

Test Design

Softwaretechnik & Entwicklungswerkzeuge, DHBW Karlsruhe, WS-2009/10
Collin Rogowski

Agenda

- Testen von GUIs
- Mocks
- Grenzfälle
- Beispiel

GUI Tests

```
public class InputCheckGUI extends JFrame {

    private final class ButtonClickedListener implements ActionListener {
        public void actionPerformed(ActionEvent event) {
            String text = textfield.getText();
            try {
                double number = Double.parseDouble(text);
                saveNumberToDatabase(number);
            } catch (NumberFormatException e) {
                showErrorDialog("The input is not a number");
            }
        }
    }

    private final JTextField textfield;
    private final JButton button;

    public InputCheckGUI() {
        textfield = new JTextField();
        button = new JButton("Save");

        createLayout();

        button.addActionListener(new ButtonClickedListener());
    }
}
```

GUI Tests

```
public class InputCheckGUI extends JFrame {  
  
    private final class ButtonClickedListener implements ActionListener {  
        public void actionPerformed(ActionEvent event) {  
            String text = textfield.getText();  
            InputChecker checker = new InputChecker(text);  
            if (checker.checkForNumber()) {  
                saveNumberToDatabase(checker.getResult());  
            } else {  
                showErrorDialog("The input is not a number");  
            }  
        }  
    }  
}  
  
private final JTextField textfield;  
private final JButton button;  
  
public InputCheckGUI() {  
    textfield = new JTextField();  
    button = new JButton("Save");  
  
    createLayout();  
  
    button.addActionListener(new ButtonClickedListener());  
}
```

GUI Tests

```
public class InputCheckerTest {  
    @Test  
    public void checkForNumber() {  
        InputChecker checker = new InputChecker("42");  
        assertTrue(checker.checkForNumber());  
        assertEquals(42.0, checker.getResult(), 0.0);  
  
        checker = new InputChecker("hallo");  
        assertFalse(checker.checkForNumber());  
  
        checker = new InputChecker("3E2");  
        assertTrue(checker.checkForNumber());  
        assertEquals(300.0, checker.getResult(), 0.0);  
    }  
}
```

GUI Tests

- minimaler View
- jegliche Logik auslagern
- Vorgehen: ohne GUI beginnen

Mocks

```
public class ServerAvailability {  
    public boolean serversAvailable(List<String> servers) {  
        for (String server : servers) {  
            try {  
                boolean isAvailable = InetAddress.getByName(server).isReachable(3000);  
                if (!isAvailable) {  
                    return false;  
                }  
            } catch (IOException e) {  
                return false;  
            }  
        }  
        return true;  
    }  
}
```

Mocks

```
public class ServerAvailability {  
    private final AvailabilityChecker checker;  
  
    public ServerAvailability(AvailabilityChecker checker) {  
        this.checker = checker;  
    }  
  
    public boolean serversAvailable(List<String> servers) {  
        for (String server : servers) {  
            boolean isAvailable = checker.check(server);  
            if (!isAvailable) {  
                return false;  
            }  
        }  
        return true;  
    }  
}
```

Mocks

```
public interface AvailabilityChecker {
    public boolean check(String server);
}

public class AvailabilityCheckerUsingPing implements AvailabilityChecker {

    private final int timeout;

    public AvailabilityCheckerUsingPing(int timeout) {
        this.timeout = timeout;
    }

    public boolean check(String server) {
        try {
            return InetAddress.getByNames(server).isReachable(timeout);
        } catch (IOException e) {
            return false;
        }
    }
}
```

Mocks

```
public class ServerAvailability {  
  
    private final AvailabilityChecker checker;  
  
    public ServerAvailability(AvailabilityChecker checker) {  
        this.checker = checker;  
    }  
  
    public boolean serversAvailable(List<String> servers) {  
        for (String server : servers) {  
            boolean isAvailable = checker.check(server);  
            if (!isAvailable) {  
                return false;  
            }  
        }  
        return true;  
    }  
  
    public static void main(String[] argv) {  
        AvailabilityChecker checker = new AvailabilityCheckerUsingPing(3000);  
        ServerAvailability availability = new ServerAvailability(checker);  
        boolean available = availability.serversAvailable(Arrays.asList(argv));  
  
        System.out.println(available);  
    }  
}
```

Mocks

```
public class ServerAvailabilityTest {  
    @Test  
    public void serversAvailable() {  
        ServerAvailability availability = new ServerAvailability(checker);  
        boolean available = availability.serversAvailable(servers);  
        assertTrue(available);  
    }  
}
```

Mocks

```
public class MockAvailabilityChecker implements AvailabilityChecker {  
    public boolean check(String server) {  
        if (server.equals("up")) {  
            return true;  
        } else if (server.equals("down")) {  
            return false;  
        } else {  
            throw new IllegalArgumentException("don't know what to do with " + server);  
        }  
    }  
}
```

Mocks

```
public class ServerAvailabilityTest {
    @Test
    public void serversAvailable() {
        ServerAvailability availability = new ServerAvailability(new MockAvailabilityChecker());

        List<String> servers = Arrays.asList(new String[] { "up", "up", "up" });
        boolean available = availability.serversAvailable(servers);
        assertTrue(available);

        servers = Arrays.asList(new String[] { "up", "up", "down" });
        available = availability.serversAvailable(servers);
        assertFalse(available);
    }
}
```

Mocks

- Funktionalität kapseln
- Zugriff per Interface
- Dependency Injection
- Steuerbaren Mock unterscheiden

Grenzfälle

```
@Test
public void testEquilateral() {
    assertEquals(1, new Triangle(2, 2, 2).check());
}

@Test
public void testIsosceles() {
    assertEquals(2, new Triangle(1, 2, 2).check());
}

@Test
public void testScalene() {
    assertEquals(3, new Triangle(2, 3, 4).check());
}

@Test
public void testIrrational() {
    try {
        new Triangle(1, 2, 3).check();
        fail();
    } catch (IllegalArgumentException e) {
    }
}

@Test
public void testNegative() {
    try {
        new Triangle(-1, 1, 2).check();
        fail();
    } catch (IllegalArgumentException e) {
    }
}
```

Grenzfälle

- MIN_VALUE, MAX_VALUE
- NaN, ∞ , $-\infty$
- null
- Methoden zweimal aufrufen (Cleanup)

Beispiel

```
public interface IntSet {  
    public void add(int i) throws DuplicateException;  
  
    @SuppressWarnings("serial")  
    class DuplicateException extends Exception {  
        public DuplicateException(String msg) {  
            super(msg);  
        }  
    }  
}
```

Beispiel

```
public class IntSetTest {
    private IntSet set;

    @Before
    public void createSet() {
        set = new IntSetImpl();
    }

    @Test
    public void addTwo() {
        try {
            set.add(1);
            set.add(2);
        } catch (IntSet.DuplicateException e) {
            fail();
        }
    }

    @Test
    public void addDuplicate() {
        try {
            set.add(1);
            set.add(1);
            fail();
        } catch (IntSet.DuplicateException e) {
        }
    }
}
```

Beispiel

```
public class IntSetImpl implements IntSet {
    private final List<Integer> list = new ArrayList<Integer>();

    public void add(int i) throws DuplicateException {
        list.add(i);
        if (listHasDuplicates()) {
            throw new DuplicateException(i + " is already in here");
        }
    }

    private boolean listHasDuplicates() {
        Collections.sort(list);
        for (int i = 0; i < list.size() - 1; i++) {
            if (list.get(i) == list.get(i + 1)) {
                return true;
            }
        }
        return false;
    }
}
```

Beispiel

```
@Test
public void addDuplicate() {
    try {
        set.add(1);
        set.add(1);
        fail();
    } catch (IntSet.DuplicateException e) {
    }
    set.remove(1);
    assertFalse(set.contains(1));
}
```