Junioraufgabe 2: Baywatch

Team-ID: 00130

Team-Name: ThinkPadHacker

Bearbeiter dieser Aufgabe:
Yannik Schiebelhut

23. November 2018

Inhaltsverzeichnis

Lösungsidee

Beide Listen von Landzungen werden aus der gegeben Datei in das Programm eingelesen. Anschließend wir die vollständige Eintrag für Eintrag mit der anderen verglichen. Stimmen die Einträge an einer Stelle nicht mit den entsprechenden in der unvollständigen, im Norden beginnenden Liste überein, wird der letzte Wert der vollständigen Liste an deren Anfang geschrieben und die Prozedur so lange wiederholt, bis alle Einträge der beiden Listen mit ihrem jeweiligen „Pendant“ übereinstimmen.

Umsetzung

Ein Scanner liest die beiden Zeilen der Datei ein, die anschließend auf zwei ArrayListen des Typs „Integer“ verteilt werden, welche sich aufgrund ihrer hohen Flexibilität für diesen Zweck hervorragend eignen. Die in der lückenhaften Liste vorhandenen Fragezeichen werden hierbei durch die Ziffer „0“ ersetzt.

Anschließend wird mithilfe von zwei for-Schleifen der oben beschriebene Vergleich, sowie die Verschiebung gegeneinander durchgeführt.

Beispiel (baywatch1.txt)

Liste des Papagei aus der Datei: 1 1 2 3 4 1 5 1 2 6 4 5 3

Umsetzung als ArrayList: [1, 1, 2, 3, 4, 1, 5, 1, 2, 6, 4, 5, 3]

Liste des Piraten aus der Datei: ? 1 ? ? ? ? 1 5 ? 2 ? ? ?

Umsetzung als ArrayList: [0, 1, 0, 0, 0, 0, 1, 5, 0, 2, 0, 0, 0]

Vergleich:

[1, 1, 2, 3, 4, 1, 5, 1, 2, 6, 4, 5, 3]
[0, 1, 0, 0, 0, 0, 1, 5, 0, 2, 0, 0, 0] // nicht alle Daten abgesehen von „0“ stimmen überein

// Umschichten der vollständigen Liste

[1, 1, 2, 3, 4, 1, 5, 1, 2, 6, 4, 5, 3] => [3, 1, 1, 2, 3, 4, 1, 5, 1, 2, 6, 4, 5]

Vergleich:

[3, 1, 1, 2, 3, 4, 1, 5, 1, 2, 6, 4, 5]
[0, 1, 0, 0, 0, 0, 1, 5, 0, 2, 0, 0, 0] // alle Daten außer „0“ stimmen überein

// fertig

// gebe vollständige Liste aus

Quellcode

```
import java.io.File;


public class J2 {


	/**
	 * The main method of the program. All essential work is done here.
	 *
	 * @param args files to be observe, given as command line arguments
	 */
	public static void main(String[] args) {
		// Verarbeiten der Kommandozeilenargumente siehe J2.java
			try {
				// set source file to the parameter, currently being processed
				File sourceFile = new File(args[x]);
				// create scanner on file
				Scanner sc = new Scanner(sourceFile);


				// process parrot line
				// get first line from sourceFile
				String parrotStr = sc.nextLine();
				System.out.printf("parrotStr:\t%s\n", parrotStr);
				// create ArrayList
```

```
List<Integer> parrot = new ArrayList<>();
// create scanner on string
Scanner parrotSc = new Scanner(parrotStr);
// information is divided with spaces here ...
parrotSc.useDelimiter(" ");
// fill ArrayList with data
while (parrotSc.hasNextInt()) {
        parrot.add(parrotSc.nextInt());
}

// process pirate line
// get second line from sourceFile
String pirateStr = sc.nextLine();
System.out.printf("pirateStr:\t%s\n", pirateStr);
// create ArrayList
List<Integer> pirate = new ArrayList<>();
// create scanner on string
Scanner pirateSc = new Scanner(pirateStr);
// information is divided with spaces here ...
pirateSc.useDelimiter(" ");
// fill ArrayList with data
while (pirateSc.hasNext()) {
        String tmp = pirateSc.next();
        // checks if current token is a question mark
        if (Pattern.matches("[0-9]", tmp)) {
                pirate.add(Integer.parseInt(tmp));
        } else { // add a zero to the ArrayList, if it is a question
mark
                pirate.add(0);
        }
}

// adjust arrays
```

```
                    // check every value in the pirate list
                    for (int i = 0; i < pirate.size(); i++) {
                        // compare to every value in the parrot list
                        for (int j = 0; j < pirate.size(); j++) {
                            // if the current pirate token was a zero, we don't
have to compare
                            if (pirate.get(j) != 0) {
                                // if the lists don't match, make the last
parrot value the first and start a
                                // new comparison from the first value
                                if (pirate.get(j) != parrot.get(j)) {
                                    parrot.add(0,
parrot.get(parrot.size() - 1));
                                    parrot.remove(parrot.size() - 1);
                                    break;
                                }
                            }
                        }
                    }

                    // this is the actual solution
                    System.out.printf("\nDie korrekte Reihenfolge beginnend im Norden
lautet:\t(%s)\n%s\n",
                            sourceFile.getName(), parrot);

                    // close scanners
                    parrotSc.close();
                    pirateSc.close();
                    sc.close();
                } catch (FileNotFoundException e) { // needed if the given file path is invalid
                    System.out.println("WARNING!!! Given file doesn't exists or
filepath is invalid!");
                }
            }
```

```
                }
            }
        }
```