

Debugging – eine kurze Einführung

Einführung in die Programmierung 1
Wintersemester 21/22



Debugging

- Ausgangspunkt: Programm hat Fehler
 - Ziel: Programm ohne Fehler
- „Werkzeuge“
 - Systematische Überlegungen
 - print-Anweisungen
 - Debugger in IDE

Vorgehensweise bei Fehlerbehebung

Daten analysieren (Testresultate, Ausgaben, Fehlermeldungen, Programmtext)

Hypothese aufstellen

Wiederholbare Experimente festlegen

Einfachste Eingabe für Tests festlegen

println-Anweisungen

- Einfache Methode zum Überprüfen von Hypothesen
- Typische Punkte für Ausgaben
 - Betreten einer Methode
 - Parameterwerte
 - Rückgabewerte
 - Direkt vor und nach einer Schleife

println in großen Programmen?

- Problemzone mit Hilfe der „binären Suche“ eingrenzen
 - println-Anweisung nach der Hälfte des Programms
 - println-Anweisung abhängig von der Ausgabe neu in eine Hälfte platzieren usw.
- Meist umständlich

Debugger

- Debugger
 - Werkzeug zum Diagnostizieren und Auffinden von Fehlern in Programmen
- Typische Funktionen
 - Steuerung des Programmablaufs (Haltepunkte, engl. Breakpoints)
 - Schrittweise Durchführung von Programmen
 - Inspizieren und modifizieren von Variablen

Debugger in IntelliJ

The screenshot shows the IntelliJ IDEA interface with the following components:

- Project View (Left):** Shows the project structure with folders like `.idea`, `out`, and `src`. The `src` folder contains several Java files, including `StarPattern.java`.
- Code Editor (Center):** Displays the `StarPattern.java` file. The code is as follows:

```
public class StarPattern {  
    public static void main(String[] args) {  
        int n = 5;  
        for (int i = -n; i <= n; i++) {  
            for (int j = -n; j <= n; j++) {  
                if (i * i <= j * j) {  
                    System.out.print("* ");  
                } else {  
                    System.out.print(" ");  
                }  
            }  
        }  
    }  
}
```


A breakpoint is set on line 6, which is the start of the inner `for` loop. The line is highlighted in blue.

The `Debugger` window at the bottom shows the `main` method is running. The `Variables` section shows the current state of the program:
 - `args`: `{String[0]@807} []`
 - `n`: `5`
 - `i`: `-5`
- Debugger Window (Bottom):** Contains the `Debugger` tab, which shows the current state of the program. The `Frames` section shows the `main` method is running. The `Variables` section shows the current state of the program.

Callouts in the image point to the following elements:

- Breakpoint:** Points to the breakpoint icon on line 6.
- Debugger einschalten:** Points to the `Debugger` button in the top toolbar.
- Einzelne Programmschritte ausführen:** Points to the `Step Into` button in the debugger toolbar.
- Variablen inspizieren:** Points to the `Variables` section in the debugger window.

Beispiele

- Demonstration der Funktionalität mit unterschiedlichen Beispielen aus dem letzten Foliensatz (Arrays)

Fehler finden – einfache Fälle

- Ausnahmen beachten, z. B.
 - Indexfehler bei Arrays oder Strings
 - Nullpointer
 - Falsches Format bei Eingaben

Logische Fehler – schwierig

- Programm überdenken
 - Nicht einfach Zeilen verändern
 - Schlechte Beispiele
 - Programming by permutation
 - https://en.wikipedia.org/wiki/Programming_by_permutation
 - Shotgun debugging
 - https://en.wikipedia.org/wiki/Shotgun_debugging
- Zusätzliche Hilfe
 - Diagramme zeichnen
 - Programm anderen Personen erklären

Debuggen größerer Programme (1)

Schlecht

- Ganzes Programm schreiben
- Ganzes Programm testen
- Ganzes Programm debuggen

Besser

- Einzelne Methode implementieren
- Einzelne Methode testen und debuggen
- Einzelne Methode implementieren
- Einzelne Methode testen und debuggen
- ...
- Alle Methoden integrieren

Debuggen größerer Programme (2)

Schlecht

- Programm ändern
- Änderung sich „merken“
- Testen
- ...
- Keine Ahnung, wo der Fehler war!

Besser

- Backup des Programms anlegen (bzw. Versionsverwaltung)
- Programm ändern
- Potentiellen Fehler dokumentieren (z. B. Kommentare)
- Programm testen
- Neue Version mit alter vergleichen
- Bei Problemen – Backup!

Weitere Informationen zu Debugging in IntelliJ

- <https://www.jetbrains.com/help/idea/debugging-code.html>