

# Contents

1	Basic Test Results	2
2	README	3
3	Food.jack	4
4	List.jack	6
5	Main.jack	8
6	Node.jack	9
7	Snake.jack	10
8	SnakeGame.jack	12

# 1 Basic Test Results

```
1 ***** TEST START *****
2
3     preparing sub.tar
4 dos2unix: converting file /tmp/bodek._MLOGg/nand2tet/Project09/roigreenberg/presubmission/testdir/stud/sub.tar/README to Uni
5 compiling...
6 Game compiled successfully. I hope it's fun!
7
8 ***** TEST END *****
```

## 2 README

```
1  roigreenberg,inbaravni
2
3  =====
4
5  Roi Greenberg, ID 30557123, roi.greenberg@mail.huji.ac.il
6  Inbar Avni, ID 201131760, inbar.avni@mail.huji.ac.il
7
8  =====
9
10
11
12          Project 9 - High Level Language
13          -----
14
15
16
17
18
19
20
21 Submitted Files
22 -----
23
24 README - This file.
25
26 Main.jack - the main.
27 SnakeGame.jack - The game
28 Snake.jack - Represent the snake head in the game
29 Food.jack - Represent the food in the game
30 List.jack - A linked List class
31 Node.jack - A node class
32
33
34 Remarks
35 -----
36
37
38 To start the game, you should choose the game speed, a number between 0 and 9.
39 Any other choose will be treated as 9(fastest).
40 The snake is always moving and can turn with pressing on the arrows keys.
41 Not that the pressing sometimes need to be little long for the game to recognize the press.
42 The game will be over if the snake will touch the walls or bite itself
43
44 To quit from the game press Q.
```

## 3 Food.jack

```
1  class Food {
2      field int x;
3      field int y;
4      //field Food food;
5
6      constructor Food new() {
7          let x = 100;
8          let y = 100;
9          do next();
10         return this;
11     }
12
13     method void dispose() {
14         do Memory.deAlloc(this);
15         return;
16     }
17
18     method void next() {
19         do Screen.setColor(false);
20         do Screen.drawRectangle(x, y, x + 4, y + 4);
21         do nextX();
22         do nextY();
23         do Screen.setColor(true);
24         do Screen.drawRectangle(x, y, x + 4, y + 4);
25
26         return;
27     }
28
29     method boolean hit(int hx,int hy) {
30         var int i, j;
31         let i = 0;
32         let j = 0;
33
34         if (((x - 1) < hx)&((y - 1) < hy)&((x + 5) > hx)&((y + 5) > hy)) {
35             return true;
36         }
37
38         return false;
39     }
40
41     method void nextX () {
42         while (true) {
43             let x = Math.multiply(x , 373) - Math.multiply(Math.divide(Math.multiply(x , 373), 491), 491) + 15;
44             if ((x < 496)&(x > 15)){
45                 return;
46             }
47         }
48         return;
49     }
50
51     method void nextY () {
52         while (true) {
53             let y = Math.multiply(y , 151) - Math.multiply(Math.divide(Math.multiply(y , 151), 227), 227) + 15;
54             if ((y < 236)&(y > 15)){
55                 return;
56             }
57         }
58         return;
59     }
```

60  
61        }  
62    }

## 4 List.jack

```
1  class List {
2
3      field Node head;
4      field Node tail;
5
6      constructor List new(int givenX, int givenY){
7
8          var int i;
9          let i = 20;
10         while (i > 0){
11             let i = i - 1;
12             do newLink(givenX, givenY + i);
13         }
14         return this;
15     }
16
17
18
19     method void newLink(int givenX, int givenY){
20
21         var Node newNode;
22
23         if (head = null){
24             let newNode = Node.new(givenX, givenY, null, null);
25             let tail = newNode;
26         } else {
27             let newNode = Node.new(givenX, givenY, head, null);
28             let head[3] = newNode;
29         }
30
31         let head = newNode;
32         do Screen.setColor(true);
33         do Screen.drawPixel(givenX, givenY);
34         return;
35     }
36
37
38     method void removeLink(){
39
40
41         var Node nodeToBeTail;
42         let nodeToBeTail = tail[3];
43         let nodeToBeTail[2] = null;
44         do tail.dispose();
45         let tail = nodeToBeTail;
46         do Screen.setColor(false);
47         do Screen.drawPixel(nodeToBeTail[0], nodeToBeTail[1]);
48         return;
49     }
50
51
52     method boolean isInList(int givenX, int givenY){
53
54         var Node curNode;
55         let curNode = head;
56
57         while (~(curNode = null)){
58
59             if ((curNode[0] = givenX) & (curNode[1] = givenY)){
```

```

60         return true;
61     }
62     let curNode = curNode[2];
63 }
64 return false;
65 }
66
67
68 method void dispose(){
69
70     var Node curNode, tmpNode;
71     let curNode = head;
72
73     while (~(curNode = null)){
74         let tmpNode = curNode[2];
75         do curNode.dispose();
76         let curNode = tmpNode;
77     }
78     do Memory.deAlloc(this);
79     return;
80 }
81
82
83 }

```

## 5 Main.jack

```
1  class Main {
2      /** Initializes a new game and starts it. */
3      function void main() {
4          var SnakeGame game;
5
6          let game = SnakeGame.new();
7          do game.game();
8          do game.dispose();
9          return;
10     }
11
12
13 }
```



## 6 Node.jack

```
1  class Node {
2
3      field int x, y;
4      field Node next, prev;
5      //static Node head;
6
7
8      /*constructor Node new(int givenX, int givenY){
9
10         let x = givenX;
11         let y = givenY;
12
13         if (head = null){
14             let head = this;
15             let next = this;
16         } else {
17             let next = head;
18         }
19
20         let prev = null;
21
22         return this;
23     }*/
24
25
26     constructor Node new(int givenX, int givenY, Node givenNext, Node givenPrev){
27
28         let x = givenX;
29         let y = givenY;
30         let next = givenNext;
31         let prev = givenPrev;
32
33         return this;
34     }
35
36
37     method void dispose(){
38
39         do Memory.deAlloc(this);
40
41         return;
42     }
43
44 }
```

## 7 Snake.jack

```
1  class Snake {
2      field int x;
3      field int y;
4      field int curDir;
5      field List body;
6      field Food food;
7      field int count, grow;
8
9      constructor Snake new(int X, int Y) {
10         let x = X;
11         let y = Y;
12         let curDir = 0;
13         let count = 0;
14         let grow = 0;
15         let body = List.new(x,y);
16         let food = Food.new();
17
18         //do Screen.setColor(true);
19         //do Screen.drawPixel(x, y);
20         return this;
21     }
22
23     method void dispose() {
24         do body.dispose();
25         do food.dispose();
26         do Memory.deAlloc(this);
27         return;
28     }
29
30
31     method boolean move(String dir) {
32         if (dir = 0) {
33             if (~(curDir = 1)) {
34                 let y = y - 1;
35                 let curDir = dir;
36             }
37             else {
38                 let y = y + 1;
39             }
40         }
41         if (dir = 1) {
42             if (~(curDir = 0)) {
43                 let y = y + 1;
44                 let curDir = dir;
45             }
46             else {
47                 let y = y - 1;
48             }
49         }
50         if (dir = 3) {
51             if (~(curDir = 2)) {
52                 let x = x + 1;
53                 let curDir = dir;
54             }
55             else {
56                 let x = x - 1;
57             }
58         }
59         if (dir = 2) {
```

```

60         if (~(curDir = 3)) {
61             let x = x - 1;
62             let curDir = dir;
63         }
64         else {
65             let x = x + 1;
66         }
67     }
68 }
69
70
71 if (x = 15) {
72     return false;
73 }
74 if (x = 500) {
75     return false;
76 }
77 if (y = 15) {
78     return false;
79 }
80 if (y = 240) {
81     return false;
82 }
83 if (body.isInList(x,y)){
84     return false;
85 }
86 do body.newLink(x,y);
87 if (hit()) {
88     let count = count + 1;
89     let grow = 10;
90     do food.next();
91 } else {
92     if(grow > 0){
93         let grow = grow - 1;
94     } else {
95         do body.removeLink();
96     }
97 }
98 }
99 return true;
100 }
101
102 method boolean hit() {
103
104     return food.hit(x,y);
105 }
106
107 }

```

## 8 SnakeGame.jack

```
1  class SnakeGame {
2
3      // The Snake
4      field Snake snake;
5      //field List body;
6      field boolean cont;
7      field char key;
8      field int fx;
9      field int score;
10
11
12
13
14      // The snake's movement direction
15      field int direction; // 0=none,1=up,2=down,3=left,4=right
16
17      /** Constructs a new Snake Game. */
18      constructor SnakeGame new() {
19          let cont = true;
20          let direction = 0;
21          let score = 0;
22
23          return this;
24      }
25
26      /** Deallocates the object's memory. */
27      method void dispose() {
28          do snake.dispose();
29          do Memory.deAlloc(this);
30          return;
31      }
32
33      method void delay(int speed) {
34          var int time1, time2;
35          var char tempKey;
36          if (speed < 0){
37              let speed = 1;
38          }
39          if (speed > 9) {
40              let speed = 10;
41          }
42          let time1 = 1000/speed;
43          let time2 = 1000/speed;
44
45          while (time1 > 0) {
46              while (time2 > 0) {
47                  let tempKey = Keyboard.keyPressed();
48                  if (~(tempKey = 0)) {
49                      let key = tempKey;
50                  }
51                  let time2 = time2 - 1;
52              }
53              let time1 = time1 - 1;
54          }
55          return;
56      }
57
58      method void game() {
59
```

```

60     var boolean cont;
61     var int i;
62     let cont = true;
63
64
65     let cont = false;
66     do printStart();
67     let snake = Snake.new(267,127);
68
69     while (key = 0) {
70         let key = Keyboard.keyPressed();
71     }
72
73     let i = 0;
74     while (i < 30) {
75         do Output.moveCursor(5,25+i);
76         let i = i + 1;
77     }
78
79
80     do run(snake, (key - 47));
81     do snake.dispose();
82
83     return;
84 }
85
86 method void run(Snake s, int speed) {
87
88     var boolean exit;
89     var int direction;
90     var boolean cont, hit;
91     let direction = 0;
92     let exit = false;
93
94     while (~exit) {
95
96
97         // waits for a key to be pressed.
98         while (key = 0) {
99             let key = Keyboard.keyPressed();
100             do delay(speed);
101             if (~(s.move(direction))){
102                 do Output.moveCursor(10,25);
103                 do Output.printString("Game Over :(");
104                 return;
105             }
106             do Output.moveCursor(0,38);
107             do Output.moveCursor(0,37);
108             do Output.moveCursor(0,36);
109             do Output.printInt(s[5]);
110
111         }
112
113         if (key = 140) {
114             let exit = true;
115         }
116         if (key = 131) {
117             let direction = 0;
118         }
119         if (key = 133) {
120             let direction = 1;
121         }
122         if (key = 130) {
123             let direction = 2;
124         }
125         if (key = 132) {
126             let direction = 3;
127         }

```

```

128
129      // waits for the key to be released.
130      while (~(key = 0)) {
131          let key = Keyboard.keyPressed();
132          do delay(speed);
133
134          if (~(s.move(direction))){
135              do Output.moveCursor(10,25);
136              do Output.printString("Game Over :(");
137              return;
138          }
139          do Output.moveCursor(0,38);
140          do Output.moveCursor(0,37);
141          do Output.moveCursor(0,36);
142          do Output.printInt(s[5]);
143      }
144  }
145
146      return;
147  }
148
149  method void printStart(){
150      do Screen.clearScreen();
151      do Screen.setColor(true);
152      do Screen.drawLine(15, 15, 15, 240);
153      do Screen.drawLine(500, 15, 500, 240);
154      do Screen.drawLine(15, 15, 500, 15);
155      do Screen.drawLine(15, 240, 500, 240);
156      do Output.moveCursor(5,25);
157      do Output.printString("Please choose speed (0-9):");
158      do Output.moveCursor(0,30);
159      do Output.printString("Score:");
160
161
162      return;
163  }
164
165
166
167  }

```