Universität
Bremen

Praktische Informatik 2
Spezifikation und Verifikation

Thomas Röfer

Cyber-Physical Systems
Deutsches Forschungszentrum für
Künstliche Intelligenz

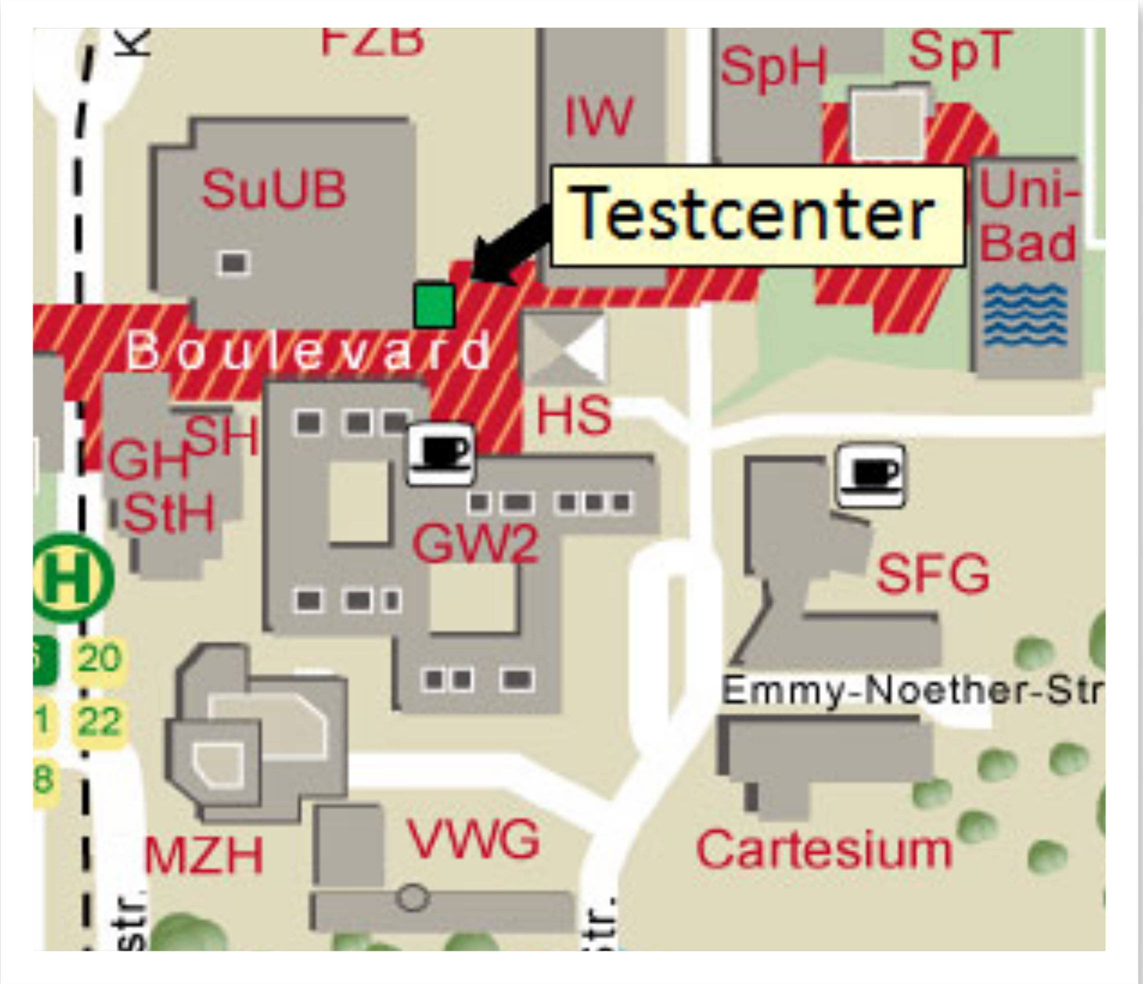
Multisensorische Interaktive Systeme
Fachbereich 3, Universität Bremen

Bild: ©Michael Ihle

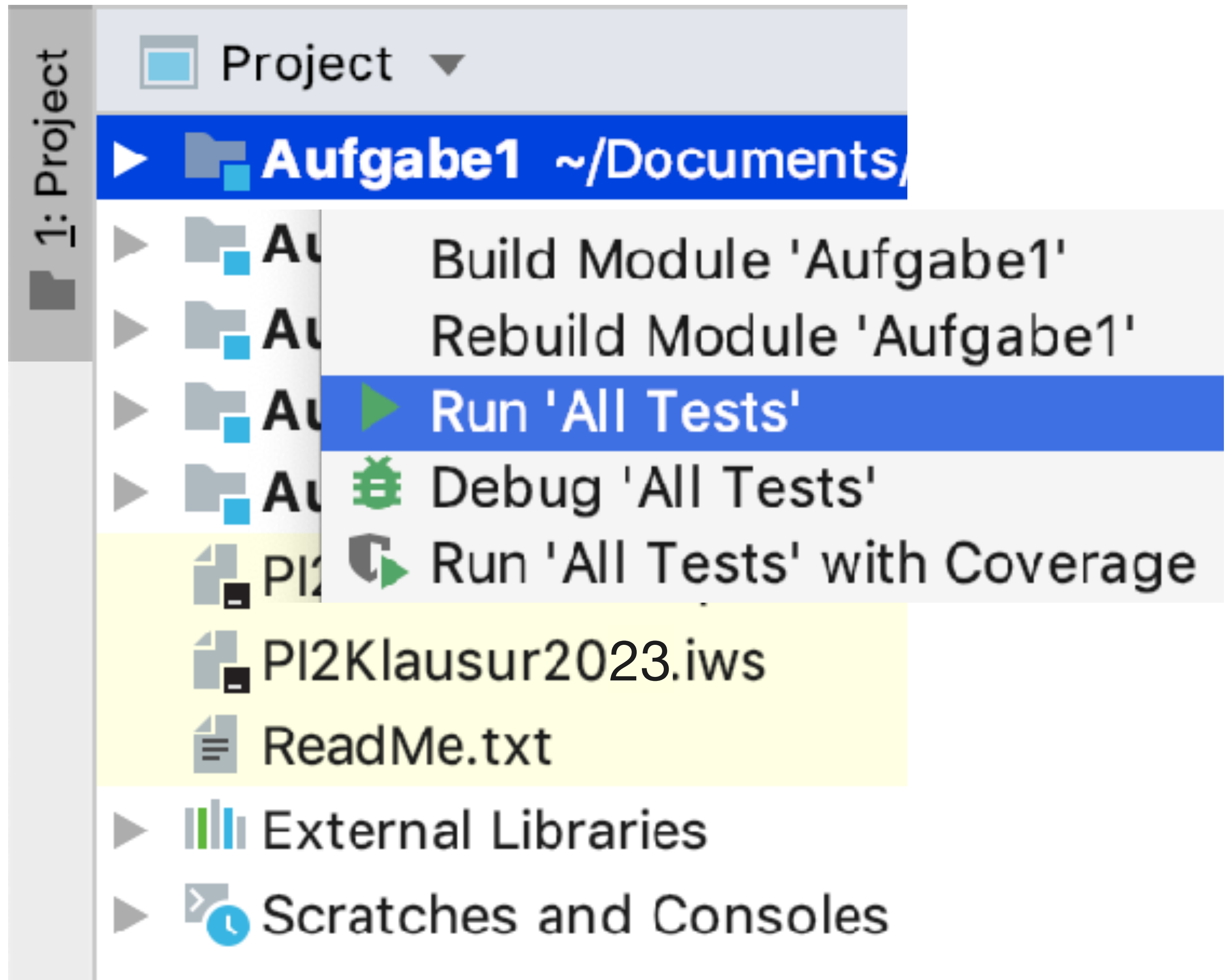
RoboCup 2023 SPL Finale

Universität Bremen

Klausur: Inhalt



- Klausur im Testcenter am 05.09.2023, Dauer 120 Minuten
- Anmeldung per Terminvergabe in Stud.IP (ihr müsst auch bei PABO angemeldet sein), wird in vorlesungsfreier Zeit freigeschaltet und per E-Mail angekündigt
- Fragenkatalog (20%): Themen quer durch das Semester
- Programmieraufgaben (5 x 20%)
  - Analoge, kürzere Varianten von (Teilen von) Aufgaben, die bereits auf Übungszetteln gestellt wurden
  - Fünf Aufgaben in einem IntelliJ-Projekt als Module
  - Zu jeder Aufgabe existieren bereits einige Tests



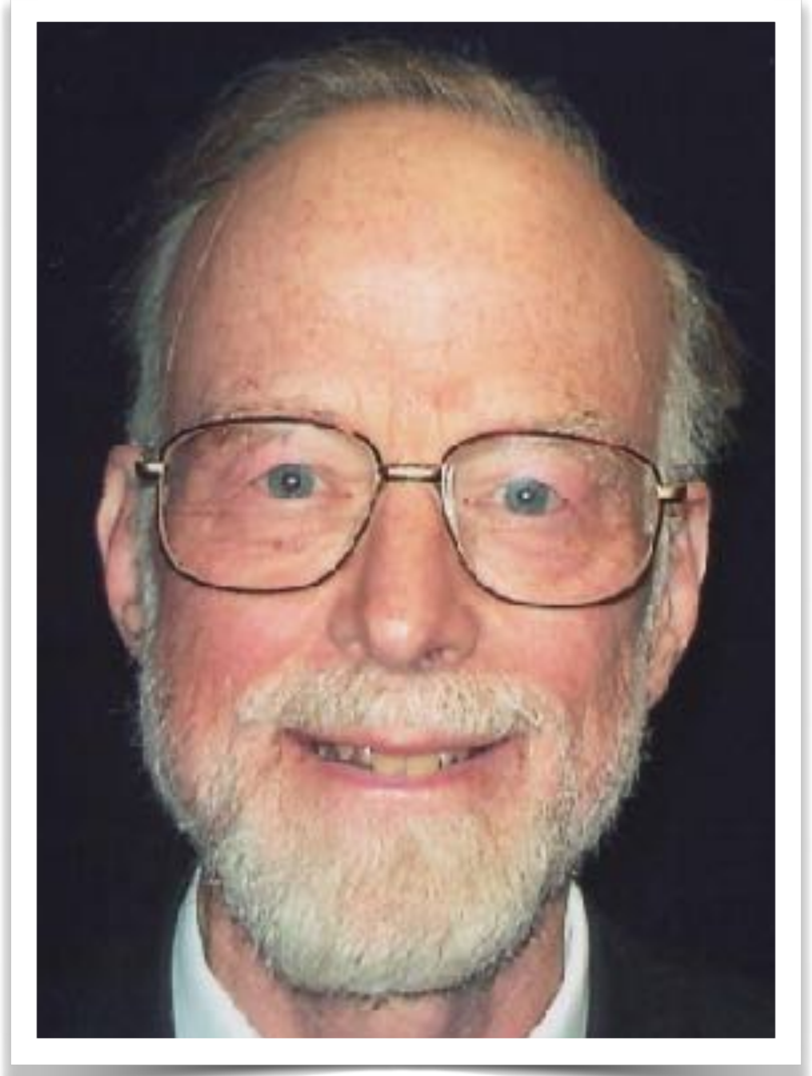
Universität Bremen

Klausur: Bewertung

- 50% zum Bestehen notwendig
- Programmieraufgaben werden automatisch getestet
- Nur jeweilige Hauptklasse einer Aufgabe wird in vorbereitete Umgebung kopiert, dort kompiliert und getestet
- Übersetzt Quelltext dabei nicht → 0% für diese Aufgabe
- Die Java-Laufzeitbibliothek darf nur verwendet werden, wenn explizit erlaubt (Ausnahme: Testausgaben mit **System.out.** sind ok)
- Testfälle sind andere als in der Klausur

Verifikation mit dem Hoare-Kalkül

- Formales System, um die Korrektheit von Programmen zu beweisen
- Hoare-Tripel: **{P} S {Q}**
  - **S**: Anweisung(en) der Programmiersprache (statement)
  - **P**: Vorbedingung (precondition)
  - **Q**: Nachbedingung (postcondition)
- **P** und **Q** sind logische Aussagen, die Bezeichner des Programms verwenden
- Hoare-Regeln definieren für verschiedene **S**, wie **P** und **Q** zueinander in Beziehung stehen

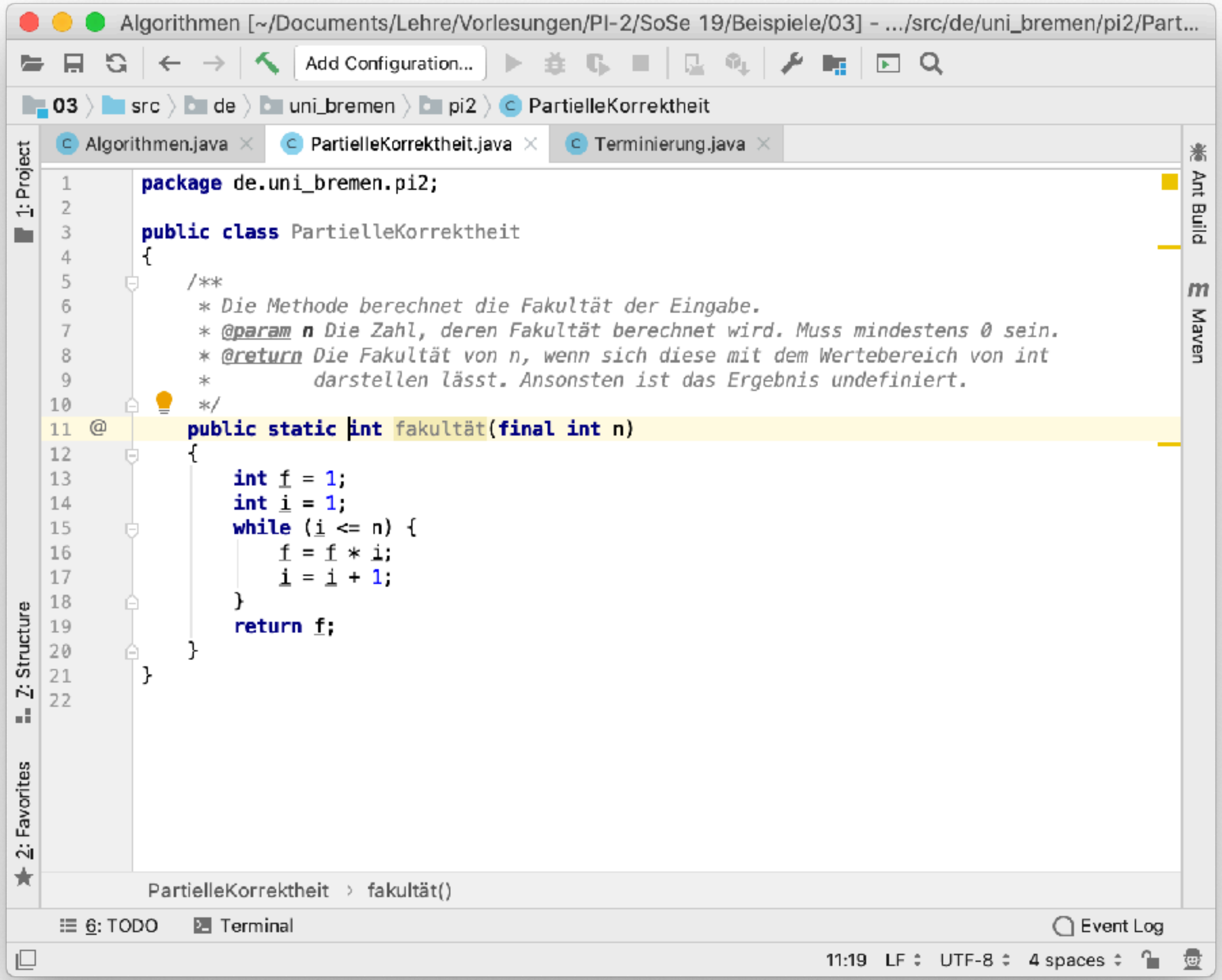


Tony Hoare

Verifikation mit dem Hoare-Kalkül: Grundidee

- Vor- und Nachbedingungen werden auch als Zusicherungen bezeichnet
- Die Vorbedingung der ersten Anweisung enthält üblicherweise Teile der Eingabespezifikation
- Die Nachbedingung der letzte Anweisung sollte die Ausgabespezifikation widerspiegeln
- Ein Beweis ist die Anwendung der Hoare-Regeln von der ersten Vorbedingung zur letzten Nachbedingung
- **Partielle Korrektheit** und **Terminierung** werden getrennt bewiesen

Beweis der partiellen Korrektheit: Demo



Universität Bremen

Hoare-Kalkül: Axiom der leeren Anweisung

- Generelle Notation: Wenn Ausdrücke über dem „Bruchstrich“ gelten (d.h. bewiesen wurden), dann gilt das Hoare-Tripel unter dem Bruchstrich

- Wenn eine Anweisung keine Variablen verändert, gelten nach der Anweisung dieselben Zusicherungen wie davor

  - $\overline{\{P\}S\{P\}}$

- Dies gilt auch, wenn **S** zwar Variablen verändert, diese aber nicht in **P** vorkommen

Hoare-Kalkül: Zuweisungsaxiom

- Nach einer Zuweisung gilt für die geänderte Variable, was vorher für die rechte Seite der Zuweisung galt
  - $\overline{\{P[E/x]\}x := E\{P\}}$
  - **P[E/x]**: In **P** wurde jedes freie Vorkommen von **x** durch **E** ersetzt
- Beispiel: **{ i+1 ≤ n+1 } i := i+1 { i ≤ n+1 }**
- Diese Regel erfordert Vorbereitung, damit sie überhaupt anwendbar ist
  - Die Vorbedingung muss in eine geeignete Form gebracht werden

Universität Bremen

Hoare-Kalkül: Kompositions- und Sequenzregel

- Um sequenzielle Programme zu beweisen, muss die Nachbedingung einer Anweisung auch die Vorbedingung der nächsten sein
  - $$\frac{\{P\}S\{R\}, \{R\}T\{Q\}}{\{P\}S; T\{Q\}}$$
- Beispiel
  - **{ i+1 ≤ n+1 ∧ f · i = (i+1-1)! }**
    f := f * i
    { i+1 ≤ n+1 ∧ f = (i+1-1)! } Gleichzeitig Vor- und Nachbedingung!
    i := i + 1
    { i ≤ n+1 ∧ f = (i-1)! }

Hoare-Kalkül: Konsequenzregel

- Vorbedingungen dürfen verstärkt werden, z.B. durch Verschärfung bestehender Aussagen oder Ergänzung zusätzlicher, wahrer Bedingungen

- Nachbedingungen dürfen abgeschwächt werden, z.B. durch Abschwächen bestehender Aussagen und Entfernen von Bedingungen

  - $$\frac{P \Rightarrow P', \{P'\}S\{Q'\}, Q' \Rightarrow Q}{\{P\}S\{Q\}}$$

- Beispiel: **{true} x := 0 {x ≥ 0}**, da $\dfrac{true \Rightarrow 0 = 0, \{0 = 0\}x := 0\{x = 0\}, x = 0 \Rightarrow x \geq 0}{\{true\}x := 0\{x \geq 0\}}$

- In Vor- und Nachbedingungen dürfen außerdem beliebige Umformungen vorgenommen werden, solange die Aussagen äquivalent bleiben

Hoare-Kalkül: Auswahlregel

- Für Verzweigungen mit 2 Alternativen muss die Nachbedingung für beide Alternativen gelten
  - $$\frac{\{P \wedge B\}S\{Q\},\ \{P \wedge \neg B\}T\{Q\}}{\{P\}\text{if } B \text{ then } S \text{ else } T\{Q\}}$$
  - Zusätzlich zur Vorbedingung gilt zu Beginn des **then**-Zweiges die **if**-Bedingung, zu Beginn des **else**-Zweiges das Gegenteil
- Beispiel: **{0 ≤ x ≤ 3} if x < 3**
  then {0 ≤ x ≤ 3 ∧ x < 3} {0 ≤ x < 3} {1 ≤ x+1 < 4} x := x+1 {1 ≤ x < 4} **{0 ≤ x ≤ 3}**
  else {0 ≤ x ≤ 3 ∧ ¬(x < 3)} {x = 3} { } {0 = 0} x := 0 {x = 0} **{0 ≤ x ≤ 3}**
- Fehlt der **else**-Zweig, muss sich **{Q}** aus **{P ∧ ¬B}** ergeben (**T** ist leere Anweisung)

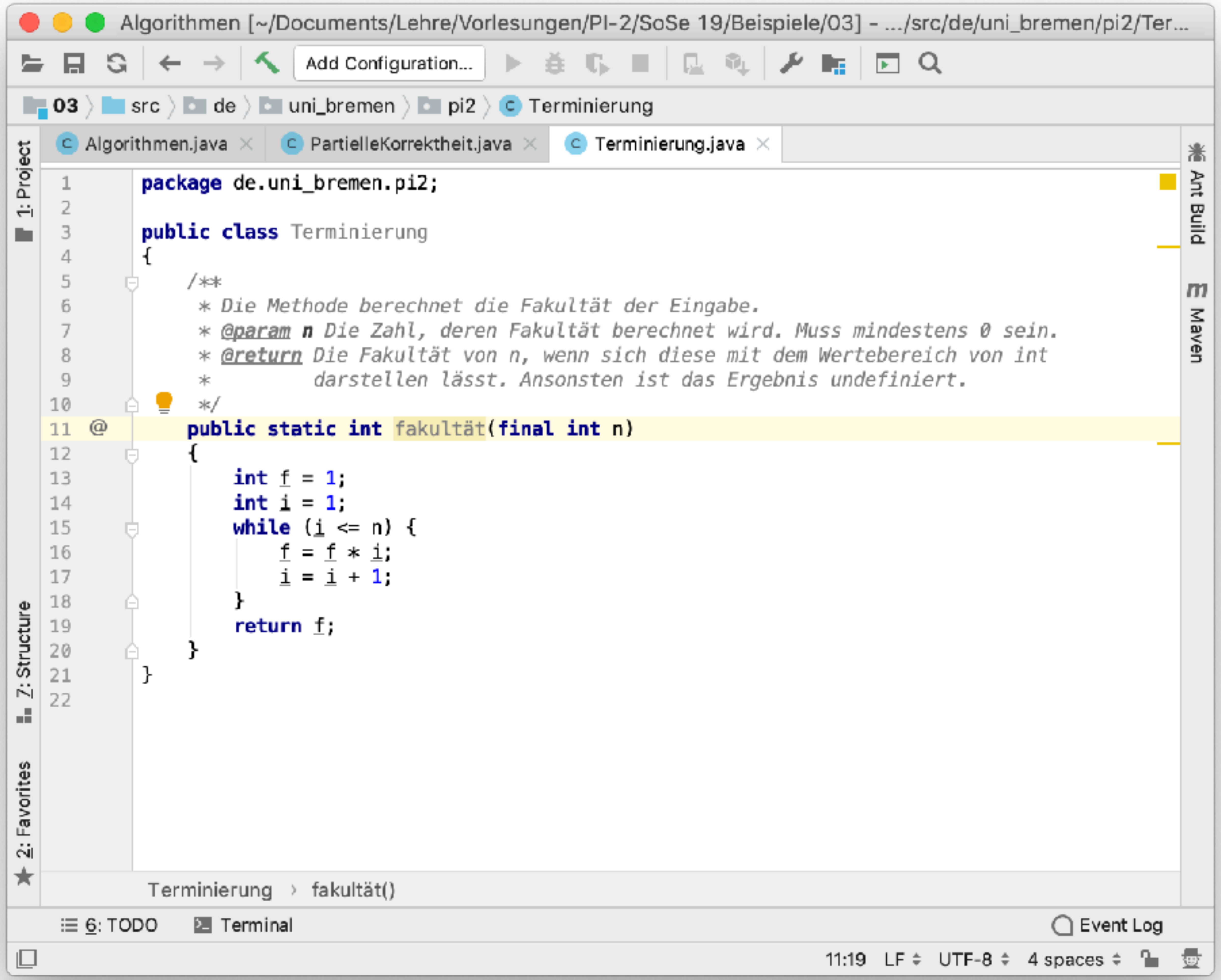
Universität Bremen

Hoare-Kalkül: Iterationsregel

- Zum Korrektheitsbeweis einer Schleife wird eine **Schleifeninvariante** definiert, die vor der Schleife, am Anfang und Ende des Schleifenrumpfes und nach der Schleife gelten muss
  - $$\frac{\{I \land B\}S\{I\}}{\{I\}\text{while } B \text{ do } S \text{ done}\{I \land \neg B\}}$$
- Zu zeigen ist, dass sie am Ende des Rumpfes gilt, wenn sie am Anfang galt
- Die Invariante muss manuell gewählt werden und beschreibt üblicherweise die Grenzen von Laufvariablen und wieviel des Ergebnisses bereits aufgebaut wurde
- Beispiel: **{i ≤ n + 1 ∧ f = (i-1)!} while i <= n do**
  {i ≤ n + 1 ∧ f = (i-1)! ∧ i ≤ n} f := f*i; i := i+1 {i ≤ n + 1 ∧ f = (i-1)!} done
  {i ≤ n + 1 ∧ f = (i-1)! ∧ ¬(i ≤ n)}

Universität Bremen

Beweis der Terminierung: Demo



Hoare-Kalkül: Terminierung

- Für den Beweis der Terminierung einer Schleife wird ein Term **$t \geq 0$** definiert (**Schleifenvariante**), der die Anzahl der noch zu erwartenden Schleifendurchläufe abschätzt

- Dieser Term muss am Ende des Schleifenrumpfes kleiner als am Anfang sein
  - $$\frac{\{I \land B \land t = z\}S\{I \land t < z\}, I \Rightarrow t \geq 0}{\{I\}\text{while } B \text{ do } S \text{ done}\{I \land \neg B\}}$$

- Um zu zeigen, dass er am Ende des Rumpfes kleiner ist, wird sein Wert zu Beginn des Rumpfes in einer ansonsten nicht verwendeten Variable **z** gespeichert

- Beispiel: **t = n+1-i**
  {n+1-i = z ∧ i <= n} {n-i < z} f = f*i {n+1-(i+1) < z} i = i+1 {n+1-i < z}

Zusammenfassung der Konzepte

- **Hoare-Kalkül**
- **Vorbedingung** und **Nachbedingung**
- **Partielle Korrektheit** und **Terminierung**