1. KARA steht in einer Reihe, an deren Ende ein Baum steht. KARA soll bis zum Baum laufen, dabei alle Blätter einsammeln und sich dort umdrehen.

```
1  import javakara.JavaKaraProgram;
2  public class BlaetterSammeln extends JavaKaraProgram
3  {
4    void zumBaum()
5    {
6     if (!kara.treeFront())
7          {  if (kara.onLeaf()) { kara.removeLeaf(); }
8             kara.move();
9             zumBaum();
10         }
11    else { if (kara.onLeaf()) { kara.removeLeaf(); }
12           kara.turnLeft();
13           kara.turnLeft();
14         }
15   } // Ende zumBaum
16
17   public void myProgram()
18   { zumBaum(); }
19 }  // Ende von BlaetterSammeln
```

2. KARA steht in einer Reihe, an deren Ende ein Kleeblatt liegt. KARA soll bis zum Kleeblatt laufen und das Blatt an seinen Startpunkt bringen.

```
1  import javakara.JavaKaraProgram;
2  public class BlattHolen extends JavaKaraProgram
3  {
4    void holeBlatt()
5    {
6     if (!kara.onLeaf())
7          {  kara.move();
8             holeBlatt(); // rekursiver Aufruf
9             kara.move();
10         }
11    else { kara.removeLeaf();
12           kara.turnLeft();
13           kara.turnLeft();
14         }
15   }
16
17   public void myProgram()
18   { holeBlatt(); }
19 }  // Ende von BlattHolen
```

3. KARA steht in einer Reihe, an deren Ende ein Baum steht. KARA soll bis zum Baum laufen, dabei alle Blätter einsammeln, sich dort umdrehen, zurücklaufen und die Blätter an den alten Plätzen ablegen.

```
1  import javakara.JavaKaraProgram;
2  public class BlaetterAblegen extends JavaKaraProgram
3  {
4    void zumBaum()
5    {
6     if (!kara.treeFront())
7          {   if (kara.onLeaf())
8                   {   kara.removeLeaf();
9                       kara.move();
10                      zumBaum();
11                      kara.move();
12                      kara.putLeaf();
13                  }
14             else {
15                      kara.move();
16                      zumBaum();
17                      kara.move();
18                  }
19         }
20    else { if (kara.onLeaf())
21                 { kara.removeLeaf();
22                   kara.putLeaf();
23                 } // eigentlich unnoetig
24           kara.turnLeft();
25           kara.turnLeft();
26         }
27   } // Ende zumBaum
28
29   public void myProgram()
30   { zumBaum();
31   }
32 }  // Ende von BlaetterAblegen
```

4. KARA steht in einer Reihe, an deren Ende ein Baum steht. KARA soll bis zum Baum laufen, dabei alle Blätter einsammeln und die Blätter hinter dem Baum spiegelbildlich ablegen. Es gibt keine weiteren Bäume, die stören könnten.

```
1  import javakara.JavaKaraProgram;
2  public class BlaetterGespiegeltAblegen extends JavaKaraProgram
3  {
4    void umBaumHerum()
5    {
6     kara.turnRight();
7     kara.move();
8     kara.turnLeft();
9     kara.move();
10    kara.move();
11    kara.turnLeft();
12    kara.move();
13    kara.turnRight();
14   }
15
16   void zumBaum()
17   {
18    if (!kara.treeFront())
19          {  if (kara.onLeaf())
20                {  kara.removeLeaf();
21                   kara.move();
22                   zumBaum();
23                   kara.move();
24                   kara.putLeaf();
25                }
26             else {
27                   kara.move();
28                   zumBaum();
29                   kara.move();
30                }
31          }
32    else { if (kara.onLeaf())
33                { kara.removeLeaf();
34                  umBaumHerum();
35                  kara.putLeaf();
36                }
37           else umBaumHerum();
38          }
39   } // Ende zumBaum
40
41   public void myProgram()
42   { zumBaum(); }
43 }  // Ende von BlaetterGespiegeltAblegen
```

5. Schreibe eine rekursive Version des Programms „Pacman“. Die Kleeblattspur geht durch einen Wald hindurch. Am Ende der Spur steht ein Pilz.

   Erweiterung: KARA läuft die Spur hin und wieder zurück.

```
1  import javakara.JavaKaraProgram;
2  public class PacmanRek extends JavaKaraProgram
3  {  // Anfang von PacmanRek
4
5    void turnAround()
6    { kara.turnLeft();
7      kara.turnLeft();
8    }
9
10   void moveBack()
11   { turnAround();
12     kara.move();
13     turnAround();
14   }
15
16   boolean leafFront()
17   {
18     boolean blatt=false; // auch false, wenn vorne Baum steht
19     if (!kara.treeFront())
20         { // jetzt muss man nachschauen
21           kara.move();
22           blatt=kara.onLeaf();
23           moveBack();
24         }
25     return blatt;
26   }
27
28   boolean leafLeft()
29   {
30     boolean blatt=false; // auch false, wenn links Baum steht
31     if (!kara.treeLeft())
32         { // jetzt muss man nachschauen
33           kara.turnLeft();
34           kara.move();
35           blatt=kara.onLeaf();
36           moveBack();
37           kara.turnRight();
38         }
39     return blatt;
40   }
41
42   void naechstesKleeblattSuchen()
43   { if (kara.mushroomFront())
```

```
44            {
45              tools.showMessage("Ich bin so satt, \n" +
46                                "ich mag kein Blatt!");
47              turnAround();
48            }
49        else { // Ende noch nicht erreicht
50              if (leafFront())
51                    {
52                      kara.move();
53                      kara.removeLeaf();
54                      naechstesKleeblattSuchen();
55                      kara.move();
56                    }
57              else { // nach links oder nach rechts
58                      if (leafLeft())
59                            {
60                             kara.turnLeft();
61                             kara.move();
62                             kara.removeLeaf();
63                             naechstesKleeblattSuchen();
64                             kara.move();
65                             kara.turnRight();
66                            }
67                      else { // dann eben nach rechts
68                             kara.turnRight();
69                             kara.move();
70                             kara.removeLeaf();
71                             naechstesKleeblattSuchen();
72                             kara.move();
73                             kara.turnLeft();
74                            }
75                    }
76            }
77    } // naechstesKleeblattsuchen
78
79    public void myProgram()
80    {  // Anfang von myProgram
81      kara.removeLeaf(); // Start auf erstem Kleeblatt
82      naechstesKleeblattSuchen();
83    }  // Ende von myProgram
84  }  // Ende von PacmanRek
```

6. Schreibe eine rekursive Version von „Labyrinth“. Das Labyrinth ist so gebaut, dass der Weg immer nur einen Baum breit ist und es keine „Löcher“ auf dem Weg gibt. Es gibt auch keine Verzweigungen.

   Erweiterung: KARA läuft den Weg im Labyrinth hin und wieder zurück.

```
1  import javakara.JavaKaraProgram;
2  public class LabyrinthRek extends JavaKaraProgram
3  {  // Anfang von LabyrinthRek
4
5    void zumNaechstenFeld()
6    {
7      if (!kara.treeFront())
8           {  kara.move();
9              zumNaechstenFeld();
10             kara.move();
11          }
12     else {  if (!kara.treeLeft())
13                  {  kara.turnLeft();
14                     kara.move();
15                     zumNaechstenFeld();
16                     kara.move();
17                     kara.turnRight();
18                  }
19             else {  if (!kara.treeRight())
20                           {  kara.turnRight();
21                              kara.move();
22                              zumNaechstenFeld();
23                              kara.move();
24                              kara.turnLeft();
25                           }
26                      else {  kara.turnLeft();
27                              kara.turnLeft();
28                           }
29                  }
30          }
31   } // zumNaechstenFeld
32
33   public void myProgram()
34   { zumNaechstenFeld();
35   }
36 }  // Ende von LabyrinthRek
```

7. KARA steht vor einer Treppe mit einer Stufenbreite von zwei „Bäumen“(siehe Abbildung). Der Beginn der Treppe ist ihm nicht bekannt. Die Höhe der Treppe ist auch unbekannt. KARA soll die Treppe hochlaufen und auf der obersten Stufe stehen bleiben.

```
1  import javakara.JavaKaraProgram;
2  public class TreppeHoch extends JavaKaraProgram
3  {
4    void stufeHoch()
5    {
6     if (kara.treeFront())
7          {  kara.turnLeft();
8             kara.move();
9             kara.turnRight();
10            kara.move();
11            kara.move();
12            stufeHoch();
13         }
14    else { kara.turnLeft();
15           kara.turnLeft();
16         }
17   }  // Ende von stufeHoch
18
19   public void myProgram()
20   { while (!kara.treeFront()) kara.move();  // Zur Treppe
21     stufeHoch();
22   }
23 }  // Ende von TreppeHoch
```

8. KARA soll wie vorher die Treppe hochsteigen und die gleiche Stufenanzahl auf der anderen Seite herabsteigen.

```
1  import javakara.JavaKaraProgram;
2  public class TreppeHochUndRunter extends JavaKaraProgram
3  {
4    void stufeHoch()
5    {
6     if (kara.treeFront())
7          {  kara.turnLeft();    // hoch
8             kara.move();
9             kara.turnRight();
10            kara.move();
11            kara.move();
12            stufeHoch();        // Rekursion
13            kara.move();        // und runter
14            kara.turnRight();
15            kara.move();
16            kara.turnLeft();
17            kara.move();
```

```
18            }
19    }  // Ende von stufeHoch
20
21    public void myProgram()
22    { while (!kara.treeFront()) kara.move(); // Zur Treppe
23      stufeHoch();
24    }
25 }  // Ende von TreppeHochUndRunter
```

9. KARA hat rechts von sich einen Baum stehen, der zu einem abgeschlossenen, von Bäumen umrandeten Feld gehört. Irgendwo am Rande dieses Feldes liegt ein Kleeblatt, das er finden und an seinen Startpunkt zurückbringen soll.

```
1  import javakara.JavaKaraProgram;
2  public class BlattAmFeldHolen extends JavaKaraProgram
3  {
4    void zumBlatt()
5    {
6     if (kara.onLeaf())
7           {  kara.removeLeaf();
8              kara.turnLeft();
9              kara.turnLeft();
10          }
11    else { if (!kara.treeRight())
12                 { kara.turnRight();
13                   kara.move();
14                   zumBlatt();
15                   kara.move();
16                   kara.turnLeft();
17                 }
18           else {  if (kara.treeFront())
19                         {  kara.turnLeft();
20                            zumBlatt();
21                            kara.turnRight();
22                         }
23                   else {  kara.move();
24                           zumBlatt();
25                           kara.move();
26                         }
27                 }
28          }
29   }  // Ende von weiter
30
31   public void myProgram()
32   { zumBlatt(); }
33 }  // Ende von BlattAmFeldHolen
```

10. KARA befindet sich in einem nach außen abgeschlossenen Irrgarten, der nur an einer Stelle, die durch ein Kleeblatt markiert ist, verlassen werden kann. Der Irrgarten enthält keine „Inseln“, d.h. von jeder Stelle im Innern ist der Rand des Irrgartens erreichbar. Schreibe ein Programm, das KARA von jedem beliebigen Punkt im Innern den Ausgang finden lässt.

    Erweiterung: KARA läuft nach dem Finden des Ausgangs wieder an seinen Ausgangspunkt zurück.

```
1  import javakara.JavaKaraProgram;
2  public class Irrgarten extends JavaKaraProgram
3  {  // Anfang von Irrgarten
4
5    void zumErstenBaum()
6    {
7     if (!kara.treeFront())
8           {  kara.move();
9              zumErstenBaum();
10             kara.move();
11          }
12    else {  kara.turnLeft(); // jetzt ist rechts ein Baum
13             zumAusgang();
14             kara.turnRight();
15          }
16   }
17
18   void zumAusgang()
19   {  // immer rechts an der Wand entlang
20    if (kara.onLeaf())
21          {  kara.removeLeaf();
22             kara.turnLeft();
23             kara.turnLeft();
24          }
25    else { if (!kara.treeRight())
26                 { kara.turnRight();
27                   kara.move();
28                   zumAusgang();
29                   kara.move();
30                   kara.turnLeft();
31                 }
32           else {  if (kara.treeFront())
33                         {  kara.turnLeft();
34                            zumAusgang();
35                            kara.turnRight();
36                         }
37                   else {  kara.move();
38                           zumAusgang();
39                           kara.move();
```

```
40                              }
41                          }
42                      }
43      }  // Ende von zumAusgang
44
45      public void myProgram()
46      {  // Anfang von myProgram
47        zumErstenBaum();
48      }  // Ende von myProgram
49  }  // Ende von Irrgarten
```

11. KARA bewacht ein Mobilé. Die Drähte sind hier als Kleeblätter dargestellt. Manchmal macht er einen Kontrollgang über alle Drähte und stößt diverse Insekten, die sich an den Enden der Drähte niedergelassen haben, vom Drahtgestell (Die Insekten sind hier als Pilze dargestellt). Schreibe ein Programm für einen Kontrollgang mit Rückkehr.

```
1   import javakara.JavaKaraProgram;
2   public class Mobile extends JavaKaraProgram
3   {  // Anfang von Mobile
4
5     void turnAround()
6     { kara.turnLeft();
7       kara.turnLeft();
8     }
9
10    void moveBack()
11    { turnAround();
12      kara.move();
13      turnAround();
14    }
15
16    boolean leafFront()
17    { boolean blatt=false; // auch false, wenn vorne Baum steht
18      if (!kara.treeFront())
19          { // jetzt muss man nachschauen
20            kara.move();
21            blatt=kara.onLeaf();
22            moveBack();
23          }
24      return blatt;
25    }
26
27    boolean leafLeft()
28    { boolean blatt=false; // auch false, wenn links Baum steht
29      if (!kara.treeLeft())
30          { // jetzt muss man nachschauen
31            kara.turnLeft();
```

```
32              kara.move();
33              blatt=kara.onLeaf();
34              moveBack();
35              kara.turnRight();
36            }
37      return blatt;
38    }
39
40    boolean leafRight()
41    { boolean blatt=false; // auch false, wenn links Baum steht
42      if (!kara.treeRight())
43            { // jetzt muss man nachschauen
44              kara.turnRight();
45              kara.move();
46              blatt=kara.onLeaf();
47              moveBack();
48              kara.turnLeft();
49            }
50      return blatt;
51    }
52
53    void zurLaus()
54    {
55     if (kara.mushroomFront()) // Ende erreicht
56           {  kara.move();
57              turnAround(); // Laus abwerfen
58              kara.move();
59           }
60     else {   if (leafFront()) // normal weiter
61                  { kara.move();
62                    zurLaus();
63                    kara.move();
64                  }
65              else {   if (leafLeft() && leafRight())//Verzweigung
66                          { kara.turnLeft();
67                            kara.move(); // nach links
68                            zurLaus();
69                            kara.move(); // von links zurueck
70                            kara.move(); // nach rechts
71                            zurLaus();
72                            kara.move(); // von rechts zurueck
73                            kara.turnLeft();
74                          }
75                       else {   if (leafLeft()) // nur nach links
76                                   {   kara.turnLeft();
77                                       kara.move();
78                                       zurLaus();
```

```
79                                          kara.move();
80                                          kara.turnRight();
81                                      }
82                                  else { if (leafRight())
83                                                  // nur nach rechts
84                                          { kara.turnRight();
85                                            kara.move();
86                                            zurLaus();
87                                            kara.move();
88                                            kara.turnLeft();
89                                          }
90                                        else { // vorne kein Blatt
91                                            turnAround();
92                                          }
93                                      }
94                                  }
95                              }
96                      }
97      } // Ende von zurLaus
98
99      public void myProgram()
100     {  // Anfang von myProgram
101       zurLaus();
102     }  // Ende von myProgram
103 }  // Ende von Mobile
```