Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Programmiervorkurs
Wintersemester 20/21

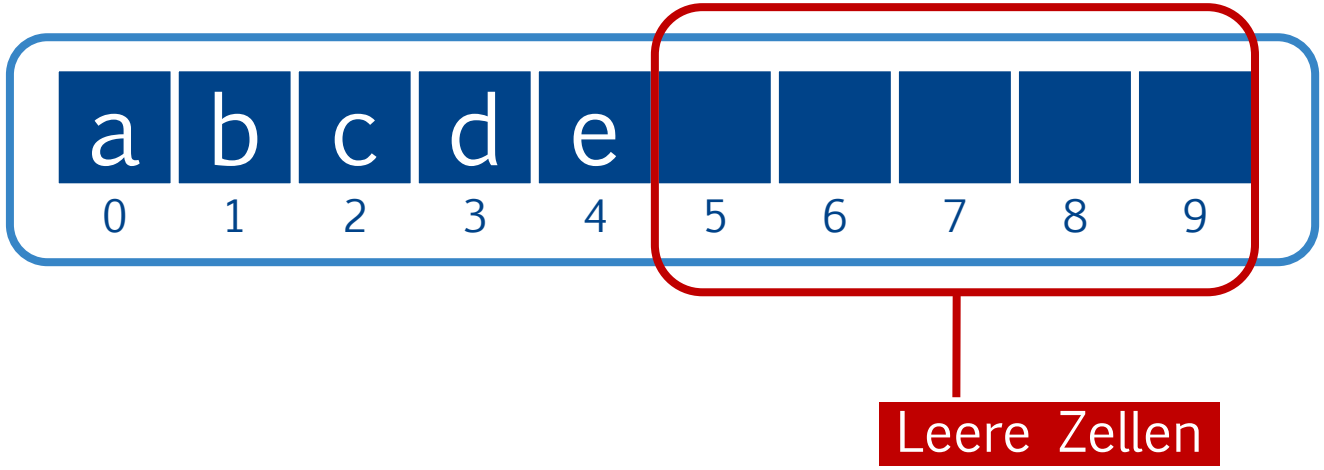
Tag 7

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

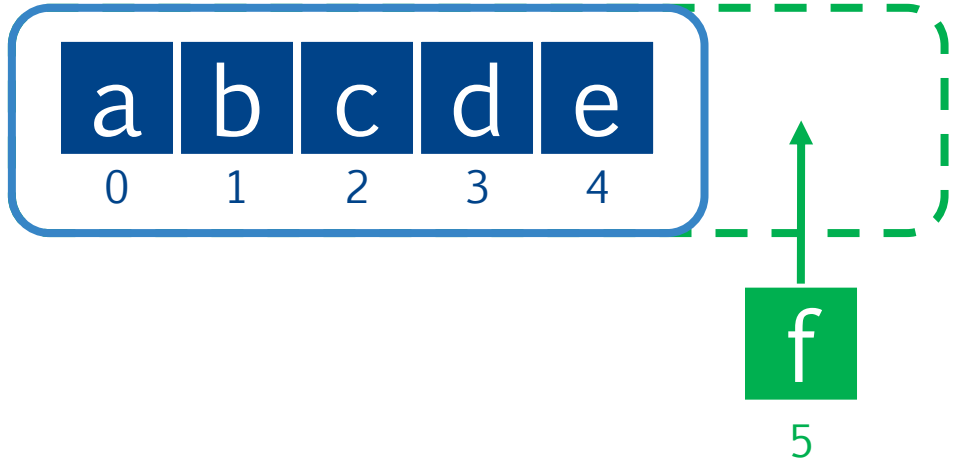
Collections

Array oder Listen?

› Array: feste Größe



› Listen:
können dynamisch
erweitert werden



ArrayList vs LinkedList

- ArrayList:
  - Speichert die Elemente intern in einem Array
  - Zugriff auf spezielle Elemente (Positionen) sehr schnell
  - Einfügen und Löschen in der Listenmitte sehr aufwendig

- LinkedList:
  - Speichert die Elemente in einer verketteten Liste
  - Für Zugriffe auf bestimmte Positionen muss Liste durchsucht werden
  - Schnelles Löschen und Hinzufügen von Elementen

- Mehr Infos:
- https://stackoverflow.com/questions/322715/when-to-use-linkedlist-over-arraylist-in-java

Umgang mit ArrayLists I

Deklaration

ArrayList <Typ> Bezeichner = new ArrayList <Typ> ();

z.B.
```
ArrayList<String> liste = new ArrayList<String>();
```

Werte hinzufügen (add)

z.B.
```
liste.add("a");
```

Werte auf einen vorgegebenen Index setzen (set)

z.B.
```
liste.set( 0 , "z");
```

Gibt den Index an

Werte entfernen (remove)

z.B.
```
liste.remove( 0 );

liste.remove( "b" );
```

Gibt den zu entfernenden Wert an

Umgang mit ArrayLists II

Größe der Liste abfragen (size)

z.B.
```
liste.size();
```

Prüfen, ob Liste bestimmten Wert enthält (contains)

z.B.
```
liste.contains(„a");
```

Werte mittels Index auslesen (get)

z.B.
```
liste.get(1);
```

Ausgabe mittels for-Schleife und Index

z.B.
```
for (int i = 0; i < liste.size(); i++) {
      System.out.println(liste.get(i));
}
```

Aufgabe: Einkaufsliste

- Für eure WG wollt ihr eine interaktive Einkaufsliste erstellen.
- Schreibt dazu ein Programm, dass den Nutzer zur Eingabe von beliebig vielen Artikeln auffordert. Diese sollen in einer ArrayList gespeichert werden und die Liste anschließend ausgegeben werden.
- Bietet außerdem die Möglichkeit abzufragen, ob ein bestimmter Artikel schon auf der Liste vorhanden ist und diesen gegebenenfalls noch hinzuzufügen.
- Wenn ihr vom Einkaufen zurück kommt, muss es natürlich eine Möglichkeit geben, Artikel von der Liste zu löschen.

Exkurs Set

- Funktionalität zur Abfrage, ob Artikel bereits vorhanden ist musste aufwändig von Hand programmiert werden

- Es gibt Datenstrukturen, in denen jedes Element nur einmal vorkommen kann

  → bringen diese Funktionalität automatisch mit

- Mengen oder auch Sets: Ungeordnete Sammlung von Elementen

- Deklaration:

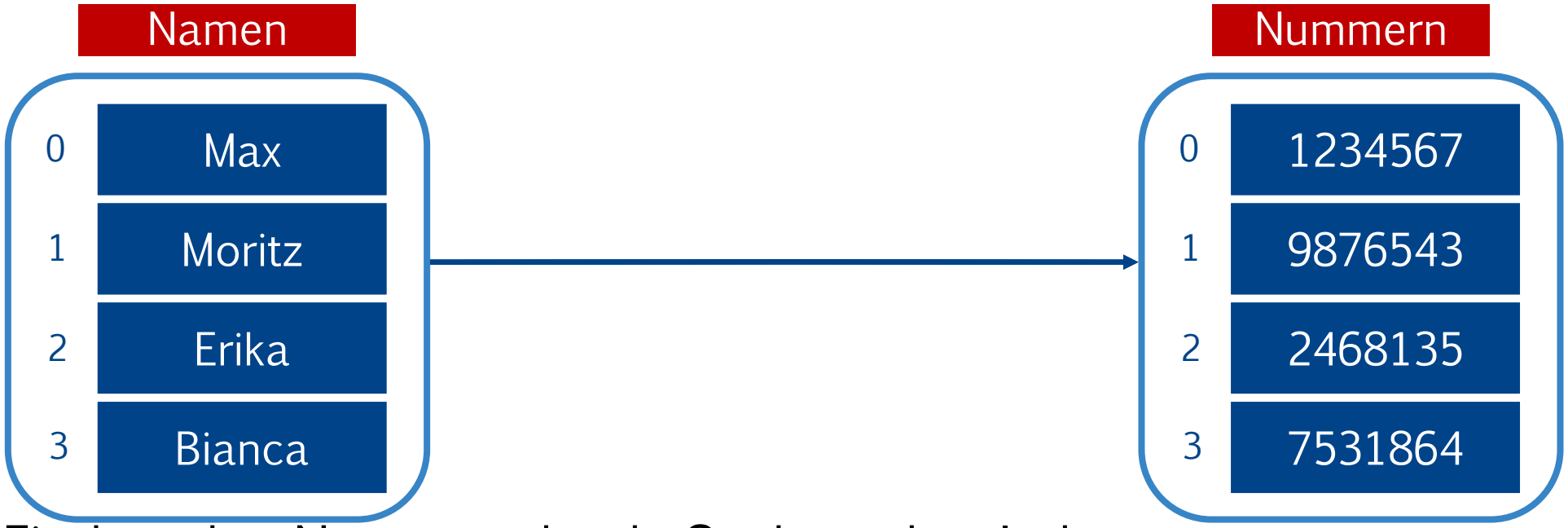
```
HashSet<String> set = new HashSet<String>();
```

- Hinzufügen (add):

```
set.add("Bananen");
```

Grenzen von Listen

› Implementierung Telefonbuch durch zwei getrennte Listen

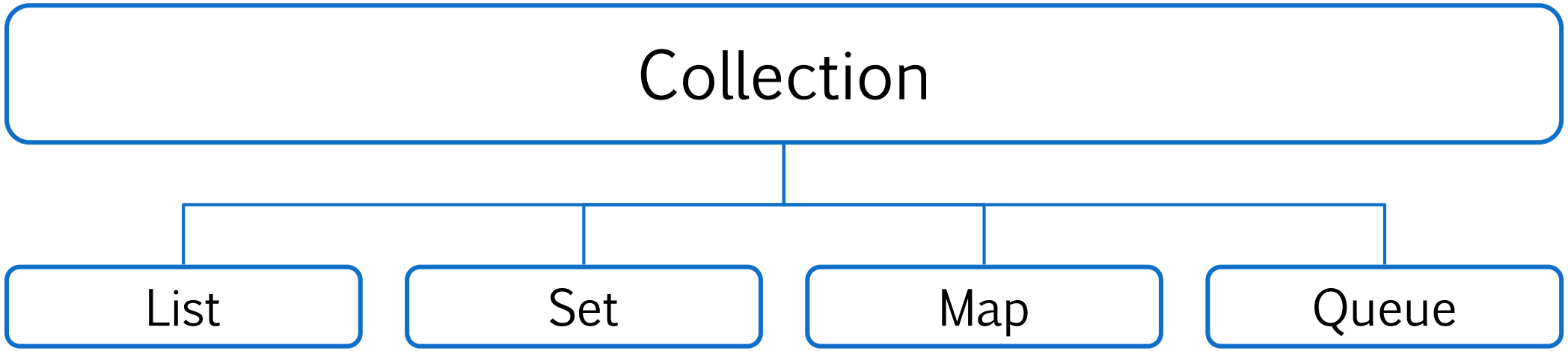


› Finden der Nummer durch Suchen des Index

› Problem: Einfügen und Löschen wird schnell sehr komplex

› Idee: Verbindung von Schlüssel (key) → Wert (value) wäre sinnvoll

Collections - Hierarchie

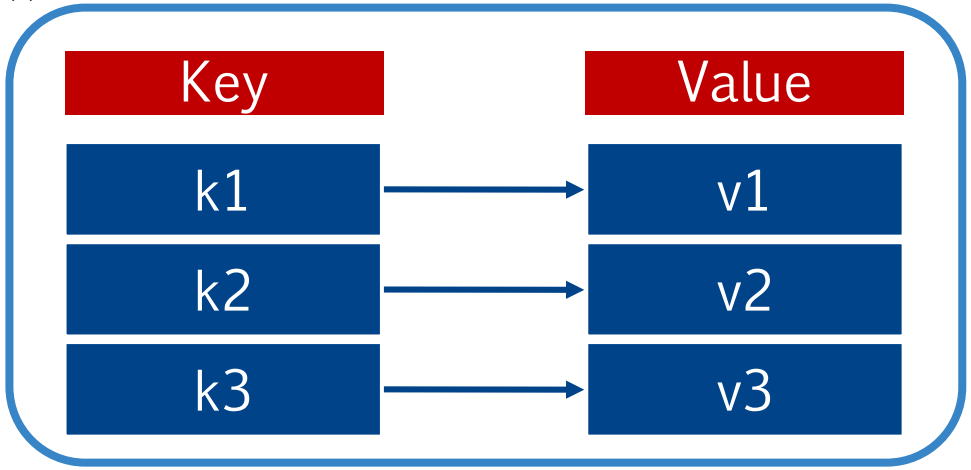


- ArrayList
- LinkedList
- HashSet
- TreeSet
- HashMap
- TreeMap

HashMap

› Datenstruktur bestehend aus Schlüssel-Wert-Paaren
→ „Map“

› Direkter Zugriff auf Wert ohne Schleifen mittels Schlüssel
→ „Hash“



Umgang mit HashMaps

Deklaration

HashMap < TypK , TypV > Bezeichner = new HashMap < TypK , TypV > ( );

```
HashMap<String, Integer> map = new HashMap<String, Integer>();
```

Werte hinzufügen (put: key, value)

```
map.put( "Max" , 1234567 );
```

Werte auslesen (get)

```
map.get( "Max" );
```

Gibt den Schlüssel an

Werte entfernen (remove)

```
map.remove( "Max" );
```

Aufgabe: Kunden

- Ein Mobilfunkunternehmen will seine Datenhaltung bei den PrePaid-Kunden verändern.
- Dazu soll zu der Handy-Nummer jedes Kunden der aktuelle Stand des Guthabens dieser Nummer gespeichert werden.
- Schreibt hierzu ein Programm das den Nutzer zur Eingabe beliebig vieler Handy-Nummern und deren Guthaben auffordert und entsprechend abspeichert.
- Nach der Eingabe aller Werte soll der Nutzer die Guthaben beliebig vieler Handy-Nummern durch Eingabe der Nummer abfragen können.

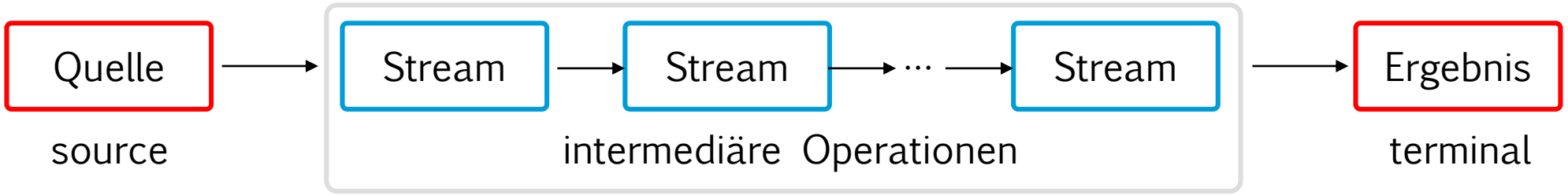
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Streams

Datenströme und Pipelines

Ein *Datenstrom (Stream)* ist eine Folge gleichstrukturierter Elemente, deren Ende nicht im Voraus festgelegt ist.

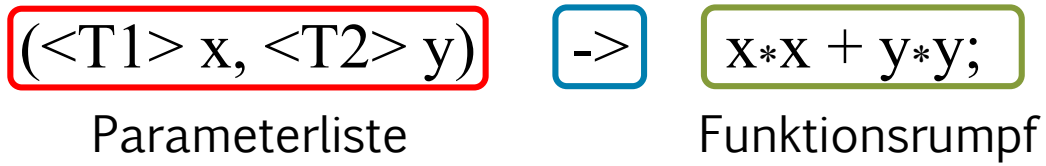
Stream = x_1 x_2 x_3 x_4 x_5 ⋯



lazy = „so viel wie nötig“ eager

Stream != Datenspeicher sondern verweist auf Quelle

Lambda-Ausdrücke



Zwei Integer Parameter mit Funktion

z.B.
```
(<Integer> x, <Integer> y)        ->        x*x + y*y;
```

Ein Integer, der vom Compiler als solcher erkannt wird

z.B.
```
x        ->        x*x +1;
```

Integer mit Methode im Funktionsrumpf

z.B.
```
(x)        ->        {return x*x + 1;};
```

Umgang mit Streams I

Deklaration

```
List <String> text = ... ;
Stream <String> Bezeichner = text.stream();
Stream <String> Bezeichner = text.parallelStream();
```

Quelle filtern

z.B. `stream.filter(x -> x <5);`

Funktion auf Quelle anwenden, Ausgabe ist ein Int-Stream

z.B. `Stream.mapToInt(x -> x * x);`

Werte zu einem Zeitpunkt im Stream ausgeben (Debugging)

z.B. `stream.peek(x -> System.out.print(x));`

Intermediäre Operationen

Umgang mit Streams II

Alle Ergebnisse aufaddieren

z.B.
```
stream.sum();
```

Auf alle Elemente eine Funktion anwenden

z.B.
```
stream.forEach(x -> System.out.print(x));
```

Übrigen Werte im Stream zählen

z.B.
```
stream.count();
```

terminale Operationen

Beispiel

Funktion $f(x) = x^2$

```
List<Integer> liste = Arrays.asList(1, 2, 3, 4, 5);
return liste.stream()
        .mapToInt(ergebnis -> ergebnis * ergebnis)
        .peek(x -> System.out.print(x + ", ")) // 9, 4, 16, 25, 1,
        .sum();
```

Erweiterung: Kunden

- Unser Unternehmen plant eine Aktion für seine Kunden:
- Jeder Kunde mit einem Guthaben unter 20€ soll 20% davon zu seinem Guthaben geschenkt bekommen.
- Erweitert das Programm um eine Funktion, welche die Kosten dieser Aktion für das Unternehmen berechnet.
- Zusatz: Umsetzung als Stream!

Abschluss

Vielen Dank!