

Praktische Informatik 2

Organisatorisches und Überblick

Thomas Röfer

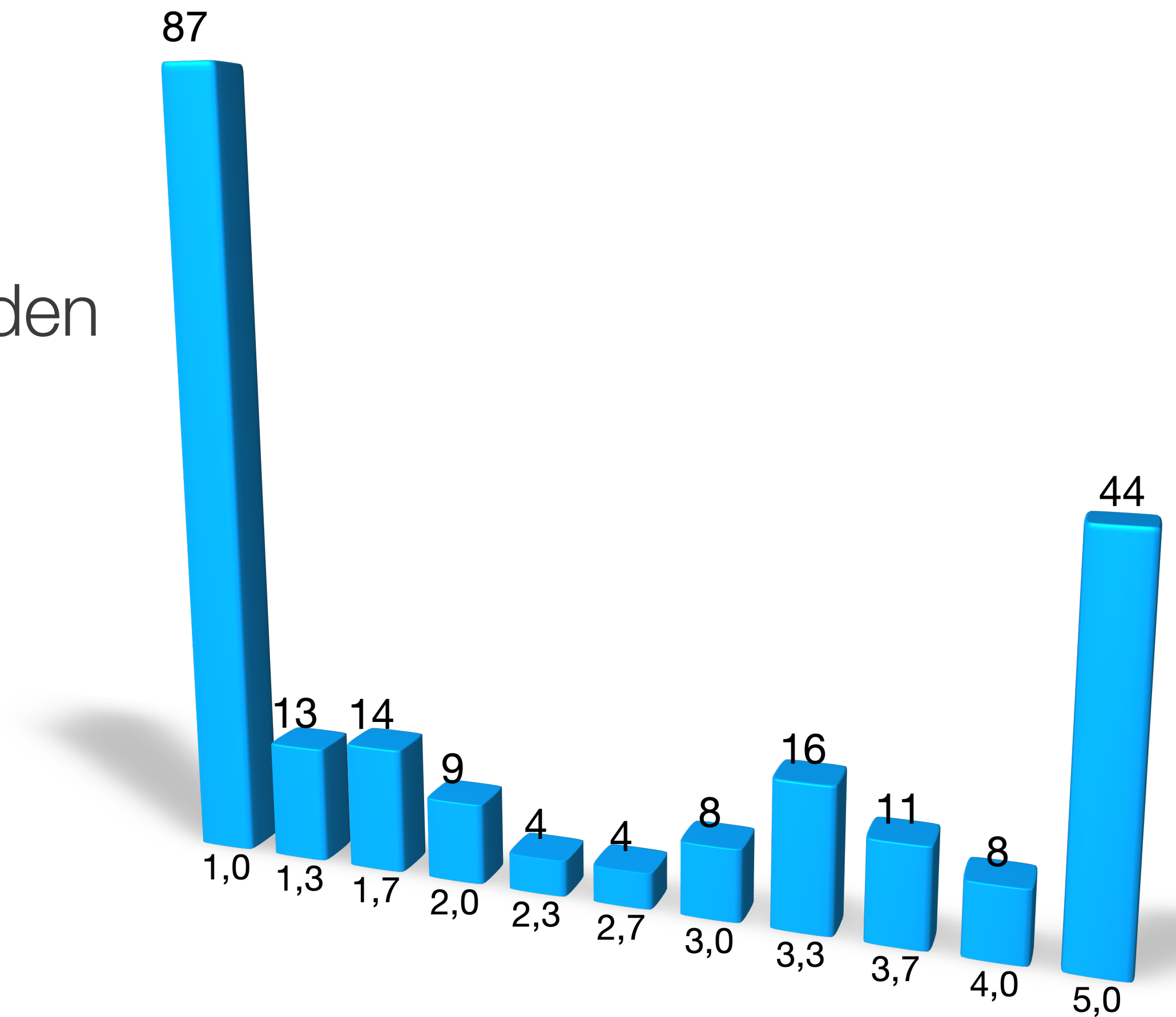
Cyber-Physical Systems
Deutsches Forschungszentrum für
Künstliche Intelligenz

Multisensorische Interaktive Systeme
Fachbereich 3, Universität Bremen



Rückblick: PI-1-Klausur

- 317 haben mindestens 1 Übungsblatt abgegeben
- 247 (78 %) haben den Übungsbetrieb bestanden
- 218 (69 %) haben die Klausur mitgeschrieben
- 174 (55 %) haben die Klausur bestanden
 - 26 haben ihre Note verbessert
 - 51 haben ihre Note verschlechtert



Rückblick: PI-1-Klausur

- 200 haben das Feld „sinnvoll“ bearbeitet
 - 117 antworteten mit „15“
 - 83 antworteten mit „16“

Nach Ausführung des obigen Codes wird der Wert von `b.get()` 16 sein. Zunächst wird eine Instanz der Klasse `Value` erstellt und der Wert 15 mit der Methode `set()` gesetzt: `a.set(15)`. Dann wird die Referenz `a` der Variable `b` zugewiesen: `Value b = a`. Da beide Variablen nun auf dasselbe Objekt zeigen, haben sie auch denselben Wert. Schließlich wird der Wert von `a` mit der Zeile `a.set(a.get() + 1)` auf 16 erhöht, da zu dem ursprünglichen Wert von 15 eins addiert wird. Da `a` und `b` dieselbe Referenz auf dasselbe Objekt haben, ist der Wert von `b.get()` nun auch 16.

ChatGPT

Werte und Referenzen

Gegeben sei:

```
int a = 15;  
int b = a;  
a = a + 1;
```

Was ist der Wert von **a**?

Was ist der Wert von **b**?

Gegeben sei:

```
class Value  
{  
    private int value;  
  
    void set(final int val)  
    {  
        value = val;  
    }  
  
    int get()  
    {  
        return value;  
    }  
}  
// ...  
Value a = new Value();  
a.set(15);  
Value b = a;  
a.set(a.get() + 1);
```

Was ist der Wert von **a.get()**?

Was ist der Wert von **b.get()**?

Worum geht's dieses Semester?

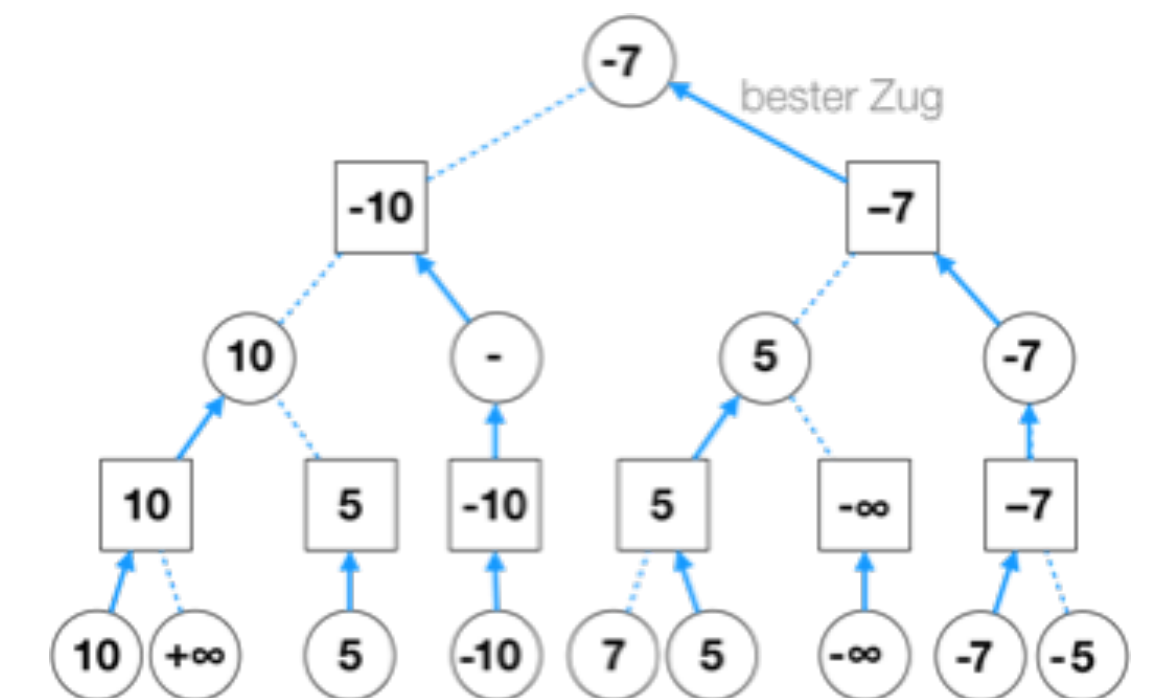
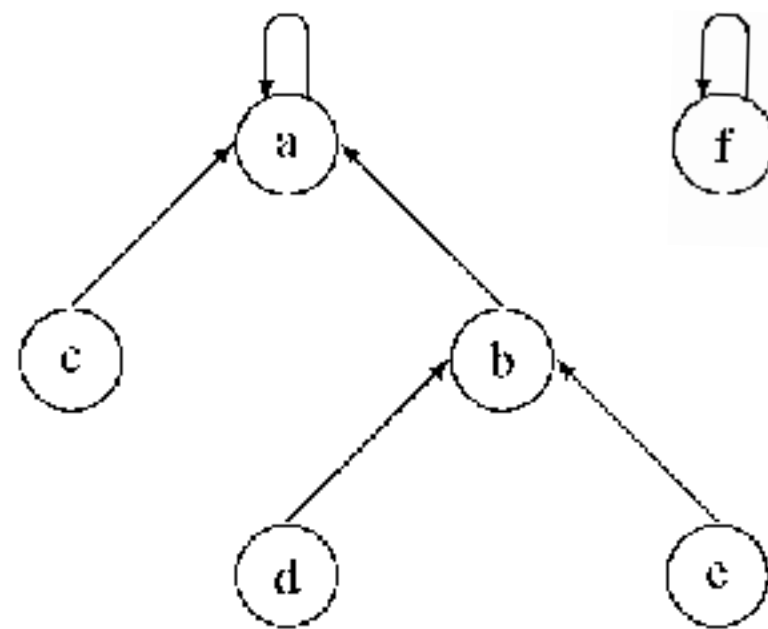
- Algorithmen und Datenstrukturen
 - Was ist ein Algorithmus?
 - Wie lange rechnet er in Abhängigkeit von der Größe der Eingabe (Komplexität)?
 - Wann ist er korrekt und wie kann das gezeigt werden?
 - Warum gibt es verschiedene Algorithmen für dasselbe Problem?
 - Welche Problemlösungsparadigmen gibt es?
- Standardalgorithmen zum Speichern und Wiederfinden von Daten
- Noch ein bisschen Java: Generische Klassen

Worum geht's dieses Semester?

- Wie funktionieren (grob): **ArrayList<E>**, **Stack<E>**, **(Priority)Queue<E>**, **LinkedList<E>**, **Arrays.binarySearch(...)**, **Arrays.sort(...)**, **TreeSet<E>**, **HashSet<E>**

- Spielprobleme (Minimax)

- Union-Find



- Ausgewählte Algorithmen auf Graphen, z.B. kürzeste Wege

Inhaltliche Voraussetzungen

- Praktische Informatik 1
- Programmierung in **Java** beherrschen
 - Wer kein **Java** kann, wird nicht bestehen und gefährdet das Bestehen anderer!
- Umgang mit **git** / **GitLab**
- Rudimentärer Umgang mit $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$

Überblick

- Übungen in Zweiergruppen aus **demselben** Tutorium, abwechselnd
 - 1 Person Implementierung
 - 1 Person Tests
- Ausgabe Donnerstag abends
- Abgabe sonntags, 10 Tage später

Woche	Vorlesung	Übung
17.04	Generisches Programmieren	1
24.04	Algorithmische Grundkonzepte	
01.05	Komplexität	2
08.05	Grundlegende Datenstrukturen	
15.05	Suchen	3
22.05	Sortieren	
29.05		4
05.06	Bäume	
12.06		5
19.06	Manipulation von Mengen	
26.06	Graphenalgorithmen	6
03.07	Hashing	
10.07	Spezifikation und Verifikation	

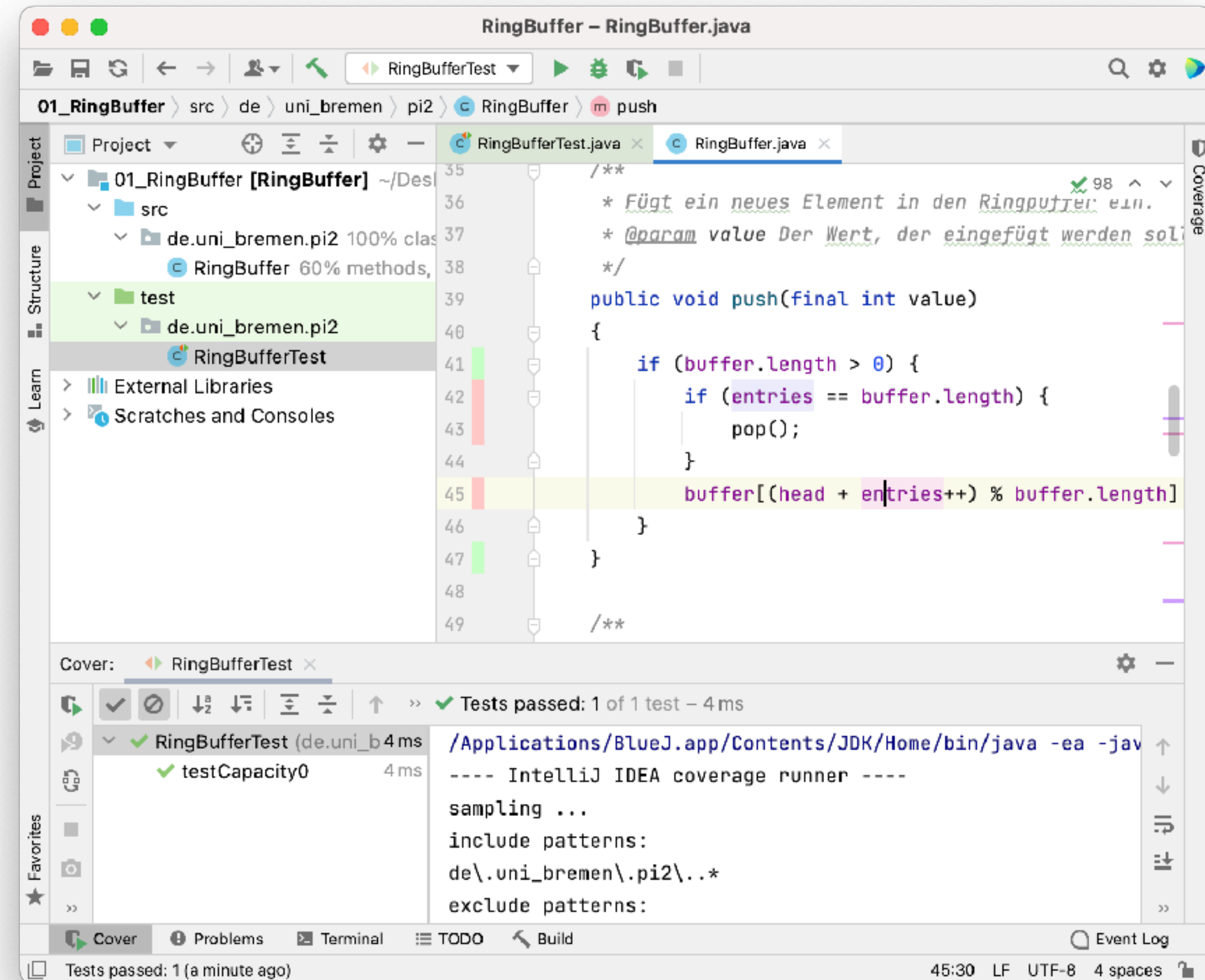
Stud.IP

- Karteikarte „Informationen“
 - Vorlesungsfolien und Code-Beispiele
 - Übungsblätter und Musterlösungen
 - Scheinbedingungen
 - Verweise auf zu installierende Software
 - Literatur
- Per Stud.IP an mich gestellte Fragen werden per E-Mail beantwortet
→ Mail-Account **@uni-bremen.de** regelmäßig abrufen!

Algorithm-Support: PI-Ergänzungskurs (03-ISONST-ASPI)

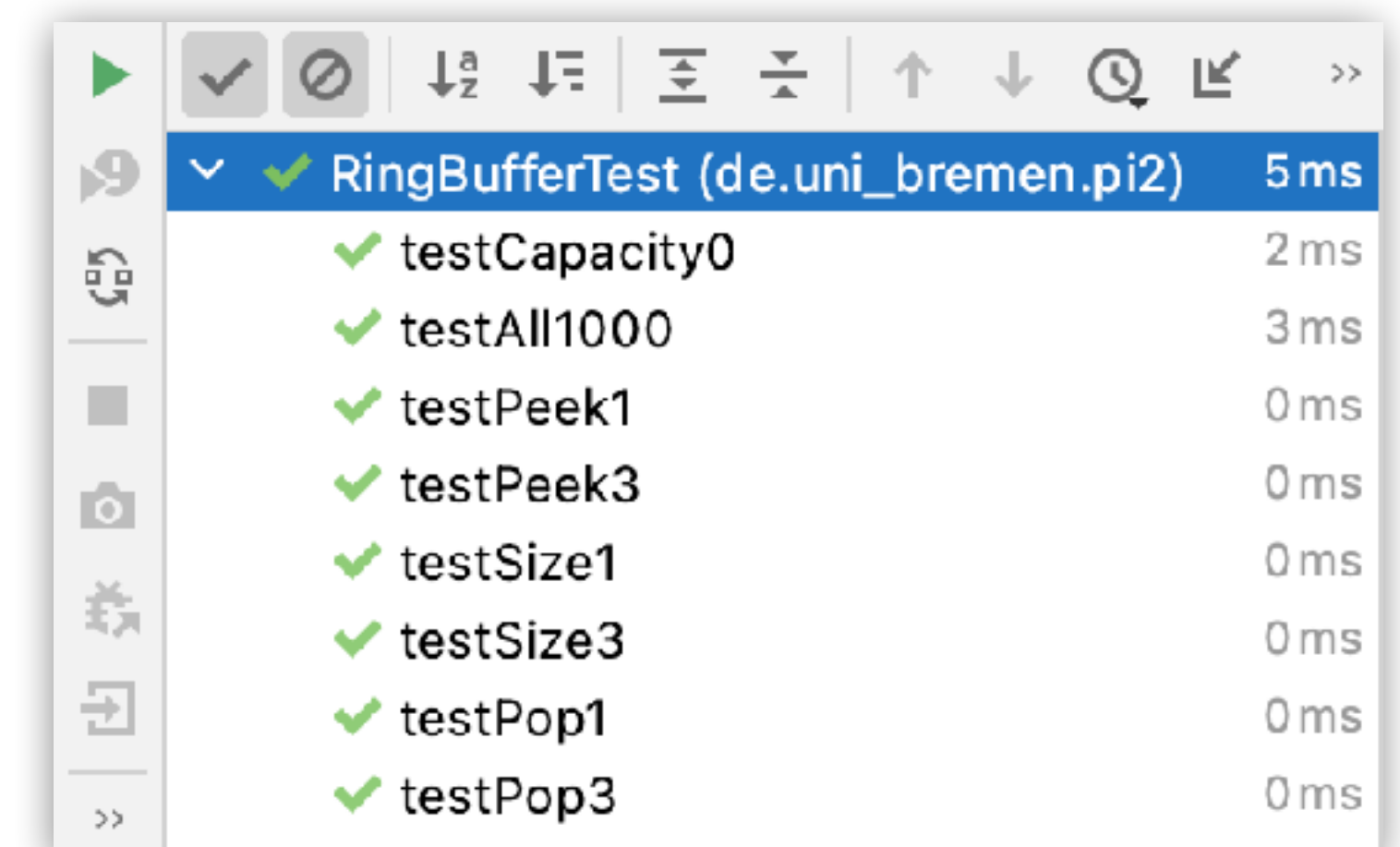
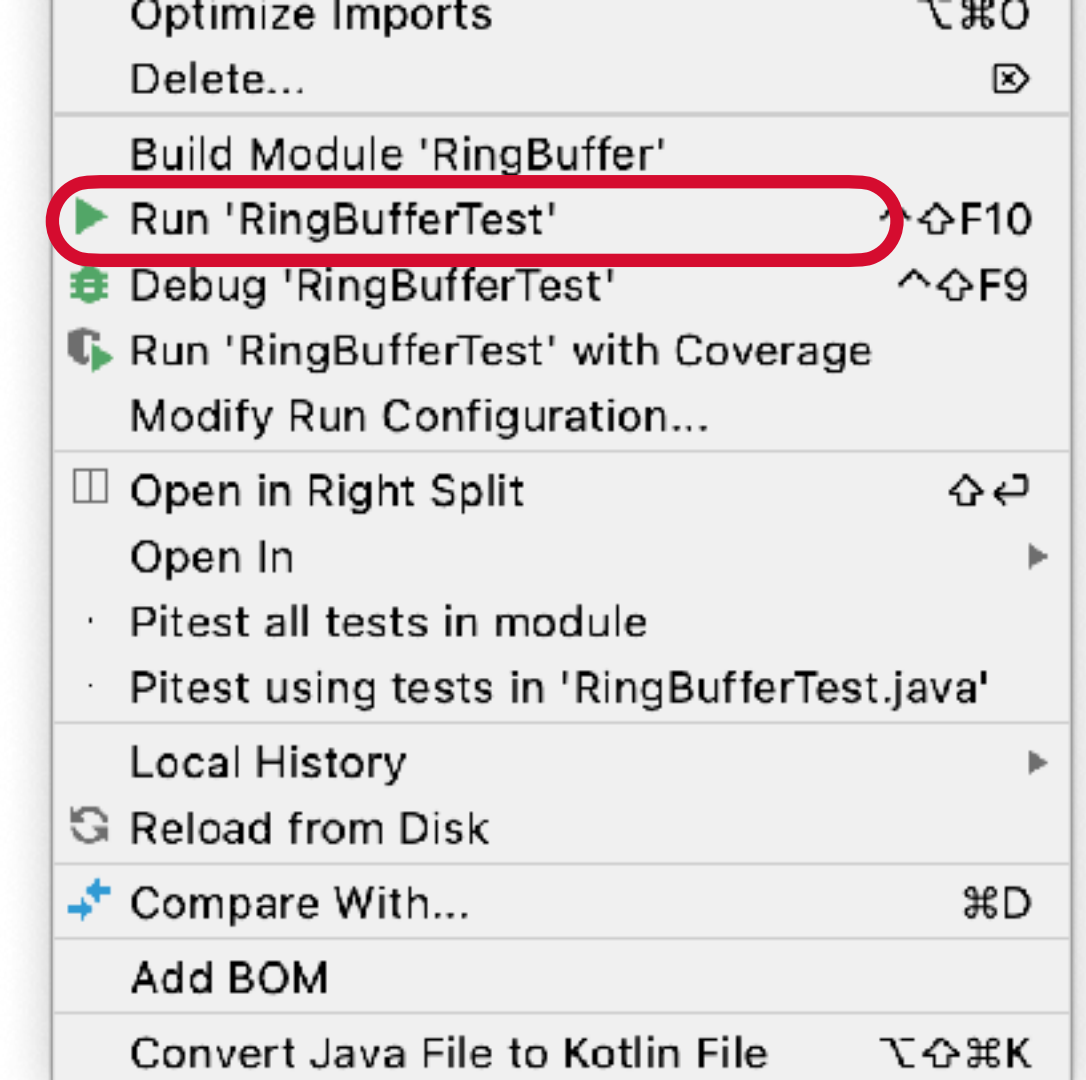
- Geleitet von Pascal Himmelmann (himmelmann.pascal@uni-bremen.de)
- Praktische Ergänzung zu PI2
- Gemeinsame Umsetzung von Aufgaben sowie individuelles Feedback
- Geeignet für Personen, die sich über PI2-Tutorien hinaus in vertiefter Form mit den Themen auseinandersetzen wollen
- Termin: Donnerstag um **16:00, MZH 4140** (Nicht jede Woche)
- Immer ein spezielles Thema pro Termin (Individuelle Teilnahme möglich)
- Ist auch insbesondere zur Klausurvorbereitung für PI2 gut geeignet :)

IntelliJ: Demo



Automatisiertes Prüfen von Testergebnissen

- Manuelles Überprüfen der Richtigkeit von Testergebnissen ist aufwändig und fehlerträchtig
- Daher sollten Tests **selbstüberprüfend** sein
- **JUnit**: Rahmenwerk zur Unterstützung von Modul- und Regressionstests in Java
- In **IntelliJ** integriert
- Wie in PI-1 nutzen wir **JUnit5**
 - Wenn also **JUnit** nachinstalliert werden muss, immer **JUnit5...** auswählen

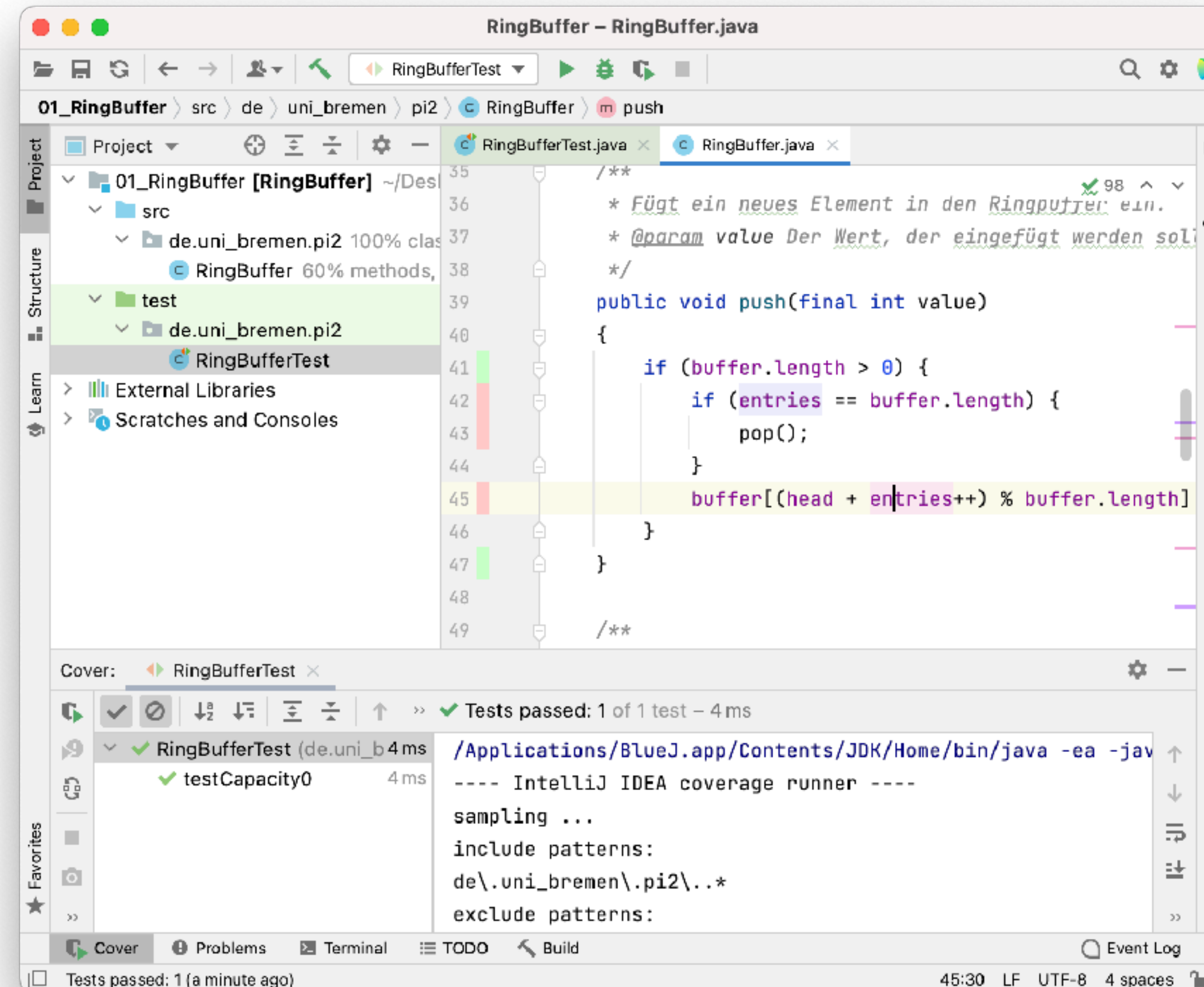


Teststärke: Testabdeckung

- Alle Tests laufen erfolgreich durch, aber testen sie auch alles?
- Es kann eine Statistik über den Code erstellt werden, der während der Tests tatsächlich ausgeführt wird (**Testabdeckung**, Ziel: 100%)
- **Methodenabdeckung**: Anteil der Methoden, die ausgeführt werden
- **Anweisungsabdeckung**: Anteil der Anweisungen (manchmal auch Zeilen), die ausgeführt werden (**Code Coverage** in **IntelliJ**)
- **Zweigabdeckung**: Anteil der Zweige (z.B. auch leere **else**-Zweige), die ausgeführt werden
- **Bedingungsabdeckung**: Anteil der Teilbedingungen, die sowohl **true** als auch **false** werden
- **Pfadabdeckung**: Anteil der möglichen Pfade (d.h. Abfolgen aus verschiedenen Zweigen), die durchlaufen werden

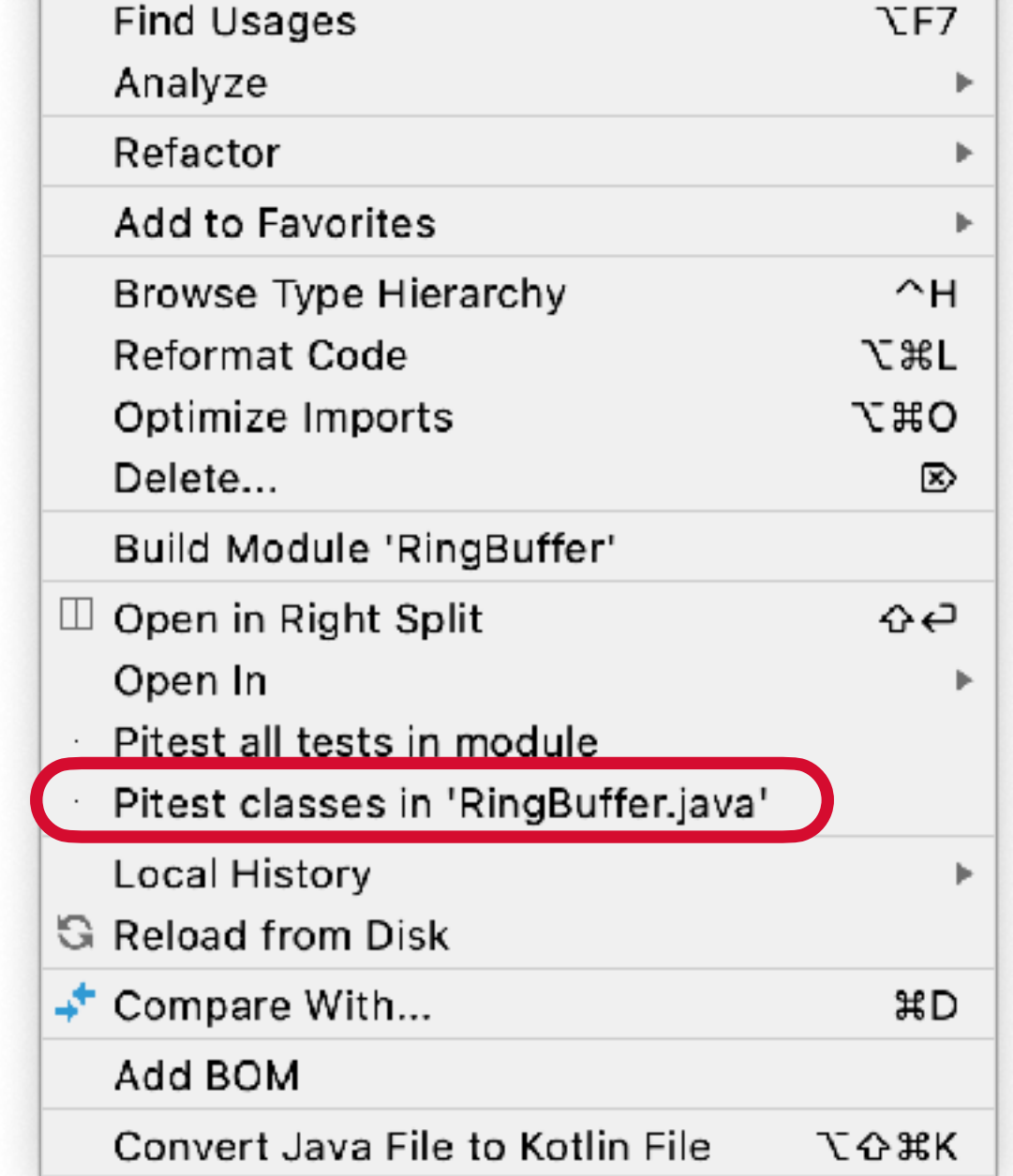


Code Coverage: Demo



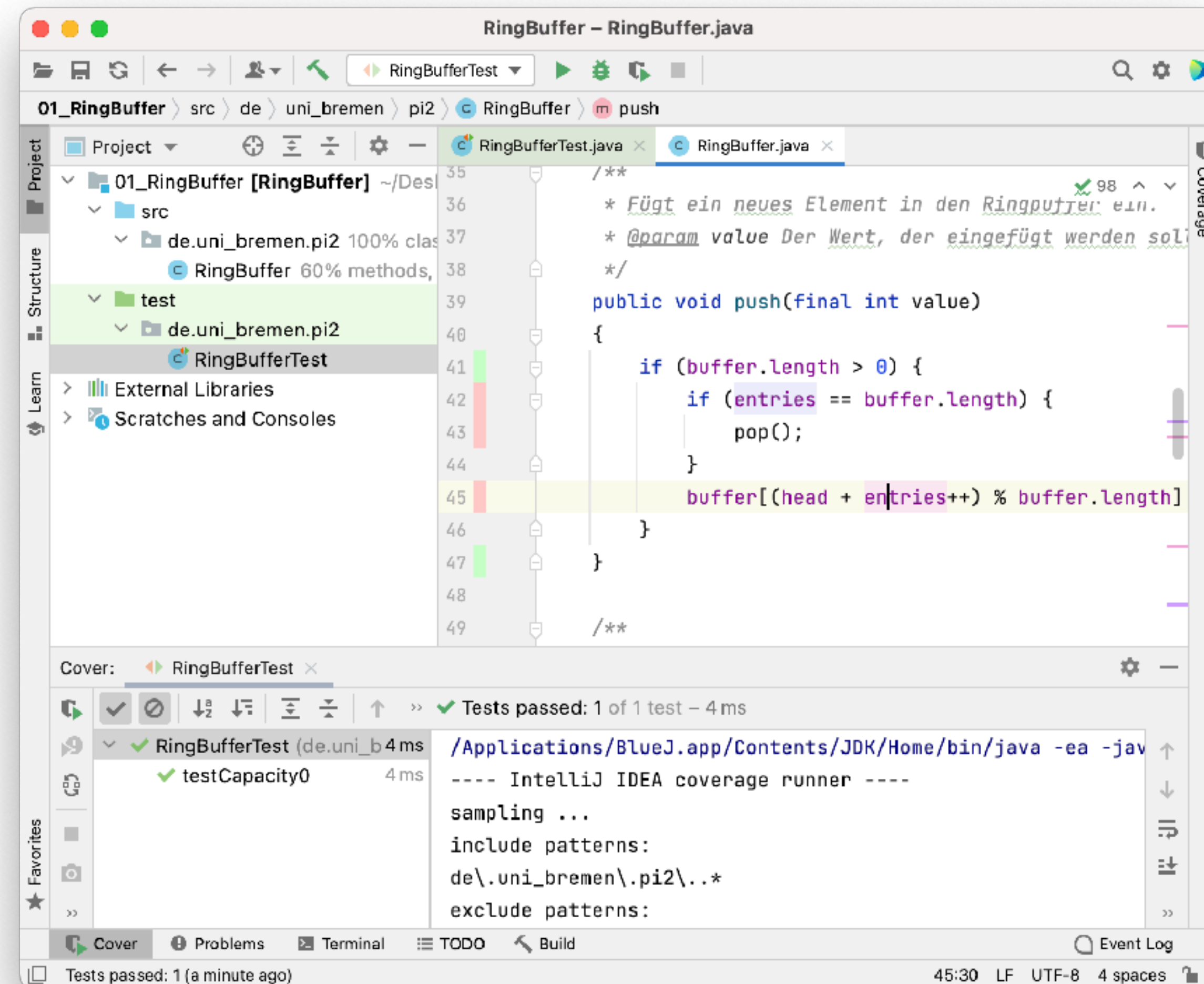
Teststärke: Mutationstests

- Mutationstests testen die Stärke von Tests
- Sie verändern den ursprünglich erfolgreich getesteten Code an einzelnen Stellen und erwarten, dass diese Änderungen durch die Tests als Fehler aufgedeckt werden
 - Allerdings macht nicht jede Änderung die Ursprungsimplementierung falsch
- Mutationen können z.B. sein:
 - Bedingungen auf **true** oder **false** setzen
 - Operatoren durch ähnliche ersetzen, z.B. **>** durch **>=** oder **+** durch **-**
 - Teile von Ausdrücken oder ganze Anweisungen weglassen



Mutationstests: Demo

Wenn Quelltext im Testprotokoll fehlt,
dann **File** → **Invalidate Caches** und
nochmal versuchen



Nächste Schritte

- **IntelliJ** installieren und sich einarbeiten
- Tutorium unter **Teilnehmende** → **Gruppen** wählen (bis Freitag, 14.04)
 - Wenn möglich, in Gruppe **Übernahme Übungsbetrieb 2022** eintragen
- Am Tutorium teilnehmen (ab Montag, 17.04)
 - Aufteilung in Gruppen
 - Gemeinsames **git**-Repository auf **GitLab** für Gruppe
 - Tutor:in dazu einladen
- Es gibt auch eine PI-2-L^AT_EX-Klasse