Praktikum "SEP: Java-Programmierung" WS 2018/19

Grundlagen: Input und Output mit der Shell

Thomas Lemberger und Martin Spießl

Basierend auf Folien von Matthias Dangl und Karlheinz Friedberger

Nutzung der Shell für User-Interaktion

Vorgehen bei Nutzung einer Shell für User-Interaktion per Texteingabe:

1. Ausgabe des Prompts
2. Lesen einer „Zeile“ von Texteingabe
3. Zerlegen der Eingabe in Tokens
4. Fallunterscheidung nach erstem Token
   - Überprüfung auf korrekte Syntax
     (Anzahl und Format der Parameter)
   - Überprüfung auf semantische Korrektheit
     (Kommando in diesem Programmzustand legitim?)
   - Ausführung der Aktion(en)

Verarbeitung von Texteingaben in der Shell

Einlesen von Texteingabe aus Input-Stream `System.in`:

```
BufferedReader stdin = new BufferedReader(
    new InputStreamReader(System.in));

String input = stdin.readLine();
```

- Problem: IOExceptions können auftreten

$\Rightarrow$ Im Methodenkopf deklarieren (`throws IOException`)

- Notwendig: Imports aus java.io
  (`BufferedReader`, `InputStreamReader`, `IOException`)

Ausgabe auf Shell mit `System.out.println`:

```
System.out.println("Hello, World!")
```

Grundgerüst der Shell (Beispiel)

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public final class Shell {

  // utility class
  private Shell() {}

  public static void main(String[] args)
      throws IOException {
    BufferedReader stdin =
      new BufferedReader(
        new InputStreamReader(System.in));

    execute(stdin);
  }
}
```

Grundgerüst der Shell (Beispiel)

```
private static void execute(
    BufferedReader stdin) throws IOException {
  boolean quit = false;
  while (!quit) {
    System.out.print("prompt> ");
    // read one line
    String input = stdin.readLine();
    // no more input?
    if (input == null) { break; }
    // split input on white spaces
    String[] tokens = input.trim().split("\\s+");
    ...
  }
```

Strg-D (Linux-Konsole) bzw. Strg-Z (Windows-Konsole) schließt Eingabestrom.

Grundgerüst der Shell (Beispiel)

Alternative: `java.util.Scanner`

```
String input = stdin.readLine(); ...
Scanner scanner = new Scanner(input);
scanner.useDelimiter("\\s+");
if (scanner.hasNext()) {
  String command = scanner.next();
} else {
  System.out.println("Error! No command.");
} ...
if (scanner.hasNextInt()) {
  int value = scanner.nextInt();
} else {
  System.out.println("Error! No number.");
} ...
scanner.close();
```