Software Design & Programmier Techniken

Wintersemester 2016/17

Lab 02 - 24. Oct 2016

Jonathan Immanuel Brachthäuser

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät
Fachbereich Informatik
Programmiersprachen und Softwaretechnik

Overview

- exercise organisation
- last homework
- last lecture
- testing
- domain analysis
- homework

Exercise Organisation

- no lab session / lecture next week
- homework grading
  - 0-2pt per exercise
  - min. 50% exercise points to participate in the exam.
  - in the range of 60% - 100% achieved exercise points you get a linear 20% bonus for the exam (after passing).
- next homework:
  - you have two weeks
  - hand in as a group of 3
  - hand in using github education.

Software Design & Programmier Techniken

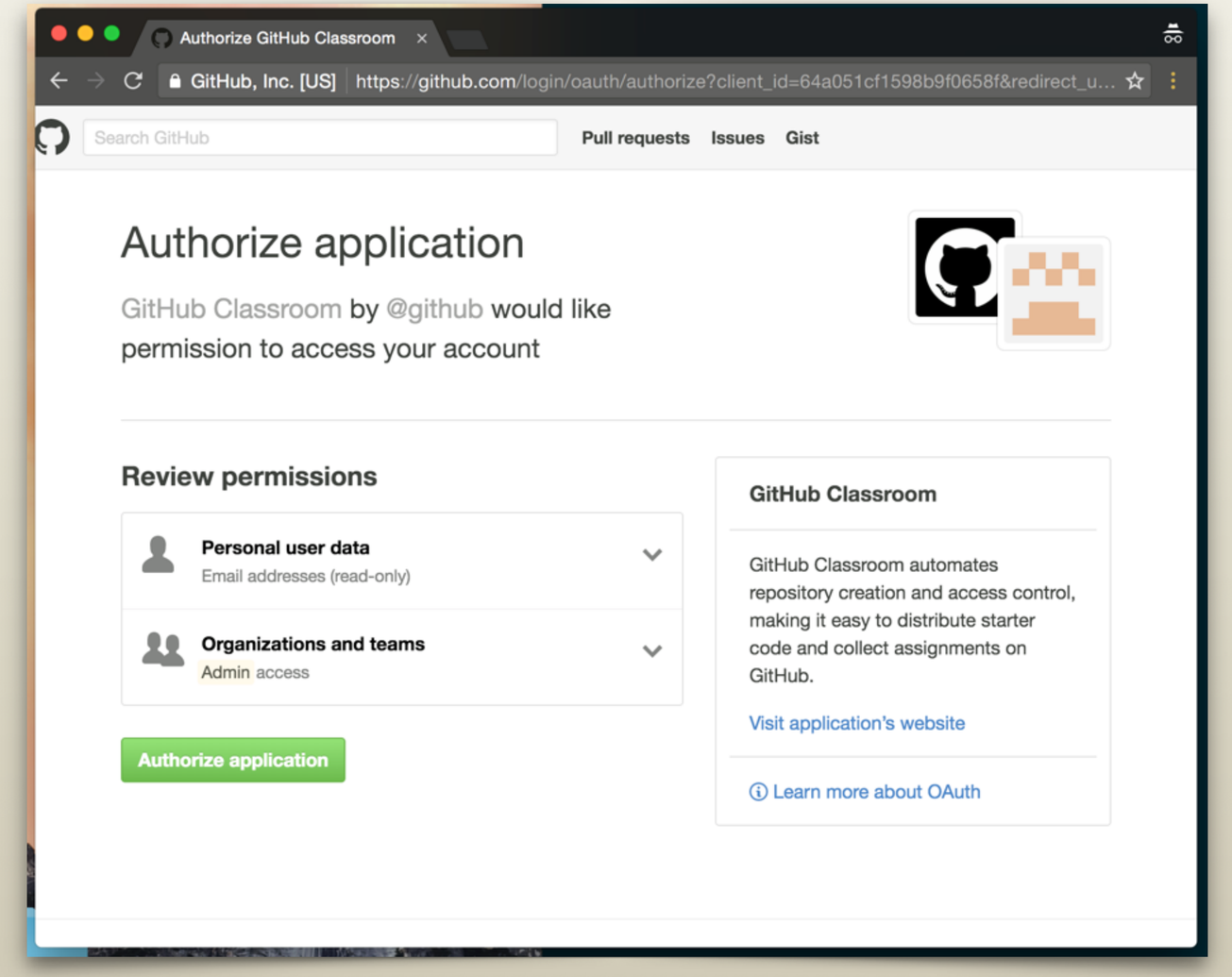
Wintersemester 2016/17

Handing in your Homework

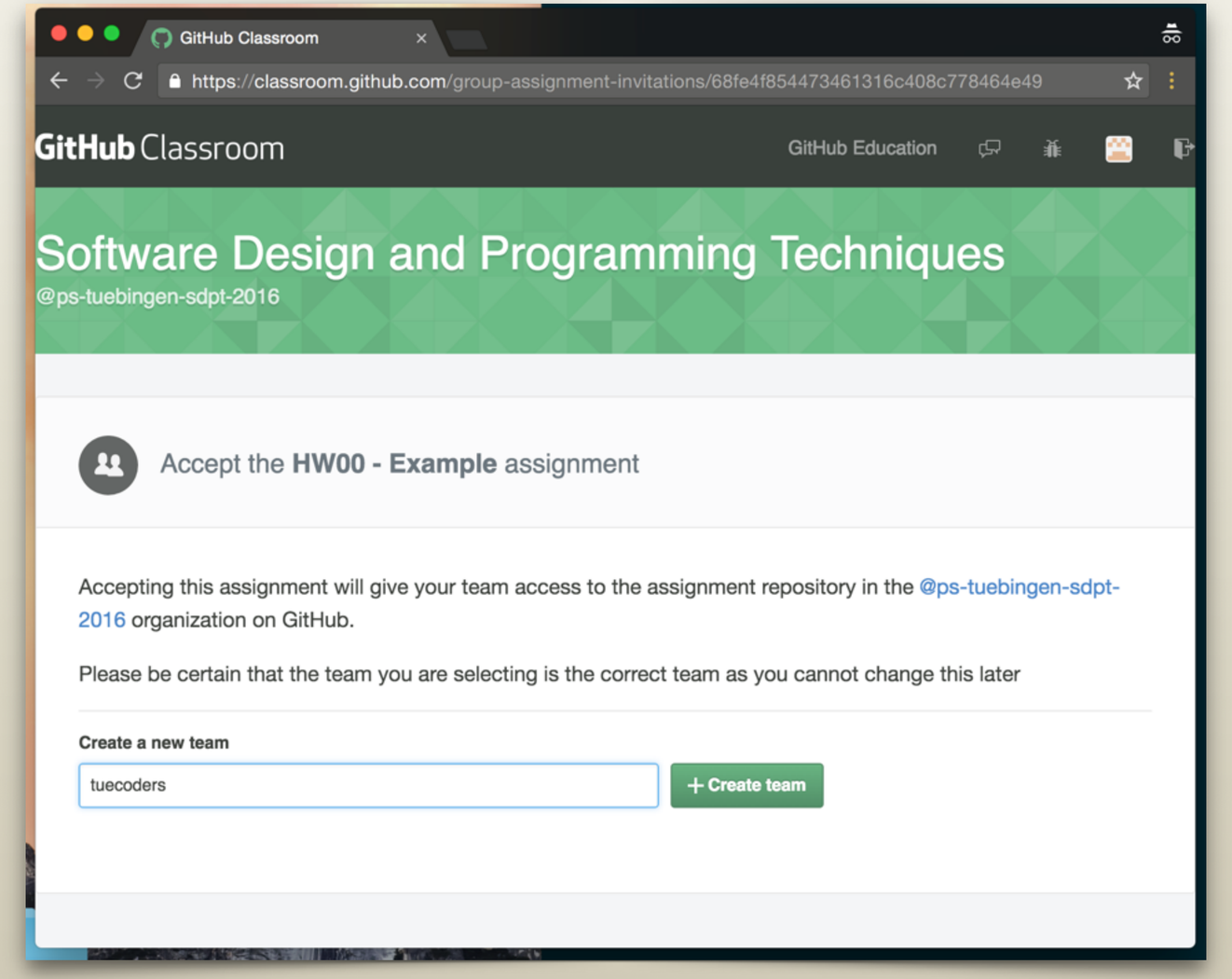
Using github education

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

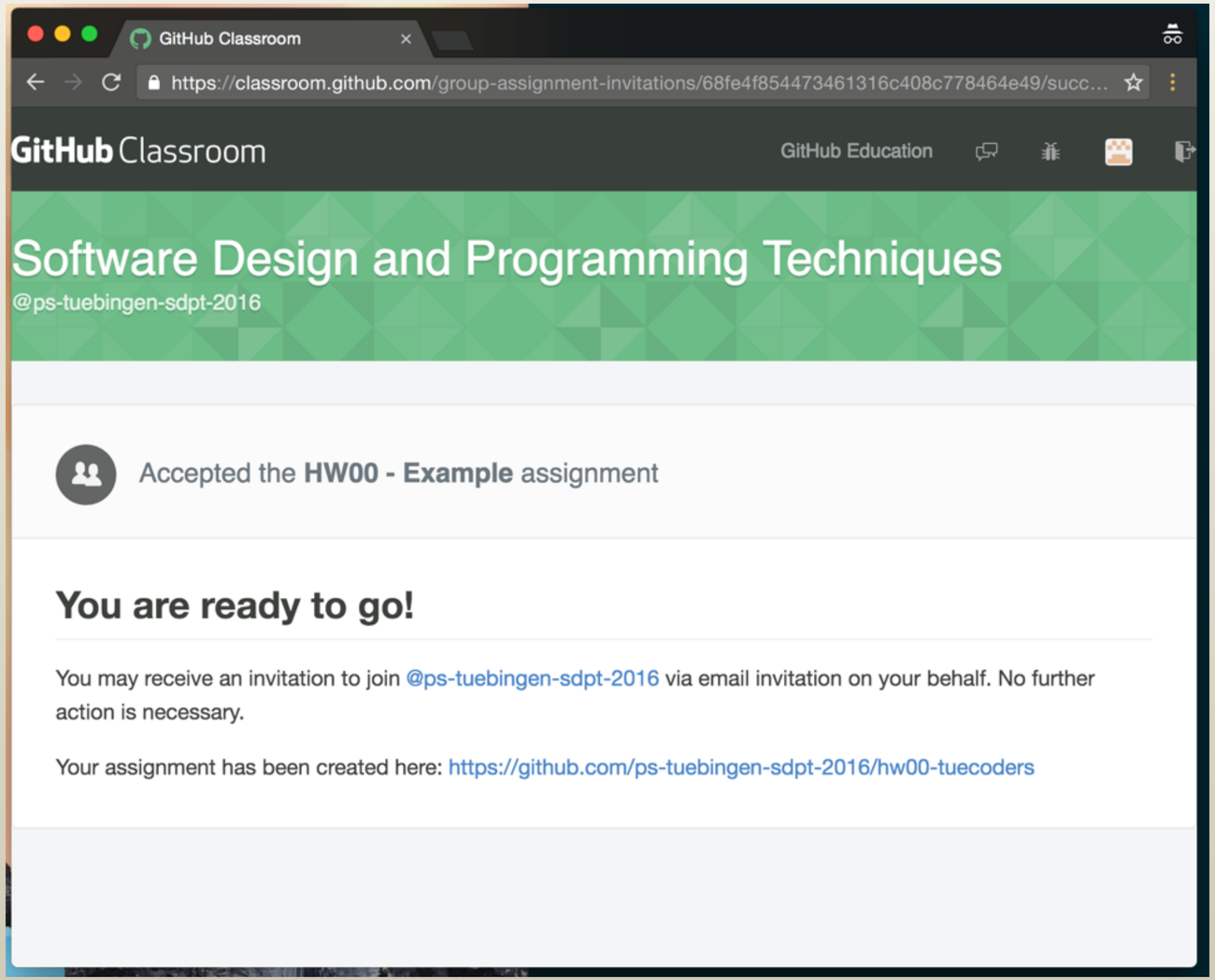
Mathematisch-Naturwissenschaftliche Fakultät
Fachbereich Informatik
Programmiersprachen und Softwaretechnik



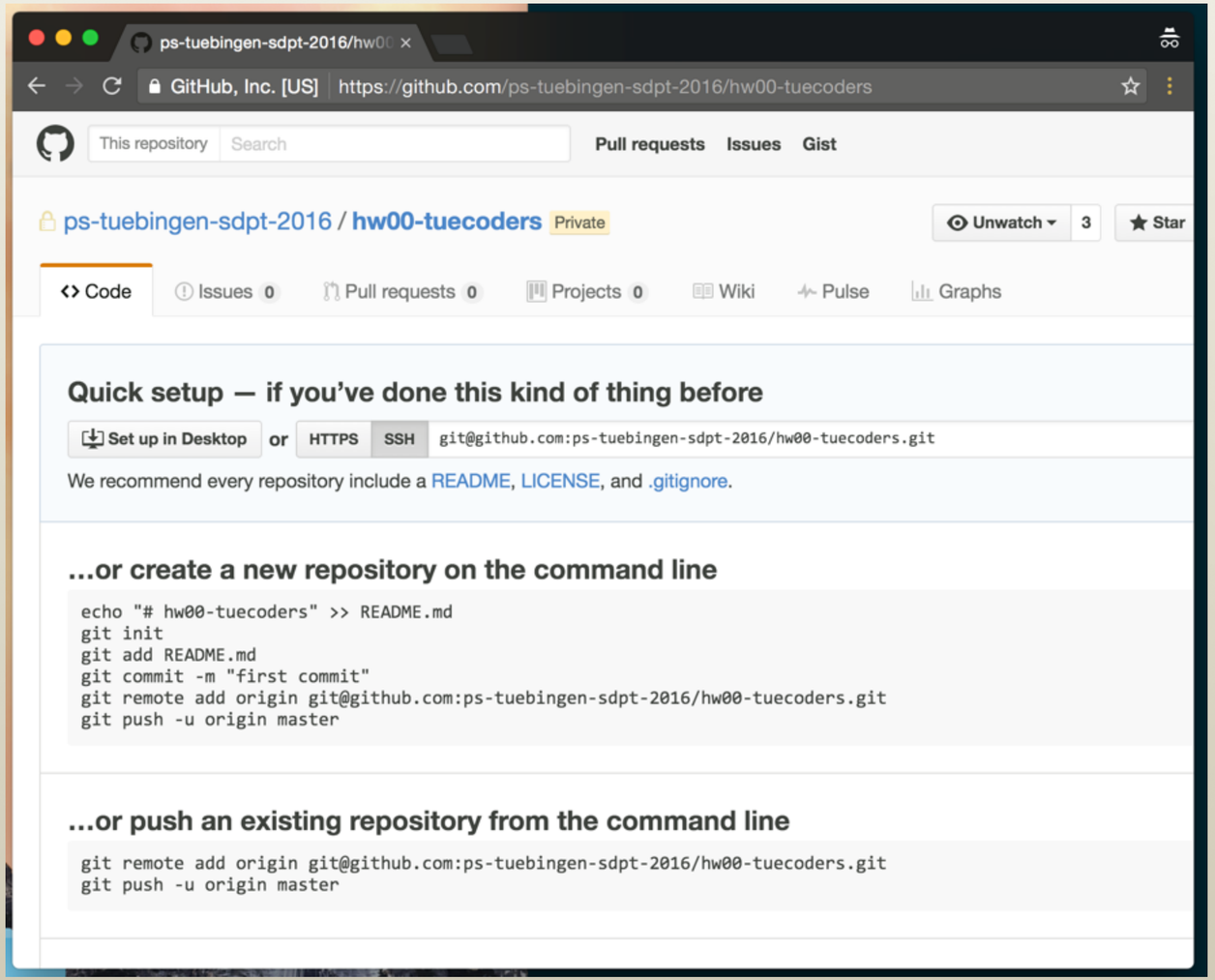
After clicking the assignment link the first time you need to authorize GH classroom.



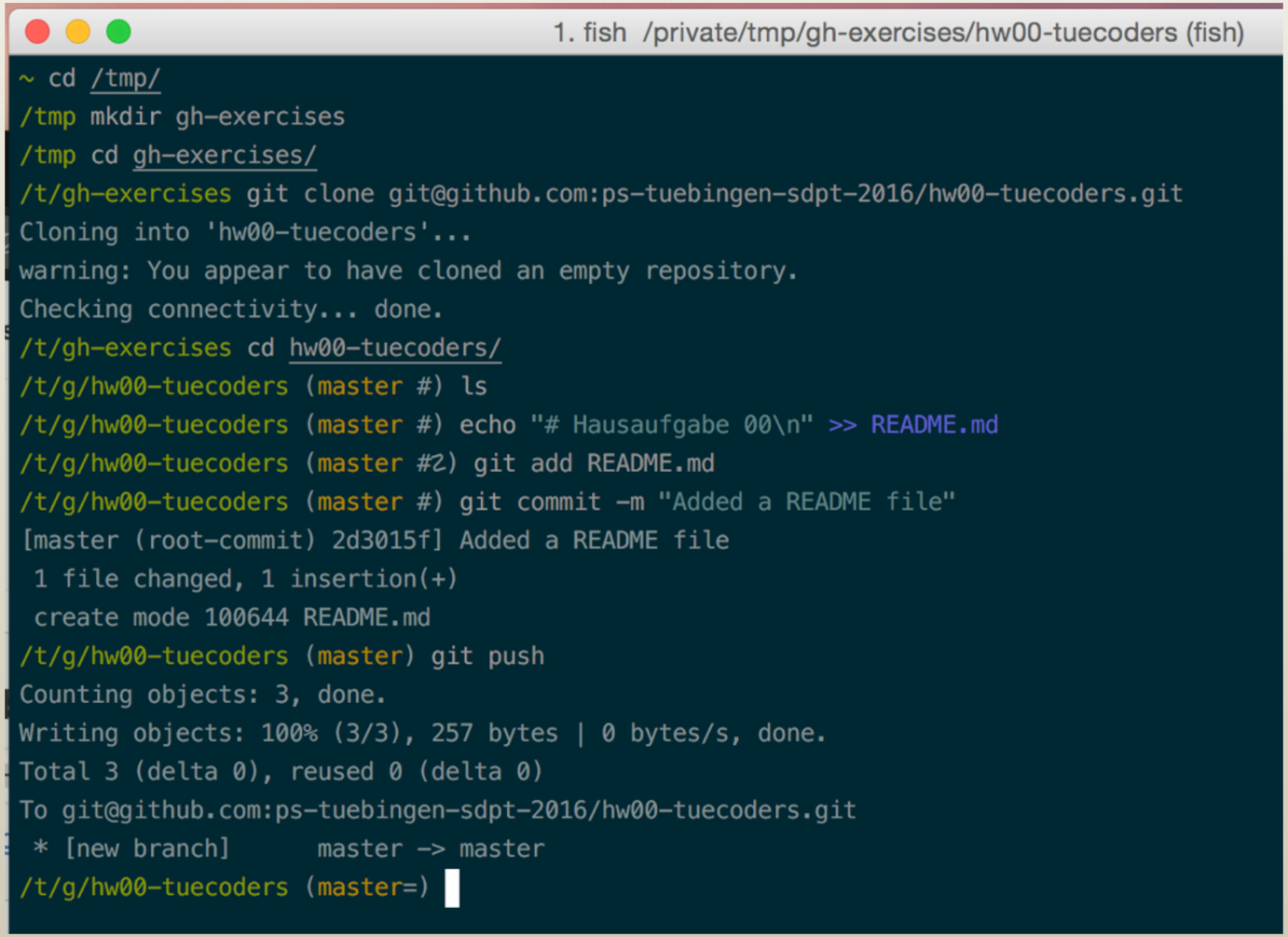
To accept the assignment you need to create or join a team.



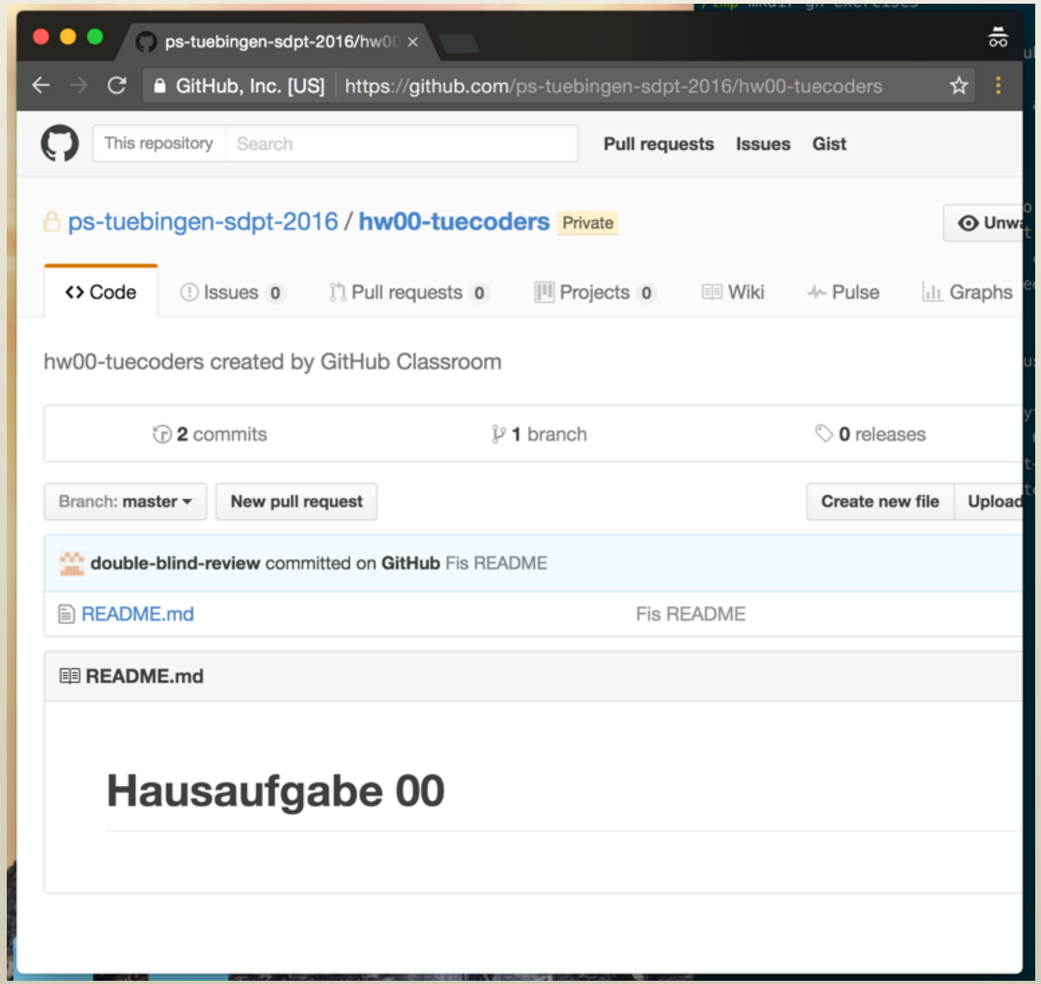
A private repository will be automatically created for your team.



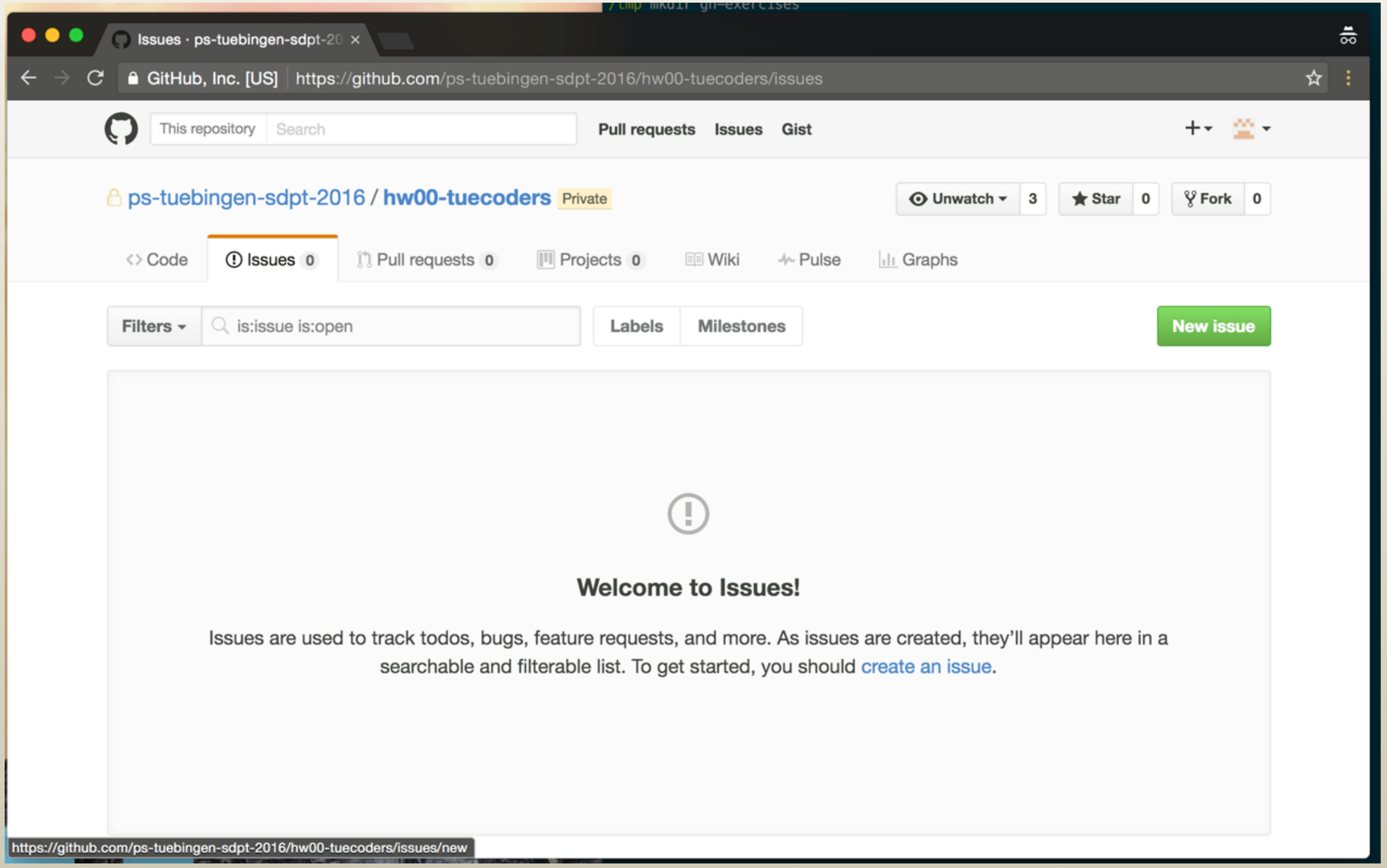
You can start collaborating with your team members on the assignment.



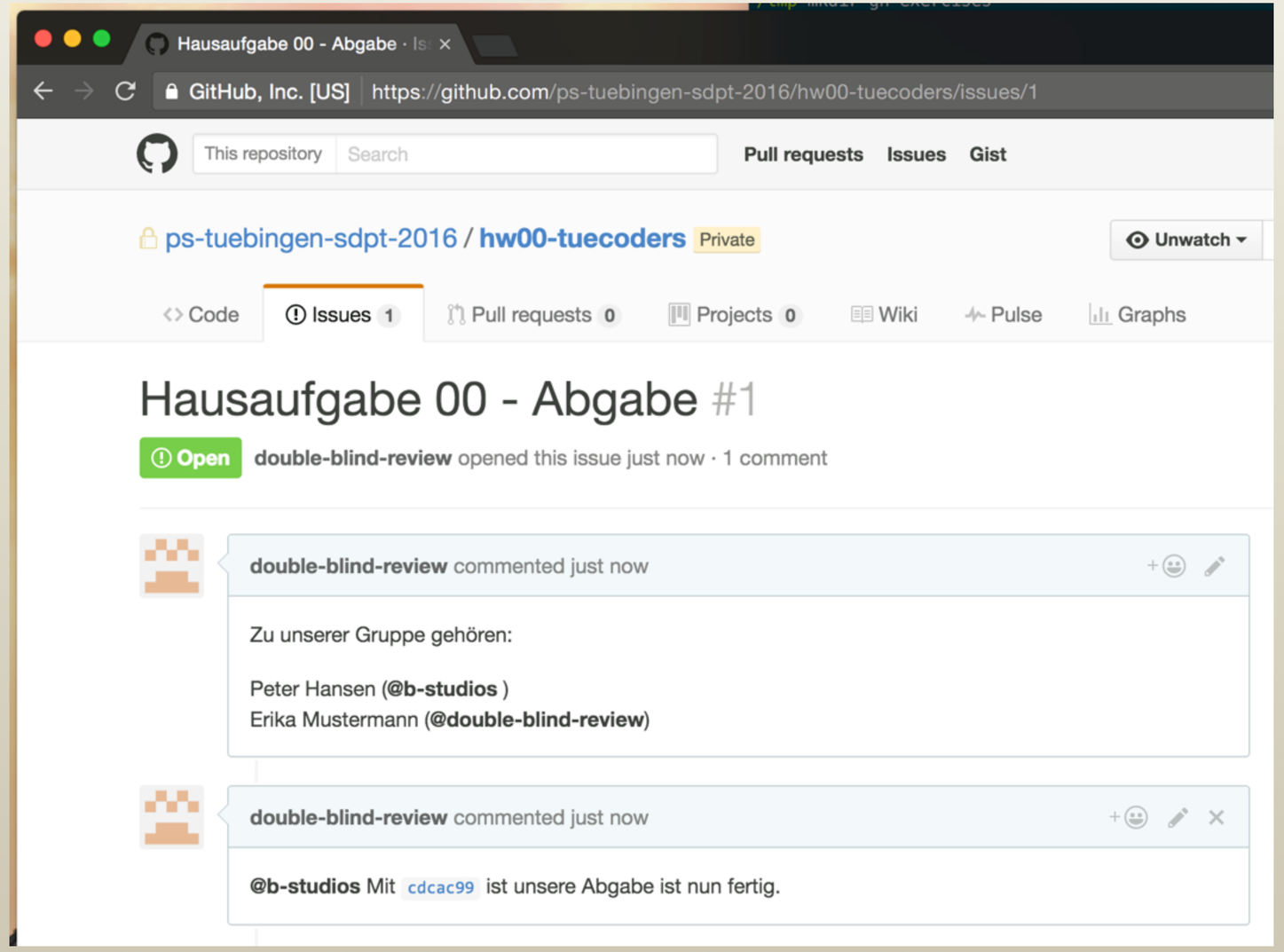
Clone the repository, contribute to the handin and push your changes.



Your contributions will be visible on the GH page of your group's repository.



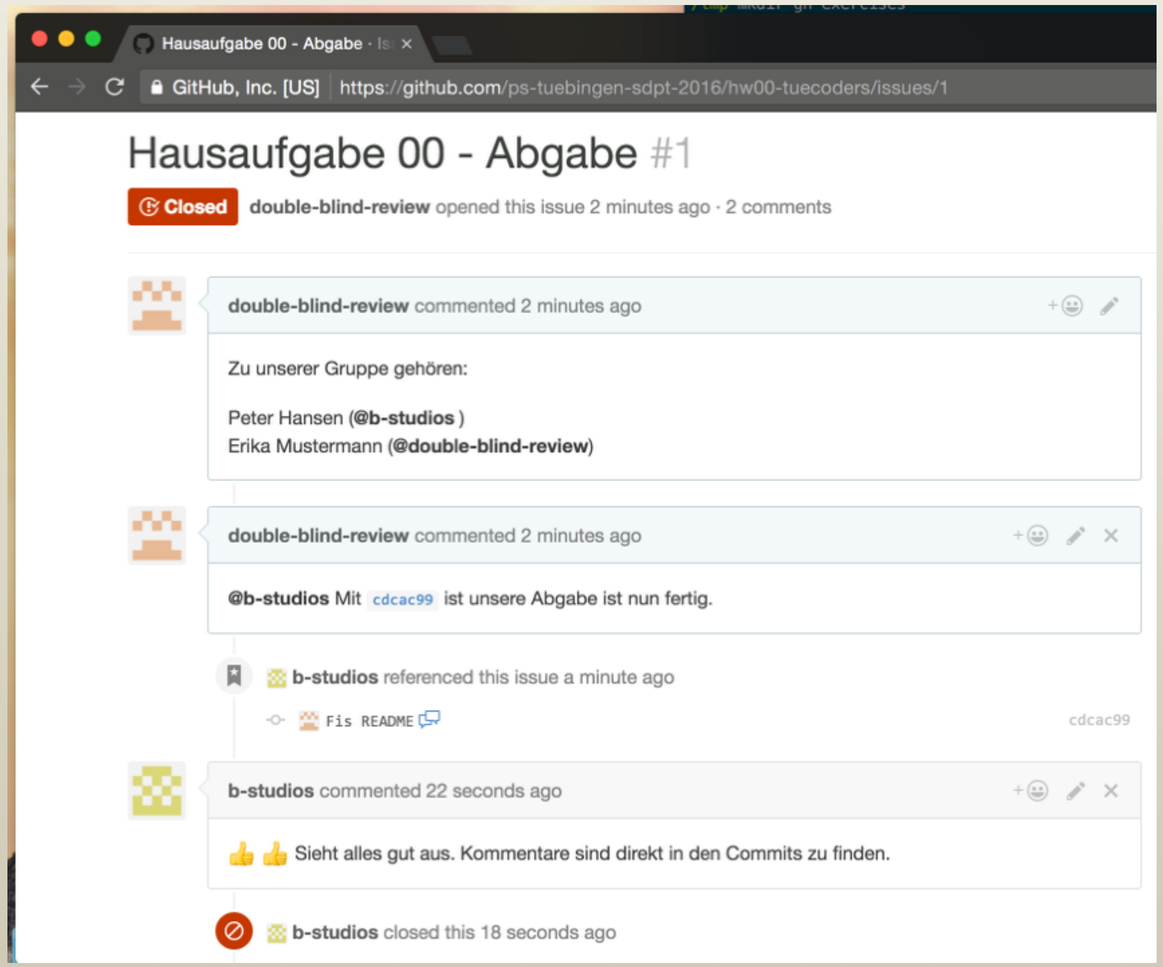
To hand in your assignment, create an "Issue".



1. *Mention your team members in the issue description.*

2. *When you are done, mention **@b-studios** and the hash of your final commit, like cdac99.*

3. *You can still add commits until the deadline, make sure to mention the issue number ("#1" here) in the commit, so it is listed in the issue history.*



Your handin will be graded and the issue will be closed.

Hand-in Format (3)

- Most of the time, we will program in **Java**, sometimes in **Scala**.
- You can use whatever editor you prefer, to work on you homework, however, make sure *before sending a PR* that your project compiles with **sbt** (http://www.scala-sbt.org)

- Sometimes you need to **write text** as part of your homework. Feel free to edit the homework assignment and inline your answers.

- Always make sure that you add all the files, that belong to your handin before committing.

Software Design & Programmier Techniken

Wintersemester 2016/17

Last Lecture

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät
Fachbereich Informatik
Programmiersprachen und Softwaretechnik

Homework: Generics

1. **Read up Generics in Java.**
   Read the chapter *"Generics (Updated)"* up to *"Generic Methods and Bounded Type Parameters"* (inclusive).
   https://docs.oracle.com/javase/tutorial/java/generics/index.html

2. **Program using Generics**
   Import the `Pair<K, V>` class from generics tutorial into your project.
   Program an interface / abstract class `Showable<T>` with a method `String show(T t)`
   Provide static instances for `Showable<String>`, `Showable<Integer>` and a static method to create `Showable<Pair<S, T>>`

3. **Reason about your Program**
   Compare this design with having an interface `Showable { String show(); }` which needs to be implemented by the classes that we want to be "Showable". Compare the two designs with respect to Extensibility. Consider multiple scenarios where you want to (a) add new classes and make them "Showable", (b) make existing classes "Showable", (c) adding a second, different implementation of Showable for some classes.

The actual assignment can be found at: **https://github.com/ps-tuebingen-sdpt-2016/homework/tree/master/hw01**

Software Design & Programmier Techniken

Wintersemester 2016/17

Testing

based on slides by Tillmann Rendel and Prof. Dr. Mira Mezini

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät
Fachbereich Informatik
Programmiersprachen und Softwaretechnik

Unit Testing

Project Goals

- tests act as specification
- tests act as documentation
- tests prevent regression of bugs
- tests help localizing defects
- tests lead to more modularity
- tests act as safety net for refactorings

=> Tests help us to improve quality

Unit Testing

Project Goals

- tests act as specification
- tests act as documentation
- tests prevent regression of bugs
- tests help localizing defects
- tests lead to more modularity
- tests act as safety net for refactorings

=> Tests help us to improve quality

Test Writing Goals

- tests reduce and not introduce risk
- tests should be fully automated
  - tests should be self-checking
  - tests should be repeatable
- tests should be simple / small
- tests should be expressive
- tests should be robust

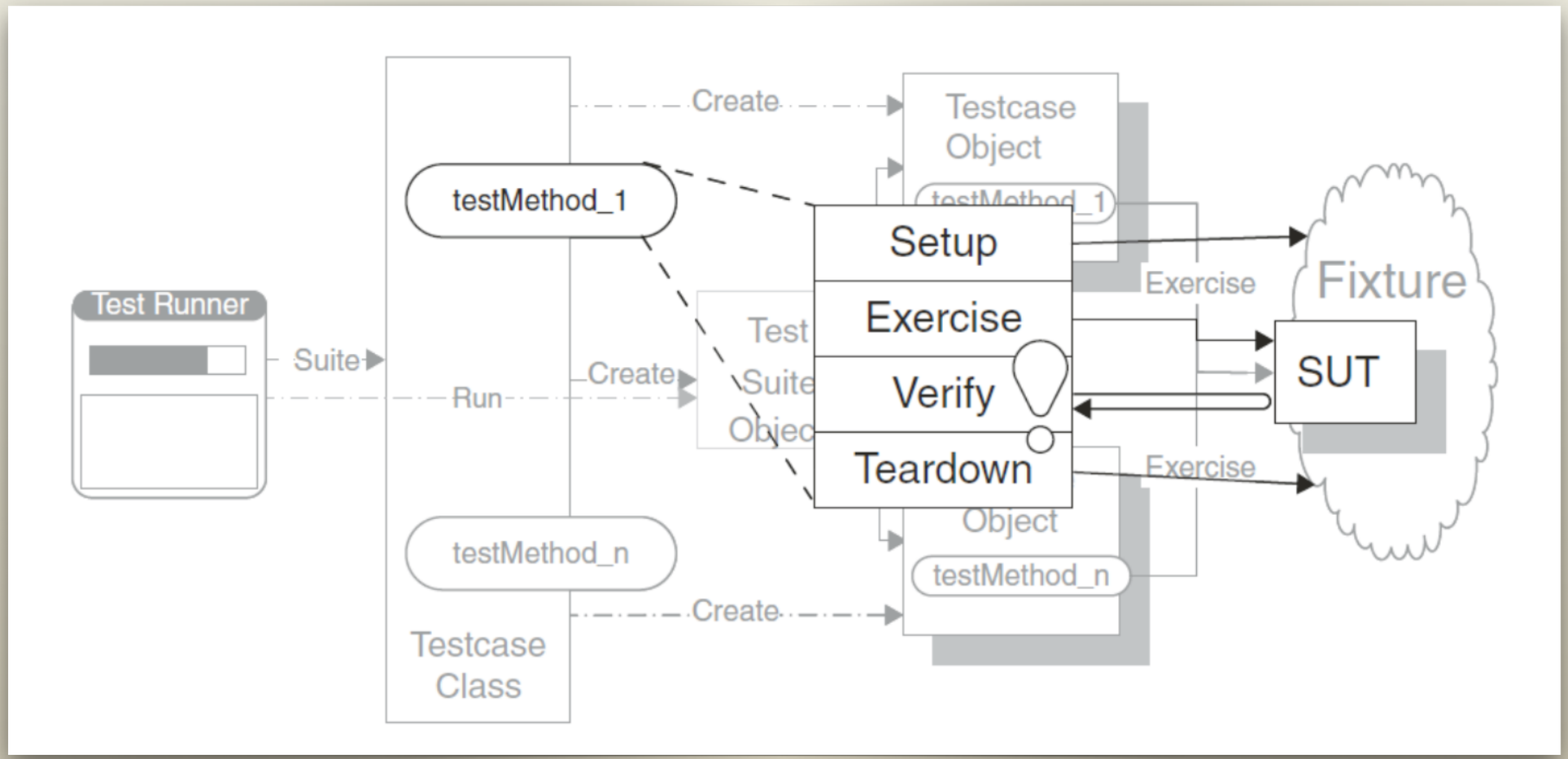
=> Reducing the testing barrier takes effort

Unit Testing

Fixture

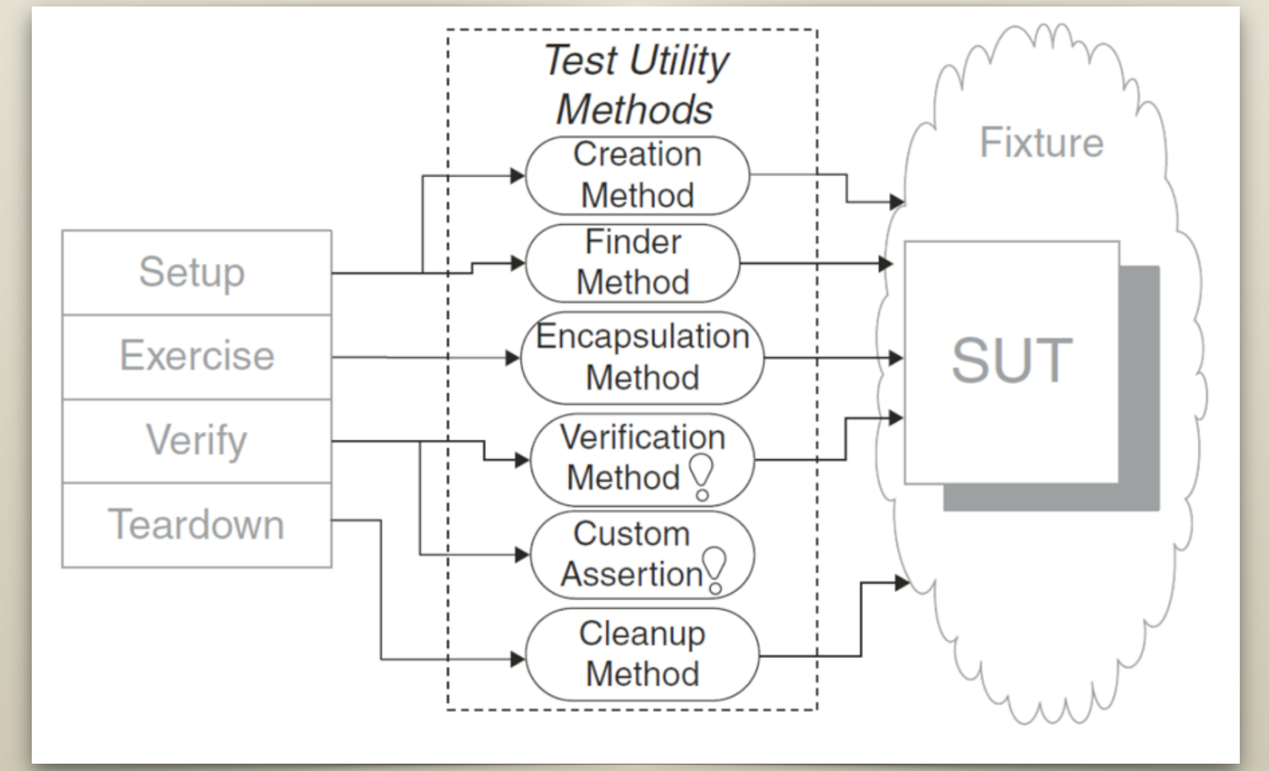
System Under Test (SUT)

Unit Testing - Four steps of test execution



Unit Testing - Test Utility Methods

Don't repeat yourself in writing tests!

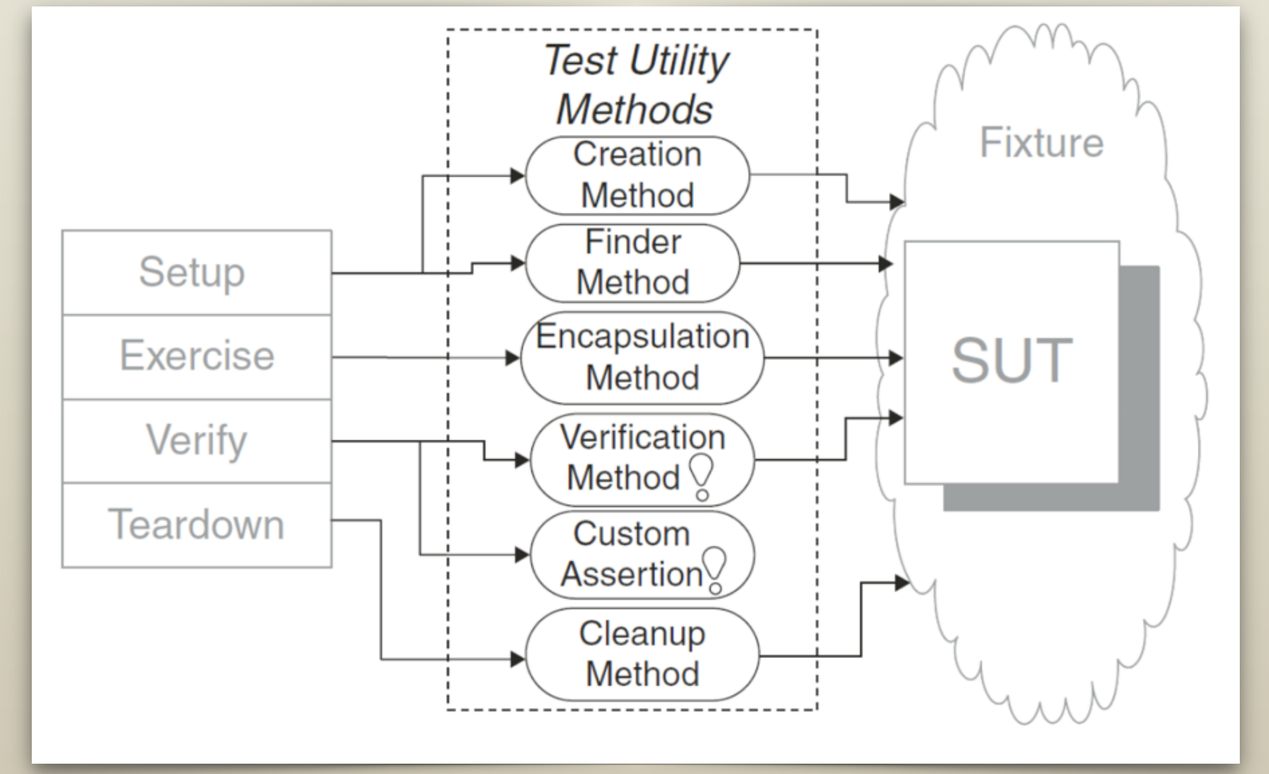


Creation Methods
... create ready-to-use objects as part of fixture setup. They hide the complexity of object creation and interdependencies from the test.

Finder Methods
... find an existing shared fixture object that meets some criteria and we want to avoid a fragile fixture and high test maintenance cost.

Unit Testing - Test Utility Methods

Don't repeat yourself in writing tests!

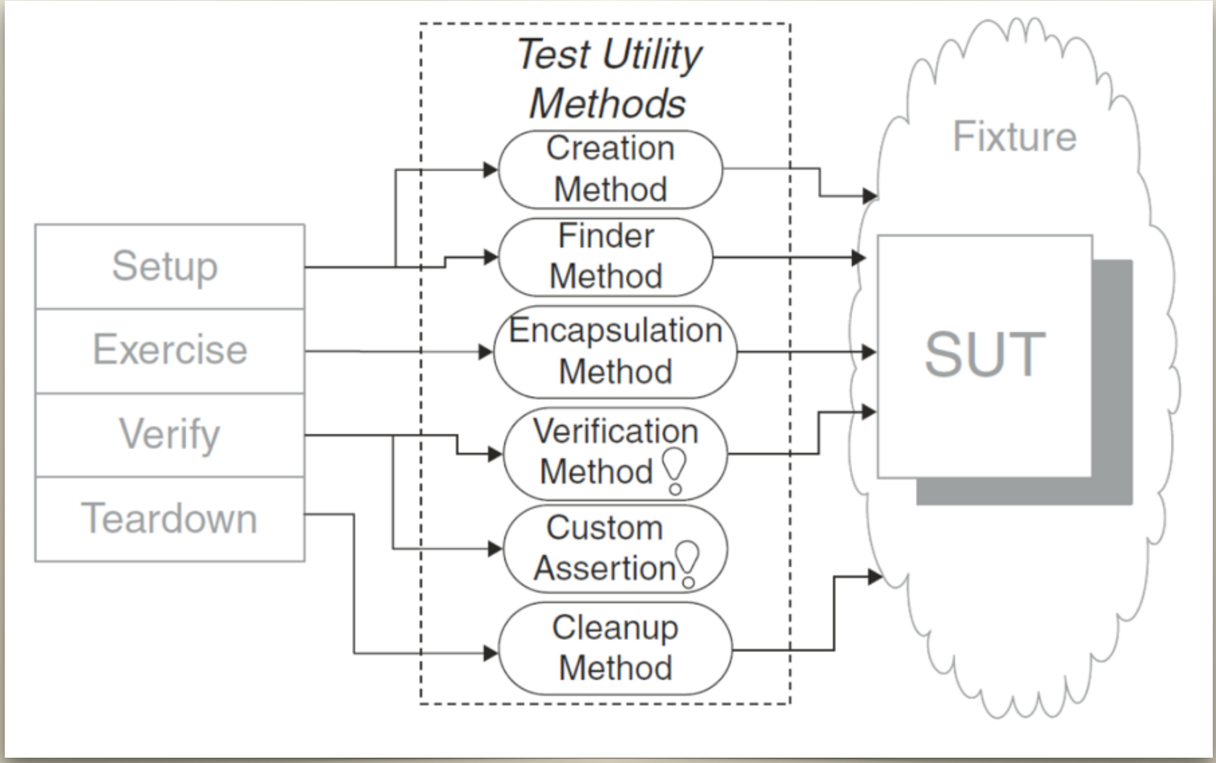


Encapsulation Methods
... encapsulate unnecessary knowledge of the API of the SUT that complicates the understanding of the test.

Custom Assertions
... specify test-specific equality in a way that is reusable across many tests. They hide the complexity of comparing the expected outcome with the actual outcome. Custom Assertions are typically side effect free and do not use the SUT.

Unit Testing - Test Utility Methods

Don't repeat yourself in writing tests!



Verification Methods
... verify that the expected outcome has occurred. They hide the complexity of verifying the outcome from the test. Unlike Custom Assertions, Verification Methods interact with the SUT .

Cleanup Methods
... used during the fixture teardown phase of the test to clean up any resources that might still be allocated after the test ends.

Software Design & Programmier Techniken

Wintersemester 2016/17

HW: Develop a Survey Library

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät
Fachbereich Informatik
Programmiersprachen und Softwaretechnik

Story #1

As a client developer I want to be able to describe the structure of a survey to deliver it as running application to the inquirer.

Description:

Using an application programming interface provided by the survey library, I want to create individual questions and compose them to full surveys.
This should be done in Java, since I want to abstract over reoccurring questions and parts of surveys.

Acceptance Criteria:

- library is written in Java
- the API offered by the library allows to describe questions
- the API offered by the library allows to describe full surveys

Story #2

As a client developer I want to be able to automatically validate surveys to avoid mistakes in defining them.

Description:

For large surveys, sometimes questions occur twice or we forget to add a text for questions. It should be possible to perform a validation run and check at least for the acceptance criteria.

Acceptance Criteria:

- questions that occur multiple times are recognized and rejected
- questions that have no question text are recognized and rejected
- empty surveys should be recognized and rejected

Story #3

As a inquirer I want to print a survey in order to hand it to users on paper.

Description:

Sometimes users cannot come to our office, so we need to send them the survey on paper. I need some simple textual representation of a survey in order to print it.

Acceptance Criteria:

- the text includes the name and description of the survey
- the text includes all question texts
- after every question text there should be some space to fill in the question by hand

Story #4

As a user I want to be prompted by the running survey application to answer questions one-after-another in order to fill in a survey form.

Description:

Seeing all questions at once can be very confusing and reduces motivation. Hence, clients should only be exposed to small fragments of the survey one-at-a-time. In the end a report should be printed.

Acceptance Criteria:

- For each question in a survey, the running application should prompt the user with the question text.
- The answers of the user are collected in a report.
- The report is printed, associating question identifiers with the given answer.
- Answers should not be longer than 50 words or 300 characters.