Assert using `assertThat` method
- Use hamcrest logical matchers like *is*, *not*
- Use hamcrest object matchers like *nullValue*, *notNullValue*, *sameInstance*
- Use hamcrest number matchers like *closeTo*, *lessThan*, *lessThanOrEqualTo*
- Use hamcrest collection matchers like *hasItem*, *hasItems*, *isIn*



- Allow time for questions from participants



# Test Your Understanding



- A variable *result* should be one of the collection (1, 2, 3). What is the matcher(s) to be used to state one of multiple choices?
- Which matcher is like the Java conditional && , || operator?
- How to test String equality ignoring case?
- How to test object's type is compatible?



# Hamcrest -Session 9: Summary

- Hamcrest was born from the popular feature of jMock called *constraints*
- Hamcrest comes with a library of useful matchers:
  1. Core
  2. Logical
  3. Object
  4. Numbers
  5. Collections
- Core matchers – *anything, is*
- Logical matchers – *not, allOf, anyOf*
- Object matchers – *equalTo, instanceOf, notNullValue, nullValue*
- Number – *closeTo, greaterThan, greaterThanOrEqualTo*
- Collections matchers – *hasItem, hasItems*



# Hamcrest Session 9: Source



- Books:

- » JUnit Recipes: Practical Methods for Programmer Testing by *J. B. Rainsberger, Scott Stirling*
- » JUnit in Action by *Vincent Massol, Ted Husted*

- Web:

- » Wiki: <http://en.wikipedia.org/wiki/JUnit>
- » JUnit: <http://www.junit.org/>
- » Hamcrest:  
<http://code.google.com/p/hamcrest/wiki/Tutorial>
- » Hamcrest API: <http://www.jmock.org/javadoc/2.5.1/>

**Disclaimer:** Parts of the content of this course is based on the materials available from the Web sites and books listed above. The materials that can be accessed from linked sites are not maintained by Cognizant Academy and we are not responsible for the contents thereof. All trademarks, service marks, and trade names in this course are the marks of the respective owner(s).





**Cognizant** | **Academy**  
Passion for making a difference

You have completed  
the Session 9  
Hamcrest

