Aufgabe 9-1

```
public class ArrayProgrammierung {

    public static int summe(int[] array) {
        int count = 0;
        for (int i = 0; i < array.length; i++) {
            count = count + array[i];
        }
        return count;
    }

    public static int min(int[] array) throws Exception {
        // Falls das Array leer ist, keine Berechnung von Minimum möglich.
        // --> Fehlerbehandlung durch Exception (werden noch in der Vorlesung vorgestellt!)
        if (array.length == 0) {
            throw new Exception("Das Array ist leer. Kein Minimum vorhanden.");
        }

        // Ansonsten berechne das Minimum
        int minimum = array[0];
        for (int i = 1; i < array.length; i++) {
            if (array[i] < minimum) {
                minimum = array[i];
            }
        }
        return minimum;
    }

    public static int[] palindrom(int[] array) {
        int[] palindrom = new int[array.length];
        for (int i = 0; i < array.length; i++) {
            palindrom[palindrom.length - 1 - i] = array[i];
        }
        return palindrom;
    }

    public static void main(String[] args) {
        // Mit try-catch werden „Exceptions“ abgefangen.
        // Damit sind Fehlerbehandlungen möglich. Mehr dazu später in der Vorlesung!
        try {
            System.out.println("summe(7,4,8,42,13) = " + summe(new int[]{7, 4, 8, 42, 13}));
            System.out.println("min(7,4,8,42,13) = " + min(new int[]{7, 4, 8, 42, 13}));
            System.out.print("palindrom(7,4,8,42,13) = ");
            printArray(palindrom(new int[]{7, 4, 8, 42, 13}));
        } catch (Exception ex) {
            System.out.println("Es ist ein Fehler aufgetreten.");
        }
    }

    public static void printArray(int[] a) {
        if (a.length == 0) {
            return;
        }
        System.out.print("[");
        for (int i = 0; i < a.length; i++) {
            System.out.print(a[i]);
            if (i != a.length - 1) {
                System.out.print(",");
            }
        }
        System.out.println("]");
    }
}
```

Aufgabe 9-2

```
public class Klausurpunkte {

    public static int[][] punkte = new int[42][8];

    public static void main(String[] args) {
        setPunkte(21, 0, 6);
        System.out.println(getPunkte(21, 0));           // Ausgabe: 6
        setPunkte(21, 1, 3);
        setPunkte(21, 2, 5);
        setPunkte(21, 3, 7);
        setPunkte(21, 4, 11);
        setPunkte(21, 5, 2);
        setPunkte(21, 6, 0);
        setPunkte(21, 7, 5);
        System.out.println(gesamtPunktezahl(21));       // Ausgabe: 39
        setPunkte(12, 4, 5);
        setPunkte(34, 4, 5);
        System.out.println(durchschnittsPunktezahl(4)); // Ausgabe: 0.5
    }

    /**
     * Gibt die Punktezahl einer bestimmten Aufgabe für einen bestimmten
     * Studenten zurück.
     *
     * @param student Spezifiziert die Nummer des Studenten dessen Punktezahl
     * zurückgegeben werden soll.
     * @param aufgabe Spezifiziert die Nummer der Aufgabe deren Punktezahl
     * zurückgegeben werden soll.
     *
     * @return Die von Student student bei Aufgabe aufgabe erreichten Punkte.
     */
    public static int getPunkte(int student, int aufgabe) {
        return punkte[student][aufgabe];
    }

    /**
     * Setzt die Punktezahl einer bestimmten Aufgabe für einen bestimmten
     * Studenten auf einen bestimmten Wert.
     *
     * @param student Spezifiziert die Nummer des Studenten dessen Punktezahl
     * gesetzt werden soll.
     * @param aufgabe Spezifiziert die Nummer der Aufgabe deren Punktezahl
     * gesetzt werden soll.
     * @param punkte Spezifiziert die Anzahl der erreichten Punkte.
     */
    public static void setPunkte(int student, int aufgabe, int punkte) {
        punkte[student][aufgabe] = punkte;
    }
```

```
    /**
     * Berechnet die Summe von Übungspunkten von Student student.
     *
     * @param student Spezifiziert die Nummer des Studenten dessen
     * Gesamtpunktezahl zurückgegeben werden soll.
     * @return Die Summe aller Übungspunkte von Student student.
     */
    public static int gesamtPunktezahl(int student) {
        int[] punkteStudent = punkte[student];
        int pkt = 0;
        for (int i = 0; i < punkteStudent.length; i++) {
            pkt = pkt + punkteStudent[i];
        }
        return pkt;
    }

    /**
     * Berechnet die Anzahl der Punkte die im Schnitt von allen Studenten bei
     * der spezifizierten Aufgabe aufgabe erreicht wurden.
     *
     * @param aufgabe Spezifiziert die Aufgabe deren durchschnittlich erreichte
     * Punktezahl zurückgegeben werden soll.
     * @return Die im Durchschnitt erreichte Punktezahl bei Aufgabe aufgabe.
     */
    public static double durchschnittsPunktezahl(int aufgabe) {
        int pkt = 0;
        for (int i = 0; i < punkte.length; i++) {
            pkt = pkt + punkte [i][aufgabe];
        }
        return pkt / (punkte.length * 1.0);
    }
}
```

Aufgabe 9-3

```
/**
 * Medoid Klasse. Diese Klasse dient zur Berechnung des Medoids einer Menge von
 * Zahlen.
 */
public class Medoid {

    /**
     * Die Methode medoid berechnet den Medoid eines Arrays vom Typ double. Der
     * Medoid ist die Zahl, die dem Mittelwert aller Zahlen am n&auml;chsten
     * liegt.
     *
     * @param numbers Die Menge von Zahlen, deren Medoid berechnet werden soll.
     * @return Gibt den Medoid der Eingabemenge von Zahlen zur&uuml;ck
     */
    public static double medoid(double[] numbers) {
        double sum = 0;
        for (int i = 0; i < numbers.length; i++) {
            sum = sum + numbers[i];
        }
        double mean = sum / numbers.length;
        double diff;
        double minDiff = Math.abs(numbers[0] - mean);
        int index = 0;
        for (int i = 1; i < numbers.length; i++) {
            diff = Math.abs(numbers[i] - mean);

            if (diff < minDiff) {
                minDiff = diff;
                index = i;
            }
        }
        return numbers[index];
    }

    /**
     * Die Methode main erh&auml;lt eine beliebige Menge von Parametern vom Typ
     * String von der Konsole, wandelt diese in Array vom double um und gibt den
     * Medoid dieses Arrays auf der Konsole aus. Achtung: Es d&uuml;rfen als
     * Parameter nur Strings angegeben werden, die als double geparst werden
     * k&ouml;nnen {@link Double#parseDouble(String)}.
     *
     * @param args Array, dessen Medoid berechnet werden soll.
     */
    public static void main(String[] args) {
        double[] numbers = new double[args.length];

        for (int i = 0; i < args.length; i++) {
            numbers[i] = Double.parseDouble(args[i]);
        }

        System.out.println(medoid(numbers));
    }
}
```

Aufgabe 9-5

```
/**
 * Klasse zur Bestimmung der Primfaktorzerlegung.
 *
 * @author Arthur Zimek
 */
public class Primfaktoren {

    /**
     * Gibt die Primfaktorzerlegung fuer eine gegebene Zahl > 1 aus.
     *
     * @param n eine Zahl > 1, deren Primfaktorzerlegung ausgegeben werden soll
     */
    public static void printPrimfaktoren(int n) {
        System.out.print(n + " = ");
        int p = 2;
        while (p <= n) {
            if (n % p == 0) {
                System.out.print(p);
                n /= p;
                if (p <= n) {
                    System.out.print(" * ");
                }
            } else {
                p++;
            }
        }
        System.out.println();
    }


    /**
     * Gibt die Primfaktorzerlegung fuer eine gegebene Zahl > 1 aus.
     *
     * Als Teiler werden hierbei nur Primzahlen getestet.
     *
     * @param n eine Zahl > 1, deren Primfaktorzerlegung ausgegeben werden soll
     */
    public static void printGesiebtePrimfaktoren(int n) {
        System.out.print(n + " = ");
        int p = 2;
        boolean[] primzahlen = Primzahlen.siebDesEratosthenes(n);
        while (p <= n && p < primzahlen.length) {
            if (primzahlen[p] && n % p == 0) {
                System.out.print(p);
                n /= p;
                if (p <= n) {
                    System.out.print(" * ");
                }
            } else {
                p++;
            }
        }
        System.out.println();
    }
```

```
    /**
     * Liest das erste Argument ein, erzeugt daraus einen Integer und gibt
     * dessen Primfaktorzerlegung durch zwei unterschiedliche Methoden aus.
     *
     * Die Verwendung der Methode {@link #printGesiebtePrimfaktoren(int)}
     * begrenzt die Anwendbarkeit st&auml;rker hinsichtlich des verf&uuml;gbaren
     * Speicherplatzes.
     *
     * @param args das erste Element <code>args[0]</code> wird als Integer
     * interpretiert
     */
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Erwartet einen Parameter: "
                    + "Zahl, deren Primfaktoren zu ermitteln sind.");
        } else {
            int argument = Integer.parseInt(args[0]);
            printPrimfaktoren(argument);
            printGesiebtePrimfaktoren(argument);
        }
    }
}


/**
 * Die Klasse Primzahlen stellt das Sieb des Eratosthenes zu Verf&uuml;gung.
 *
 * @author Arthur Zimek
 */
public class Primzahlen {

    /**
     * Implementiert den Algorithmus &quot;Sieb des Eratosthenes&quot;.
     *
     * Das resultierende boolesche Array der L&auml;nge <code>n+1</code>
     * enth&auml;lt den Wert <code>true</code> genau an den Stellen, deren
     * Indizes Primzahlen sind.
     *
     * @param n Maximum der gesuchten Primzahlen
     * @return Array der L&auml;nge <code>n+1</code> mit dem Wert
     * <code>true</code> an genau den Feldern, deren Index eine Primzahl ist
     */
    public static boolean[] siebDesEratosthenes(int n) {
        boolean[] primzahlen = new boolean[n + 1];
        for (int i = 2; i < primzahlen.length; i++) {
            primzahlen[i] = true;
        }
        for (int i = 0; i < primzahlen.length; i++) {
            if (primzahlen[i]) {
                int m = 2;
                while (i * m < primzahlen.length) {
                    primzahlen[i * m] = false;
                    m++;
                }
            }
        }
        return primzahlen;
    }
```

```
    /**
     * Gibt die Primzahlen &le; <code>n</code> aus.
     *
     *
     * @param n Maximum der gesuchten Primzahlen
     */
    public static void printPrimzahlen(int n) {
        boolean[] primzahlen = siebDesEratosthenes(n);
        System.out.println("Die Primzahlen <= " + n + " sind:");
        for (int i = 0; i < primzahlen.length; i++) {
            if (primzahlen[i]) {
                System.out.println(i);
            }
        }
    }

    /**
     * Liest das erste Argument ein, erzeugt daraus einen Integer und gibt alle
     * Primzahlen aus, die kleiner-gleich dieser Zahl sind.
     *
     * @param args das erste Element <code>args[0]</code> wird als Integer
     * interpretiert
     */
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Erwartet einen Parameter: "
                    + "Maximum der gesuchten Primzahlen");
        } else {
            printPrimzahlen(Integer.parseInt(args[0]));
        }
    }

}
```