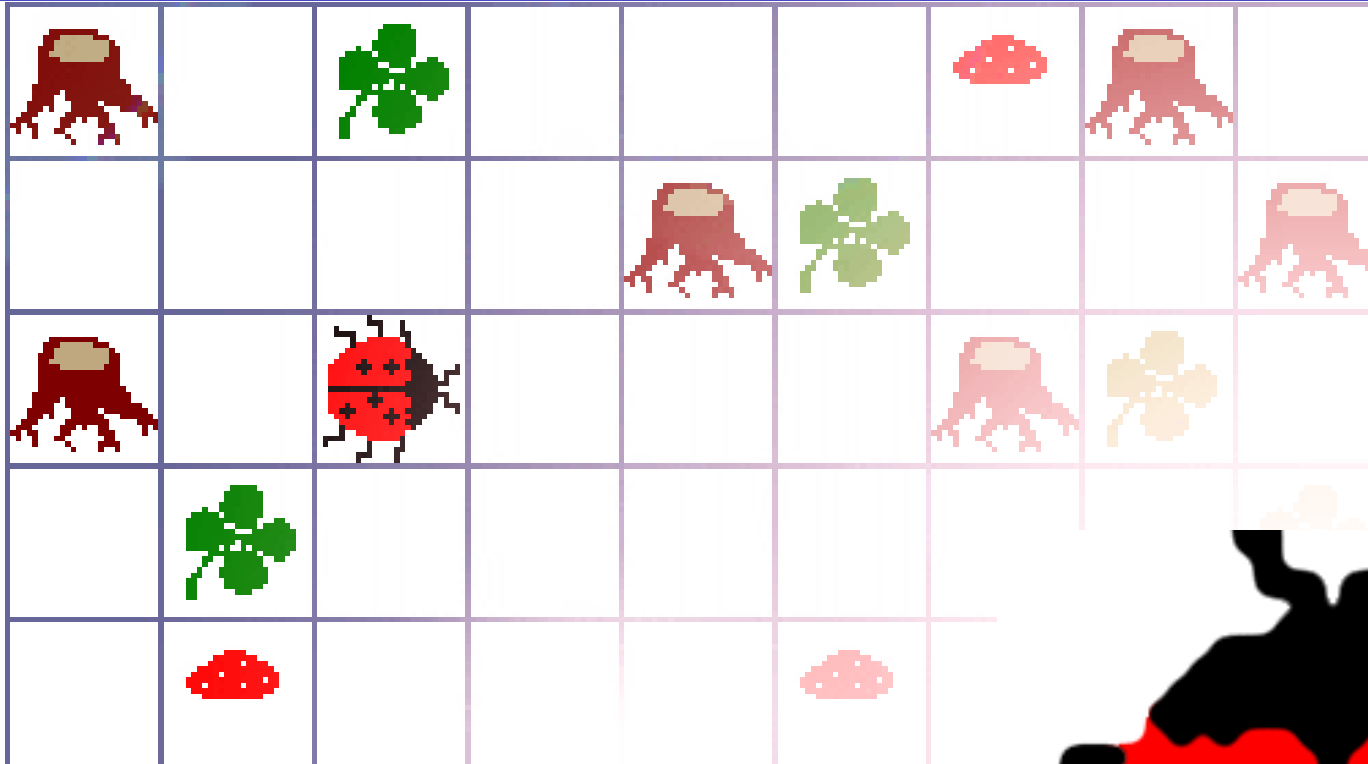


Kara, der programmierbare Marienkäfer !



<http://www.swisseduc.ch/informatik/karatojava/download.html>

Kara soll eine Spur von x Kleeblättern legen.

Aufgabe:

Kara soll vor sich eine Spur von Kleeblättern legen. Die Anzahl x sei variabel.

Das Neue: Karas Auftrag soll durch eine Methode *legeKleeblattzeile* erledigt werden. Ihr müsst der Methode dazu die gewünschte Anzahl als **Parameter** übergeben.

Schreibe ein möglichst allgemeines, gegliedertes Javakaraprogramm!

```
public class ParameterBeispiel extends JavaKaraProgram {
```

```
    void legeKleeblattzeile(int anzahl) {  
        for (int i=1; i<=anzahl; i=i+1) {  
            kara.putLeaf();  
            kara.move();  
        }  
    }
```

```
    public void myProgram() {  
        legeKleeblattzeile(7);  
        kara.turnRight();  
        legeKleeblattzeile(5);  
    }  
}
```

Kara soll eine Spur von x Kleeblättern legen.

Erläuterungen:

- Bei den bisherigen Programmen war es mitunter lästig, die Methoden immer mit einem Klammerpaar () zu versehen. Nun wird klar, dass die bisher eingesetzten Methoden nur Spezialfälle darstellen, bei denen kein **Parameter** übergeben wird.

- In der Klammer wird der **Parameter** (hier anzahl) mit einem Typ angegeben, hier int. Mehrere **Parameter** werden durch Kommata getrennt.

Beispiel: void zeichneRechteck(int breite, int hoehe)

- Beim Aufruf einer Methode wird der beim Aufruf in der Klammer stehende Wert in die Variable (hier anzahl) kopiert.

```
public class ParameterBeispiel extends JavaKaraProgram {
```

```
    void legeKleeblattzeile(int anzahl) {  
        for (int i=1; i<=anzahl; i=i+1) {  
            kara.putLeaf();  
            kara.move();  
        }  
    }  
}
```

```
public void myProgram() {  
    legeKleeblattzeile(7);  
    kara.turnRight();  
    legeKleeblattzeile(5);  
}
```

Kara dreht sich so oft nach links bzw. rechts.

Aufgabe:

Schreibe eine Methode `turnLeft(int anzahl)` und eine Methode `turnRight(int anzahl)`, die Kara veranlasst, sich so oft nach links bzw. rechts zu drehen, wie `anzahl` angibt. Programmiere ein passendes Verhalten für negative Werte von `Anzahl`.

Kara legt Muster.

Kara soll Muster aus Kleeblättern legen: **Quadrate, Rechtecke, Dreiecke** usw. Jedes Muster soll durch eine Methode realisiert werden, z.B.

```
void quadratZeichnen(int seitenlaenge).
```

Folgende Methoden sollen vorher zur Verfügung gestellt werden:

1. **void turnAround()**

Dreht Kara um 180 Grad.

2. **void legeX(int anzahl)**

Kara startet auf der Position des ersten abzulegenden Blattes und legt danach in seiner aktuellen Blickrichtung anzahl Kleeblätter ab. Am Ende steht er einen Schritt hinter dem letzten Blatt.

3. **void geheX(int anzahl)**

Kara startet auf seiner aktuellen Position und geht danach in seiner aktuellen Blickrichtung anzahl Schritte vorwärts.

Aufgaben:

1. Kara soll ein mit Kleeblättern gefülltes Quadrat mit variabler Seitenlänge ablegen. Der Methodenaufruf `quadratZeichnen(5)` soll demnach ein Quadrat mit der Seitenlänge 5 erzeugen.

2. Kara soll ein mit Kleeblättern gefülltes Rechteck mit variabler Breite und Höhe ablegen. Der Methodenaufruf `rechteckZeichnen(5,3)` soll demnach ein Rechteck mit der Breite 5 und der Höhe 3 erzeugen.

3. Kara soll ein mit Kleeblättern gefülltes gleichseitiges Dreieck mit variabler Seitenlänge ablegen. Eine Seite des Dreiecks soll waagrecht liegen.

4. Kara soll ein mit Kleeblättern umrandetes Rechteck mit variabler Breite und Höhe ablegen.

5. Kara soll ein mit Kleeblättern gefülltes Quadrat mit variabler Seitenlänge in „Kegelaufstellung“ ablegen. Die Seiten des Quadrates sind Diagonalen in der Kara-Welt. Dazu soll eine Methode `legeKegelreihe(int anzahl)` entwickelt werden, die eine diagonale Kegelreihe ablegt. Eine weitere Methode `zurNaechstenReihe()` ist ebenso zu programmieren.

Kara soll eine Spur von Kleeblättern zählen.

Aufgabe:

Kara steht am Anfang einer langen Spur von Kleeblättern. Er soll herausfinden, wie viele Blätter in der Spur liegen.

Das Neue:

Karas Auftrag soll durch eine Methode `zaehle()`, die das Ergebnis des Zählvorgangs zurück geben soll, erledigt werden.

Schreibe ein möglichst allgemeines, gegliedertes Javakaraprogramm!

```
public class RückgabeBeispiel extends JavaKaraProgram {
```

```
    int zaehle(int laenge) {  
        int anzahl=0;  
        for (int i=1; i<=laenge; i=i+1) {  
            if (kara.onLeaf()) { anzahl++; }  
            kara.move();  
        }  
        return anzahl; // Rueckgabe des Funktionswertes  
    } // Ende von zaehle
```

```
public void myProgram() {  
    int Kleeblattanzahl = 0;  
    ...  
    Kleeblattanzahl=zaehle(10);  
    ...  
}
```

Kara soll eine Spur von Kleeblättern zählen.

Erläuterungen:

1. Bei den bisherigen Programmen war es mitunter lästig, den Methoden immer das Wort **void**, was so viel wie **leer** bedeutet, voran zu stellen. Nun wird klar, dass die bisher eingesetzten Methoden nur Spezialfälle darstellen, bei denen kein Wert zurück gegeben wird.
2. Bei der Methode `zaehle` wird deutlich, dass Methoden in Java als Funktionen (wie in der Mathematik) angesehen werden können. Der Parameter z.B. `laenge` ist das Argument der Funktion (in der Mathematik meistens x), die einen Funktionswert (hier `zaehle`) eines bestimmten Typs (hier `int`) zurück liefert, den man einer Variablen zuweisen kann (hier: **`kleeblattAnzahl = zaehle(wieWeit)`**). In der Mathematik würde man $y = f(x)$ schreiben.
3. Dem Namen der Methode wird der **Typ des Rückgabewertes** voran gestellt. Die Rückgabe eines Funktionswertes innerhalb einer Methode geschieht mit `return`, gefolgt vom Funktionswert. Man muss darauf achten, dass eine Methode mit einem Rückgabewert immer mit einem entsprechenden `return` verlassen wird. Einfache Methoden **ohne einen Return** können auch mit einem `return` Rückgabewert verlassen werden. Dann steht eben nichts hinter `return`.

Kara bildet Sensoren nach.

Aufgabe:

In der in JavaKara eingebauten Methodensammlung fehlen einige Methoden.

Es gibt: `treeFront()`, `treeLeft()` und `treeRight()` für Bäume, weil Kara dafür Sensoren besitzt. Für Kleeblätter gibt es nur den Sensor `onLeaf()`. Wir können aber einen Sensor für Kleeblätter um Kara herum nachbilden, in dem wir Kara dort hinschicken, mit `onLeaf()` nachsehen und zurück kehren lassen. Das Ergebnis seiner Erkundung (`true` oder `false`) soll Kara zurück liefern.

Schreibe den Java-Code für **`leafLeft()`**, **`leafRight()`** und **`leafFront()`**. Findet Kara an der entsprechenden Stelle einen Baum, so soll `false` zurück geliefert werden.

Kara bildet Sensoren nach.

Aufgabe:

In der in JavaKara eingebauten Methodensammlung fehlen einige Methoden.

Es gibt: `treeFront()`, `reeLeft()` und `treeRight()` für Bäume, weil Kara dafür Sensoren besitzt. Für Kleeblätter gibt es nur den Sensor `onLeaf()`. Wir können aber einen Sensor für Kleeblätter um Kara herum nachbilden, in dem wir Kara dort hinschicken, mit `onLeaf()` nachsehen und zurück kehren lassen. Das Ergebnis seiner Erkundung (`true` oder `false`) soll Kara zurück liefern.

Schreibe den Java-Code für **`leafLeft()`**, **`leafRight()`** und **`leafFront()`**. Findet Kara an der entsprechenden Stelle einen Baum, so soll `false` zurück geliefert werden.

```
void moveBack() {
    turnAround();
    kara.move();
    turnAround();
}

boolean leafFront() {
    boolean blatt=false; // damit false fuer Baum
    if (!kara.treeFront()) {
        kara.move();
        blatt=kara.onLeaf();
        moveBack();
    }
    return blatt;
}
```

Kara hat ein Gedächtnis.

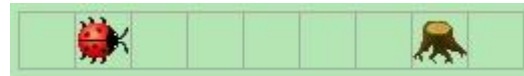
Aufgabe:

Schreibe ein Programm, das Kara geradeaus bis zum nächsten Baum führt. Kara soll dann zum Ausgangspunkt zurückkehren.

Das Neue:

Kara soll weder den Ausgangspunkt durch ein Kleeblatt markieren, noch soll Kara sich die Anzahl der Schritte bis zum Baum merken.

Schreibe ein möglichst allgemeines, gegliedertes Javakarapogramm!



Kara hat ein Gedächtnis.

Aufgabe:

Schreibe ein Programm, das Kara geradeaus bis zum nächsten Baum führt. Kara soll dann zum Ausgangspunkt zurückkehren.

Das Neue:

Kara soll weder den Ausgangspunkt durch ein Kleeblatt markieren, noch soll Kara sich die Anzahl der Schritte bis zum Baum merken.

Schreibe ein möglichst allgemeines, gegliedertes Javakaraprogramm!

```
public void DreheUm() {  
    kara.turnLeft();  
    kara.turnLeft();  
}
```

```
public void RueckeVorEinFeld() {  
    if (!kara.treeFront()) {  
        kara.move();  
        RueckeVorEinFeld();  
        kara.move();  
    }  
    else {  
        DreheUm();  
    }  
}
```

```
public void myProgram() {  
    RueckeVorEinFeld();  
    DreheUm();  
}
```