Aufgabe 10-1

```
public class Algorithmen {

    public static boolean istQuadratzahl(int zahl) {
        for (int i = 1; i <= zahl; i++) {
            if (i * i == zahl) {
                return true;
            }
        }
        return false;
    }


    public static int[] fib(int n) {
        int[] result = new int[n];
        int a = 0;
        int b = 1;
        for (int i = 0; i < n; i++) {
            result[i] = a;
            a = b;
            b = result[i] + b;
        }
        return result;
    }

    public static int[] countElements(int[] werte, int max, int numBuckets) {
        int[] result = new int[numBuckets];
        int werteProBucket = max / numBuckets;
        for (int i = 0; i < werte.length; i++) {
            result[werte[i] / werteProBucket] += 1;
        }
        return result;
    }
}
```

Aufgabe 10-2

```
public class Matrix {

    static int[][] matrix = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9},
        {10, 11, 12}
    };

    public static boolean gleich(int[] a1, int[] a2) {
        if (a1.length != a2.length) {
            return false;
        }
        boolean gleich = true;
        for (int i = 0; i < a1.length && gleich; i++) {
            gleich &= a1[i] == a2[i];
        }
        return gleich;
    }

    public static boolean enthaeltZeile(int[][] matrix, int[] zeile) {
        boolean enthaelt = false;
        for (int i = 0; i < matrix.length && !enthaelt; i++) {
            enthaelt = enthaelt || gleich(matrix[i], zeile);
        }
        return enthaelt;
    }

    public static boolean enthaeltSpalte(int[][] matrix, int[] spalte) {
        if (matrix.length != spalte.length || matrix.length == 0) {
            return false;
        }
        boolean enthaelt = false;

        int[] spalte_j = new int[matrix.length];
        for (int j = 0; j < matrix[0].length && !enthaelt; j++) {
            for (int i = 0; i < matrix.length; i++) {
                spalte_j[i] = matrix[i][j];
            }
            enthaelt = enthaelt || gleich(spalte_j, spalte);
        }
        return enthaelt;
    }

    public static int[][] transponieren(int[][] matrix) {
        int[][] result = new int[matrix[0].length][matrix.length];
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                result[j][i] = matrix[i][j];
            }
        }
        return result;
    }

}
```

Aufgabe 10-3

```
/**
 * Klasse Addition dient zur Addition zweier nat&uuml;rlicher Zahlen in
 * p-adischer Darstellung f&uuml;r 2 &le; p &le; 62.
 *
 * @author Arthur Zimek
 */
public class Addition {

    /**
     * Das Alphabet zur Darstellung der Zahlen. Die Zeichen in diesem Alphabet
     * sind die Ziffern 0-9 gefolgt von den Buchstaben A-Z gefolgt von den
     * Buchstaben a-z.
     */
    public static final char[] ALPHABET = alphabet();

    /**
     * Erzeugt ein Alphabet als Array von Zeichen der Form
     * 0,1,2,...,9,A,B,C,...,Z,a,b,...,z.
     *
     * @return Array von Zeichen der Form 0,1,2,...,9,A,B,C,...,Z,a,b,...,z
     */
    public static char[] alphabet() {
        int anzahlDezimalZiffern = 10;
        int anzahlGrossBuchstaben = 'Z' - 'A' + 1;
        int anzahlKleinBuchstaben = 'z' - 'a' + 1;
        char[] alphabet = new char[anzahlDezimalZiffern + anzahlGrossBuchstaben
                + anzahlKleinBuchstaben];
        // Initialisieren der Ziffern von 0 bis 9
        for (int i = 0; i < anzahlDezimalZiffern; i++) {
            alphabet[i] = (char) ('0' + i);
        }
        // Initialisieren der Ziffern von A bis Z
        for (int i = 0; i < anzahlGrossBuchstaben; i++) {
            alphabet[i + anzahlDezimalZiffern] = (char) ('A' + i);
        }
        // Initialisieren der Ziffern von a bis z
        for (int i = 0; i < anzahlKleinBuchstaben; i++) {
            alphabet[i + anzahlGrossBuchstaben + anzahlDezimalZiffern]
                    = (char) ('a' + i);
        }
        return alphabet;
    }
```

```
/**
 * Berechnet die Summe zweier Zahlen in p-adischer Darstellung zur Basis 2
 * &le; p &le; 62.
 *
 * @param ziffern1 Darstellung des ersten Summanden
 * @param ziffern2 Darstellung des zweiten Summanden
 * @param p Basis der Darstellungen {@code ziffern1} und {@code ziffern2} (2
 * &le; p &le; 62)
 * @return Das Ergebnis der Addition von {@code ziffern1} und
 * {@code ziffern2} zur Basis {@code p} ohne f&uumlhrende Nullen
 */
public static char[] summe(char[] ziffern1, char[] ziffern2, int p) {
    int argumentLaenge = Math.max(ziffern1.length, ziffern2.length);
    char[] zwischenergebnis = new char[argumentLaenge + 1];
    int uebertrag = 0;

    for (int i = 1; i <= argumentLaenge; i++) {
        char ziffer1 = i <= ziffern1.length
                ? ziffern1[ziffern1.length - i] : ALPHABET[0];
        char ziffer2 = i <= ziffern2.length
                ? ziffern2[ziffern2.length - i] : ALPHABET[0];

        int ziffer1Wert = binaereSuche(ALPHABET, ziffer1);
        int ziffer2Wert = binaereSuche(ALPHABET, ziffer2);
        // Die Verwendung von
        // java.util.Arrays.binarySearch(ALPHABET,zifferX)
        // waere auch moeglich!

        int summe = ziffer1Wert + ziffer2Wert + uebertrag;
        zwischenergebnis[zwischenergebnis.length - i] = ALPHABET[summe % p];
        uebertrag = summe / p;
    }
    zwischenergebnis[0] = ALPHABET[uebertrag];

    int anzahlFuehrendeNullen = 0;
    while (anzahlFuehrendeNullen < zwischenergebnis.length - 1
            && zwischenergebnis[anzahlFuehrendeNullen] <= ALPHABET[0]) {
        anzahlFuehrendeNullen++;
    }

    char[] ergebnis = new char[zwischenergebnis.length - anzahlFuehrendeNullen];
    for (int i = 0; i < zwischenergebnis.length - anzahlFuehrendeNullen; i++) {
        ergebnis[i] = zwischenergebnis[i + anzahlFuehrendeNullen];
    }

    return ergebnis;
}
```

```
    /**
     * Bin&auml;re Suche in einem sortierten Array {@code a} nach dem Eintrag {@code q}.
     *
     * @param a aufsteigend sortiertes Array
     * @param q Anfrage
     * @return Index des Eintrags {@code q}, falls in {@code a} vorhanden, andernfalls -1
     */
    public static int binaereSuche(char[] a, char q) {
        int first = 0;
        int last = a.length - 1;
        while (last >= first) {
            int m = (last + first) / 2;
            if (q == a[m]) {
                return m;
            } else if (q < a[m]) {
                last = m - 1;
            } else { // q > a[m]
                first = m + 1;
            }
        }
        return -1; // Element nicht gefunden
    }

    /**
     * Addiert zwei Zahlen mit Basis 2 &le; p &le; 62 und gibt das Ergebnis der
     * Addition ohne f&uuml;hrende Nullen aus.
     *
     * G&uuml;ltige Ziffern für String-Darstellungen sind die ersten {@code p}
     * Zeichen aus dem Alphabet {@link #ALPHABET}.
     *
     * @param args drei Argumente werden erwartet:
     * <ul>
     * <li>{@code args[0]} : String-Darstellung einer nat&uuml;rlichen Zahl mit
     * Basis p</li>
     * <li>{@code args[1]} : String-Darstellung einer nat&uuml;rlichen Zahl mit
     * Basis p</li>
     * <li>{@code args[2]} : String-Darstellung einer nat&uuml;rlichen Zahl mit
     * Basis 10 als Basis p der zu addierenden Zahlen</li>
     * </ul>
     */
    public static void main(String[] args) {
        if (args.length != 3) {
            System.out.println("Erwartete Eingabe:\n"
                    + " java Addition s1 s2 p\n\n"
                    + "    Argumente:\n"
                    + "    s1: String-Darstellung einer nat\u00FCrlichen Zahl mit Basis p\n"
                    + "    s2: String-Darstellung einer nat\u00FCrlichen Zahl mit Basis p\n"
                    + "    p:  String-Darstellung mit Basis 10 einer nat\u00FCrlichen Zahl als
                                                    Basis von s1 und s2 (2 <= p <= 62)\n"
                    + "\nG\u00FCltige Zeichen f\u00FCr die String-Darstellungen sind "
                    + "jeweils die ersten p Zeichen aus einem Alphabet mit "
                    + "Zeichen in der Reihenfolge 0,1,...,9,A,B,C,...,Z,a,b,c,...,z.");
        } else {
            String s1 = args[0];
            String s2 = args[1];
            int p = Integer.parseInt(args[2]);
            char[] ziffern1 = s1.toCharArray();
            char[] ziffern2 = s2.toCharArray();

            String ergebnis = String.copyValueOf(summe(ziffern1, ziffern2, p));
            System.out.println(ergebnis);
        }
    }
}
```