Chapter 2: Program Flow (Solutions)

Solution for Task 9:

public void act()

{

if (treeFront())

{

goAroundTree();

}

else

{

move();

}

if (onLeaf())

{

removeLeaf();

}

}

public void goAroundTree()

{

turnLeft();

move();

turnRight();

move();

move();

turnRight();

move();

turnLeft();

}

Solution for Task 11:

public void act()

{

if (treeLeft())

{

move();

}

else

{

if (onLeaf())

{

removeLeaf();

move();

}

else

{

move();

}

}

}

Solution for Task 12:

public void act()

{

if (treeLeft() && treeRight())

{

putLeaf();

}

move();

}

Solution for Task 13:

public void act()

{

if (treeLeft() || treeRight())

{

putLeaf();

move();

}

else

{

move();

}

}

Solution for Task 14:

public void act()

{

if (!onLeaf())

{

putLeaf();

}

if (!treeFront())

{

move();

}

}

Solution for Task 15:

public void act()

{

if (onLeaf())

{

removeLeaf();

}

else

{

if (!treeFront())

{

move();

}

else

{

if (!treeLeft())

{

turnLeft();

move();

}

else

{

turnRight();

move();

}

}

}

}

Solution for Task 16:

public void act()

{

if (!treeFront())

{

removeLeaf();

findNextLeaf();

}

}

public void findNextLeaf()

{

// look for leaf in front

if (!onLeaf())

{

// no leaf in front, go back and look left

turnAndGoBack();

turnRight();

move();

if (!onLeaf())

{

// no leaf left; leaf must be on right side

turnAndGoBack();

move();

}

}

}

public void turnAndGoBack()

{

turnLeft();

turnLeft();

move();

}

Solution for Task 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();

}

Solution for Task 19:

public void act()

{

while (treeFront())

{

oneStepUp();

}

}

public void oneStepUp()

{

turnLeft();

move();

turnRight();

move();

}

Solution for Task 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();

}

}

}

}