Kapitel 2: Programmfluss (Lösungen)

Lösung zu Aufgabe 9:

public void act()

{

if (treeFront())

{

goAroundTree();

}

else

{

move();

}

if (onLeaf())

{

removeLeaf();

}

}

public void goAroundTree()

{

turnLeft();

move();

turnRight();

move();

move();

turnRight();

move();

turnLeft();

}

Lösung zu Aufgabe 11:

public void act()

{

if (treeLeft())

{

move();

}

else

{

if (onLeaf())

{

removeLeaf();

move();

}

else

{

move();

}

}

}

Lösung zu Aufgabe 12:

public void act()

{

if (treeLeft() && treeRight())

{

putLeaf();

}

else

{

move();

}

}

Lösung zu Aufgabe 13:

public void act()

{

if (treeLeft() || treeRight())

{

putLeaf();

move();

}

else

{

move();

}

}

Lösung zu Aufgabe 14:

public void act()

{

if (!onLeaf())

{

putLeaf();

}

if (!treeFront())

{

move();

}

}

Lösung zu Aufgabe 15:

public void act()

{

if (onLeaf())

{

removeLeaf();

}

else

{

if (!treeFront())

{

move();

}

else

{

if (!treeLeft())

{

turnLeft();

move();

}

else

{

turnRight();

move();

}

}

}

}

Lösung zu Aufgabe 16:

public void act()

{

if (!treeFront())

{

removeLeaf();

findNextLeaf();

}

}

public void findNextLeaf()

{

// look for leaf in front

// (erst mal vorne schauen)

move();

if (!onLeaf())

{

// kein Blatt vorne, also zurueck und links schauen

turnAndGoBack();

turnRight();

move();

if (!onLeaf())

{

// links ist auch kein Blatt; dann muss es rechts liegen

turnAndGoBack();

move();

}

}

}

public void turnAndGoBack()

{

turnLeft();

turnLeft();

move();

}

Lösung zu Aufgabe 18:

public void act()

{

while (!onLeaf())

{

if (treeFront())

{

goAroundTree();

}

else

{

move();

}

}

// Found leaf --> eat it

removeLeaf();

}

public void goAroundTree()

{

turnLeft();

move();

turnRight();

move();

while (treeRight())

{

move();

}

turnRight();

move();

turnLeft();

}

Lösung zu Aufgabe 19:

public void act()

{

while (treeFront())

{

oneStepUp();

}

}

public void oneStepUp()

{

turnLeft();

move();

turnRight();

move();

}

Lösung zu Aufgabe 20:

public void act()

{

makeOneStep();

}

public void makeOneStep()

{

if (!treeRight())

{

// no tree right --> go right

turnRight();

move();

}

else

{

// there is a tree right

if (!treeFront())

{

// no tree in front --> move

move();

}

else

{

// trees right and front

if (! treeLeft())

{

// no tree left --> go left

turnLeft();

move();

}

else

{

// trees right, front and left: dead end

turnLeft();

turnLeft();

move();

}

}

}

}