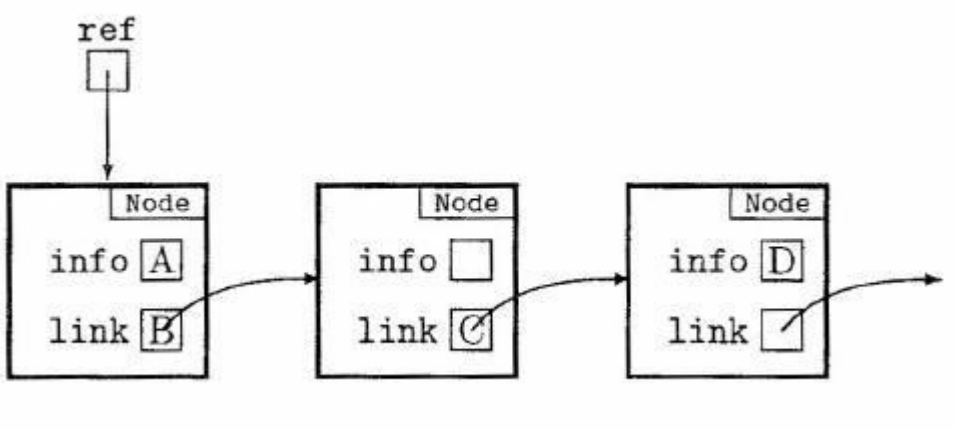
Linked List – Exercise

Section 1:

1. In the diagram, ref is a reference to an object of the type Node discussed in this section. The field marked A can be referred to as ref. info. Find similar names for the fields marked B, C, and D.



- The field marked A can be referred to as ref Info
- The field marked B can be referred to as the second Node's info
- The field marked C can be referred to as D info

2. Rewrite the method printList shown in Example 3 so that, as well as printing the values in the list, it also prints a message if the list is empty.

```
/*
 * This method prints the list
 * pre: none
 * post: Outputs list
*/
public void printList () {
	Node current = head;
	if (current == null){
		System.out.println("The list is EMPTY");
	}
	else {
		while (current.next != null || current != null) {
			System.out.print(current.data + " \t");
			if (current.next != null)
				current = current.next;
			else {
				current = current;
				break;
			}
		}
```

```
                        System.out.println();
                }

        }
```

3. Write an instance method sum for the List class that returns the sum of the values in the info fields of its implicit List object.

```
        /*
         * This method returns the sum of list
         * pre: none
         * post: returns sum
        */
        public int sum () {
                Node current = head;
                int sum = 0;
                while (current != null || current.next != null) {
                        sum = sum + current.data;
                        if (current.next != null)
                                current = current.next;
                        else {
                                current = current;
                                break;
                        }
                }
                return sum;
        }
```

4. Write an instance method deleteFirst for the List class that deletes the first node of a linked list. If the list is empty, the method should print a warning message

```
        /*
         * This method deletes the first node of the list
         * pre: none
         * post: none
        */
        public void deleteFirst () {
                if (head == null) {
                        System.out.println("WARNING - This list is empty
!");
                }
                else {
                        head = head.next;
                }

        }
```

5. Write an instance method deleteLast for the List class that deletes the last node of a linked list. If the list is empty, the method should print a warning message.

```
        /*
         * This method deletes the LAST node of the list
```

```
         * pre: none
         * post: none
        */
       public void deleteLast () {
              Node current = head;
              if (head == null) {
                     System.out.println("WARNING - This list is empty
!");
              }
              else {
                     while (current.next.next != null) {

                            current = current.next;
                     }
                     current.next = null;
              }
       }
```

6. Write a toString method for the List class. The method should return a string that contains all the inf 0 fields of the list, separated by / /. For example, if the list contained the integers 3, 5, and 8 in its info fields, the method should return "3//5//8".

```
       /*
        * This method returns a string that contains all the info fields
of the list
        * pre: none
        * post: returns string
       */
       public String toString () {
              Node current = head;
              String string = "";
              if (head == null) {
                     string = "WARNING - This list is empty";
              }
              else {
                     while (current.next != null || current != null) {
                            string = string + current.data + "//";
                            if (current.next != null)
                                   current = current.next;
                            else {
                                   current = current;
                                   break;
                            }
                     }
              }
              return string;

       }
```

7. Write an instance method addAtRear for the List class. The method should have a single int parameter called item. The method should first locate the last node in a list and then create and attach a new node, containing the value of item, at that

end of the list. In writing your method, be sure that it works correctly for an empty list.

```
public void addAtRear (int item) {
            Node current = head;
            if (head == null) {
                  head = new Node (item,null);
            }
            else {
                  while (current.next != null) {
                        current = current.next;
                  }
                  System.out.println(current.data);
                  current.next = new Node (item,null);
            }

}
```

Section 2:

1) Define a linked list (called List) of the class fraction. Ensure that fraction has the following private fields: int num and int den. Ensure that all instance methods associated with the class fields are included. Create the following constructors:
   a. fraction ()
      A constructer that will initialize num to 1 and den to 1 and links to null.
   b. fraction (int n, int d, fraction f)
      A constructer that will initialize num to n and den to d and links to f.

2) By making the class fields in fraction private, how does this affect the rest of your code?

It effects the rest of the code as I can no longer access and change the fields in the class Fraction directly. Hence, I have to use the accessor methods in order to access the den and num from Fraction.

3) Create the following instance methods in your List class:

   a. insertFirst(int n, int d)
      The following instance method (in class **List)** will insert *n(num) and d(den)* to an empty linked list.

```
public void insertFirst (int n, int d){
            if (head == null) {
                  head  = new Fraction (n,d,null);
            }
            else {
                  head = head  = new Fraction (n,d,head.link);
            }
}
```

b. insertSecond(int n, int d)

The following instance method (in class List) will insert *n(num) and d(den)* as the second fraction.

```
public void insertSecond (int n, int d){
		if (head.link == null) {
			head.link = new Fraction (n,d,null);
		}
		else {
			head.link = new Fraction (n,d,head.link.link);
		}
}
```

c. insertThird(int n, int d)

The following instance method (in class List) will insert *n(num) and d(den)* as the third fraction

```
public void insertThird (int n, int d){
		if (head.link.link == null) {
			head.link.link = new Fraction (n,d,null);
		}
		else {
			head.link.link = new Fraction (n,d,head.link.link.link);
		}
}
```

d. addatEnd(int n, int d)

The following instance method (in class List) will insert *n(num) and d(den)* at the end of the linked list.

```
public void addatEnd(int n, int d) {
		if (head == null) {
			head = new Fraction (n,d,null);
		}
		else {
			Fraction current = head;
			while (current.link != null) {
				current = current.link;
			}
			current.link = new Fraction (n,d,null);
		}
	}

	public void addatFront () {
		head = new Fraction (1,1,head.link);
	}

	public void addatFront(int n, int d) {
		head = new Fraction (n,d,head.link);
}
```

e. addatFront()

The following instance method (in class List) will insert a fraction (1,1) at the beginning of the Linked List

```
public void addatFront () {
		head = new Fraction (1,1,head.link);
}
```

f. addatFront(int n, int d)

The following instance method (in class List) will insert a fraction (n,d) at the beginning of the Linked List

```
public void addatFront(int n, int d) {
		Fraction temp = head;
		head = new Fraction (n,d,temp);
}
```

g. boolean emptyList()

The following class method returns a value of true if the list of students is empty. Otherwise, it returns false.

```
public boolean emptyList () {
		if (head == null)
			return true;
		else
			return false;
}
```

h. printList()

The following class method will print the fraction contained in the list.

```
public void printList() {
		Fraction current = head;
		if (current == null){
			System.out.println("The list is EMPTY");
		}
		else {
			while (current.link != null || current != null) {
				System.out.print(current.num() + "/" + current.den()
+ " \t");
				if (current.link != null)
					current = current.link;
				else {
					break;
				}
			}
			System.out.println();
		}
```

```
	}
```

i. printFraction(int x)

This method traverses through the linked list and prints the x^{th} fraction.

```
public void printFraction(int x) {
		Fraction current = head;
		int count = 0;
		if (head == null) {
			System.out.println("The list is EMPTY");
		}
		else {
			while (count < x && (current.link != null || current != 
null)) {
				count++;
				if (current.link != null && count < x) {
					current = current.link;
				}
				else {
					break;
				}
			}
			if (count < x) {
				System.out.println(x + "th fraction does not EXIST 
!");
			}
			else {
				System.out.println(x + "th fraction is " + 
current.num() + "/" + current.den());
			}
		}
	}
```

j. int getNum(int x)

This method returns num from the x^{th} fraction in the linked list. This method will return -1 if the there is no x^{th} fraction.

```
public int getNum (int x) {
		Fraction current = head;
		int count = 1;
		int num = 0;
		if (head == null) {
			num = -1;
		}
		else {
			while (count < x && (current.link != null || current != 
null)) {
				count++;
				if (current.link != null) {
					current = current.link;
				}
				else {
```

```
                                    break;
                              }
                        }
                        if (count < x) {
                              num =  -1;
                        }
                        else {
                              num = current.num();
                        }
                  }
                  return num;
            }
```

k. putNum(int x, int n)

This method assigns num from the x^{th} fraction in the linked list to n.

```
public void putNum(int x, int n) {
                  Fraction current = head;
                  int count = 0;
                  if (head == null) {
                        System.out.println("The list is EMPTY");
                  }
                  else {
                        while (count < x && (current.link != null || current !=
null)) {
                              count++;
                              if (current.link != null && count < x) {
                                    current = current.link;
                              }
                              else {
                                    break;
                              }
                        }
                        if (count < x) {
                              System.out.println(x + "th fraction does not EXIST
!");
                        }
                        else {
                              current = new Fraction (n,current.den(),
current.link);
                        }
                  }
            }
```

l. int getDen(int x)

This method returns den from the x^{th} fraction in the linked list. This method will return -1 if the there is no x^{th} fraction.

```
public int getDen (int x) {
                  Fraction current = head;
                  int count = 1;
                  int den = 0;
```

```
		if (head == null) {
			den = -1;
		}
		else {
			while (count < x && (current.link != null || current != 
null)) {
				if (current.link != null)
					current = current.link;
				else
					break;
				count++;
			}
		}

		if (count < x)
			den = -1;
		else
			den = current.den();

		return den;
	}
```

m. putDen(int x, int d)

This method assigns num from the x^{th} fraction in the linked list to d.

```
	public void putDen(int x, int d) {
		Fraction current = head;
		int count = 1;

		if (head == null) {
			System.out.println("The list is EMPTY");
		}
		else {
			while (count < x && (current != null || current != null))
{
				count++;
				if (current.link != null && count < x) {
					current = current.link;
				}
				else {
					break;
				}

				if (count < x) {
					System.out.println(x + "th fraction does not
EXIST !");
				}
				else {
					current = new Fraction (d,current.den(),
current.link);
				}
			}
```

```
}
```

4) Write a main program that will prompt for the numerator and the denominator of three fractions and link them together in a list using the first three instance methods of the previous question.

```
//QUESTION 4
LinkedList fraction = new LinkedList();
for (int x = 1; x <= 3; x++){
	System.out.println("Enter fraction " + x + " NUMERATOR:");
	int numerator = input.nextInt();
	System.out.println("Enter fraction " + x + "
DENOMINATOR:");
	int denominator = input.nextInt();

	if (x == 1)
		fraction.insertFirst(numerator, denominator);
	else if (x == 2)
		fraction.insertSecond(numerator, denominator);
	else
		fraction.insertThird(numerator, denominator);
}
```

5) Write a main program that will continually prompt for the numerator and the denominator of a fraction until the user decides to stop. After each fraction,
   a. Add fraction to beginning of the list
   b. Print the list

```
//QUESTION 5
LinkedList fraction1 = new LinkedList();
for (int x = 1; x <= 3; x++){
	System.out.println("Enter fraction " + x + " NUMERATOR:");
	int numerator = input.nextInt();
	System.out.println("Enter fraction " + x + "
DENOMINATOR:");
	int denominator = input.nextInt();
	fraction1.addatFront(numerator, denominator);
	fraction1.printList();
}
```

6) Add the following functionality to the main program:
   a. Allow the user to specify any fraction to be printed out
   b. Allow the user to modify any specified fraction in the list.
      Prompt for the n^{th} fraction from the linked list to be changed
      i. Change numerator
      ii. Change denominator

```
//QUESTION 6
```

```
		int option = 0;
		do {
			System.out.println("\t\t *MENU*");
			System.out.println("1 - Specify Any Fraction To Be Printed
Out.");
			System.out.println("2 - Modify Any Specified Fraction In
The List.");
			System.out.println("3 - Exit.");
			option = input.nextInt();
			if (option == 1) {
				System.out.println("Enter the nth fraction from the
linked list to be printed out:");
				int location = input.nextInt();
				fraction.printFraction(location);
			}

			else if (option == 2) {
				System.out.println("Enter the nth fraction from the
linked list to be changed:");
				int location = input.nextInt();
				System.out.println("Enter the NUMERATOR of nth
fraction:");
				int numerator = input.nextInt();
				fraction.putNum(location, numerator);
				System.out.println("Enter the DENOMINATOR of nth
fraction:");
				int denominator = input.nextInt();
				fraction.putDen(location, denominator);
			}
			else if (option == 3)
				System.out.println("Have a good day !");
			else
				System.out.println("Invalid option, Please try again
!");
		}
		while (option != 3);
```