Übungen zur Vorlesung

Algorithmen und Datenstrukturen

Übungsblatt 00

Cryptoplexity
Cryptography & Complexity Theory
Technische Universität Darmstadt
www.cryptoplexity.de

Arbeitsgruppe Kryptographie und Komplexitätstheorie
Prof. Dr. Marc Fischlin
Moritz Huppert
Tobias Schmalz

Sommersemester 2024
Veröffentlicht: 12.04.2024, 14:00 Uhr MESZ
Abgabe: 26.04.2024, 23:59 Uhr MESZ

Hausübungen

In diesem Bereich finden Sie die praktische Hausübung von Blatt 00. Bitte beachten Sie die allgemeinen Hinweise zu den Hausübungen und deren Abgabe im Moodle-Kurs.

Bitte reichen Sie Ihre Abgabe bis spätestens *Freitag, 26.04.2024, 23:59 Uhr MESZ* ein. Verspätete Abgaben können *nicht* berücksichtigt werden.

H1 Matrikelnummer in Moodle

Bitte tragen Sie Ihre Matrikelnummer in Ihrem Moodle-Profil im Feld „ID-Nummer“ ein. Vergewissern Sie sich, dass Sie die Matrikelnummer korrekt eingetragen haben und nicht etwa aus Versehen die TU-ID, da sich das Feld im Nachhinein nicht durch Sie ändern lässt. Sollte in dem Feld nicht Ihre Matrikelnummer stehen, müssen Sie die Administratoren des Lernportals kontaktieren. Zum Kontaktformular kommen Sie über den Fragezeichen-Button unten rechts auf den Seiten des Lernportals.

H2 Java IDE

Zum Bearbeiten unserer Hausübungen benötigen Sie eine Java-Entwicklungsumgebung. Die Verwendung von IntelliJ wird von uns *dringend* empfohlen und unterstützt.

Im Studierenden-Guide finden Sie im Abschnitt *Vorbereitung → Installieren von Java* und *Vorbereitung → Installieren von IntelliJ* Guides zur Installation von Java bzw. IntelliJ.

H3 Vorlagen für die praktischen Hausübungen

In der AuD stellen wir Ihnen für jede praktische Hausübung eine Vorlage bereit, die zur Bearbeitung der jeweiligen Hausübung verwendet werden *muss*.

Beachten Sie weiter die für jedes Übungsblatt relevante Seite *Verbindliche Anforderungen für alle Abgaben*, sowie die mitgelieferten Hinweise.

In den Vorlagen sind Klassen- / Methodenköpfe und ähnliches bereits vorgegeben. Sie müssen also nur in den mit „`// TODO`“ markierten Bereichen Ihre Anweisungen schreiben und gegebenenfalls Aufrufe von `crash()` entfernen. Sie dürfen natürlich weitere Klassen, Methoden, etc. hinzufügen, sofern die verbindlichen Anforderungen eingehalten werden.

H4 Bearbeiten der praktischen Hausübungen

Die praktischen Hausübungen sind von Ihnen alleine zu lösen. Gruppenabgaben sind nicht zulässig und werden von uns als Plagiat gewertet. Wir verfolgen „Abschreiben“ und andere Arten von Täuschungsversuchen! Disziplinarische Maßnahmen treffen nicht nur die, die abschreiben, sondern auch die, die abschreiben lassen.

Den Prozess zum Exportieren und Hochladen der praktischen Hausübungen finden Sie auf den Seiten des Studierenden-Guides im Abschnitt Hausübungen → Exportieren und Hochladen.

H5 Beispielaufgabe - Fibonacci-Zahlen berechnen (0 Punkte)

Die Fibonacci-Folge ist eine unendliche Folge natürlicher Zahlen. Die n-te Fibonacci-Zahl f_n ist definiert durch: $f_n = f_{n-1} + f_{n-2}$ für $n \geq 3$. Für $n = 1$ und $n = 2$ gilt $f_n = 1$. Die n-te Zahl der Folge ist also die Summe der zwei vorherigen Zahlen: $1, 1, 2, 3, 5, 8, 13, \ldots$

Im Folgenden implementieren Sie nun zwei Ansätze, um eine beliebige Fibonacci-Zahl zu berechnen: iterativ und rekursiv.

Die Klassen `IterativeFibonacciCalculator` und `RecursiveFibonacciCalculator` implementieren beide das Interface `FibonacciCalculator`. Dieses Interface hat eine Methode `get(int)`, die benutzt wird, um die n-te Fibonacci Zahl zu berechnen. Ihre Aufgabe ist nun diese Methode in den oben genannten Klassen zu implementieren. Ihre Implementation von `get(int)` in `IterativeFibonacciCalculator` darf dabei nur rein iterativ arbeiten, das heißt, Rekursion ist nicht erlaubt. Bei `RecursiveFibonacciCalculator` ist es genau anders herum: `get(int)` darf keine Schleifen verwenden, sondern muss rein rekursiv arbeiten. Sie dürfen die Rekursion allerdings in eine Hilfsmethode auslagern.