Matching Strings

```
assertThat(s, is(@"FooBar"));
assertThat(s, startsWith(@"Foo"));
assertThat(s, endsWith(@"Bar"));
assertThat(s, containsString(@"oo"));
assertThat(s,
equalToIgnoringCase(@"foobar"));
assertThat(@" X \n Y \t\t Z \n",
equalToIgnoringWhiteSpace(@"X Y Z"));
```

Given NSString *s = @"FooBar";
is â€“ match the complete string
startsWith â€“ match the beginning of a string
endsWith â€“ match the end of a string
containsString â€“ match part of the string
equalTo â€“ match the complete string
equalToIgnoringCase â€“ match the complete string but ignore case
equalToIgnoringWhiteSpace â€“ match the complete string but ignore extra whitespace (new line, tab, or double spaces)

Matching Dictionaries

```
NSDictionary *d = [NSDictionary
dictionaryWithObjectsAndKeys: @"valA",
@"keyA", @"valB", @"keyB", @"valC",
@"keyC", nil];
assertThat(d, hasCountOf(3));
assertThat(d, isNot(empty()));
assertThat(d, hasKey(@"keyA"));
assertThat(d, isNot(hasKey(@"keyX")));
assertThat(d, hasValue(@"valA"));
assertThat(d, hasEntry(@"keyA", @"valA"));
assertThat(d, hasEntries(@"keyA", @"valA",
@"keyC", @"valC", nil));
```

hasKey â€“ match a key
hasValue â€“ match a value
hasEntry â€“ match a key-value pair
hasEntries â€“ match a list of k-v pairs

Matcher Error Messages

```
NSString *s = @"bar";
assertThat(s, is(@"foo")); //Expected "foo", but
was "bar"
assertThat(s, describedAs(@"doh! this should
be 'foo'", equalTo(@"foo"), nil)); //Expected
doh! this should be 'foo', but was "bar"
assertThat(s, describedAs(@"doh! this should
be foo, %0, %1", equalTo(@"foo"), @"baz",
[NSNumber numberWithInt:42], nil));
//Expected doh! this should be foo, "baz",
<42>, but was "bar"
```

Combining Matchers

```
assertThat(s, allOf(startsWith(@"Foo"),
endsWith(@"Bar"), nil));
assertThat(s, anyOf(startsWith(@"Foo"),
startsWith(@"Bar"), nil));
assertThat(s, anyOf(endsWith(@"Foo"),
endsWith(@"Bar"), nil));
```

allOf â€“ AND together all matchers
anyOf â€“ OR together all matches
The list of matchers must be nil terminated.

Matching Nil

```
assertThat(s, notNilValue());
assertThat(o, nilValue());
```

Given NSObject *o = nil;
nilValue() â€“ stands in for nil
notNilValue() â€“ stands in for !nil

Matching Numbers

```
assertThatInt(5, equalToInt(5));
assertThatFloat(3.14, equalToFloat(3.14f));
assertThatBool( false, equalToBool(NO) );
NSNumber *f = [NSNumber
numberWithFloat:3.14f];
assertThat(f, closeTo(3.0f, 0.25f));
assertThat(f, lessThan([NSNumber
numberWithInt:4]));
```

Matching Numbers (cont)

```
assertThat(f, greaterThan([NSNumber
numberWithInt:3]));
```

closeTo â€“ match a number with a target number plus or minus a delta (both params are double)
lessThan â€“ match a number less than the given number (param is NSNumber), also lessThanOrEqualTo

hasProperty Matcher

```
Person *p = [Person
personWithFirstName:@"Joe"
andLastname:@"Doe"];
assertThat(p, hasProperty(@"firstName",
@"Joe"));
NSArray *a = [NSArray arrayWithObjects:
[Person personWithFirstName:@"Joe"
andLastname:@"Doe"], [Person
personWithFirstName:@"Joe"
andLastname:@"Smith"], [Person
personWithFirstName:@"Jane"
andLastname:@"Allen"], nil];
assertThat(a, contains(
hasProperty(@"firstName", @"Joe"),
hasProperty(@"firstName", @"Joe"),
hasProperty(@"firstName", @"Jane"), nil));
```

Any method, without arguments, that returns an object

Invert Matcher

```
assertThat(s, isNot(@"foo"));
assertThat(s, isNot(endsWith(@"Baz")));
assertThat(s, isNot(allOf(startsWith(@"Baz"),
endsWith(@"Baz"), nil)));
assertThat(s, isNot(anyOf(startsWith(@"Baz"),
startsWith(@"Baz"), nil)));
```

isNot â€“ negate the matcher

Matching Classes

```
assertThat(s, instanceOf([NSString class]));
```

instanceOf â€“ match the class

Matching Arrays

```
NSArray *a = [NSArray array];
assertThat(a, is(empty()));
assertThat(a, hasCountOf(0));
NSArray *a = [NSArray arrayWithObjects:@"a", @"b", @"c", nil];
assertThat(a, hasItem(@"a"));
assertThat(a, isNot(hasItem(@"X")));
assertThat(a, hasItem(equalToIgnoringCase(@"A")));
NSArray *a = [NSArray arrayWithObjects: [NSNumber
numberWithInt:2], [NSNumber numberWithInt:3], [NSNumber
numberWithInt:5], nil];
assertThat(a, hasItem(equalToInt(2)));
assertThat(a, isNot(hasItem(equalToInt(13))));
assertThat(a, contains(equalToInt(2), equalToInt(3), equalToInt(5), nil));
NSArray *a = [NSArray arrayWithObjects:@"a", @"b", @"c", nil];
assertThat(a, hasItems(@"b", @"a", nil));
assertThat(a, contains(@"a", @"b", @"c", nil));
assertThat(a, containsInAnyOrder(@"c", @"b", @"a", nil));
assertThat([a componentsJoinedByString:@","], is(@"a,b,c"));
```

hasItem â€“ match if given item appears in the list
hasItems â€“ match if all given items appear in the list (in any order)
contains â€“ match exactly the entire array
containsInAnyOrder â€“ match entire array, but in any order
hasCountOf â€“ match the size of the array
empty â€“ match an empty array