Test 5

The fifth test will be 75 minutes, will consist of two parts and will take place this week.

The programming part will be about recursion. You will be asked to implement two recursive methods. This part will be worth 50%. If your code does not compile, you get a 50% penalty (that is, your score for the programming part will be divided by 2 if your code does not compile).

The "written" part will be about recursion (prove correctness and termination (15% each), determine the recurrence relation (5%) and a big-O proof (15%)).

During the test, you will have access to the textbook. You may bring a blank piece of paper to the test.

Implementing Data Structures

EECS 1030

`moodle.yorku.ca`

Data structures

Data structures

A data structure represents data and operations to manipulate that data.

Data structures

Data structures

A data structure represents data and operations to manipulate that data.

Data structures in Java

In Java, data structures are usually represented by classes, where the data is generally captured by means of attributes and the operations are usually represented by methods.

Data structures

Data structures

A data structure represents data and operations to manipulate that data.

Data structures in Java

In Java, data structures are usually represented by classes, where the data is generally captured by means of attributes and the operations are usually represented by methods.

Examples

ArrayList, LinkedList, HashSet, TreeSet, HashMap, and TreeMap.

ListString

The ListString interface is a simplified version of the List interface.

- The elements of the list are strings.
- Not all methods of List are included in ListString.
- Some of the methods of List are simplified.

The API of the ListString interface can be found here.

LinkedListString

Problem

Implement the `ListString` using a linked list of nodes.

LinkedListString

Problem

Implement the `ListString` using a linked list of nodes.

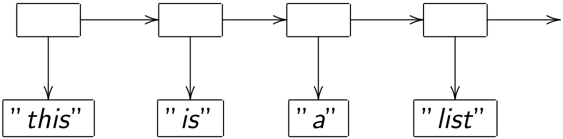
```
["this", "is", "a", "list"]
```

LinkedListString

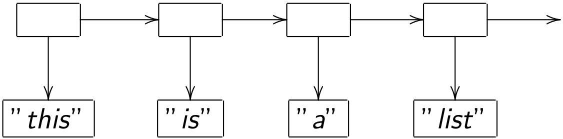
Problem

Implement the `ListString` using a linked list of nodes.

```
["this", "is", "a", "list"]
```



Node



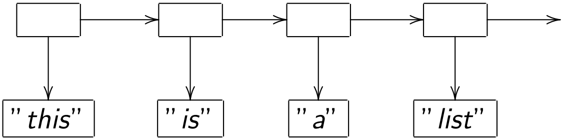
Question

The empty boxes represent the `Node` objects. The nonempty boxes represent the `String` objects.

An arrow from one box to another represents that the former object has an attribute whose value is the latter object.

How many attributes does the `Node` object have?

Node



Question

The empty boxes represent the `Node` objects. The nonempty boxes represent the `String` objects.

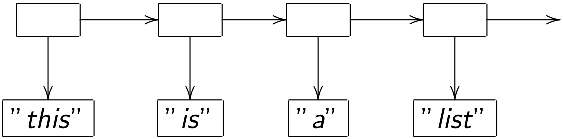
An arrow from one box to another represents that the former object has an attribute whose value is the latter object.

How many attributes does the `Node` object have?

Answer

Two.

Node



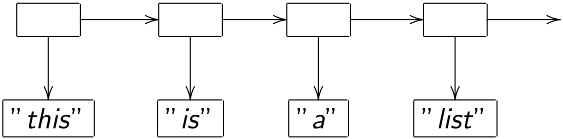
Question

The empty boxes represent the `Node` objects. The nonempty boxes represent the `String` objects.

An arrow from one box to another represents that the former object has an attribute whose value is the latter object.

What are the types of the two attributes of the `Node` class?

Node



Question

The empty boxes represent the `Node` objects. The nonempty boxes represent the `String` objects.

An arrow from one box to another represents that the former object has an attribute whose value is the latter object.

What are the types of the two attributes of the `Node` class?

Answer

`String` and `Node`.

Node

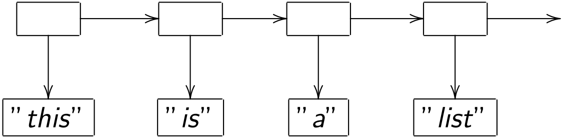
Draw the UML diagram of the Node class.

Node

Problem

Create a class named Node with attributes element of type String and next of type Node.

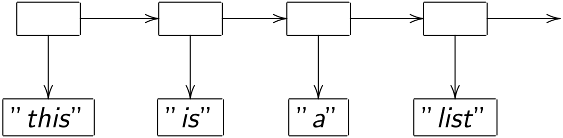
Node



Problem

Write a snippet of Java code that creates the above structure.

Node



```
Node node = new Node("list", null);
node = new Node("a", node);
node = new Node("is", node);
node = new Node("this", node);
```

Node

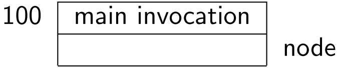
```
Node node = new Node("list", null);
node = new Node("a", node);
node = new Node("is", node);
node = new Node("this", node);
```

Problem

Draw the corresponding memory diagrams.

Memory diagrams

```
Node node = new Node("list", null);
node = new Node("a", node);
node = new Node("is", node);
node = new Node("this", node);
```

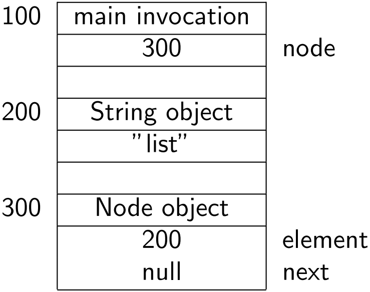


100 main invocation

node

Memory diagrams

```
Node node = new Node(”list”, null);
node = new Node(”a”, node);
node = new Node(”is”, node);
node = new Node(”this”, node);
```



100	main invocation	
	300	node
200	String object	
	”list”	
300	Node object	
	200	element
	null	next

Memory diagrams

```
Node node = new Node("list", null);
node = new Node("a", node);
node = new Node("is", node);
node = new Node("this", node);
```

100	main invocation	
	500	node
200	String object	
	"list"	
300	Node object	
	200	element
	null	next
400	String object	
	"a"	
500	Node object	
	400	element
	300	next

ListString

Problem

Create the class `LinkedListString` that implements the interface `ListString`.

Use recursion.[a]

[a] All methods can be implemented using iteration, but we want to practice recursion some more.