Example 2 (DeepSeek (v3.1)): migration test no. 1

Konvertierung 1

BeispielProgramm.java

```java
public class BeispielProgramm {

    // COBOL-like data structure implementation
    private static class CobolData {
        public byte counter = 42;

        // Month names structure
        public String jan = "Januar   ";
        public String feb = "Februar ";
        public String mar = "Maerz   ";
        public String apr = "April   ";
        public String mai = "Mai     ";
        public String jun = "Juni    ";
        public String jul = "Juli    ";
        public String aug = "August  ";
        public String sep = "September";
        public String okt = "Oktober ";
        public String nov = "November";
        public String dez = "Dezember";

        // Month array that redefines the structure
        public String[] monate = new String[12];

        // Constructor to initialize the array
        public CobolData() {
            updateMonateArray();
        }

        // Method to update the array when structure changes
        private void updateMonateArray() {
            monate[0] = jan;
            monate[1] = feb;
            monate[2] = mar;
            monate[3] = apr;
            monate[4] = mai;
            monate[5] = jun;
            monate[6] = jul;
            monate[7] = aug;
            monate[8] = sep;
            monate[9] = okt;
            monate[10] = nov;
            monate[11] = dez;
        }

        // Method to simulate COBOL MOVE for the structure
        public void moveToMyStruct(String value) {
            // Ensure the value is exactly 16 characters (2 months * 8 chars each)
            String paddedValue = String.format(format:"%-16s", value).substring(beginIndex:0, endIndex:16);

            // Split into two 8-character parts
            jan = paddedValue.substring(beginIndex:0, endIndex:8);
            feb = paddedValue.substring(beginIndex:8, endIndex:16);

            // Update the array to reflect changes
            updateMonateArray();
        }
    }

    public static void main(String[] args) {
        CobolData data = new CobolData();

        // DISPLAY #1
        System.out.println(data.counter);

        // MOVE 84 to counter
        data.counter = 84;

        // DISPLAY #2
        System.out.println(data.counter);

        // DISPLAY #3 (Note: COBOL arrays are 1-based, Java are 0-based)
        System.out.println(data.monate[8]); // September (index 8 = month 9)

        // MOVE "------" TO monate(9)
        data.monate[8] = "------  "; // Pad to 8 characters

        // DISPLAY #4
        System.out.println(data.monate[8]);

        // MOVE "Monat 01Monat 02" TO my-struct
        data.moveToMyStruct(value:"Monat 01Monat 02");

        // DISPLAY #5
        System.out.println(data.feb);

        // DISPLAY #6
        System.out.println(data.monate[1]); // February (index 1 = month 2)

        // DISPLAY #7
        System.out.println(data.monate[2]); // March (index 2 = month 3)
    }
}
```

Falsches Ergebnis:

```
42
84
September
------
Monat 02
Monat 02
Maerz
```

The data type errors mentioned with ChatGPT-4o also exist in this case. This time, the **`byte`** type is selected, but its value range does not match that of COBOL PIC 9.

In Java, the `MOVE "Monat 01Monat02" TO my-struct` command is implemented by calling the `moveToMyStruct` method, which, however, only assigns the first two fields correctly, while the initialisation of the rest with spaces is omitted. Therefore, as with ChatGPT-4o, the last output line is **Maerz** and not empty, as is the case with the COBOL programme.

Example 2 (DeepSeek (v3.1)): migration test no. 2

Konvertierung 2

BeispielProgramm.java

```java
import java.util.Arrays;

public class BeispielProgramm {

    // COBOL-like data structure with fixed-length strings
    private static class CobolData {
        public byte counter = 42; // PIC 9 equivalent

        // Individual month fields with fixed length (8 chars)
        public String jan = padToLength(str:"Januar", length:8);
        public String feb = padToLength(str:"Februar", length:8);
        public String mar = padToLength(str:"Maerz", length:8);
        public String apr = padToLength(str:"April", length:8);
        public String mai = padToLength(str:"Mai", length:8);
        public String jun = padToLength(str:"Juni", length:8);
        public String jul = padToLength(str:"Juli", length:8);
        public String aug = padToLength(str:"August", length:8);
        public String sep = padToLength(str:"September", length:8);
        public String okt = padToLength(str:"Oktober", length:8);
        public String nov = padToLength(str:"November", length:8);
        public String dez = padToLength(str:"Dezember", length:8);

        // Array that redefines the structure (COBOL OCCURS)
        public String[] monate = new String[12];

        public CobolData() {
            // Initialize the array with the month values
            updateMonateArray();
        }

        // Update the array to reflect current field values
        private void updateMonateArray() {
            monate[0] = jan;
            monate[1] = feb;
            monate[2] = mar;
            monate[3] = apr;
            monate[4] = mai;
            monate[5] = jun;
            monate[6] = jul;
            monate[7] = aug;
            monate[8] = sep;
            monate[9] = okt;
            monate[10] = nov;
            monate[11] = dez;
        }

        // Update individual fields from array changes
        private void updateFieldsFromArray() {
            jan = monate[0];
            feb = monate[1];
            mar = monate[2];
            apr = monate[3];
            mai = monate[4];
            jun = monate[5];
            jul = monate[6];
            aug = monate[7];
            sep = monate[8];
            okt = monate[9];
            nov = monate[10];
            dez = monate[11];
        }

        // COBOL MOVE equivalent for the entire structure
        public void moveToMyStruct(String value) {
            // Pad to exactly 96 characters (12 months * 8 chars)
            String paddedValue = padToLength(value, length:96);

            // Update the array with the new values
            for (int i = 0; i < 12; i++) {
                int start = i * 8;
                int end = start + 8;
                monate[i] = paddedValue.substring(start, Math.min(end, paddedValue.length()));
            }

            // Update individual fields from the array
            updateFieldsFromArray();
        }

        // COBOL MOVE equivalent for array element
        public void moveToMonate(int index, String value) {
            // COBOL uses 1-based indexing, Java uses 0-based
            int javaIndex = index - 1;
            if (javaIndex >= 0 && javaIndex < 12) {
                monate[javaIndex] = padToLength(value, length:8);
                updateFieldsFromArray(); // Update individual fields
            }
        }

        // Helper method to pad strings to fixed length
        private String padToLength(String str, int length) {
            if (str.length() > length) {
                return str.substring(beginIndex:0, length);
            } else if (str.length() < length) {
                return String.format("%-" + length + "s", str);
            }
            return str;
        }
    }

    public static void main(String[] args) {
        CobolData data = new CobolData();

        // DISPLAY #1
        System.out.println(data.counter);

        // MOVE 84 to counter
        data.counter = 84;

        // DISPLAY #2
        System.out.println(data.counter);

        // DISPLAY #3 - monate(9) (September, index 8 in 0-based)
        System.out.println(data.monate[8]);

        // MOVE "------" TO monate(9)
        data.moveToMonate(index:9, value:"------");

        // DISPLAY #4
        System.out.println(data.monate[8]);

        // MOVE "Monat 01Monat 02" TO my-struct
        // This will overwrite the first 16 characters (2 months)
        data.moveToMyStruct(value:"Monat 01Monat 02");

        // DISPLAY #5 - feb should show "Monat 02"
        System.out.println(data.feb);

        // DISPLAY #6 - monate(2) should show "Monat 02" (feb)
        System.out.println(data.monate[1]);

        // DISPLAY #7 - monate(3) should show original "Maerz" but padded
        System.out.println(data.monate[2]);
    }
}
```

Falsches Ergebnis:

```
42
84
Septembe
------
Monat 02
Monat 02
```

The type of `counter` (**`byte`**) is incorrect here as well. `String` is also used for the month names, but this time, during assignment or initialisation, it is truncated or filled up to the correct length using the `padToLength` method.

The `MOVE "Monat 01Monat02" TO my-struct` is also implemented correctly, but with a relatively large amount of Java code. If this happens frequently, the Java programme quickly becomes large and difficult to handle.