Lab 1

CSE 274

A programmer has developed a Java class LinkList that implements Linked List data structure. However, this LinkList class is buggy. We are asked to find the bugs and fix them.

I. A bug in the Constructor

Create a new project in Eclipse, import the `Application.java` file as a source file, and then run the Application. You will see several errors. One of the errors indicates that:

```
The constructor Link(int, double) is undefined
```

The compiler is expecting a constructor in the form of `Link(int, double)`. However, the class `Link` includes a constructor in the form of `Link()`. We need to edit the constructor of class `Link` so that it becomes in the specific form that the complier is expecting.

Question 1: Add proper arguments to the constructor `Link()`, so that this constructor becomes in the specific form that the compiler is expecting. Run the Application to make sure that the bug is fixed and you see no error regarding the constructor.

II. A bug in deleteFirst Method

Copy the following code to the body of `main` method in the `Application` class and then run the Application.

```
LinkList myLinkedlist = new LinkList();

myLinkedlist.insertFirst(34,17.8);
myLinkedlist.insertFirst(42,21.6);
myLinkedlist.insertFirst(39,81.3);
myLinkedlist.insertFirst(18,59.8);
myLinkedlist.insertFirst(27,56.5);
myLinkedlist.insertFirst(61,22.4);
myLinkedlist.insertFirst(55,93.7);

myLinkedlist.displayList();
```

Make sure that you see the following output:

```

{55,93.7}
{61,22.4}
{27,56.5}
{18,59.8}
{39,81.3}
{42,21.6}
{34,17.8}
```

We see that we could create a linked list, insert seven links, and print the stored data inside links of the linked list. Now, let's delete the first link. Add the following line of code at the end of main method:

```
myLinkedlist.deleteFirst();
System.out.println("After deletion of the first link:");
myLinkedlist.displayList();
```

Check the output and see that the first link was actually deleted. So far, the Java program has been working as expected.

Let's do another test. Erase everything from the body of `main` method, copy the following lines to the body of `main` method, and run the Application.

```
LinkList myLinkedlist = new LinkList();
myLinkedlist.deleteFirst();
myLinkedlist.displayList();
```

In above, we are creating an empty linked list, calling the deleteFirst() method, and then calling displayList() method. Since the linked list is empty we expect an empty output. But we see the following error:

```
java.lang.NullPointerException
```

It seems that `deleteFirst` is buggy and results in an error when the method is called for an empty linked list.

Question 2: Add new lines of code to `deleteFirst` method so that calling this method on an empty linked list does not result in an error. Perform a test to make sure that the bug is fixed.

III. A Bug in `Find` Method

Erase the body of the `main` method, copy the following lines of code to the body of `main` method, and run the application.

```
LinkList myLinkedlist = new LinkList();

myLinkedlist.insertFirst(34,17.8);
myLinkedlist.insertFirst(42,21.6);
myLinkedlist.insertFirst(39,81.3);
myLinkedlist.insertFirst(18,59.8);
myLinkedlist.insertFirst(27,56.5);
myLinkedlist.insertFirst(61,22.4);
myLinkedlist.insertFirst(55,93.7);

myLinkedlist.displayList();

int key;
boolean Result;

key=27;
Result=myLinkedlist.find(key)==null;
System.out.println(Result);
```

You should see the following output

```
{55,93.7}
{61,22.4}
{27,56.5}
{18,59.8}
{39,81.3}
{42,21.6}
{34,17.8}
false
```

Accordingly, the program is able to find a key in the linked list when the key actually exits in the key. Let's see what happens when we search for a key that does not exist in the linked list. To perform this test, replace `key=27` with `key=46` in the body of main method and run the Application again. You should see the following error:

```
java.lang.NullPointerException
```

It seems that `find` method is buggy.

Question 3: Add new lines of code to `find` method to fix the bug, and run a test to make sure that the bug is fixed.

IV. Searching over an Empty Linked List

Erase the body of the `main` method, copy the following lines of code to the body of `main` method, and run the application.

```
LinkList myLinkedlist = new LinkList();

int key;
boolean Result;
key=27;
Result=myLinkedlist.find(key)==null;
System.out.println(Result);
```

You should see the following error:

```
java.lang.NullPointerException
```

It seems that calling the `find` method for an empty linked list results in an error.

Question 4: Add new lines of code to `find` method to fix the bug, and run a test to make sure that the bug is fixed.

V. Developing `insertLast` Method

We want to develop a method `insertLast(int id, double dd)` that enables us to insert a new link at the ending part of the linked list. A draft for this method is already provided and we need to complete the draft. At the top part of the draft you see these lines of code:

```
if (first==null)
{
insertFirst(id, dd);
return;
}
```

In above we are checking if the linked list is empty. For an empty linked list the ending part is actually the beginning part. So for inserting a new link at the beginning part of an empty linked list we can simply call the `InsertFirst` method which is already developed and is fully functional.

We now focus on non-empty linked lists. If the linked list in not empty, then we should move to the last link so that we can add a new link at the ending part of the linked list. Moving to the last link can be accomplished by the following lines of code:

```
Link current=first;

while(current.next!=null)
{
current=current.next;
}
```

From the above, `current` is a reference to the last link. For inserting a new link at the ending part of the linked list following tasks should be completed:

- Creating a new link
- Making the `next` variable of the new link refer to `null`
- making the `next` variable of the current link refer to the new link

Question 5: Complete the above tasks by adding new lines of code to `insertLast` method.
To test that `insertLast` is working properly, erase the body of `main` method, copy the following lines of code to the body of `main`, run the application and see if a new link is being added at the ending part of the linked list:

```
LinkList myLinkedlist = new LinkList();

myLinkedlist.insertFirst(34,17.8);
myLinkedlist.insertFirst(42,21.6);
myLinkedlist.insertFirst(39,81.3);

myLinkedlist.displayList();

myLinkedlist.insertLast(66,59.1);

System.out.println("Linked list after new insertion:" );

myLinkedlist.displayList();
```

If `insertLast` is working properly, you should see the following output:

```
{39,81.3}
{42,21.6}
{34,17.8}
Linked list after new insertion:
{39,81.3}
{42,21.6}
{34,17.8}
{66,59.1}
```

VI. Developing CountandDisplay Method

Question 6: Develop the following method for the `LinkList` class:

```
public void CountandDisplay(int index)
```

The above method receives an `index` as an input, starts from the first link, move to the next ones, find the specific link that has the given `index`, and displays the data inside the link. It should be noted that the first link has an index of 1. If there is no link with the requested `index`, the method should not print any output and should return.
To test the `CountandDisplay` method, erase the body of `main` method, copy the following lines of code to the body of `main` method, and run the application:

```
LinkList myLinkedlist = new LinkList();

myLinkedlist.insertFirst(6,17.8);
myLinkedlist.insertFirst(5,21.6);
myLinkedlist.insertFirst(4,81.3);

System.out.println("The link with index 2:");
myLinkedlist.CountandDisplay(2);
System.out.println("The link with index 10:");
myLinkedlist.CountandDisplay(10);
```

You should see the following output:

```
The link with index 2:
{5,21.6}
The link with index 10:
```

VII. Submitting the Assignment

Once, you are done with all the above steps please erase the body of `main` method, copy the following lines of code to the body of `main` method, run the application one more time to make sure you see no error, and finally upload `Application.java` on canvas:

```
LinkList myLinkedlist = new LinkList();

System.out.println(myLinkedlist.find(4)==null);
myLinkedlist.deleteFirst();
myLinkedlist.insertLast(1, 29.2);
myLinkedlist.insertFirst(3, 13.5);
myLinkedlist.insertLast(2, 22.2);
myLinkedlist.deleteFirst();
myLinkedlist.displayList();
System.out.println(myLinkedlist.find(1)==null);
myLinkedlist.insertFirst(6,17.8);
myLinkedlist.insertFirst(5,21.6);
myLinkedlist.insertFirst(4,81.3);
myLinkedlist.displayList();
System.out.println("The link with index 4:");
myLinkedlist.CountandDisplay(4);
```

The above code should produce the following output:

```
true
{1,29.2}
{2,22.2}
false
{4,81.3}
{5,21.6}
{6,17.8}
{1,29.2}
{2,22.2}
The link with index 4:
{1,29.2}
```