Software Development Report

CSC 221: Programming II: Fall 2023

Manel Casado

Linked List Lab: Worm

Problem Summary

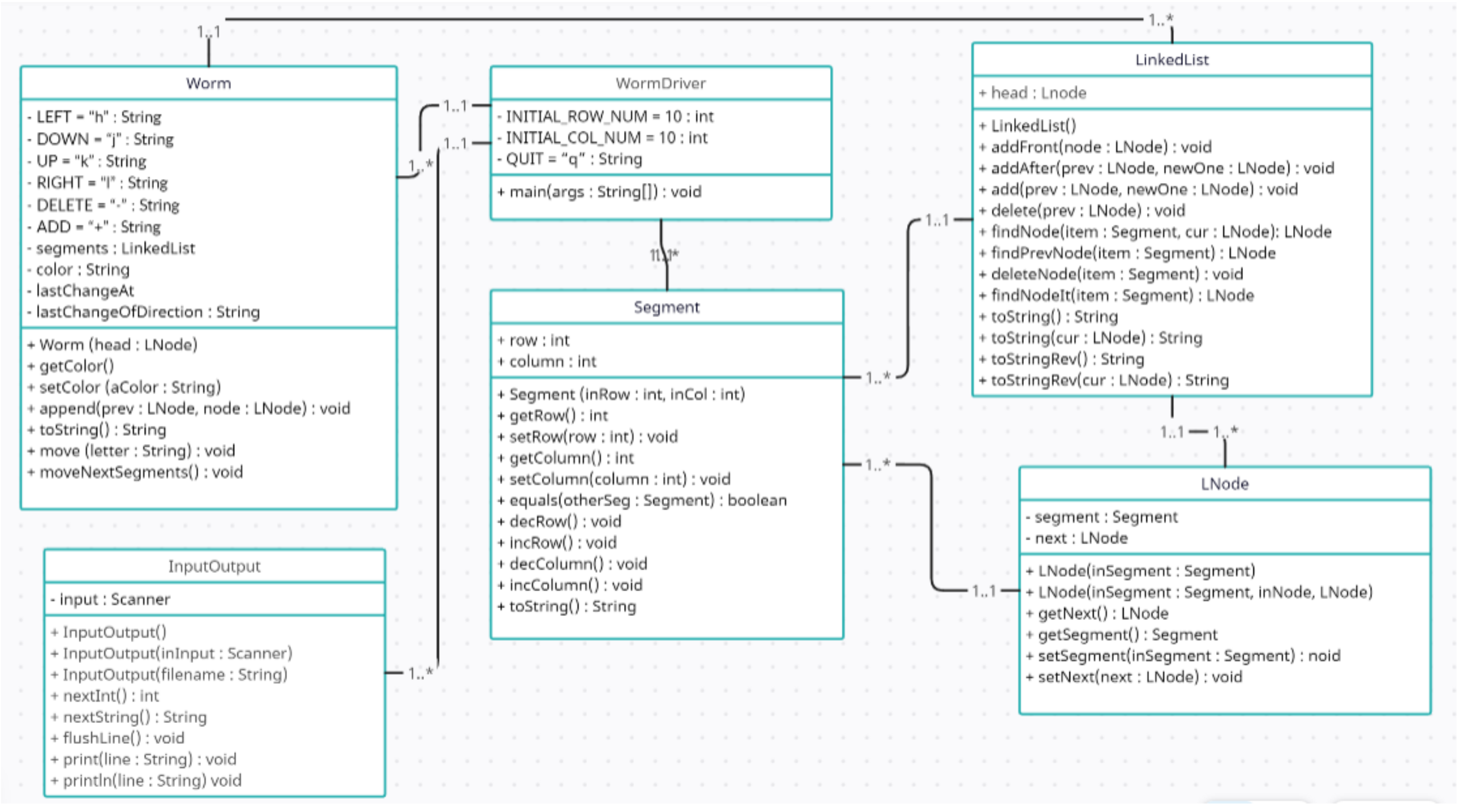
The goal of this program is to create and maintain a “worm” made of different segments using a linked list, this has to be done by positioning the segments in rows and columns in a board addressing the previous and following segments. The worm can move left (h), right (l), up (k), and down (j), and it can also increase and decrease its size by adding or deleting a segment (+/-) by inputting the specified keys.

To make this possible I needed to implement the following classes: *Worm*, *Segment*, *LinkedList*, *Lnode*, *InputOutput*, and *WormDriver*.

Implementation Requirements

- Compile all classes
- Run *WormDriver.java*
- Choose the color of the worm
- Move the worm around using the different keys

System Design



1..1
1..*
Worm
- LEFT = "h" : String
- DOWN = "j" : String
- UP = "k" : String
- RIGHT = "l" : String
- DELETE = "-" : String
- ADD = "+" : String
- segments : LinkedList
- color : String
- lastChangeAt
- lastChangeOfDirection : String
+ Worm (head : LNode)
+ getColor()
+ setColor (aColor : String)
+ append(prev : LNode, node : LNode) : void
+ toString() : String
+ move (letter : String) : void
+ moveNextSegments() : void
1..1
1..1
1..*
WormDriver
- INITIAL_ROW_NUM = 10 : int
- INITIAL_COL_NUM = 10 : int
- QUIT = “q” : String
+ main(args : String[]) : void
11..1*
Segment
+ row : int
+ column : int
+ Segment (inRow : int, inCol : int)
+ getRow() : int
+ setRow(row : int) : void
+ getColumn() : int
+ setColumn(column : int) : void
+ equals(otherSeg : Segment) : boolean
+ decRow() : void
+ incRow() : void
+ decColumn() : void
+ incColumn() : void
+ toString() : String
1..*
1..*
1..1
LinkedList
+ head : Lnode
+ LinkedList()
+ addFront(node : LNode) : void
+ addAfter(prev : LNode, newOne : LNode) : void
+ add(prev : LNode, newOne : LNode) : void
+ delete(prev : LNode) : void
+ findNode(item : Segment, cur : LNode): LNode
+ findPrevNode(item : Segment) : LNode
+ deleteNode(item : Segment) : void
+ findNodeIt(item : Segment) : LNode
+ toString() : String
+ toString(cur : LNode) : String
+ toStringRev() : String
+ toStringRev(cur : LNode) : String
1..1
1..*
LNode
- segment : Segment
- next : LNode
+ LNode(inSegment : Segment)
+ LNode(inSegment : Segment, inNode, LNode)
+ getNext() : LNode
+ getSegment() : Segment
+ setSegment(inSegment : Segment) : noid
+ setNext(next : LNode) : void
1..1
InputOutput
- input : Scanner
+ InputOutput()
+ InputOutput(inInput : Scanner)
+ InputOutput(filename : String)
+ nextInt() : int
+ nextString() : String
+ flushLine() : void
+ print(line : String) : void
+ println(line : String) void
1..*

Acceptance Testing Plan

Name	Description	Input	Expected Output	Actual Output	Pass?

Test1	Select a color for the worm and make sure it saves and outputs	green	Worm color: green	`Which color:`
				▶▶ `green`
				`Worm color: green`
				`Segments: [10,10] [11,10] [12,10] [13,10] [14,10] [15,10]`
				`Next movement and press enter:`
				▶▶
					Yes

Test2	Make sure the worm has it's head at 10,10 and 6 segments at the beginning	green	[10,10] [11,10] [12,10] [13,10] [14,10] [15,10]	`Which color:`
				▶▶ `green`
				`Worm color: green`
				`Segments: [10,10] [11,10] [12,10] [13,10] [14,10] [15,10]`
				`Next movement and press enter:`
				▶▶
					Yes

Test3	Move left	h	Worm shifts one position to the left so head is in (10, 9). The rest of the segments +1 row	`Next movement and press enter:`
				▶▶ `h`
				`Worm color: green`
				`Segments: [10,9] [10,10] [11,10] [12,10] [13,10] [14,10]`
					Yes

Test4	Move up	j	Worm moves one position upwards so head is in (11, 9). The rest of the segments move to the previous segment position	`Next movement and press enter:`
				▶▶ `j`
				`Worm color: green`
				`Segments: [11,9] [10,10] [11,10]`
					No, only head moved

				[12,10] [13,10] [14,10]	
Test5	Move right	l	Worm moves one position to the left so head is in (10, 11). The rest of the segments move to the previous segment position	│ Next movement and press enter: ▶▶│ l │ Worm color: green │ Segments: [10,11] [10,10] [11,10] [12,10] [13,10] [14,10]	Yes
Test6	Move down	k	Worm moves one position upwards so head is in (9, 11). The rest of the segments move to the previous segment position	│ Next movement and press enter: ▶▶│ k │ Worm color: green │ Segments: [9,11] [10,10] [11,10] [12,10] [13,10] [14,10]	No, only head moved
Test7	Adds a segment to the end of the worm	+	Segment added to the end	│ Next movement and press enter: ▶▶│ + │ Worm color: green │ Segments: [9,11] [10,10] [11,10] [12,10] [13,10] [14,10] [15,10]	Yes
Test8	Removes head from the form and assigns head to the first segment	-	Head removed	│ Next movement and press enter: ▶▶│ - │ Worm color: green │ Segments: [10,10] [11,10] [12,10] [13,10] [14,10] [15,10]	Yes
Test9	Quit game	q	Run ends	│ Next movement and press enter:	Yes

				▶▶│ q │ └ ----jGRASP: operation complete.	
Test10	Wrong input	a	Asks for move again	│ Next movement and press enter: ▶▶│ a │ Worm color: green │ Segments: [10,10] [11,10] [12,10] [13,10] [14,10] [15,10] │ Next movement and press enter:	Yes

Estimation of Time required

In my Software Development Report, I projected this program to take me 8 hours between coding, testing, and SDR. It ended up taking me about 9 hours.

Outside Resources used

- Dr. Bareiss lecture code from Google Drive
- Creately to make UML
- InputOutput.java from last lab

Security Report

- Using third-party code is not the safest thing to do
- Program is well structured and every class is as independent as possible

Ethical Report

- As all games, this one when perfected could cause addiction

Future Improvements

- Print board to show the user the worm
- Create a game out of this program
- Make left and right movements work correctly so not only the head is moving

Lessons Learned

- Linked Lists and Nodes

Improvements of Work

- Not aplicable