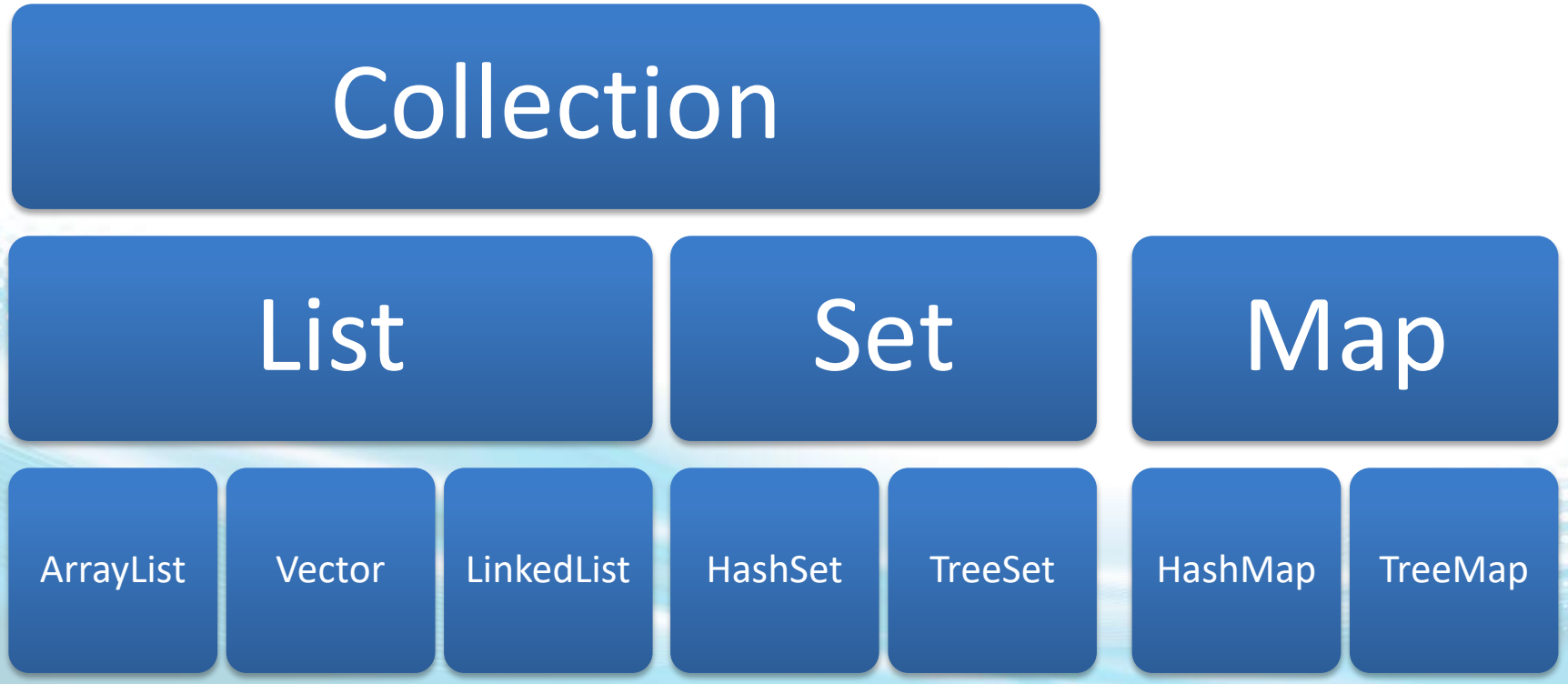
Java-Grundlagenschulung
Tag 4

SHD Einzelhandelssoftware
GmbH & Co. KG
SHD

Agenda

- Übersicht Collection-Framework
- Arrays
- Aufgabe 1 – Arrays in der Praxis
- Das List Interface
- Aufgabe 2 - Die Liste in der Praxis
- Das Set Interface
- Aufgabe 3 – Das Set in der Praxis
- Das Map Interface
- Aufgabe 4 – Die Map in der Praxis
- Hausaufgabe

Übersicht Collection-Framework



Die ‚Urliste' -> Arrays

- Mehrere Werte können in der selben Variable gespeichert werden
- Zuordnung geschieht über einen Index (beginnt mit 0)
- Beim Initialisieren muss die max. Größe des Arrays bereits bestimmt werden
- Wächst nicht dynamisch mit der Anzahl der Elemente mit
- Kann mehrdimensional sein (Matrix)

- Syntax Beispiele:

```
String[] stringArray = new String[10];
stringArray[5] = „Ich bin das 5. Element“
System.out.println(stringArray[5]);   =>  „Ich bin das 5. Element“
String[][] zweiDimStringArray = new String[5][5]; (Kann 25 Elemente halten)
zweiDimStringArray[2][2] = „Ich bin das mittlere Element“;
```

Aufgabe 1 - Arrays

1. Erzeuge ein Stalltier Array und füge diesem 100 unterschiedliche Tiere hinzu.

2. Gebe mit einer normalen Schleife und mit einer for-each Schleife den Namen jedes Stalltiers einmal auf der Konsole aus.

3. Erzeuge einen zweidimensionalen String Array mit 10 * 10 Feldern, befülle jedes davon mit einem ‚*' und zeichne mit dem Einsatz von print und println ein *-Quadrat auf der Konsole.

4. Für die fleißigen, male damit nur den Rand des Quadrats

Das List Interface

ArrayList

- Besteht aus einem dynamisch vergrößerndem Array
- Schnellen Zugriff auf einzelne Elemente, da indexbasiert.
- Langsam beim schreiben / löschen

Vector

- Gleiches Verhalten wie ArrayList
- Bietet jedoch Threadsicherheit, da synchronisiert.
- Daraus resultiert eine geringfügig langsamere Laufzeit.

LinkedList

- Doppelt verkettete Liste, nicht indiziert. (Perlenkette)
- Bei jedem Zugriff auf ein Element muss von vorne beginnend die Kette iteriert werden.
- Bearbeitung am Anfang/Ende der Liste schnell, in der Mitte langsam.

Aufgabe 2 - Die Liste in der Praxis

Führe folgende Aufgabe mit jedem Listentyp(ArrayList, Vector, LinkedList) durch und stoppe / dokumentiere die Zeit:

1. Liste mit 100.000 / 1.000.000 Stalltieren befüllen

2. Iterieren über die Liste und gebe jedes Stalltier anhand der überschriebenen toString() Methode auf der Konsole aus

3. 5 Elemente vom Anfang, der Mitte und dem Ende der jeweiligen Liste löschen (Ohne Iteration)

Das Set Interface

Allgemein gilt:

- Keine doppelten Elemente
- Einzelne Objekte lassen sich nur über Iteration auslesen

HashSet

- HashSet hohe Performance beim bearbeiten aller Elemente
- Elemente nur einmalig vorhanden durch hashCode() Vergleich (erbt Default von Klasse Object)
- Keine Garantie für eine sortierte Reihenfolge

TreeSet

- Gleiches Verhalten wie HashSet
- Ist Sortiert (Für Sortierung wird compareTo() (return -1, 0 oder 1) benötigt)
- Zugriff langsamer wegen Sortierung.

Aufgabe 3 – Das Set in der Praxis

Führe folgende Aufgabe mit jedem Settyp(HashSet, TreeSet) durch und dokumentiere das Verhalten:

1. Befülle das Set mit 10 Tieren (Gewicht zufällig)

2. Iteriere über das Set und gebe jedes Element auf der Konsole aus. Beachte dabei, dass der <Typ> des TreeSets das Comperable Interface implementieren muss. Sortiere dafür in der compareTo() Methode die Stalltiere nach dem Gewicht aufsteigend.

3. Befülle die selben Sets erneut mit den Tieren aus Aufgabe 1 und gebe die Größe der Sets auf der Konsole aus.

Das Map Interface

Allgemein gilt:

- Verbindung zwischen Key & Value (Wörterbuch)
  Über einen Key wird ein Value assoziiert.
  Arbeitet nur in eine Richtung schnell. (vom Key zum Value)

- Keine doppelten Keys möglich
- Erbt nicht von Collection
- Iteration über Key und/oder Value möglich
- Hinzufügen von Datensätzen mit put()
- Hinzufügen doppelter Keys => Vorhandener Value des Keys wird überschrieben

Das Map Interface

HashMap

- Erlaubt null Einträge (Sowohl Key, als auch Value).
- Keine Garantie für eine sortierte Reihenfolge
- Eignet sich dazu viele Elemente zu speichern und über den Schlüssel schnell wieder verfügbar zu machen.

TreeMap

- Ist Sortiert nach Key (Für Sortierung wird compareTo() benötigt)
- Zugriff langsamer wegen Sortierung.

Aufgabe 4 – Die Map in der Praxis

Führe folgende Aufgabe mit jedem Maptyp(HashMap, TreeMap) durch und dokumentiere das Verhalten:

1. Befülle die Map mit verschiedenen Keys & Values

2. Befülle die Map mit 2 gleichen Keys & unterschiedlichen Values

3. Iteriere mit 3 verschiedenen Methoden über die Map (Key, Value, EntrySet) und gebe jeweils für jedes Element den Key und/oder Value aus.
   Stoppe die Zeit und dokumentiere welche Iteration die Schnellste ist.

Hausaufgabe

- Schreibe eine Methode, welche eine übergebene Collection, mit einer übergebenen Anzahl an zufälligen Zahlen befüllt.
- Schreibe eine Methoden, welche die Collection entgegennimmt und diese anhand eines Parameters entweder aufsteigend oder absteigend sortiert (mit sort() und compare()) und anschließend die sortierte Liste wieder zurückgibt.
- Schreibe eine Methode, welche den Inhalt einer übergebenen Collection an eine zweite übergebene Collection hinzufügt und die zusammengeführte Liste zurück gibt.