

SOFTENG 254:
Quality Assurance

Lecture 2a: Overview of Testing

Paramvir Singh
School of Computer Science

Potential Assessment Question

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

Consider the following scenario:

While filling a form on a website, Bob notices that an image on the page overlaps one of the text boxes, making it hard to see what he is typing into that box.

Which of the following terms best describes what Bob has experienced?

- (a) a bug
- (b) a fault
- (c) an error
- (d) a failure

Justify your answer.

Agenda

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- Admin
- Assignment 1
- What is testing?
- Describing tests
- Defining good tests

Assignment 1

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- Given: 11 implementations, 10 are known to have faults. 1 is hoped to not have any faults.
 - Each implementation is *supposed* to provide a method that will format a piece of text “flush left”
 - Write JUnit tests to detect faults in implementations (provided as .jar files) and not detect faults in good implementation
 - Describe purpose of each test case.
 - Limit on size of test suite — at most **200** calls to one of the `assertX` methods
 - Marking will be (mostly) **automatic**. Failure to meet submissions requirements means no marks.
 - Authoritative details on Canvas
 - Be very **careful** while discussing the assignment on Piazza.
 - Worth 7.5%, Due 17:00hr Friday 14 August (Week 3)
- adb.auckland.ac.nz

Assignment 1

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- Implementation under test is the method:

```
package text;  
class Formatter {  
    public static List<String> flushLeftText(String text,  
                                              int linewidth)  
  
    ...  
}
```

- Format text into lines

Input: text=Program testing can be used to show the presence of
bugs, but never to show their absence.

linewidth=20

Output:

12345678901234567890

Program testing can
be used to show the
presence of
bugs, but never to
show their absence.

- More (probably correct) details in JavaDoc page provided.

Previously in SOFTENG 254

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- Test case — specifies the *pretest state* of the IUT, the *inputs*, and the *expected state or behaviour*

Developing Test Suites

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- how do we describe tests?
- what's the best way to run the tests?
- when is a test a “good” one?
- how do we come up with tests?
- when do we have enough tests?

Developing Test Suites

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- how do we describe tests?
- what's the best way to run the tests?
- when is a test a “good” one?
- how do we come up with tests?
- when do we have enough tests?

Describing tests

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- What is the Implementation Under Test (IUT)?
- What is the pretest state?
- What are the inputs? (test case values and other inputs)
- What is the expected state?

Test documentation

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- if testing is to be systematic, it must be *planned* — **test plan**
- a test plan might include:
 - how the testing is to be done
 - who does it
 - what the schedule is
 - what level of quality of testing is to be performed
 - what resources are required
- each individual test case has to be described
- how the test results are to be used
- there are standards for this kind of thing. ..

IEEE 829-1998

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- IEEE Standard for Software Test Documentation (1998)
- Test Plan
- Test Design Specification
- Test Case Specification
- Test Procedure Specification
- Test Item Transmittal Report
- Test Log
- Test **Incident** Report
- Test Summary Report

Test Case Specification

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- Purpose — To define a test case identified by a test design specification.
- Outline — A test case specification shall have the following structure:
 - a) Test case specification identifier;
 - b) Test items;
 - c) Input specifications;
 - d) Output specifications;
 - e) Environmental needs;
 - f) Special procedural requirements;
 - g) Intercase dependencies.

Developing Tests

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- how do we describe tests?
- **what's the best way to run the tests?**
- when is a test a “good” one?
- how do we come up with tests?
- when do we have enough tests?

Test Procedure Specification

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- a) Test procedure specification identifier
- b) Purpose
- c) Special requirements
- d) *Procedure steps*

Procedure steps

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

Include as applicable:

- Log
- Set up
- Start
- Proceed
- Measure
- Shut down
- Restart
- Stop
- Wrap up
- Contingencies

Test Scripts

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- **[Test Scripts](#)**
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- detailed sequence of steps needed to perform the test
- Example: Authenticated Edit of Wiki
 1. Navigate to test page *url* **Set up**
 2. Check that **Edit** tab is visible at top of page **Start**
 3. Check that **Create an account or log in** link is visible at top right corner *confirms that user is not already authenticated*
 4. Click **Edit** tab **Proceed**
 5. Verify that **Login required to edit** page shows
 6. Click **login** link
 7. (etc)

Running tests

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- Manually running test scripts is tedious, error prone, and expensive
⇒ Automate the process as much as possible
- XUnit family testing frameworks (SUnit, JUnit, NUnit, . . .) provide frameworks for specifying tests, executing tests, reporting the results
- Many (many!) other tools and frameworks exist to support different testing activities

JUnit

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- Developed by Erich Gamma and Kent Beck, based on SUnit, developed by Kent Beck for Smalltalk
- Two styles “3.8” and “4” (and above)
- JUnit 3.8 (and earlier) uses **reflection** to identify test cases to be executed

```
import junit.framework.TestCase;
public class TestNumZeros extends TestCase {
    public void testNoZeros() {
        int[] input = { 1, 93, 2, 5, -17 };
        IntList list = new IntList(input);
        assertEquals(0, list.numZeros());
    }
}
```

- JUnit 4 uses **annotations**

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class TestNumZeros {
    @Test
    public void noZeros() {
        int[] input = { 1, 93, 2, 5, -17 };
        IntList list = new IntList(input);
        assertEquals(0, list.numZeros());
    }
}
```

- Support for pre-test (“set up”), post-test (“tear down”) management, and more

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- **[Automation](#)**
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

Automation

Suppose this is our test case:

```
import junit.framework.TestCase;
public class TestNumZeros extends TestCase {
    public void testNoZeros() {
        int[] input = {1, 93, 2, 5, -17 };
        IntList list = new IntList(input);
        assertEquals(0, list.numZeros());
    }
}
```

IUT?

Pre-test state?

Inputs?

Expected results?

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- **[Automation](#)**
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

Automation

Suppose this is our test case:

```
import junit.framework.TestCase;
public class TestNumZeros extends TestCase {
    public void testNoZeros() {
        int[] input = {1, 93, 2, 5, -17 };
        IntList list = new IntList(input);
        assertEquals(0, list.numZeros());
    }
}
```

IUT? IntList#numZeros()

Pre-test state? new IntList(input)

Inputs? {1, 93, 2, 5, -17}

Expected results? 0

Automation

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- **[Automation](#)**
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

Suppose this is our test case:

```
import junit.framework.TestCase;
public class TestNumZeros extends TestCase {
    public void testNoZeros() {
        int[] input = {1, 93, 2, 5, -17 };
        IntList list = new IntList(input);
        assertEquals(0, list.numZeros());
    }
}
```

IUT? IntList#numZeros()

Pre-test state? new IntList(input)

Inputs? {1, 93, 2, 5, -17} **probably the clearest in intent**

Expected results? 0

Alternative

IUT? IntList

Pre-test state? new IntList(input)

Inputs? list.numZeros() *method to invoke is input*

Expected results? 0

Running tests

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

```
package se254;
import junit.framework.TestCase;
public class TestIntList extends TestCase {
    public void testNoZeros() {
        int[] input = {1, 93, 2, 5, -17 };
        IntList list = new IntList(input);
        assertEquals(0, list.numZeros());
    }
    // An easy way to run tests from commandline (3.8)
    public static void main(String[] args) {
        junit.textui.TestRunner.run(IntList.class);
    }
}
```

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

Running tests

```

/home/ewan/
  se254stuff/
    lib/
      junit.jar
    lec03/
      main/
        se254/
          IntList.java IntList.class
      test/
        se254/
          TestIntList.java TestIntList.class

```

- The JUnit jar file is in the directory `/home/ewan/se254stuff/lib`. Its *absolute path* is `/home/ewan/se254stuff/lib/junit.jar`
- The source and compiled (bytecode) for the IUT `se254.IntList` is in `/home/ewan/se254stuff/lec03/main/se254`.
- The source and compiled (bytecode) for the test class `se254.TestIntList` is in `/home/ewan/se254stuff/lec03/test/se254`.
- The package that the IUT and test class is in is `se254`
- The top of the package hierarchy can be found at both `/home/ewan/se254stuff/lec03/main` and `/home/ewan/se254stuff/lec03/test`

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

Running tests

```
/home/ewan/  
  se254stuff/  
    lib/  
      junit.jar  
    lec03/  
      main/  
        se254/  
          IntList.java IntList.class  
        test/  
          se254/  
            TestIntList.java TestIntList.class
```

- If the *current working directory* (cwd) is /home/ewan/se254stuff then:
 - The *relative* path to junit.jar is lib/junit.jar
 - The *relative* paths to the tops of the package hierarchy are lec03/main and lec03/test

Running tests

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- If the *current working directory* is `/home/ewan/se254stuff` then:
 - If the Java runtime (`java`) is asked to find the class `se254.TestIntList`, and `lib/junit.jar`, `lec03/main`, `lec03/test` is on the **classpath** (in that order) then the Java runtime will:
 1. Look for the file `se254/TestIntList.class` in `lib/junit.jar` (it knows this is a Jar file and knows how to look inside them).
 2. If it does not find it, it will then look for the file with the path `se254/TestIntList.class` *relative to* `lec03/main`.
 3. If it does not find it, it will then look for the file with the path `se254/TestIntList.class` *relative to* `lec03/test`.
 4. If it does not find it, it will throw a `ClassNotFoundException`

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

Running tests

```

/home/ewan/
  se254stuff/
    lib/
      junit.jar
    lec03/
      main/
        se254/
          IntList.java IntList.class
      test/
        se254/
          TestIntList.java TestIntList.class

```

```

prompt > cd /home/ewan/se254stuff
prompt > java -cp lib/junit.jar:lec03/main:lec03/test \
? se254.TestIntList
.
Time: 0.001

OK(1 test)
prompt >

```

Running tests

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

```
/home/ewan/  
se254stuff/  
  intlist.jar  
lib/  
  junit.jar  
lec03/  
  test/  
    se254/  
      TestIntList.java TestIntList.class
```

```
prompt > cd /home/ewan/se254stuff  
prompt > java -cp lib/junit.jar:intlist.jar:lec03/test \  
? se254.TestIntList  
.  
Time: 0.001  
  
OK(1 test)  
prompt >
```

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

Running tests

```

/home/ewan/
  se254stuff/
    intl.jar
    lib/
      junit.jar
    lec03/
      main/
        se254/
          IntList.java IntList.class
      test/
        se254/
          TestIntList.java TestIntList.class

```

```

prompt > cd /home/ewan/se254stuff
prompt > java -cp \
? lib/junit.jar:intl.jar:lec03/main:lec03/test \
? se254.TestIntList
.
Time: 0.001

OK (1 test)
prompt >

```

In a world without JUnit

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

```
// somewhere
public void test(String name, int expected, int actual) {
    if (actual != expected) {
        System.out.println(name + " should have " +
                           "returned " + expected +
                           " but instead returned " +
                           actual);
    }
}
```

```
.....
int[] input = {1, 93, 2, 5, -17 };
IntList list = new IntList(input);
test("non-empty array with no zeros", 0, list.numZeros());
```

Test Automation Issues

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- Some artifact testing cannot easily be automated (e.g., user interfaces)
- Some potential failure situations cannot be easily exercised (e.g., timing problems in real-time or concurrent software)
- Some designs cannot be easily automatically tested (e.g., legacy code)
- Still a lot of work needed to support *traceability* — relationship between tests executed and client's requirements
- An important part of testing is very difficult to automate (what?)

Developing Tests

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- how do we describe tests?
- what's the best way to run the tests?
- **when is a test a “good” one?**
- how do we come up with tests?
- when do we have enough tests?

What is a “good” test?

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

- [PAQ](#)
- [Agenda](#)
- [Assignment 1](#)
- [Previously](#)
- [Test Suites](#)
- [Describing tests](#)
- [Test documentation](#)
- [IEEE 829-1998](#)
- [Test Case](#)
- [Developing Tests](#)
- [Test Procedure](#)
- [Procedure steps](#)
- [Test Scripts](#)
- [Automation](#)
- [Automation Issues](#)
- [Developing Tests](#)
- [Good Tests](#)
- [Key Points](#)

Key Points

- To ensure that testing is done properly, we need a good process
- To do a good job of testing, we need a test suite (group of test cases)
- Ideally we would like to run the test suite automatically