CSE 12: Week 3 Discussion

1-19-21

Focus: Linked List, Generics, and 2D Arrays

LinkedLists

Linked List Implementation

```
public interface StringList {
  /* Add an element at the beginning of the list */
  void prepend(String s);
  /* Add an element at the end of the list */
  void add(String s);
  /* Get the element at the given index */
  String get(int index);
  /* Get the number of elements in the list */
  int size();
  /* Add an element at the specified index */
  void insert(int index, String s);
  /* Remove the element at the specified index */
  void remove(int index);
}
```

```
class Node {
  //FILL IN IMPLEMENTATION OF CLASS
}

public class LinkedStringList implements StringList {
  Node front;
  //IMPLEMENTATION DONE DURING LECTURE
}
```

Which of the following fields does the Node class need?

a. String value
b. Node next
c. Node previous
d. a and b
e. a, b, and c

Answer - d (String value and Node next)

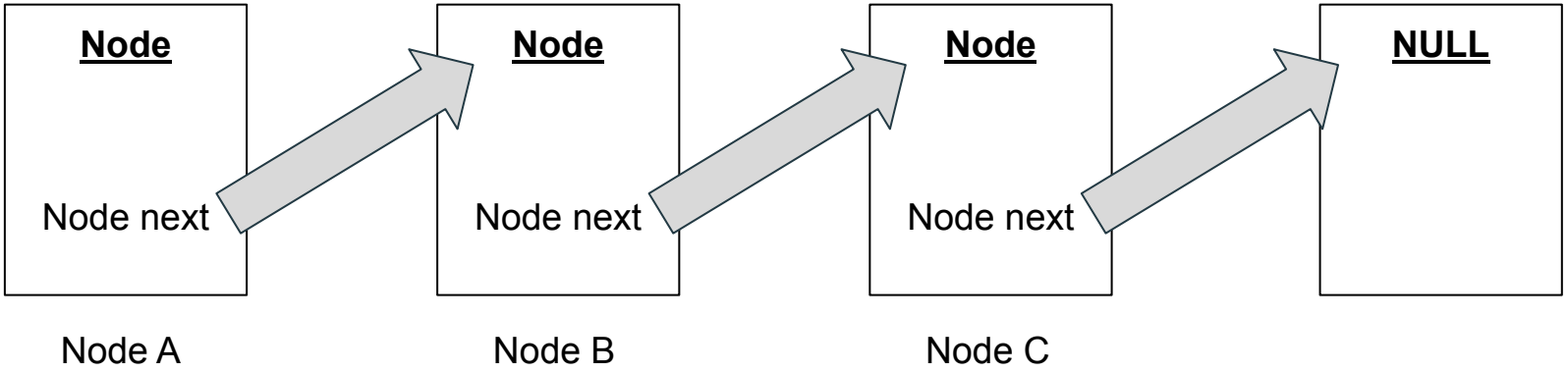
```
public interface StringList {
  /* Add an element at the beginning of the list */
  void prepend(String s);
  /* Add an element at the end of the list */
  void add(String s);
  /* Get the element at the given index */
  String get(int index);
  /* Get the number of elements in the list */
  int size();
  /* Add an element at the specified index */
  void insert(int index, String s);
  /* Remove the element at the specified index */
  void remove(int index);
}
```

```
class Node {
  String value;
  Node next;
  public Node(String value, Node next) {
    this.value = value;
    this.next = next;
}

public class LinkedStringList implements StringList {
  Node front;
  //IMPLEMENTATION DONE DURING LECTURE
}
```

We need to keep track of the data in the current node (which is a String) and keep track of the next node attached to the current node

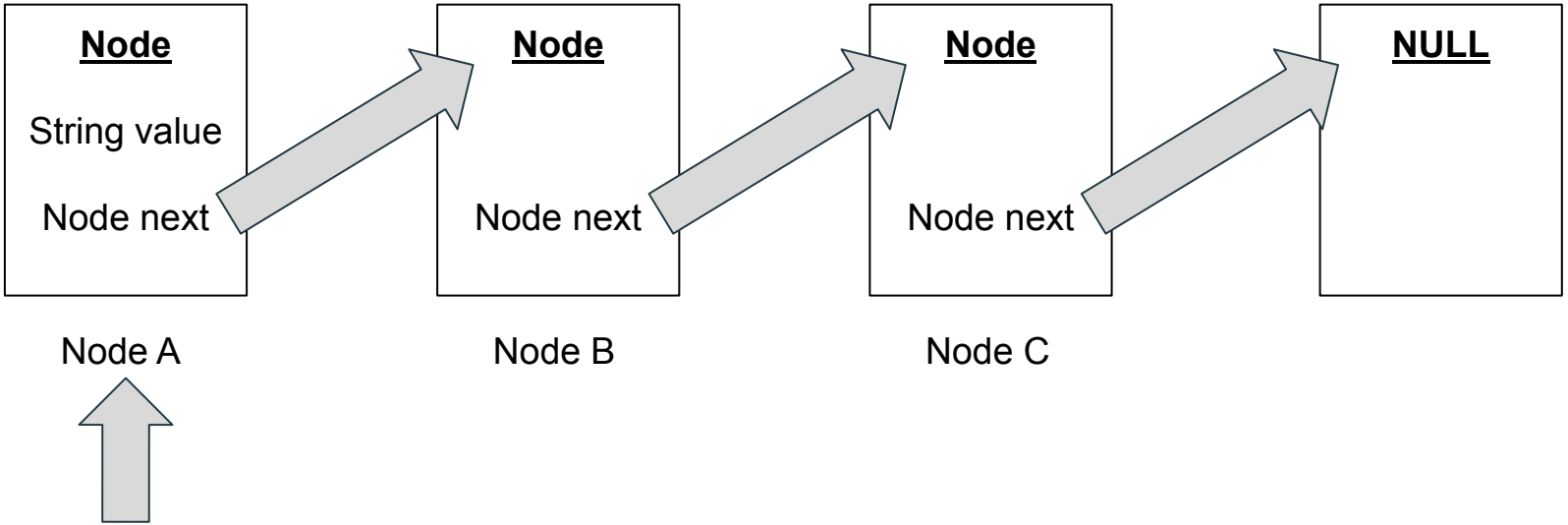
Linked List Understanding Check



If we are allowed to only keep track of one node in the LinkedStringList class, which node should we keep track of?

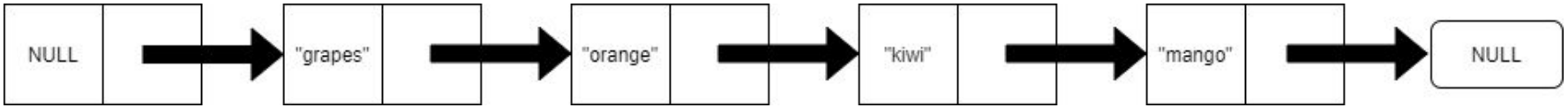
a. Node A
b. Node B
c. Node C
d. More than one of these

Answer - a (Node A)



RECALL FROM LECTURE - this starting node of the list is stored as Node first in the LinkedStringList class as a field

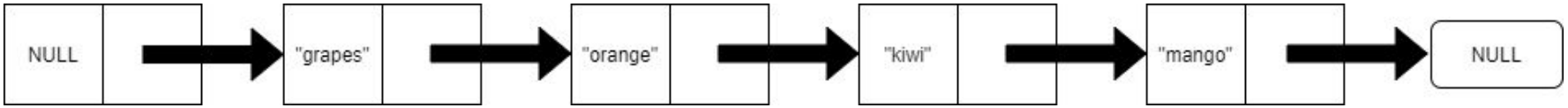
LinkedList



From the diagram above, how many elements and how many nodes are there?

A) Elements: 5, Nodes: 5
B) Elements: 5, Nodes: 6
C) Elements: 4, Nodes: 5
D) Elements: 4, Nodes: 6

LinkedList - Dummy Node

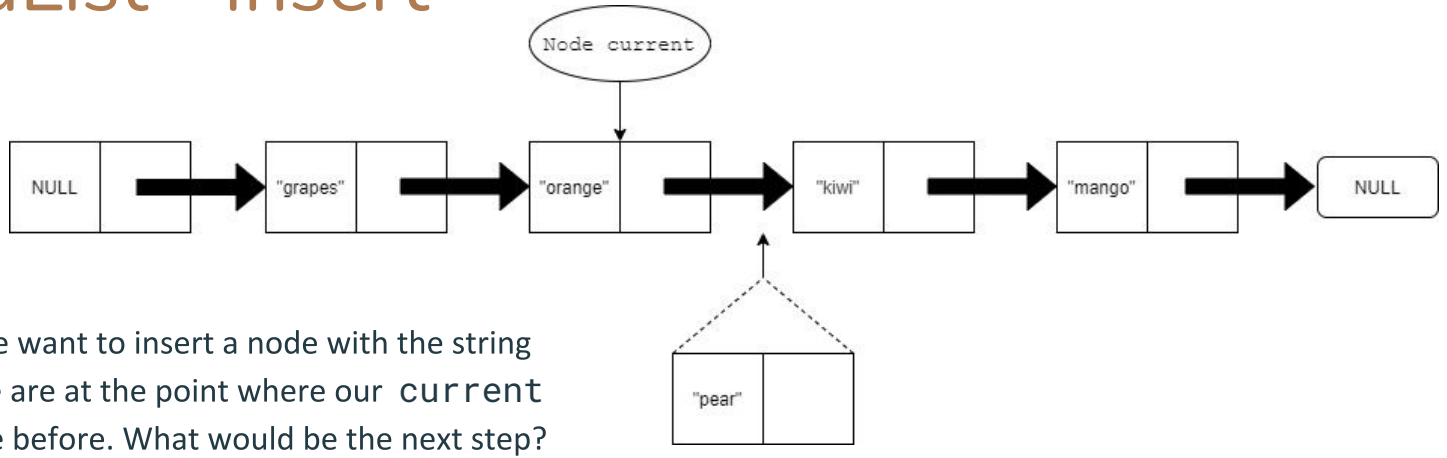


From the diagram above, how many elements and how many nodes are there?

A) Elements: 5, Nodes: 5
B) Elements: 5, Nodes: 6
C) Elements: 4, Nodes: 5
D) Elements: 4, Nodes: 6

In the diagram, we can see that a dummy node is implemented at the front. This is still a node object, however, it is not an element in the LinkedList!

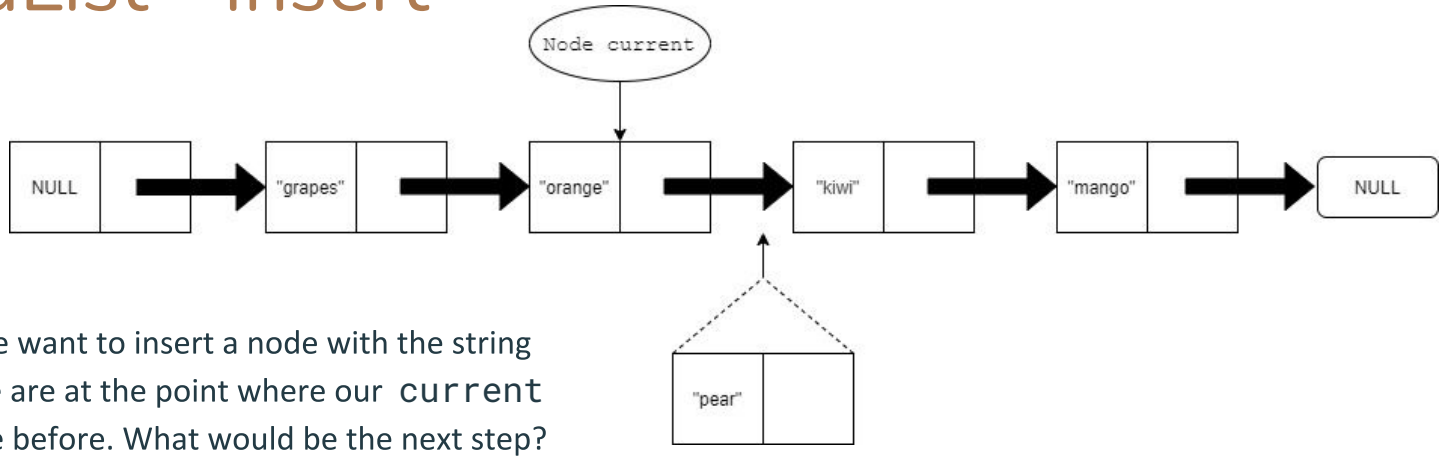
LinkedList - Insert



From the diagram, we want to insert a node with the string "pear" at index 2. We are at the point where our `current` node is set to the one before. What would be the next step?

A) `current = current.next;`
B) `current.next = new Node("pear", current.next.next);`
C) `current = current.next.next;`
D) `current.next = new Node("pear", current.next);`

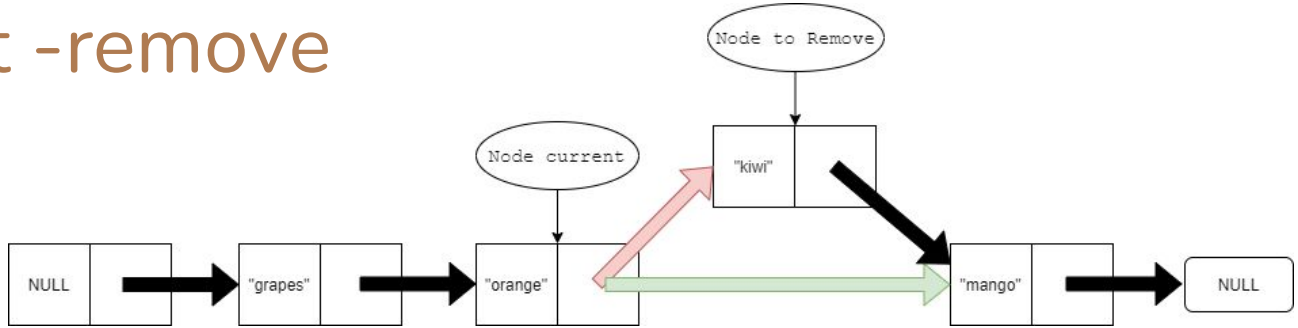
LinkedList - Insert



From the diagram, we want to insert a node with the string "pear" at index 2. We are at the point where our `current` node is set to the one before. What would be the next step?

A) `current = current.next;`
B) `current.next = new Node("pear", current.next.next);`
C) `current = current.next.next;`
D) `current.next = new Node("pear", current.next);`

LinkedList -remove



In the above diagram, we want to remove the node at index 2.

We are at the point where our `current` node is at index 1.

Which of the following lines of code achieve our goal?

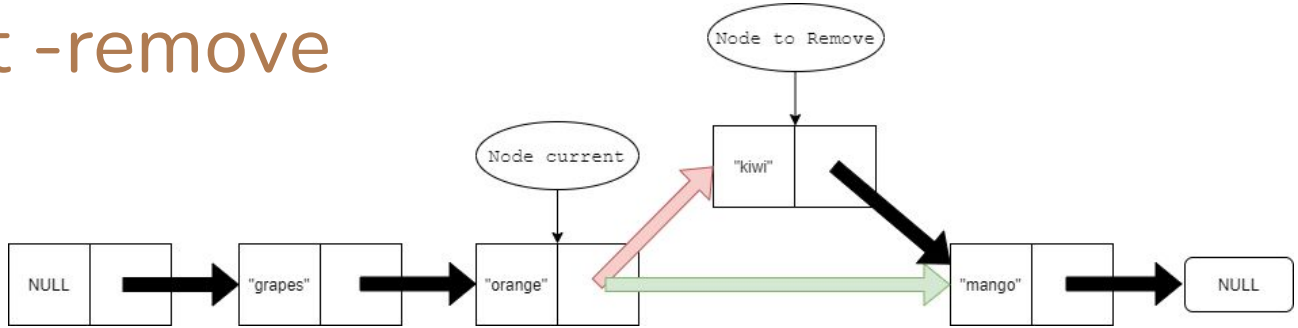
A) `current = current.next;`

B) `current.next.next = current.next;`

C) `current.next = current.next.next;`

D) `current = current.next.next;`

LinkedList -remove



In the above diagram, we want to remove the node at index 2.

We are at the point where our `current` node is at index 1.

Which of the following lines of code achieve our goal?

A) `current = current.next;`

B) `current.next.next = current.next;`

C) `current.next = current.next.next;`

D) `current = current.next.next;`

Building Intuition

How can we make the current way we are implementing singly linked lists more versatile?

a. Make the Node class hold any data type aside from just Strings
b. Add array holding all other nodes in a linked list as a field in the Node class
c. Have reference to the last node in the linked list in the LinkedStringList class

Answer - a

Our current implementation of the linked list only allows Strings to be stored in each Node instance.

Too limiting - may want to create a linked list to hold different data type.

SOLUTION - Generics!

Generics

Generics

Multiple classes may be nearly identical, differing only in their data types they contain…

Generics allow us to implement classes without limiting the data type that we can store in the class

Implement Linked List with Generics

```
public interface StringList {
  /* Add an element at the beginning of the list */
  void prepend(? s);
  /* Add an element at the end of the list */
  void add(? s);
  /* Get the element at the given index */
  ? get(int index);
  /* Get the number of elements in the list */
  int size();
  /* Add an element at the specified index */
  void insert(int index, ? s);
  /* Remove the element at the specified index */
  void remove(int index);
}
```

```
class Node {
  ? value;
  Node next;
  public Node(? value, Node next) {
    this.value = value;
    this.next = next;
}

public class LinkedStringList implements List {
  Node front;
  //IMPLEMENTATION DONE DURING LECTURE
}
```

What syntax would make the StringList interface a generic class?

a. public interface<Element> StringList
b. generic interface<Element> StringList
c. public interface StringList<Element>
d. public generic interface StringList<Element>
e. It is fine as it is

Answer - c

```
public interface StringList<Element> {
  /* Add an element at the beginning of the list */
  void prepend(? s);
  /* Add an element at the end of the list */
  void add(? s);
  /* Get the element at the given index */
  ? get(int index);
  /* Get the number of elements in the list */
  int size();
  /* Add an element at the specified index */
  void insert(int index, ? s);
  /* Remove the element at the specified index */
  void remove(int index);
}
```

```
class Node<Element> {
  ? value;
  Node next;
  public Node(? value, Node next) {
    this.value = value;
    this.next = next;
}

public class LinkedStringList<Element> implements StringList {
  Node front;
  //IMPLEMENTATION DONE DURING LECTURE
}
```

Similarly change all class headers that would require information about generic data type:

- Node class
- StringList interface
- LinkedStringList class

Change Class Names to Reflect Generic Behavior

```
public interface List<Element> {
  /* Add an element at the beginning of the list */
  void prepend(? s);
  /* Add an element at the end of the list */
  void add(? s);
  /* Get the element at the given index */
  ? get(int index);
  /* Get the number of elements in the list */
  int size();
  /* Add an element at the specified index */
  void insert(int index, ? s);
  /* Remove the element at the specified index */
  void remove(int index);
}
```

```
class Node<Element> {
  ? value;
  Node next;
  public Node(? value, Node next) {
    this.value = value;
    this.next = next;
}

public class LinkedList<Element> implements List {
  Node front;
  //IMPLEMENTATION DONE DURING LECTURE
}
```

- StringList > List
- LinkedStringList > LinkedList

Implement Linked List with Generics

```
public interface List<Element> {
  /* Add an element at the beginning of the list */
  void prepend(? s);
  /* Add an element at the end of the list */
  void add(? s);
  /* Get the element at the given index */
  ? get(int index);
  /* Get the number of elements in the list */
  int size();
  /* Add an element at the specified index */
  void insert(int index, ? s);
  /* Remove the element at the specified index */
  void remove(int index);
}
```

```
class Node<Element> {
  ? value;
  Node next;
  public Node(? value, Node next) {
    this.value = value;
    this.next = next;
}

public class LinkedList<Element> implements List {
  Node front;
  //IMPLEMENTATION DONE DURING LECTURE
}
```

What do we have to do in the class definitions to make linked list generalizable to all data types instead of just Strings?

a. Remove all String variables in the class definitions
b. Add <Element> next to each String variable
c. Replace all String variables with Element as the data type
d. Call the Node constructors using generics
e. Both c and d

Answer - e

```
public interface List<Element> {
  /* Add an element at the beginning of the list */
  void prepend(Element e);
  /* Add an element at the end of the list */
  void add(Element e);
  /* Get the element at the given index */
  Element get(int index);
  /* Get the number of elements in the list */
  int size();
  /* Add an element at the specified index */
  void insert(int index, Element e);
  /* Remove the element at the specified index */
  void remove(int index);
}
```

```
class Node<Element> {
  Element e;
  Node<Element> next;
  public Node(Element e, Node next) {
    this.value = value;
    this.next = next;
}

public class LinkedList<Element> implements List {
  Node<Element> front;
  //IMPLEMENTATION DONE DURING LECTURE
}
```

yellow = replaced String with Element

green = called Node constructor with generic Element data type

2D Arrays

What is a 2D Array?

A 2D array is a matrix where we are able to hold values given two indices.

Elements are arranged in rows and columns

Declaration of empty 2D String Array:

String[][] fruit = new String[3][4];

	0	1	2	3
0				
1				
2				

Accessing Elements

fruit[<row_index>][<col_index>]

You can reassign or you can just access the element

fruit[0][0] = “grapefruit”;

System.out.println(fruit[0][0]);

	0	1	2	3
0	“grape”	“orange”	“apple”	“lemon”
1	“peach”	“lime”	“pear”	“mango”
2	“kiwi”	“plum”	“cherry”	“melon”

Which element would be accessed?

fruit[2][1] = ?

A) “pear”
B) “cherry”
C) “peach”
D) “plum”

	0	1	2	3
0	“grape”	“orange”	“apple”	“lemon”
1	“peach”	“lime”	“pear”	“mango”
2	“kiwi”	“plum”	“cherry”	“melon”

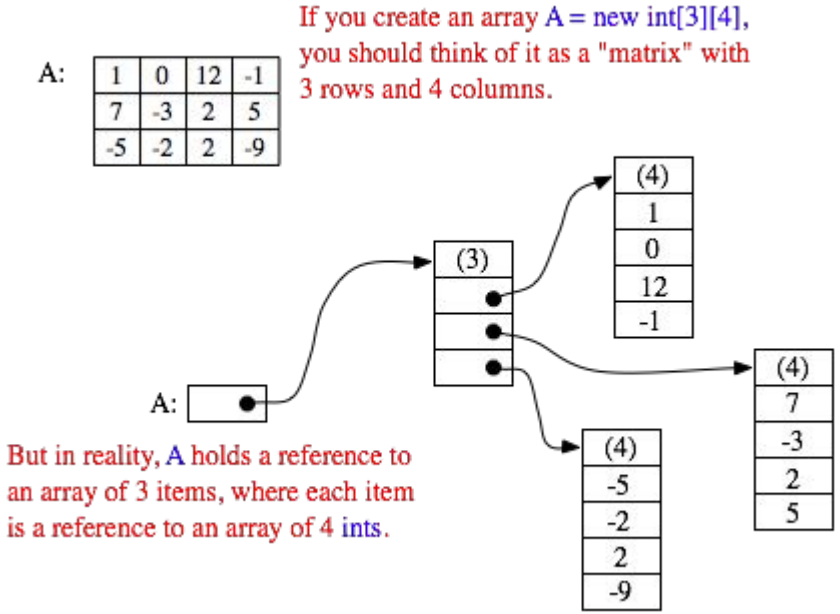
Which element would be accessed?

fruit[2][1] = ?

A) “pear”
B) “cherry”
C) “peach”
D) “plum”

	0	1	2	3
0	“grape”	“orange”	“apple”	“lemon”
1	“peach”	“lime”	“pear”	“mango”
2	“kiwi”	“plum”	“cherry”	“melon”

Memory



Reference: http://math.hws.edu/eck/cs124/javanotes5/c7/s5.html