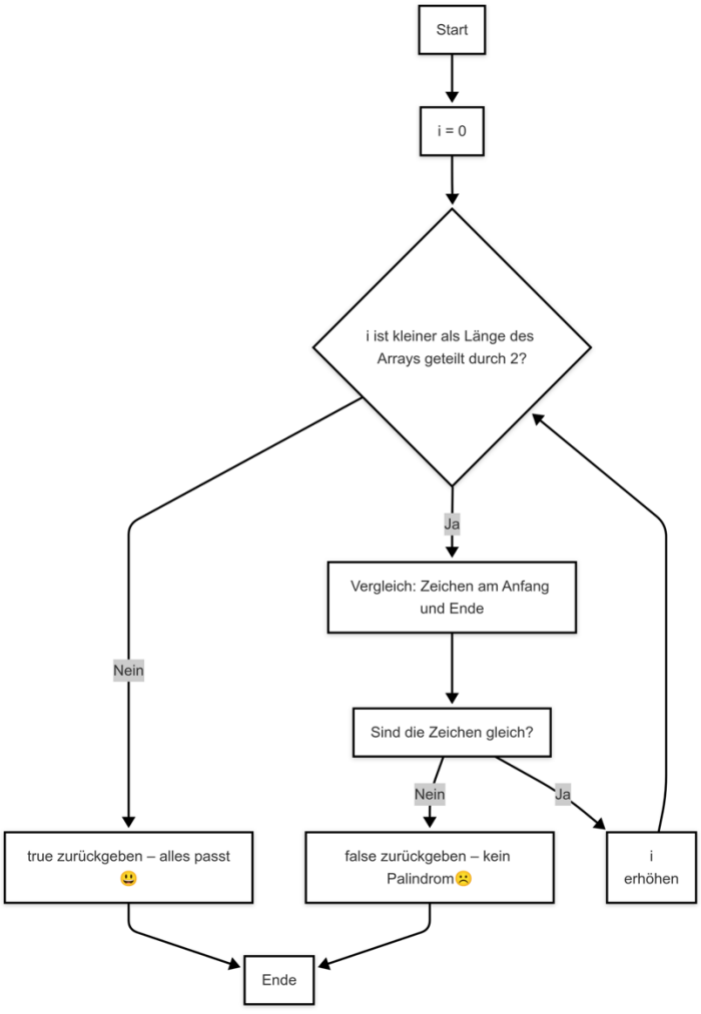
Aufgabe a3x1

```
public boolean isPalindrome(char[] arrayOfChars) {

   for (int i = 0; i < arrayOfChars.length/2; i++) {
         if(arrayOfChars[i]== arrayOfChars[arrayOfChars.length-1-i]) {
         }
         else return false;
      }

     return true;
   }
```



Beschreibung:

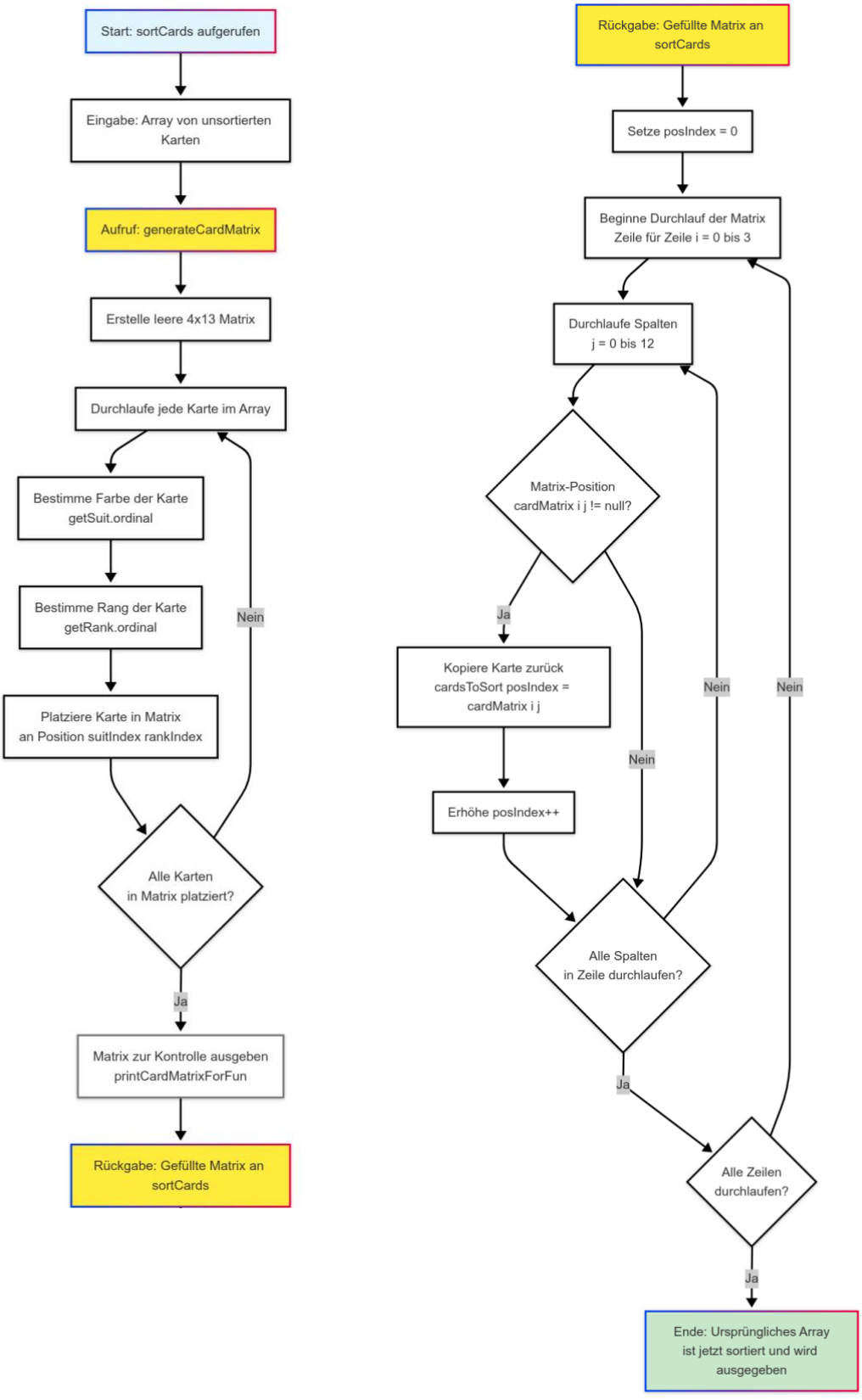
- Nach der Definition ist ein Wort ein Palindrom, wenn die Buchstaben an den Stellen ***n*** und ***(Wortlänge - 1 - n)*** übereinstimmen. Daraus folgt, dass man nur die ersten ***(Wortlänge / 2)*** Buchstabenpaare vergleichen muss;
- Sollte an den gennanten Positionen kein gleiches Buchstabenpaar stehen, ist dieses Wort kein Palindrom und somit wird sofort falsch ausgegeben, ohne die restlichen Buchstabenpaare züberprüfen.

Aufgabe a3x2

```
public static Card[][] generateCardMatrix (Card[] givenCards) {
        Card[][] cardMatrix = new Card[4][13];
    for (Card card : givenCards) {
        int suitIndex = card.getSuit().ordinal();
        int rankIndex = card.getRank().ordinal();
        cardMatrix[suitIndex][rankIndex] = card;
    }//for
    printCardMatrixForFun(cardMatrix);
    return cardMatrix;
  }//method
    public static void sortCards(Card[] cardsToSort) {
        Card[][] cardMatrix = generateCardMatrix(cardsToSort);
        int posIndex = 0;
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 13; j++) {
                if (cardMatrix[i][j]!=null) {
                    cardsToSort[posIndex] = cardMatrix[i][j];
                    posIndex++;
                }//if
            }//for inside for
        }//for
    }//method
```

Beschreibung:

- Die Methode *sortCards* wird aufgerufen. Dabei wird eine Referenz auf das unsortierte Kartenarray übergeben;
- Innerhalb der Methode wird die Hilfsmethode *generateCardMatrix* aufgerufen, der ebenfalls die Referenz auf das unsortierte Array übergeben wird;
- Da sowohl Rank als auch Suit einen Index besitzen, können die Karten entsprechend in einem zweidimensionalen Array (*cardMatrix*) einsortiert werden;
- Nach dem Befüllen wird die sortierte *cardMatrix* an *sortCards* zurückgegeben;
- Die Methode *sortCards* durchläuft alle Zeilen und Spalten der *cardMatrix.* Wenn sich an einer Stelle eine Karte befindet (!= null), wird sie an der aktuellen Position im ursprünglichen Array *cardsToSort* gespeichert.


Start: sortCards aufgerufen
Eingabe: Array von unsortierten Karten
Aufruf: generateCardMatrix
Erstelle leere 4x13 Matrix
Durchlaufe jede Karte im Array
Bestimme Farbe der Karte getSuit.ordinal
Bestimme Rang der Karte getRank.ordinal
Platziere Karte in Matrix an Position suitIndex rankIndex
Alle Karten in Matrix platziert?
Nein
Ja
Matrix zur Kontrolle ausgeben printCardMatrixForFun
Rückgabe: Gefüllte Matrix an sortCards
Rückgabe: Gefüllte Matrix an sortCards
Setze posIndex = 0
Beginne Durchlauf der Matrix Zeile für Zeile i = 0 bis 3
Durchlaufe Spalten j = 0 bis 12
Matrix-Position cardMatrix i j != null?
Ja
Kopiere Karte zurück cardsToSort posIndex = cardMatrix i j
Erhöhe posIndex++
Nein
Nein
Nein
Alle Spalten in Zeile durchlaufen?
Ja
Alle Zeilen durchlaufen?
Ja
Ende: Ursprüngliches Array ist jetzt sortiert und wird ausgegeben

Aufgabe a3x3

```
public long process( long[][] theArray ){
    int patternHeight = 2; int patternWidth = 2;
    int patternSize = patternHeight + patternWidth;
    int startPosition = 0;
    int minMiddleRowLength, currentRowLength, minEdgeRowLength;
    int limitLength;
    long totalSum = 0;
    while (startPosition+patternSize <= theArray.length) {
       currentRowLength = 0;
       limitLength = 0;
       minMiddleRowLength = Integer.MAX_VALUE;
       minEdgeRowLength = Integer.MAX_VALUE;
       for (int i = startPosition; i<patternSize+startPosition; i++) {
          if (i - startPosition == 1 || i - startPosition == 2) {
             currentRowLength = theArray[i].length;
               if (currentRowLength<minMiddleRowLength) {
                  minMiddleRowLength = currentRowLength;
             }//if inside if
          }//if inside if and for
          else {
             currentRowLength = theArray[i].length;
              if (currentRowLength < minEdgeRowLength) {
                 minEdgeRowLength = currentRowLength;
             }//if inside else
          }//else
       }//for
       if (minEdgeRowLength >= minMiddleRowLength) {
          limitLength = minMiddleRowLength;
       }//if
       else { limitLength = minEdgeRowLength+1;
       }
     if (limitLength >= 4) {
       for (int j = 0; j <= limitLength-patternSize; j++) {
          for (int i = 0; i < patternWidth; i++) {
```

```
            totalSum += theArray[startPosition][i+j+1];
            totalSum += theArray[startPosition+patternSize-1][i+j+1];
          }// for inside for and if(musterSiteLength)
          for (int i = 0; i < patternHeight; i++ ) {
            totalSum += theArray[startPosition+1+i][j];
            totalSum += theArray[startPosition+1+i][j+patternSize-1];
          }// for inside for and if (musterSiteHeigth)
        }//for inside if
      }//if
       startPosition++;
  }//while
 return totalSum;
}
```

Beschreibung:

- Die Größe des Musters (2 Zeilen × 2 Spalten) wird festgelegt;
- Es wird geprüft, ob ab der aktuellen Startposition genügend Zeilen im Array vorhanden sind – also mindestens so viele wie die Musterhöhe. Ist das nicht der Fall, wird nichts weiter ausgeführt;
- Für alle Zeilen, die vom Muster betroffen sind, wird die Länge (Anzahl der Spalten) überprüft. Dabei wird die jeweils kürzeste Zeile gespeichert;
- Falls die kürzeste dieser Zeilen groß genug ist (mindestens so viele Spalten wie die Musterbreite), wird die Verarbeitung fortgesetzt;
- An den Positionen, die dem Muster entsprechen, werden die Zahlen ausgewählt und aufaddiert.


Start: process aufgerufen
Eingabe: 2D Array theArray
Initialisiere Pattern-Parameter patternHeight=2, patternWidth=2
Setze startPosition=0 totalSum=0
startPosition + patternSize <= theArray.length?
Nein
Ja
Ende: Rückgabe totalSum
Initialisiere Variablen für aktuelle Position
Durchlaufe Pattern-Zeilen von startPosition bis patternSize
Ist aktuelle Zeile mittlere Zeile? (Position 1 oder 2)
Ja
Nein
Bestimme minimale Mittlere-Zeilen-Länge
Bestimme minimale Rand-Zeilen-Länge
Bestimme minimale Mittlere-Zeilen-Länge
Bestimme minimale Rand-Zeilen-Länge
Berechne limitLength basierend auf minimalen Längen
limitLength > 0?
Ja
Durchlaufe horizontale Positionen j=0 bis limitLength-patternSize
Summiere obere und untere Pattern-Kanten horizontal
Summiere linke und rechte Pattern-Kanten vertikal
Alle horizontalen Positionen durchlaufen?
Nein
Nein
Ja
Erhöhe startPosition++