**Diagram of LinkedList from java api**

LinkedList is a doubly linked list and it has references to first and last nodes

Empty LinkedList:

List<String> names = new LinkedList();

LinkedList

first last

null

null

LinkedList with one node

names.add(“Diana”);

LinkedList

first last

Node

data

next

prev

prev

Diana

null

null

LinkedList with many nodes

List<String> names = new LinkedList();

names.add(**"Diana"**); //add is same as addLastnames.add(**"Harry"**);  
names.add(**"Romeo"**);

LinkedList

first

last

null

Node

data

next

prev

Node

data

next

prev

Node

data

next

prev

Romeo

Harry

Diana

null

**Questions**

1. Draw a diagram for the updated list when the following code is executed:

names.addFirst(“Sam”);

A diagram of a data flow

Description automatically generated with medium confidence

Show the references that had to be updated.

The references that had to be updated are the LinkedList’s ‘first’ reference, and the ‘Diana’ Node’s ‘previous’ reference.

1. Draw a diagram for the updated list when the following code is executed:

names.add(“Sam”); //add is same as addLast – verify in api

A diagram of a data flow

Description automatically generated with medium confidence

Show the references that had to be updated.

The references that had to be updated are the LinkedList’s ‘last’ reference, and the ‘Romeo’ Node’s ‘next’ reference.

1. As we saw in the slides, a node can be inserted/deleted in the middle by using an iterator.

Suppose the following is executed on the LinkedList with 3 nodes in the diagram above:

ListIterator<String> iterator = staff.listIterator(); // |DHR

iterator.next(); // D|HR

iterator.next(); // DH|R

iterator.add("Juliet"); // DHJ|R

Draw a diagram for the updated list.

A diagram of data flow

Description automatically generated

Show the references that had to be updated to do the insert.

The references that had to be updated are the ‘Harry’ node’s ‘next’ reference, and the ‘Romeo’ node’s ‘previous reference.

1. Suppose now a node is deleted. The following code is executed after the code in Q3:

iterator.remove(); // DH|R

A screenshot of a computer

Description automatically generated

Executing iterator.remove() directly after iterator.add(‘string’) doesn’t seem to work as lastReturned=null at that moment. To get the result of DH|R I had to run the following code:

A screenshot of a computer

Description automatically generated

Draw a diagram for the updated list.

A diagram of a graph

Description automatically generated with medium confidence

Show the references that had to be updated to do the delete.

The references that had to change are the ‘Harry’ node’s ‘next’ reference, and the ‘Romeo’ node’s ‘previous’ reference.