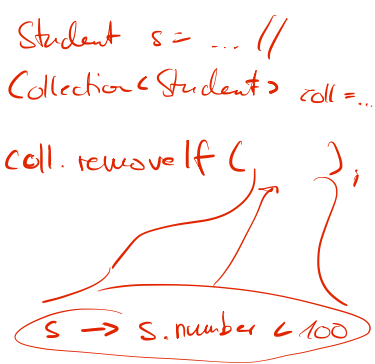
Recap: Sorting and Functional Interfaces

- Sort algorithms
  - Naive sorting takes a lot longer than needed
  - MergeSort: Recursive algorithm for sorting collections
- Functional Interfaces
  - Sometimes we need to pass code as arguments
    - E.g., sort criteria
  - Several interfaces for that (e.g., `Function<X,Y>`) and special syntax: `x -> x.someMethod()`
- Exercise 10

Collections and Streams

- Interface `Collection<E>`
  - defines method `removeIf(Predicate<? super E> filter)`
  - defines method `stream()`



Collections and Streams

- Interface `Collection<E>`
  - defines method `removeIf(Predicate<? super E> filter)`
  - defines method `stream()`
- Easy to implement and read way of dealing with collections

demo

Session 11: Stream-API and Unit-Testing

Fortgeschrittene Programmierung (Java 2)

Nils Reiter
nils.reiter@uni-koeln.de

June 28, 2023

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

IDH Summer Party 2023

05.07., 18 Uhr

Uniwiese, zwischen Mensa und IDH

Section 1

Unit-Testing

Introduction

- Automatic verification that the code works properly
- Why?
  - Large, complex projects: Changes at one place might break something else
  - Humans make mistakes

Introduction

- Automatic verification that the code works properly
- Why?
  - Large, complex projects: Changes at one place might break something else
  - Humans make mistakes
- Unit tests
  - Collection of expected output for a given input
    - E.g., for `add(2,3)`, we expect 5
  - Include edge cases
    - E.g., illegal input

JUnit

- Java library to support unit testing
- Uses code annotations and static methods
- Version 4 vs. 5
- User guide

JUnit

- Java library to support unit testing
- Uses code annotations and static methods
- Version 4 vs. 5
- User guide

```
1  import static org.junit.jupiter.api.Assertions.assertEquals;
2
3  import org.junit.jupiter.api.Test;
4
5  class TestCalculator {
6    Calculator calculator = new Calculator();
7
8    @Test
9    void addition() {
10     assertEquals(2, calculator.add(1, 1));
11   }
12 }
```

Annotations

- `@Test`
  - Marks a method that contains at least one test
- `@BeforeEach` / `@AfterEach`
  - Code to be executed before/after each test method
  - E.g., to create data structures used for the test
- `@BeforeAll` / `@AfterAll`
  - Code to be executed once before the first / after the last test method
  - E.g., to establish a database connection

Assert...

- Assert methods to specify comparison between expected and real outcome
- `assertEquals(expected, actual)`: We expect them to be equal
  - I.e., that `equals()` returns true
- `assertTrue(a)` / `assertFalse(a)`: We expect to be true/false
- `assertNotEquals(...)`: We expect them to be not equal

Assert...

- Assert methods to specify comparison between expected and real outcome
- `assertEquals(expected, actual)`: We expect them to be equal
  - I.e., that `equals()` returns true
- `assertTrue(a)` / `assertFalse(a)`: We expect to be true/false
- `assertNotEquals(...)`: We expect them to be not equal
- `assertThrows(ex, code)` We expect the code to throw an exception of type ex
  - `code` given using a functional interface (last week)
  - E.g. `assertThrows(ArithmeticException.class, () -> { calculator.divide(1, 0) })`

Assert...

- Assert methods to specify comparison between expected and real outcome
- `assertEquals(expected, actual)`: We expect them to be equal
  - I.e., that `equals()` returns true
- `assertTrue(a)` / `assertFalse(a)`: We expect to be true/false
- `assertNotEquals(...)`: We expect them to be not equal
- `assertThrows(ex, code)` We expect the code to throw an exception of type ex
  - `code` given using a functional interface (last week)
  - E.g. `assertThrows(ArithmeticException.class, () -> { calculator.divide(1, 0) })`
  - Why does `assertThrows(ArithmeticException.class, calculator.divide(1, 0))` not work?

JUnit

Maven Integration

```
<dependencies>
  <!-- Allows implementing the tests -->
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.8.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <plugins>
    <!-- Allows maven to find and execute the tests -->
    <plugin>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.22.2</version>
    </plugin>
  </plugins>
</build>
```

demo

Exercise

https://github.com/idh-cologne-java-2-summer-2023/exercise-11