# Advanced Programming (I00032)
# A DSL for testing with Matchers

Assignment 10

In this assignment you will make a DSL for testing with matchers, similar to the matchers in JUnit. There is much information about these matchers on the internet. Some useful sources are:

1. `https://code.google.com/p/hamcrest/wiki/Tutorial`
2. `http://junit.sourceforge.net/javadoc/org/junit/Assert.html`
3. `http://junit.org/apidocs/org/hamcrest/CoreMatchers.html`

It is not our intention to construct a complete implementation of the JUnit Matcher DSL, nor to follow the JUnit DSL as close as possible. We just want a similar DSL.

## Basic Matchers

Basic matchers enable assertions like:

```
a1 = AssertThat "(2*2) (Is (EqualTo (2+2)))" (2*2) (Is (EqualTo (2+2)))
a2 = AssertThat "(3*3) (EqualTo (3+3))"      (3*3) (EqualTo (3+3))              // fail
a3 = AssertThat "(length [0..3]) is not 4"   (length [0..3]) (Not (EqualTo 4))  // fail
```

These assertions can be evaluated by an expression like:

```
Start = test (a1 * a2 * a3)
```

The intended result is a report like:

```
passes = 1, fails = 2
1: AssertThat (3*3) (EqualTo (3+3))
2: AssertThat (length [0..3]) (Not (EqualTo 4))
```

The basic matcher DSL has (at least) the following constructs:

1. `Is` *matcher* this is just syntactic sugar. It has no semantics.
2. `EqualTo` *value* test value equality using $==$.
3. `LessThan` *value* test value inequality using `<`.
4. `Not` *matcher* matches if the wrapped matcher doesn't match and vice versa.
5. `Either` *matcher matcher* succeeds when any of these matchers holds.

## 1 Deep Embedding of Matchers

Define a deep embedded DSL for matchers and an implementation of `AssertThat` that is able to evaluate the given assertions by producing a report similar to the one shown above.

It is not necessary to produce output before all asserts are evaluated. Notice that it is not too difficult to print the matcher used in the assertion.

## 2 Shallow Embedding of Matchers

Make a shallow embedding of matchers that is able to evaluate the same assertions and produces a similar report.

Notice that it is harder, but not impossible, to show the matcher used in the assertion.

## 3 List Matchers

It is convenient to add special purpose matchers for data types that are frequently used. For lists we define at least:

6. `Contains` *value* check whether the given value is a member of the list.

This allows assertions like:

```
a4 = AssertThat "[0..3] (Contains 2)" [0..3] (Contains 2)
a5 = AssertThat "[0..3] (Contains 7)" [0..3] (Contains 7)        // fail
a6 = AssertThat "[0..3] (Either (EqualTo [1]) (Contains 7))"
       [0..3] (Either (EqualTo [1]) (Contains 7))                // fail
```

Extend one of your DSL implementations with this construct for list assertions.

## 4 String Matchers

In the same spirit we can add tailor-made matchers for data types like `String`:

7. `ContainsString` *substring* Check whether this substring occurs in the given string.

This allows assertions like:

```
a7 = AssertThat  "'hello world' (ContainsString 'hello')"
       "hello world" (ContainsString "hello")
a8 = AssertThat  "'hello world' (ContainsString 'world')"
       "hello world" (ContainsString "world")
a9 = AssertThat  "Red, yellow and blue"
       "Who is afraid of red, yellow and blue" (ContainsString "Red") // fail
```

Extend one of your DSL implementations with this construct for list assertions.

## 5 [Optional] Universal Quantification

For plain unit tests we were able to increase the expressional power by turing Boolean expressions into universal quantified variants. This allows us to state more general properties like $\forall x, y \in \mathsf{Int} \,.\, x + y == y + x$ instead of $3 + 4 == 4 + 3$.

Implement a similar extension for your matcher DSL. It is fine to stop the evaluation of the current assertion as soon as it is falsified. It is very helpful to show the values that falsify such an assertion.

## Deadline

The deadline for this assignment is November 30, 13:30h. Include the report of evaluating assertion `a1 .. a9` as a comment in your file.