

Table of Contents

Classes	1
Interfaces	3
Unit Tests	5

All of our Java classes use a standardized style. In addition, we use [Checkstyle](#) and the [Checkstyle Eclipse Plugin](#), with predefined rules to avoid wasting time during development.

The Checkstyle rules are saved in the **JVx Repository** under the following file name:

```
<jvx>/trunk/java/library/checkstyle_opensource.xml
```

Classes

We use the following style for classes:

[ClassTemplate.java](#)

```
/*
 * Copyright 2018 SIB Visions GmbH
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you
 may not
 * use this file except in compliance with the License. You may obtain
 a copy of
 * the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See
 the
 * License for the specific language governing permissions and
 limitations under
 * the License.
 *
 * History
 *
 * dd.MM.yyyy - [XX] - creation
 */
package com.sibvisions.foo;

/**
 * I do that and that ....
 *
 * @author First Last
 */
public class Bar
{
    //~~~~~
    // Class members
    //~~~~~
}
```

```
/** The foo type. */
public static final int TYPE_FOO = 1;

/** The value of foo bar. */
private Object oValue;

//~~~~~
// Initialization
//~~~~~

/**
 * Creates a new instance of <code>Bar</code>.
 */
public Bar()
{
}

//~~~~~
// Abstract methods implementation
//~~~~~

//~~~~~
// Interface implementation
//~~~~~

//~~~~~
// Abstract methods
//~~~~~

//~~~~~
// Overwritten methods
//~~~~~

/**
 * {@inheritDoc}
 */
@Override
public String toString()
{
    return "Foo";
}

//~~~~~
// User-defined methods
//~~~~~

/**
 * Sets the value.
 *
```

```

    * @param pValue the value.
    */
    public void setValue(Object pValue)
    {
        this.oValue = pValue;
    }

    /**
     * Gets the value.
     *
     * @return the value.
     */
    public Object getValue()
    {
        return oValue;
    }

    //*****
    // Subclass definition
    //*****

} // Bar

```

The following rules are defined by this template:

- Variable declaration at the beginning (first constants, then variables)
 - Then constructors and initialization methods
 - Then the implementation of abstract methods
 - Then the implementation of interface methods
 - Then the definition of abstract methods
 - Then all overwritten methods (marked with @Override)
 - Then all methods of the class
 - Sub/Inner classes at the end
- Each parameter of a method is marked using the prefix "p"
 - A prefix is also used for instance variables, e.g.:

```
String sValue = "bar";
```

- Important changes are documented in the header, including time stamp and author
- Documentation for the class declaration, ALL methods and instance variables and constants

Interfaces

We use the following style for interfaces:

[InterfaceTemplate.java](#)

```
/*
```

```
* Copyright 2018 SIB Visions GmbH
*
* Licensed under the Apache License, Version 2.0 (the "License"); you
may not
* use this file except in compliance with the License. You may obtain
a copy of
* the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT
* WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See
the
* License for the specific language governing permissions and
limitations under
* the License.
*
*
* History
*
* dd.MM.yyyy - [XX] - creation
*/
package com.sibvisions.foo;

/**
 * I do that and that ....
 *
 * @author First Last
 */
public interface IBar
{
    //~~~~~
    // Constants
    //~~~~~

    /** The foo type. */
    public static final int TYPE_FOO = 1;

    //~~~~~
    // Method definitions
    //~~~~~

    /**
     * Sets the value.
     *
     * @param pValue value.
     */
    public void setValue(Object pValue);
}
```

```

//*****
// Subinterface definition
//*****
} // IBar

```

The following rules are defined by this template:

- Constants are defined at the beginning
- Then interface methods
- Sub/Inner interfaces at the end
- Each interface starts with "I"
- Important changes are documented in the header, including time stamp and author
- Documentation for the interface declaration, ALL methods and constants

Unit Tests

The use of unit tests ensures that basic functionality works as expected. A unit test can never test the entire functionality in all conceivable configurations, but without it the required quality standards cannot be met. We therefore require a working set of unit tests.

Unit tests are saved separately from the core source code:

```

<jvx>/trunk/java/library/src/com/sibvisions/foo
<jvx>/trunk/java/library/test/com/sibvisions/foo

```

[JUnit](#) is used as the testing framework.

We use the following style for unit tests:

[TestTemplate.java](#)

```

/*
 * Copyright 2018 SIB Visions GmbH
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you
 may not
 * use this file except in compliance with the License. You may obtain
 a copy of
 * the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See
 the

```

```
* License for the specific language governing permissions and
limitations under
* the License.
*
*
* History
*
* dd.MM.yyyy - [XX] - creation
*/
package com.sibvisions.foo;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Assert;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

/**
 * Tests the functionality of ...
 *
 * @author First Last
 */
public class TestBar
{
    //~~~~~
    // Class members
    //~~~~~

    //~~~~~
    // Initialization
    //~~~~~

    /**
     * Initializes the unit test.
     *
     * @throws Exception if initialization fails
     */
    @BeforeClass
    public static void beforeClass() throws Exception
    {
    }

    /**
     * Resets the unit test.
     *
     * @throws Exception if reset fails
     */
    @AfterClass
    public static void afterClass() throws Exception
    {
    }
}
```

```

}

/**
 * Sets values before each test.
 *
 * @throws Exception if set values fails
 */
@Before
public void beforeTest() throws Exception
{
}

/**
 * Reset values after each test.
 *
 * @throws Exception if reset values fails
 */
@After
public void afterTest() throws Exception
{
}

// ~~~~~
// Abstract methods implementation
// ~~~~~

// ~~~~~
// Interface implementation
// ~~~~~

// ~~~~~
// Overwritten methods
// ~~~~~

// ~~~~~
// User-defined methods
// ~~~~~

// ~~~~~
// Test methods
// ~~~~~

/**
 * Tests the ... method.
 */
@Test
public void testGet()
{
}

// *****

```

```
// Subclass definition
//*****
} // TestBar
```

The following rules are defined by this template:

- Variable declaration at the beginning (first constants, then variables)
- Then methods for the test initialization
- Then the implementation of abstract methods
- Then the implementation of interface methods
- Then all overwritten methods (marked with @Override)
- Then all methods of the class
- Then all test methods (marked with @Test)
- Sub/Inner classes at the end

- Each test class starts with "Test"
- Each test method starts with "test"
- Important changes are documented in the header, including time stamp and author
- Documentation for the class declaration, ALL methods and instance variables and constants

From:
<http://doc.sibvisions.com/> - **Documentation**

Permanent link:
http://doc.sibvisions.com/jvx/join/style_java



Last update: **2018/02/06 09:20**