Informatik II

Übung 5

Andreas Bärtschi, Andreea Ciuprina, Felix Friedrich, Patrick Gruntz,
Hermann Lehner, Max Rossmannek, Chris Wendler

FS 2018

Heutiges Programm

1. Feedback letzte Übung
2. Wiederholung Vorlesung

Letzte Woche: Lesen vom Scanner

```
static void printByIndex(List<Measurement> data, Scanner scanner){
    int index;
    while (scanner.hasNextInt()) {
        index = scanner.nextInt();
        Measurement m = data.get(index);
        System.out.println(m);
    }
}
```

Letzte Woche: Lesen vom Scanner

```
static void printByIndex(List<Measurement> data, Scanner scanner){
    int index;
    while (scanner.hasNextInt()) {
        index = scanner.nextInt();
        Measurement m = data.get(index);
        System.out.println(m);
    }
}
```

Runtime exceptions müssen wir hier nicht behandeln:
Bei der ersten Nicht-Integer-Eingabe beenden wir die While-Schleife und lesen in der Folge nichts mehr ein.

Letzte Woche: Lesen aus einem File

```
private static List<Measurement> readCSV(String csvFile) {
    try (Filereader fr = new FileReader(csvFile);
         BufferedReader br = new BufferedReader(fr);) {
        List<Measurement> data = new ArrayList<Measurement>();
        String line = br.readLine();                    // CSV header line
        while ((line = br.readLine()) != null) {    // All other lines
            Measurement m = new Measurement(line); // Constructor
            data.add(m);
        }
        return data;
    } catch (IOException e) {
        System.err.println("Could not read earthquake data: " + e);
        return null;
    }
}
```

Letzte Woche: Generisches Sortieren

Durch Implementieren des Comparable Interface:

```
public int compareTo(Measurement that) {
    int BEFORE = -1; int EQUAL = 0; int AFTER = 1;
    // exactly the same reference?
    if (this == that) return EQUAL;
    // Compare Magnitude DESCENDING
    if (this.getMagnitude() > that.getMagnitude())
        return BEFORE;
    else if (this.getMagnitude() < that.getMagnitude())
        return AFTER;
    else // Compare Date/Time DESCENDING
        return -this.getDateTime().compareTo(that.getDateTime());
}
```

Letzte Woche: Generisches Sortieren

Alternative: Übergeben eines Comparators.

```
import java.util.Comparator;

...

Collections.sort(data, // + Comparator
    Comparator.comparing(Measurement::getMagnitude, // DESC Magnitude
                         Comparator.reverseOrder())
          .thenComparing(Measurement::getDateTime, // DESC Date/Time
                         Comparator.reverseOrder()));
```

Wiederholung Inhalte Vorlesung: Beispiel 1

```
try(Stream<String> stream = Files.lines(Paths.get("fruits.csv"))) {

    Fruit someFruit = stream.skip(1)
        .map(Fruit::new)
        .max(Fruit::compareTo)


        .get();
    System.out.println(someFruit);

} catch(IOException e) {
    System.err.println("Oh, that's too bad "+e);
}
```

Wiederholung Inhalte Vorlesung: Beispiel 1

```
try(Stream<String> stream = Files.lines(Paths.get("fruits.csv"))) {

    Fruit someFruit = stream.skip(1) // skip CSV header line
        .map(Fruit::new)             // map lines to Fruit Objects
        .max(Fruit::compareTo)       // use implemented Comparable
        //.max((a,b) -> a.compareTo(b)) // equivalent lambda expr.
                                     // .max returns Optional<Fruit>
        .get();                      // Optional<Fruit> to Fruit
    System.out.println(someFruit);

} catch(IOException e) {
    System.err.println("Oh, that's too bad "+e);
}
```

Wiederholung Inhalte Vorlesung: Beispiel 2

```
...

stream.skip(1)
    .map(String::toUpperCase)
    .map(Fruit::new)
    .filter(m -> m.getTaste() >= 5)
    .filter(m -> m.getCalories()-55 <= 7 && 55-m.getCalories() <= 7)

    .sorted((a,b) -> Integer.compare(a.getTaste(), b.getTaste()))

    .forEach(System.out::println);

...
```

Wiederholung Inhalte Vorlesung: Beispiel 2

```
...

stream.skip(1)
    .map(String::toUpperCase)        // all CAPS
    .map(Fruit::new)                 // map lines to fruits, only keep
    .filter(m -> m.getTaste() >= 5) // - good tasting fruits
    .filter(m -> m.getCalories()-55 <= 7 && 55-m.getCalories() <= 7)
                                     // - with |calories-55| <= 7
    .sorted((a,b) -> Integer.compare(a.getTaste(), b.getTaste()))
                                     // sort by Taste ascendingly
    .forEach(System.out::println);   // print remaining sorted fruits

...
```

Wiederholung Inhalte Vorlesung: Beispiel 3

```
...

Double someDouble =
stream.skip(1)
    .map(Fruit::new)
    .mapToDouble(m -> 1d*m.getCalories()/m.getTaste())

    .peek(System.out::println)
    .average()
    .getAsDouble();
System.out.println(someDouble);

...
```

Wiederholung Inhalte Vorlesung: Beispiel 3

```
...

Double someDouble =
stream.skip(1)
    .map(Fruit::new)            // map lines to fruits
    .mapToDouble(m -> 1d*m.getCalories()/m.getTaste())
                                // map to Calories per Unit of Taste
    .peek(System.out::println) // observe what's happening in stream
    .average()                  // compute mean, as OptionalDouble
    .getAsDouble();             // OptionalDouble to Double
System.out.println(someDouble);

...
```

Fragen oder Anregungen?