1 Zuerst Informationen:

1.1 while-Schleife

Die Syntax der while-Schleife:

```
while ( Bedingung )
   { Auszufuehrende Methoden,
     wenn Bedingung erfuellt ist.
   }
```

Der Ablauf: Zuerst wird die Bedingung ausgewertet. Wenn das Ergebnis „ja“ lautet, werden die Methoden, die in `{ }` eingeschlossen sind, ausgeführt. Nach der Ausführung wird die Bedingung wieder ausgewertet, usw. Wird die Bedingung nicht erfüllt, macht das Programm mit der nächsten Anweisung nach dem in `{ }` eingeschlossenen Block weiter.
Wird die Bedingung schon beim ersten Mal nicht erfüllt, wird der Anweisungsblock kein einziges Mal ausgeführt.

1.2 do-while-Schleife

Die Syntax der do-while-Schleife:

```
do {
     Auszufuehrende Methoden
   } while ( Bedingung );
```

Der Unterschied zwischen den beiden Schleifenformen besteht im Zeitpunkt der Prüfung der Bedingung. Bei der do-while-Schleife findet die Prüfung am Ende der Schleife statt. Sie wird also mindestens einmal durchlaufen. Das kann gefährlich sein. Beachte das Semikolon am Ende der Zeile mit `while`!

1.3 Zählschleife oder for-Schleife

Die Syntax der for-Schleife:

```
for ( Startwert; Bedingung; Zaehlweise )
{
   Anweisungen
}
```

1. Ablauf: Zuerst wird die Schleifenvariable mit dem Startwert initialisiert und dann die Bedingung geprüft. Ist sie wahr, werden die Anweisungen ausgeführt. Am Ende der Schleife wird die Schleifenvariable nach Angabe der Zählweise verändert. Danach werden nur noch zu Beginn der Schleife die Bedingung überprüft, die Anweisungen ausgeführt und jeweils am Ende die Schleifenvariable verändert, bis die Bedingung falsch wird.

```
int i;
for (i=1; i<=5; i++)
{     System.out.print("Die " + i + ". Zahl ist " + i + ".");
}
```

2. Bei Variante 2 wird eine Schleifenvariable `i` sogar innerhalb der Klammer von `for` deklariert und initialisiert. Dies hat zur Folge, das die Variable `i` nur innerhalb der Schleife Gültigkeit hat. Man sollte sich angewöhnen, nur nach Variante 2 zu arbeiten! Beispiel:

```
for (int i=1; i<=5; i++)
{     System.out.print("Die " + i + ". Zahl ist " + i + ".");
}
```

1.4 Einfache Konsolenausgabe

Mit

```
System.out.println("Das ist ein Text fuer die Konsole.");
int x = 7;
System.out.print(" Der Wert von x ist " + x + ".");
```

können einfache Ausgaben gemacht werden. Nach println() wird eine neue Zeile begonnen.

1.5 Ganze Zahlen (int)

1. Mit `int i;` wird Speicherplatz für eine Variable mit dem Namen `i` und dem Typ `int`, d.h. INTEGER, reserviert. Man sagt: Die Variable `i` wird *deklariert*. Java kennt fünf verschiedene INTEGER-Typen:

Typ	**von**	**bis einschließlich**	**Größe**
`byte`	-128	127	8 Bit
`short`	-32.768	32.767	16 Bit
`int`	-2.147.483.648	2.147.483.647	32 Bit
`long`	-9.223.372.036.854.775.808	9.223.372.036.854.775.807	64 Bit
`char`	0	65535	16 Bit

2. Durch `i = 0` wird der Variablen `i` der Wert 0 zugewiesen. Man sagt: Die Variable `i` wird *initialisiert.*

3. Deklaration und Initialisierung können zusammengefasst werden durch

   `int i = 0;`

4. Für die Bedingung `i<5` wird der Vergleichsoperator `<` benutzt. Java kennt die folgenden Vergleichsoperatoren:

Zeichen	Bedeutung
`<`	kleiner als
`>`	größer als
`<=`	kleiner oder gleich
`>=`	größer oder gleich
`==`	gleich
`!=`	ungleich

5. Die Zuweisung `i = i + 1;` muss man von rechts nach links lesen: „Nimm den aktuellen Wert von `i`, addiere 1 dazu und speichere den neuen Wert wieder unter dem Namen `i` ab.

   Für eine Addition von 1 gibt es eine abkürzende Schreibweise: `i++;`

   Für die Subtraktion gilt alles entsprechend.

6. Es ist möglich, eine Variable mit einem endgültigen, unveränderlichen Wert zu versehen, d.h. sie zu einer Konstanten zu machen:

   ```
   final int ANZAHL=5;
   ```

   Konstanten schreibt man komplett mit Großbuchstaben.

7. Variablen schreibt man wie Methoden beginnend mit einem Kleinbuchstaben.

8. In einer Zeile lassen sich mehrere Variablen zugleich deklarieren bzw. initialisieren:

   ```
   int x, y, z, a=1, zahl;
   ```

9. **Integer-Division:** Mit Java kann man wie in der Grundschule mit Rest und ohne Nachkommastellen dividieren. Beispiele:

   `int ergebnis = 23/8;` liefert in der Variablen `ergebnis` eine 2.

   `int rest = 23 % 8;` liefert in der Variablen `rest` eine 7.

1.6 Mini-Programm zum Austesten:

```
1  public class MiniProgramm
2  {
3    public void schleife1()
4    { int n = 1; // Startwert
5      while (n <= 5)
6          { System.out.println(n);
7            n = n + 1;
8          }
9    }
10
11   public void action()
12   {
13     schleife1();
14   }
15 }
```

2 Aufgaben:

1. Gib die Zahlen von 1 bis 10 aus
   (a) mit einer while-Schleife.
   (b) mit einer do-while-Schleife.
   (c) mit einer beliebigen Schleife, aber in umgekehrter Reihenfolge.
2. Gib mit einer Schleife die geraden Zahlen von 2 bis 10 aus
   (a) mit einer while-Schleife.
   (b) mit einer for-Schleife.
3. Gib die ungeraden Zahlen von 1 bis 30 in umgekehrter Reiehenfolge aus.
4. Gib alle Zahlen von 1 bis 100 aus ohne die Vielfachen von 5 und 7.
5. Addiere alle Zahlen von 1 bis 100 und gib die Summe aus.
6. Gib alle Zahlen von 1 bis 100 aus, die die Ziffer 3 nicht enthalten.
7. Starte mit einer beliebigen Zahl. Wenn die Zahl gerade ist, dann halbiere sie. Andernfalls ersetze sie durch das Dreifache ergänzt um 1. Fahre mit der neuen Zahl fort, bis die Zahl 1 erreicht wird.
8. Gib die Vor- und Nachteile der einzelnen Schleifentypen an.