



Im Rahmen dieses Praktikums werden die in der Vorlesung durchgenommenen Hashing Verfahren implementiert. Ziel ist es, die in der Vorlesung vermittelten Kenntnisse zu vertiefen und praktisch anzuwenden. Als Programmiersprache kommt Java zum Einsatz. Alle Algorithmen sind im Rahmen der Vorlesung besprochen worden. Den zugehörigen Pseudocode findet man in den Vorlesungsunterlagen.

Auf dem Vorlesungsserver finden Sie ein ZIP Archiv mit dem Java Quellcode zur Bearbeitung dieses Praktikums. Das Archiv enthält folgende Klassen:

Name	Paket	Beschreibung
HashTableBase	hashing	Basisklasse für Hashtabellen
HashEntry	hashing	Interface für Einträge von Hashtabellen
PhonebookEntry	hashing	Klasse zum Speichern von Namen mit zugehörigen Telefonnummern
ChainLink	hashing	Glied einer verketteten Liste
HashChain	hashing	Verkettete Liste für den Einsatz von Hashing mit Chaining
HashTableChaining	hashing	Hashtabelle mit Verkettung
HashTableOpenAddressing	hashing	Hashtabelle mit Open Addressing
HashFunction	hashing	Interface für Hashfunktionen
DivisionMethod	hashing	Hashfunktion auf Basis der Divisionsmethode
MultiplicationMethod	hashing	Hashfunktion auf Basis der Multiplikationsmethode
OpenAddressingHashFunction	hashing	Interface für Hashfunktionen mit Open Addressing
LinearProbing	hashing	Hashfunktion auf Basis der Linear Probing Technik
QuadraticProbing	hashing	Hashfunktion auf Basis der Quadratic Probing Technik
DoubleHashing	hashing	Hashfunktion auf Basis der Double Hashing Technik

Viele dieser Klassen enthalten Methoden, die im Rahmen dieses Praktikums vervollständigt werden.

Aufgabe 1. Die Klasse `HashTableBase` ist eine Basisklasse für Hashtabellen. Sie stellt abstrakte Methoden zum Einfügen, Suchen und Löschen eines Elements bereit. Voraussetzung ist, dass das einzufügende Element das Interface `HashEntry` implementiert.

- a) Welche Attribute enthält `HashTableBase`?
- b) Welche Methoden stellt `HashTableBase` bereit? Welche der Methoden sind abstrakt?

Aufgabe 2. Im folgenden wird eine Hashtabelle eingesetzt, um Personen mit zugehörigen Telefonnummern abzuspeichern. Hierzu steht die Klasse `PhonebookEntry` bereit.

- a) Implementieren Sie in der Klasse `PhonebookEntry` das Interface `HashEntry`. Als Schlüssel soll der Inhalt der Variable `name` verwendet werden.
- b) Erzeugen Sie `PhonebookEntry` Instanzen für die folgenden Telefonnummern:

<i>Name</i>	<i>Telefonnummer</i>
Winfried Bantel	07361/576-4235
Roland Dietrich	07361/576-4140
Roland Hellmann	07361/576-4138
Christoph Karg	07361/576-4205
Ulrich Klauck	07361/576-4184
Detlef Küpper	07361/576-4375
Manfred Rössle	07361/576-4377

Wie lauten die Schlüssel der jeweiligen Instanzen?

Aufgabe 3. Im Rahmen dieser Aufgabe wird eine verkettete Liste implementiert, die bei Hashing mit Chaining zum Einsatz kommt. Die Glieder der Liste werden in Instanzen der Klasse `ChainLink` abgelegt. Die verkettete Liste selbst wird durch die Klasse `HashChain` realisiert.

- a) Implementieren Sie die Methode `insert()`, die ein Element an den Anfang der Liste hängt.
- b) Implementieren Sie die Methode `search()` zum Suchen eines Elements mit passendem Schlüssel. Falls das Element nicht in der Liste enthalten ist, dann soll `null` zurückgegeben werden.
- c) Implementieren Sie die Methode `delete()` zum Löschen eines Eintrags der Hashtabelle.
- d) Testen Sie Ihre Implementierung mit den Daten von Aufgabe 2.

Aufgabe 4. In den folgenden drei Aufgaben wird eine Hashtabelle mit Chaining realisiert. Die einzusetzenden Hashfunktionen müssen das Interface `HashFunction` bereitstellen. Implementieren Sie in der Klasse `DivisionMethod` eine Hashfunktion auf Basis der Divisionsmethode.

Hinweis: Um einen String in einen `int` zu konvertieren, bietet sich die Methode `hashCode()` der String Klasse an. Achten Sie darauf, dass sie keine negativen Zahlen verwenden.

Aufgabe 5. Die Klasse `HashTableChaining` enthält alle Attribute und Methoden für eine Hashtabelle mit Verkettung. Die einzusetzende Hashfunktion wird über den Konstruktor übergeben.

- a) Implementieren Sie die Methode `insert()`.
- b) Implementieren Sie die Methode `search()`.
- c) Implementieren Sie die Methode `delete()`.
- d) Testen Sie Ihre Implementierung mit den Daten von Aufgabe 2 unter Verwendung der Divisionsmethode ($m = 13$).

Hinweis: Mit der Methode `print()` kann man den Inhalt der Hashtabelle ausgeben.

Aufgabe 6. Implementieren Sie in der Klasse `MultiplicationMethod` eine Hashfunktion auf Basis der Multiplikationsmethode. Testen Sie die Hashfunktion zusammen mit der Klasse `HashTableChaining` und den Daten von Aufgabe 2. Verwenden Sie hierzu eine Hashtabelle mit $m = 32$ Slots und den Wert $A = (\pi - 3)/2$.

Aufgabe 7. Der letzte Teil des Praktikums ist dem Hashing mit Open Addressing gewidmet. Als Hashfunktionen kommen Klassen zum Einsatz, deren Funktionalität über das Interface `OpenAdressingHashFunction` festgelegt ist.

- a) Implementieren Sie in der Klasse `LinearProbing` eine Hashfunktion gemäß dem Linear Probing Verfahren.
- b) Welche Sondierungssequenz durchläuft die Hashfunktion für den Telefonbuch Eintrag (Name: "Carsten Lecon", Nummer: "07361/576-4365"), wenn eine Hashtabelle der Größe $m = 13$ eingesetzt wird?

Aufgabe 8. Die Klasse `HashTableOpenAddressing` dient der Implementierung einer Hashtabelle mit Open Addressing. Der Aufbau dieser Klasse ist analog zu dem der Klasse `HashTableChaining`.

- a) Implementieren Sie die Methode `insert()`.
- b) Implementieren Sie die Methode `search()`.

- c) Implementieren Sie die Methode `delete()`.
- d) Testen Sie Ihre Implementierung mit den Daten von Aufgabe 2 unter Einsatz von Linear Probing ($m = 13$).
Hinweis: Mit der Methode `print()` kann man den Inhalt der Hashtabelle ausgeben.

Aufgabe 9. Ziel dieser Aufgabe ist die Implementierung einer Hashfunktion mit Quadratic Probing.

- a) Implementieren Sie in der Klasse `QuadraticProbing` eine Hashfunktion gemäß dem Quadratic Probing Verfahren.
- b) Welche Sondierungssequenz durchläuft die Hashfunktion für den Telefonbuch Eintrag (Name: "Carsten Lecon", Nummer: "07361/576-4365"), wenn eine Hashtabelle der Größe $m = 16$ und Quadratic Probing mit $c_1 = \frac{1}{2}$ und $c_2 = \frac{1}{2}$ eingesetzt wird?
- c) Testen Sie die Klasse mit `HashTableOpenAddressing` und den Daten von Aufgabe 2.

Aufgabe 10. Die letzte Aufgabe beschäftigt sich mit Double Hashing.

- a) Implementieren Sie in der Klasse `DoubleHashing` eine Hashfunktion gemäß des Double Hashing Verfahrens. Implementieren Sie unter Einsatz dieser Klasse sowohl Methode 1 als auch Methode 2 des Double Hashing. Erstellen Sie hierzu (falls notwendig) passende Klassen auf Basis des `HashFunction` Interfaces.
- b) Welche Sondierungssequenz durchläuft die Hashfunktion für den Telefonbuch Eintrag (Name: "Carsten Lecon", Nummer: "07361/576-4365"), wenn eine Hashtabelle der Größe $m = 19$ und Double Hashing mit den Hashfunktionen $h_1(k) = k \bmod 19$ und $h_2(k) = 1 + (k \bmod 18)$ eingesetzt wird?
- c) Welche Sondierungssequenz durchläuft die Hashfunktion für den Telefonbuch Eintrag (Name: "Carsten Lecon", Nummer: "07361/576-4365"), wenn eine Hashtabelle der Größe $m = 32$ und Double Hashing mit den Hashfunktionen $h_1(k) = k \bmod 32$ und $h_2(k) = (2k + 1) \bmod 32$ eingesetzt wird?
- d) Testen Sie die Klasse mit `HashTableOpenAddressing` und den Daten von Aufgabe 2.