Informatik II

Übung 4

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

FS 2018

Program Today

1. Feedback of last exercise
2. Repetition Lectures
3. Next Exercise

Sums

Given a data vector of length $n \in \mathbb{N}$: $(y_i)_{i=1\ldots n} \in \mathbb{R}^n$

Sum $m_n := \sum_{i=1}^{n} y_i$
Sum of Squares $s_n := \sum_{i=1}^{n} y_i^2$

New data point ($n \to n+1$):

$$m_{n+1} = m_n + y_{n+1}$$
$$s_{n+1} = s_n + y_{n+1}^2$$

Provisional Means

Mean $\mu_n := m_n/n$

$$\begin{aligned}
\mu_{n+1} &= \frac{1}{n+1}\sum_{i=1}^{n+1} y_i \\
&= \frac{1}{n+1}(n \cdot \mu_n + y_{n+1}) \\
&= \frac{1}{n+1}((n+1)\mu_n + y_{n+1} - \mu_n) \\
&= \mu_n + \frac{y_{n+1} - \mu_n}{n+1}.
\end{aligned}$$

Mean Squared Error

$$\sigma_n^2 := \sum_{i=1}^{n}(y_i - \mu_n)^2 \quad = \sum_{i=1}^{n} y_i^2 - 2\mu_n y_i + \mu_n^2 = s_n - n \cdot \mu_n^2$$

$$\begin{aligned}
\sigma_{n+1}^2 &= s_{n+1} - (n+1) \cdot \mu_{n+1}^2 \\
&= s_n + y_{n+1}^2 - (n+1)\left(\mu_n + \frac{y_{n+1} - \mu_n}{n+1}\right)^2 \\
&= \sigma_n^2 + (y_{n+1} - \mu_n) \cdot (y_{n+1} - \mu_{n+1})
\end{aligned}$$

Result

```
public void addValue(double x){
  ++n;
  if (n == 1){
    mean = x;
    ssq = 0;
  } else {
    double oldMean = mean;
    mean = mean + (x − mean) / n;
    ssq = ssq + (x − oldMean) ∗ (x − mean);
  }
}
```

Example: Reading from the Scanner

```
Scanner scanner = new Scanner(System.in);
int number;
while (scanner.hasNextInt()) {
        number = scanner.nextInt();
        ...
}
```

Example: Reading from a File

```
try ( FileReader fr = new FileReader(filename);
      BufferedReader bufr = new BufferedReader(fr);) {
        ...
        String line;
        // read line by line
        while ((line = br.readLine()) != null){
                // do something with line
        }
        ...
} catch (IOException e){
        // do some recovery handling
}
```

Generic Sorting with Java

```
public class Fruit {
        private String name;          // Attributes
        private int taste;
        private int calories;
        Fruit(String name, int taste, int calories) {
                this.name = name;     // Constructor
                this.taste = taste;
                this.calories = calories;
        }
        public String getName() {     // Name Getter
                return name;
        }
        public int getTaste() {       // Taste Getter
                return taste;
        ...
```

Generic Sorting with Java

```
import java.util.ArrayList;
import java.util.List;
import java.util.Collections;
public class Main {
	public static void main(String[] args) {
		List<Fruit> fruits = new ArrayList<Fruit>();
		fruits.add(new Fruit("apple", 5, 52));
		fruits.add(new Fruit("pear", 5, 57));
		fruits.add(new Fruit("banana", 6, 89));
		fruits.add(new Fruit("kiwi", 6, 61));
		Collections.sort(fruits);
		for (Fruit fruit : fruits)
			System.out.println(fruit.getName());
	}
}
```

Generic Sorting with Java

```
import java.util.ArrayList;
import java.util.List;
import java.util.Collections;
public class Main {
        public static void main(String[] args) {
                List<Fruit> fruits = new ArrayList<Fruit>();
                fruits.add(new Fruit("apple", 5, 52));
                fruits.add(new Fruit("pear", 5, 57));
                fruits.add(new Fruit("banana", 6, 89));
                fruits.add(new Fruit("kiwi", 6, 61));
Sort criteria? →Collections.sort(fruits);
                for (Fruit fruit : fruits)
                        System.out.println(fruit.getName());
        }
}
```

Generic Sorting: Comparable Interface

```
public class Fruit implements Comparable<Fruit> {
        private String name;          // Attributes
        private int taste;
        private int calories;
        Fruit(String name, int taste, int calories) {
                this.name = name;     // Constructor
                this.taste = taste;
                this.calories = calories;
        }
        public String getName() {     // Name Getter
                return name;
        }
        public int getTaste() {       // Taste Getter
                return taste;
        ...
```

Generic Sorting: Comparison Method

```
// Comparison Method for Comparable Interface:
public int compareTo(Fruit that) {
        int BEFORE = -1; int EQUAL = 0; int AFTER = 1;
        // Check if same fruit
        if (this == that)
                return EQUAL;
        // DESCENDING by Taste
        if (this.getTaste() > that.getTaste())
                return BEFORE;
        else if (this.getTaste() < that.getTaste())
                return AFTER;
        else    // Same Taste -> ASCENDING by Calories
                return this.getCalories()-that.getCalories();
}
```

Generic Sorting: Comparison Method

```
// Comparison Method for Comparable Interface:
public int compareTo(Fruit that) {
    int BEFORE = -1; int EQUAL = 0; int AFTER = 1;
    // Check if same fruit
    if (this == that)
        return EQUAL;
    // DESCENDING by Taste
    if (this.getTaste() > that.getTaste())
        return BEFORE;
    else if (this.getTaste() < that.getTaste())
        return AFTER;
    else    // Same Taste -> ASCENDING by Calories
        return this.getCalories()-that.getCalories();
}
```

negative value
⇔ "this < that"

positive value
⇔ "this > that"

Questions / Suggestions?