# Java

## Collections part 2

Vincent Gerber, Tilman Hinnerichs

Java Kurs

20. Juni 2018

# Overview

# Repetition

What we learned last time:

- How to use generics
- How to handle Javas lists, sets and iterators

What we will try to achieve today:

- How to use iterators on sets and lists
- How to use maps and what to with them
- What exceptions are and how to handle them

# A quiz!

| | Set | List |
|---|---|---|

# A quiz!

| | Set | List |
|---|---|---|
| Same item twice in it? | | |
| Ordered? | | |
| Iterable? | | |
| What package to import | | |
| Declaring set type (variable type) | | |
| Building an instance (example) | | |
| Add an item | | |
| Removing an item | | |

# A quiz!

| | Set | List |
|---|---|---|
| Same item twice in it? | No! | Yes! |
| Ordered? | No! | Yes! |
| Iterable? | Yes! | Also yes! |
| What package to import | import java.util.* | import java.util.* |
| Declaring set type (variable type) | Set<T> set | List<T>list |
| Building an instance (example) | = new HashSet<T>() | = new ArrayList<T> |
| Add an item | set.add(item) | list.add(item) |
| Removing an item | set.remove(item) | list.remove(item) |

# Another quiz!

The iterator:

| | Iterator |
|---|---|
| How to declare | |
| How to build an instance | |
| First main function (With data type) | |
| Second main function (With data types) | |
| Third main function (With data type) | |
| How to get from collection? | |

# How to iterate over sets and lists

# How to iterate over sets and lists

```
1  Set<T> mySet = new HashSet<T>();
2  foreach(T item:mySet){
3      item.doSomething();
4  }
5
6  List<T> myList = new ArrayList<T>();
7  foreach(T item:myList){
8  item.doSomething();
9  }
10
```

# Another quiz!

The iterator:

| | Iterator |
|---|---|
| How to declare | Iterator<T> iter |
| How to build an instance | = new Iterator<T>() |
| First main function (With data type) | boolean iter.hasNext() |
| Second main function (With data types) | T iter.next() |
| Third main function (With data type) | T iter.remove() |
| How to get from collection(e.g. set)? | set.iterator() |

# How to iterate over sets and lists using iterators

```
1  Set<T> mySet = new HashSet<T>();
2  Iterator<T> myIter = mySet.iterator();
3
4  while(myIter.hasNext()){
5      T item = myIter.next();
6      item.doSomething();
7  }
8
```

# Exercise

- Create an array with 10 elements. Create a list and fill the list with the array elments. Create a set and fill the set with the list elments and create a map with the set elments as values and the index as key.
- Extend our vending machine with an internal storage

# Map

The interface *Map* is not a subinterface of *Collection*.
A map contains pairs of key and value. Each key refers to a value. Two keys can refer to the same value. There are not two equal keys in one map. *Map* is part of the package `java.util`.

```
1   public static void main (String[] args) {
2
3   Map<Integer, String> map =
4   new HashMap<Integer, String>();
5
6   map.put(23, "foo");
7   map.put(28, "foo");
8   map.put(31, "bar");
9   map.put(23, "bar"); // "bar" replaces "foo" for key = 23
10
11  System.out.println(map);
12  // prints: {23=bar, 28=foo, 31=bar}
13  }
14
```

# Key, Set and Values

You can get the set of keys from the map. Because one value can exist multiple times a collection is used for the values.

```
1   public static void main (String[] args) {
2
3   // [...] map like previous slide
4
5   Set<Integer> keys = map.keySet();
6   Collection<String> values = map.values();
7
8   System.out.println(keys);
9   // prints: [23, 28, 31]
10
11  System.out.println(values);
12  // prints: [bar, foo, bar]
13  }
14
```

# Iterator

To iterate over a map use the iterator from the set of keys.

```
1   public static void main (String[] args) {
2
3   // [...] map, keys, values like previous slide
4   Iterator<Integer> iter = keys.iterator();
5
6   while(iter.hasNext()) {
7   System.out.print(map.get(iter.next()) + " ");
8   } // prints: bar foo bar
9
10  System.out.println(); // print a line break
11
12  for(Integer i: keys) {
13  System.out.print(map.get(i) + " ");
14  } // prints: bar foo bar
15  }
16
```

# Nested Maps

Nested maps offer storage with key pairs.

```
1   public static void main (String[] args) {
2
3   Map<String, Map<Integer, String>> addresses =
4   new HashMap<String, Map<Integer, String>>();
5
6   addresses.put("Noethnitzer Str.",
7   new HashMap<Integer, String>());
8
9   addresses.get("Noethnitzer Str.").
10  put(46, "Andreas-Pfitzmann-Bau");
11  addresses.get("Noethnitzer Str.").
12  put(44, "Fraunhofer IWU");
13  }
14
```

# Maps and For Each

You can interate through the entry set of a map (available before Java 1.8)

```
1     Map<String, String> map = ...
2     for (Map.Entry<String, String> entry : map.entrySet()) {
3     System.out.println("Key: " + entry.getKey() +
4     ", value" + entry.getValue());
5     }
6
```

# Overview

| | |
|---|---|
| List | • Keeps order of objects<br>• Easily traversible<br>• Search not effective |
| Set | • No duplicates<br>• No order - still traversible<br>• Effective searching |
| Map | • Key-Value storage<br>• Search super-effective<br>• Traversing difficult |

# Easy and some more complex exercises

Hier könnten Ihre Aufgaben zu Mpas und Iteratoren stehen!