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.

```
import JavaKaraProgram;

public class BlaetterSammeln extends JavaKaraProgram
{
  void zumBaum()
  {
   if (!kara.treeFront())
        {  if (kara.onLeaf()) { kara.removeLeaf(); }
           kara.move();
           zumBaum();
        }
   else { if (kara.onLeaf()) { kara.removeLeaf(); }
          kara.turnLeft();
          kara.turnLeft();
        }
  } // Ende zumBaum

  public void myProgram()
  { zumBaum();
  }
}  // 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.

```
import JavaKaraProgram;
public class BlattHolen extends JavaKaraProgram
{
  void holeBlatt()
  {
   if (!kara.onLeaf())
        {  kara.move();
           holeBlatt(); // rekursiver Aufruf
           kara.move();
        }
   else { kara.removeLeaf();
          kara.turnLeft();
          kara.turnLeft();
        }
  }

  public void myProgram()
  { holeBlatt();
  }
}  // 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.

```
import JavaKaraProgram;
public class BlaetterAblegen extends JavaKaraProgram
{
  void zumBaum()
  {
   if (!kara.treeFront())
        {  if (kara.onLeaf())
```

```
                {  kara.removeLeaf();
                   kara.move();
                   zumBaum();
                   kara.move();
                   kara.putLeaf();
                 }
           else {
                   kara.move();
                   zumBaum();
                   kara.move();
                 }
      }
   else { if (kara.onLeaf())
              { kara.removeLeaf();
                kara.putLeaf();
              } // eigentlich unnoetig
        kara.turnLeft();
        kara.turnLeft();
      }
  } // Ende zumBaum

  public void myProgram()
  { zumBaum();
  }
}  // 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.

```
import JavaKaraProgram;
public class BlaetterGespiegeltAblegen extends JavaKaraProgram
{
  void umBaumHerum()
  {
   kara.turnRight();
   kara.move();
   kara.turnLeft();
   kara.move();
   kara.move();
   kara.turnLeft();
   kara.move();
   kara.turnRight();
  }

  void zumBaum()
  {
   if (!kara.treeFront())
        {  if (kara.onLeaf())
                {  kara.removeLeaf();
                   kara.move();
                   zumBaum();
                   kara.move();
                   kara.putLeaf();
                 }
           else {
                   kara.move();
                   zumBaum();
                   kara.move();
                 }
      }
   else { if (kara.onLeaf())
              { kara.removeLeaf();
                umBaumHerum();
                kara.putLeaf();
              }
        else { umBaumHerum();
              }
```

```
      }
  } // Ende zumBaum

  public void myProgram()
  { zumBaum();
  }
}  // 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.

```
import JavaKaraProgram;
public class PacmanRek extends JavaKaraProgram
{  // Anfang von PacmanRek

  void turnAround()
  { kara.turnLeft();
    kara.turnLeft();
  }

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

  boolean leafFront()
  {
    boolean blatt=false; // auch false, wenn vorne Baum steht
    if (!kara.treeFront())
         { // jetzt muss man nachschauen
           kara.move();
           blatt=kara.onLeaf();
           moveBack();
         }
    return blatt;
  }

  boolean leafLeft()
  {
    boolean blatt=false; // auch false, wenn links Baum steht
    if (!kara.treeLeft())
         { // jetzt muss man nachschauen
           kara.turnLeft();
           kara.move();
           blatt=kara.onLeaf();
           moveBack();
           kara.turnRight();
         }
    return blatt;
  }

  void naechstesKleeblattSuchen()
  { if (kara.mushroomFront())
         {
           tools.showMessage("Ich bin so satt, \n " +
                             "ich mag kein Blatt!");
           turnAround();
         }
    else { // Ende noch nicht erreicht
           if (leafFront())
                {
                  kara.move();
                  kara.removeLeaf();
                  naechstesKleeblattSuchen();
                  kara.move();
                }
```

```
          else { // nach links oder nach rechts
                 if (leafLeft())
                      {
                       kara.turnLeft();
                       kara.move();
                       kara.removeLeaf();
                       naechstesKleeblattSuchen();
                       kara.move();
                       kara.turnRight();
                      }
                 else { // dann eben nach rechts
                       kara.turnRight();
                       kara.move();
                       kara.removeLeaf();
                       naechstesKleeblattSuchen();
                       kara.move();
                       kara.turnLeft();
                      }
               }
        }
  } // naechstesKleeblattsuchen

  public void myProgram()
  {  // Anfang von myProgram
    kara.removeLeaf(); // Start auf erstem Kleeblatt
    naechstesKleeblattSuchen();
  }  // Ende von myProgram
}  // 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.

```
import JavaKaraProgram;
public class LabyrinthRek extends JavaKaraProgram
{  // Anfang von LabyrinthRek

  void zumNaechstenFeld()
  {
    if (!kara.treeFront())
         {  kara.move();
            zumNaechstenFeld();
            kara.move();
         }
    else {  if (!kara.treeLeft())
                 {  kara.turnLeft();
                    kara.move();
                    zumNaechstenFeld();
                    kara.move();
                    kara.turnRight();
                 }
            else {  if (!kara.treeRight())
                         {  kara.turnRight();
                            kara.move();
                            zumNaechstenFeld();
                            kara.move();
                            kara.turnLeft();
                         }
                    else {  kara.turnLeft();
                            kara.turnLeft();
                         }
                 }
         }
  } // zumNaechstenFeld

  public void myProgram()
```

```
  { zumNaechstenFeld();
  }
}  // 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.

```
import JavaKaraProgram;
public class TreppeHoch extends JavaKaraProgram
{
  void stufeHoch()
  {
   if (kara.treeFront())
        {  kara.turnLeft();
           kara.move();
           kara.turnRight();
           kara.move();
           kara.move();
           stufeHoch();
        }
   else { kara.turnLeft();
          kara.turnLeft();
        }
  }  // Ende von stufeHoch

  public void myProgram()
  { while (!kara.treeFront())
      { kara.move(); } // Zur Treppe
    stufeHoch();
  }
}  // Ende von TreppeHoch
```

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

```
import JavaKaraProgram;
public class TreppeHochUndRunter extends JavaKaraProgram
{
  void stufeHoch()
  {
   if (kara.treeFront())
        {  kara.turnLeft();   // hoch
           kara.move();
           kara.turnRight();
           kara.move();
           kara.move();
           stufeHoch();       // Rekursion
           kara.move();       // und runter
           kara.turnRight();
           kara.move();
           kara.turnLeft();
           kara.move();
        }
  }  // Ende von stufeHoch

  public void myProgram()
  { while (!kara.treeFront())
      { kara.move(); } // Zur Treppe
    stufeHoch();
  }
}  // 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.

```
import JavaKaraProgram;
public class BlattAmFeldHolen extends JavaKaraProgram
{
  void zumBlatt()
  {
   if (kara.onLeaf())
        {  kara.removeLeaf();
           kara.turnLeft();
           kara.turnLeft();
        }
   else { if (!kara.treeRight())
               { kara.turnRight();
                 kara.move();
                 zumBlatt();
                 kara.move();
                 kara.turnLeft();
               }
          else {  if (kara.treeFront())
                        {  kara.turnLeft();
                           zumBlatt();
                           kara.turnRight();
                        }
                   else {  kara.move();
                           zumBlatt();
                           kara.move();
                        }
               }
        }
  }  // Ende von weiter

  public void myProgram()
  { zumBlatt();
  }
}  // 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.

```
import JavaKaraProgram;
public class Irrgarten extends JavaKaraProgram
{  // Anfang von Irrgarten

  void zumErstenBaum()
  {
   if (!kara.treeFront())
        {  kara.move();
           zumErstenBaum();
           kara.move();
        }
   else {  kara.turnLeft(); // jetzt ist rechts ein Baum
           zumAusgang();
           kara.turnRight();
        }
  }

  void zumAusgang()
  {  // immer rechts an der Wand entlang
   if (kara.onLeaf())
        {  kara.removeLeaf();
           kara.turnLeft();
           kara.turnLeft();
        }
   else { if (!kara.treeRight())
```

```
              { kara.turnRight();
                kara.move();
                zumAusgang();
                kara.move();
                kara.turnLeft();
              }
          else {  if (kara.treeFront())
                       {  kara.turnLeft();
                          zumAusgang();
                          kara.turnRight();
                       }
                  else {  kara.move();
                          zumAusgang();
                          kara.move();
                       }
              }
        }
  }  // Ende von zumAusgang

  public void myProgram()
  {  // Anfang von myProgram
    zumErstenBaum();
  }  // Ende von myProgram
}  // 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.

```
import JavaKaraProgram;
public class Mobile extends JavaKaraProgram
{  // Anfang von Mobile

  void turnAround()
  { kara.turnLeft();
    kara.turnLeft();
  }

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

  boolean leafFront()
  { boolean blatt=false; // auch false, wenn vorne Baum steht
    if (!kara.treeFront())
         { // jetzt muss man nachschauen
           kara.move();
           blatt=kara.onLeaf();
           moveBack();
         }
    return blatt;
  }

  boolean leafLeft()
  { boolean blatt=false; // auch false, wenn links Baum steht
    if (!kara.treeLeft())
         { // jetzt muss man nachschauen
           kara.turnLeft();
           kara.move();
           blatt=kara.onLeaf();
           moveBack();
           kara.turnRight();
         }
    return blatt;
  }
```

```
  boolean leafRight()
  { boolean blatt=false; // auch false, wenn links Baum steht
    if (!kara.treeRight())
         { // jetzt muss man nachschauen
           kara.turnRight();
           kara.move();
           blatt=kara.onLeaf();
           moveBack();
           kara.turnLeft();
         }
    return blatt;
  }

  void zurLaus()
  {
   if (kara.mushroomFront()) // Ende erreicht
        {  kara.move();
           turnAround(); // Laus abwerfen
           kara.move();
        }
   else {  if (leafFront()) // normal weiter
                { kara.move();
                  zurLaus();
                  kara.move();
                }
           else {  if (leafLeft() && leafRight()) // Verzweigung
                        { kara.turnLeft();
                          kara.move(); // nach links
                          zurLaus();
                          kara.move(); // von links zurueck
                          kara.move(); // nach rechts
                          zurLaus();
                          kara.move(); // von rechts zurueck
                          kara.turnLeft();
                        }
                   else {  if (leafLeft()) // nur nach links
                                {  kara.turnLeft();
                                   kara.move();
                                   zurLaus();
                                   kara.move();
                                   kara.turnRight();
                                }
                           else { if (leafRight()) // nur nach rechts
                                       { kara.turnRight();
                                         kara.move();
                                         zurLaus();
                                         kara.move();
                                         kara.turnLeft();
                                       }
                                  else { // vorne kein Blatt
                                         turnAround();
                                       }
                                }
                        }
                }
        }
  }  // Ende von zurLaus

  public void myProgram()
  {  // Anfang von myProgram
    zurLaus();
  }  // Ende von myProgram
}  // Ende von Mobile
```