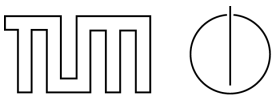
Tutorial for Software Quality Management
Summer Term 2010
Dr. S. Wagner
L. Heinemann, S. Islam
18.05.2010



Technische Universität München
Fakultät für Informatik

Tutorial Sheet 4: Testing

A computer with internet access is required for this tutorial sheet.

Assignment 1: Black-box testing

Consider the following (simplified) rules for valid e-mail addresses.

- An e-mail address consists of two non-empty parts separated by an @-sign.
- Both parts may contain only the characters a-z, A-Z, 0-9.
- A dot is allowed, except as the first or last character of a part and there must not be any 2 consecutive dots.

Define test cases for a function that validates e-mail addresses according to these rules. It takes a string with an e-mail address as an input and returns a boolean result that indicates whether the e-mail address is valid. List pairs of inputs and expected outputs.

Example: john.doe@example.com → true

Assignment 2: White-box testing

The following function computes the solution for quadratic equations of the form $a \cdot x^2 + b \cdot x + c = 0$. Design test cases by listing pairs of input values for the three parameters and expected output values. Try to cover all possible execution paths.

```
1  /**
2   * Computes the real solutions of the quadratic equation
3   * a*x*x + b*x + c = 0. The coefficient a must not be 0,
4   * else an {@link IllegalArgumentException} is thrown;
5   *
6   * @param a the a coefficient, must not be 0
7   * @param b the b coefficient
8   * @param c the c coefficient
9   * @return the solutions for the equation. There can be
10  *   either 0, 1 or 2 solutions.
11  */
12 public static double[] computeRoots(double a, double b, double c) {
```

```
13    if (a == 0) {
14      throw new IllegalArgumentException("a must not be 0");
15    }
16    double d = b * b - 4 * a * c;
17    if (d == 0) {
18      return new double[] { -b / 2 * a };
19    } else if (d > 0) {
20      return new double[] { (-b + Math.sqrt(d)) / (2 * a),
21                            (-b - Math.sqrt(d)) / (2 * a) };
22    } else {
23      return new double[] {};
24    }
25  }
```

Assignment 3: Unit Testing with JUnit

- Download and install the Eclipse IDE on your computer from the following URL if it is not already installed: http://www.eclipse.org/downloads/

  Choose the package *Eclipse Classic 3.5.2*.

- Create a new Java project and copy the Java file `Range` from the following URL into your project: http://www4.in.tum.de/lehre/vorlesungen/sqm/ss10/sheets/Range.java. This Java file contains the stub for an interval class over integers. In contains method stubs including their Javadoc documentation.

- Create a new JUnit Test Case `RangeTest` in the project. Write test cases for the Range class. Try to cover all functionality that is specified in the Javadoc and also think of corner cases. *Do not yet write the implementation of the Range class.*

- Implement the Java class so that all your test cases pass.

- With the implementation of the *Range* class at hand, reconsider your test cases. Try to find cases that may not be covered by your tests yet and complete your tests accordingly.

- Install the coverage tool *Eclemma* by performing the following steps: From your Eclipse menu select Help → Install New Software. In the Install dialog enter http://update.eclemma.org/ at the Work with field. Check the latest EclEmma version and press Next. Follow the steps in the installation wizard.

- After the restart of Eclipse there is a new launch button in your tool bar. Start the RangeTest test case with the new coverage launch button. Inspect the line coverage of the `Range` class in the *Coverage* view that opens automatically. If you encounter uncovered implementation code in the Range class, write additional test cases to cover the code.