Semestralklausur Einführung in die Programmierung, WS 2007/08, 28.1.2008	Seite 1/5
Name, Vorname, Matrikelnummer:	Unterschrift:

Gruppe A

Aufgabe 1 (12 Punkte)

Schreiben Sie eine Klassenmethode mit einem Parameter n vom Typ `int`, von dem Sie annehmen dürfen, dass er eine positive Zahl enthält, und Ergebnistyp `boolean`; das Ergebnis soll genau dann `true` sein, wenn n eine Primzahl ist.
Probieren Sie dazu für alle ganzen Zahlen $2 \leq i \leq \sqrt{n}$ aus, ob i Teiler von n ist (`n % i` dürfte hierbei eine nützliche Größe sein).

```
static boolean istPrim(int n) {
    int i = 2;
    if (n == 1) return false;
    while (i*i <= n) {
        if (n%i == 0) return false;
        i++;
    }
    return true;
}
```

Schreiben Sie eine weitere Klassenmethode mit einem Parameter n vom Typ `int`, von dem Sie wieder annehmen dürfen, dass er eine positive Zahl enthält. Diese Methode soll kein Funktionsergebnis haben, sondern soll von den Zahlen $1 \ldots n$ genau diejenigen ausdrucken, die Primzahl sind.

```
static void erstePrimzahlen(int n) {
    for (int i=2; i<=n; i++) {
        if (istPrim(i)) {
            System.out.println(i);
        }
    }
}
```

Aufgabe 2 (14 Punkte)

Schreiben Sie eine Klassenmethode mit einem Parameter a vom Typ „Feld von `int`“ und einem Parameter n vom Typ `int`, wobei Sie annehmen dürfen, dass n ein zulässiger Index des Feldes a ist. Das Ergebnis soll angeben, wieviele aufeinanderfolgende Komponenten von a, beginnend mit a_n, aufsteigend sortiert sind, also das maximale i mit $a_j \leq a_k$ für alle $n \leq j \leq k < n + i$.
Beispiel: Für das Feld `{1,2,3,2,2,2,2,3,2,1}` würden sich in Abhängigkeit von n folgende Werte ergeben:

n	0	1	2	3	4	5	6	7	8	9
Ergebnis	3	2	1	5	4	3	2	1	1	1

```
static int sortAbschnitt(int[] a, int n) {
    int i=1;
    while (n+i<a.length && a[n+i]>=a[n+i-1]) i++;
    return i;
}
```

Schreiben Sie nun eine Klassenmethode mit einem Parameter a vom Typ „Feld von `int`“, die die Länge des längsten aufsteigend sortierten Abschnitts von a bestimmt (im Beispiel also 5), indem Sie für jeden Index von a die Methode aus der vorigen Teilaufgabe aufrufen und das Maximum bestimmen (diese Aufgabe ist also auch dann vollständig lösbar, wenn Sie die vorige Teilaufgabe nicht bearbeitet haben).

```
static int laengsterAbschnitt(int[] a) {
    int max=0;
    for (int i=0; i<a.length; i++) {
        int s = sortAbschnitt(a, i);
        if (s>max) max = s;
    }
    return max;
}
```

Verbessern Sie die Methode aus der vorigen Teilaufgabe, indem Sie für solche Indizes, für die Sie aus den bis dahin angefallenen Ergebnissen schon wissen, dass der längste aufsteigend sortierte Abschnitt hier *nicht* beginnt, die Länge des zugehörigen Abschnitts gar nicht mehr bestimmen.

```
static int laengsterAbschnittEff(int[] a) {
    int max=0, s;
    for (int i=0; i<a.length-max; i = i+s) {
        s = sortAbschnitt(a, i);
        if (s>max) max = s;
    }
    return max;
}
```

Aufgabe 3 (14 Punkte)

Schreiben Sie eine Klasse zur Darstellung von Quadern $[x_0, x_1] \times [y_0, y_1] \times [z_0, z_1]$ mit ganzzahligen Koordinaten $x_0, x_1, y_0, y_1, z_0, z_1 \in \mathbb{Z}$.
Deklarieren Sie Instanzvariablen für die Koordinaten, und schreiben Sie einen Konstruktor mit 6 ganzzahligen Parametern, die den Werten x_0, x_1, y_0, y_1, z_0 und z_1 (in dieser Reihenfolge) entsprechen sowie einen parameterlosen Konstruktor, der alle Koordinaten auf 0 setzt.

```
public class Quader {
// hier geht's los...

    private int x0, x1, y0, y1, z0, z1;

    public Quader(int xx0, int xx1,
                  int yy0, int yy1,
                  int zz0, int zz1) {
        x0 = xx0; x1 = xx1;
        y0 = yy0; y1 = yy1;
        z0 = zz0; z1 = zz1;
    }

    public Quader() {
        this(0, 0, 0, 0, 0, 0);
    }
```

Schreiben Sie eine Instanzmethode ohne Parameter, die als Ergebnis das Volumen des Quaders liefert.

```
    public int volumen() {
        return (x1-x0)*(y1-y0)*(z1-z0);
    }
```

Schreiben Sie eine Instanzmethode `groesser` mit einem Quader als Parameter und Ergebnistyp `boolean`, die das Volumen dieses „Parameterquaders“ vergleicht mit dem durch das Objekt selber dargestellten Quader (also der Quader, dessen Instanzmethode `groesser` aufgerufen wird); das Ergebnis soll genau dann `true` sein, wenn der „Parameterquader“ größer ist.

```
public boolean groesser(Quader q) {
    return q.volumen() > volumen();
}
```

Schreiben Sie nun noch eine Klassenmethode, die zwei Quader q_1 und q_2 mit von Ihnen zu wählenden Koordinaten erzeugt, ihre Volumina ausdruckt; dann mittels der Instanzmethode `groesser` von q_1 überprüft, welcher Quader größer ist, und eine entsprechende Meldung ausdruckt.

```
public static void test() {
    Quader q1 = new Quader(1,2,3,5,6,9);
    System.out.println("q1: " + q1.volumen());
    Quader q2 = new Quader(1,2,3,5,6,8);
    System.out.println("q2: " + q2.volumen());
    if (q1.groesser(q2))
        System.out.println("q2 ist groesser");
    else
        System.out.println("q1 ist groesser");
}
```