

Softwaretechnik - Schlange

Softwaretechnik

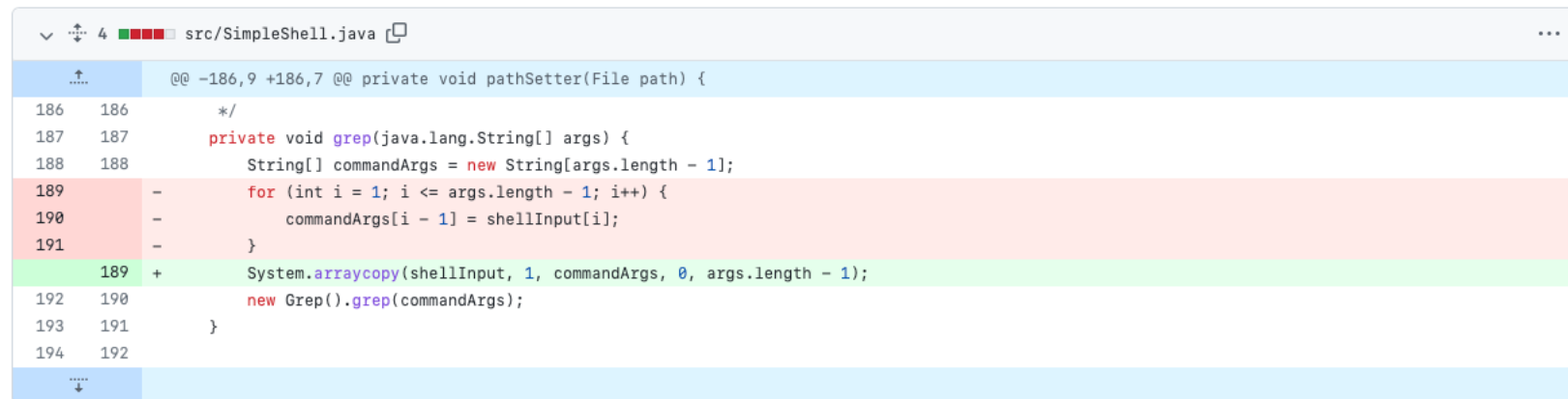
Einsendeaufgabe 5

- 3 Schreiben Sie ganz kurz etwas über die 1-2 Refactorings, die Sie besonders beeindruckt haben und über je eines, das Sie für überflüssig halten und eines, das Sie nicht verstanden haben!

3.1 Beeindruckend:

1. **Extract Method** ist für mich beeindruckend, da es viel Schreibarbeit spart.
2. **Replace Exception with Precheck** – sorgt mit einem Klick für Sicherheit im Code und spart einem oft viele Checks (bspw. auf *null*).
3. **Replace Loop with Pipeline**: dies ist beeindruckend, da es potenziell langsame Loops mit einfachen Funktionsaufrufen ersetzt.

Ein Beispiel hierfür habe ich in Code aus dem GP 2 Modul gefunden:



```
src/SimpleShell.java
@@ -186,9 +186,7 @@ private void pathSetter(File path) {
186 186      */
187 187      private void grep(java.lang.String[] args) {
188 188          String[] commandArgs = new String[args.length - 1];
189 -         for (int i = 1; i <= args.length - 1; i++) {
190 -             commandArgs[i - 1] = shellInput[i];
191 -         }
189 +         System.arraycopy(shellInput, 1, commandArgs, 0, args.length - 1);
192 190         new Grep().grep(commandArgs);
193 191     }
194 192 }
```

Abbildung 1: Replace Loop with Pipeline

3.2 Überflüssig:

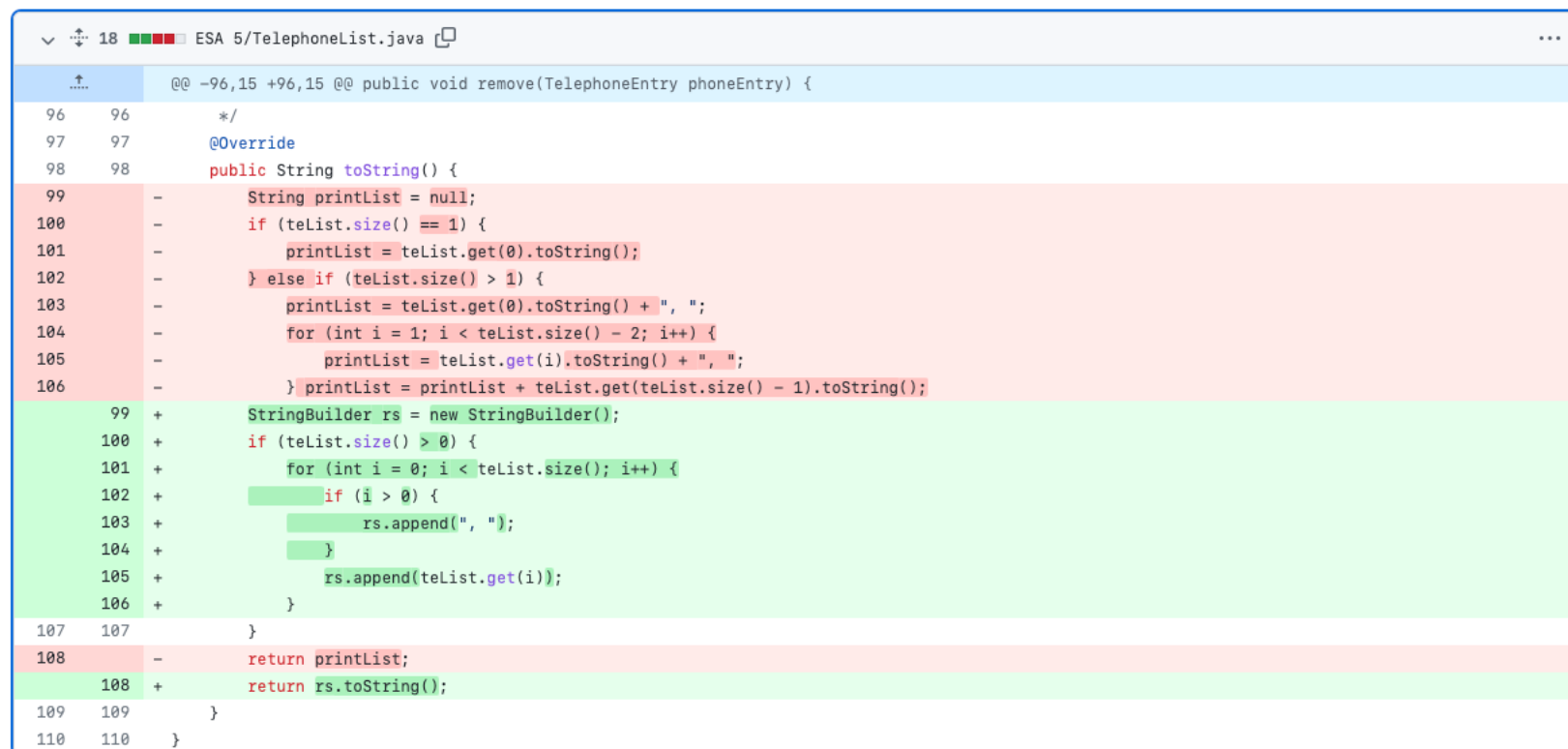
1. Nicht per se überflüssig, aber oftmals trägt [Replace Magic Literal](#) nicht zur Lesbarkeit von Code bei. Wenig nützlich ist es insbesondere, wenn die Magic Number nur einmal benutzt wird.
2. [Replace Temp with Query](#) ist im Beispiel von Fowler nicht hilfreich, da, statt eine Konstante zu befüllen nur eine Methode aufgerufen wird.

3.3 Nicht verstanden:

Ich habe noch nicht mit Factories gearbeitet und daher den Sinn und Zweck von [Replace Constructor with Factory Function](#) nicht verstanden.

4 Erstellen sie mindestens ein nicht triviales eigenes Code-Beispiel, welches den Zustand „Vorher und Nachher“ zeigt.

Ich habe mir eine alte Einsendeaufgabe aus dem Modul GP 2 genommen und darin eine nicht sehr elegante toString() Methode gefunden um eine Liste ausgeben zu lassen. Diese habe ich (manuell) refactored (der Commit ist [hier](#) zu finden). Das Ergebnis:



```
18  ESA 5/TelephoneList.java
@@ -96,15 +96,15 @@ public void remove(TelephoneEntry phoneEntry) {
96 96      */
97 97      @Override
98 98      public String toString() {
99 -        String printList = null;
100 -        if (teList.size() == 1) {
101 -            printList = teList.get(0).toString();
102 -        } else if (teList.size() > 1) {
103 -            printList = teList.get(0).toString() + ", ";
104 -            for (int i = 1; i < teList.size() - 2; i++) {
105 -                printList = teList.get(i).toString() + ", ";
106 -            } printList = printList + teList.get(teList.size() - 1).toString();
107
109 +        StringBuilder rs = new StringBuilder();
110 +        if (teList.size() > 0) {
111 +            for (int i = 0; i < teList.size(); i++) {
112 +                if (i > 0) {
113 +                    rs.append(", ");
114 +                }
115 +                rs.append(teList.get(i));
116 +            }
117
118 -        return printList;
119 +        return rs.toString();
120
121     }
122 }
```

Abbildung 2: Refactoring einer toString() Methode zur Ausgabe einer Liste