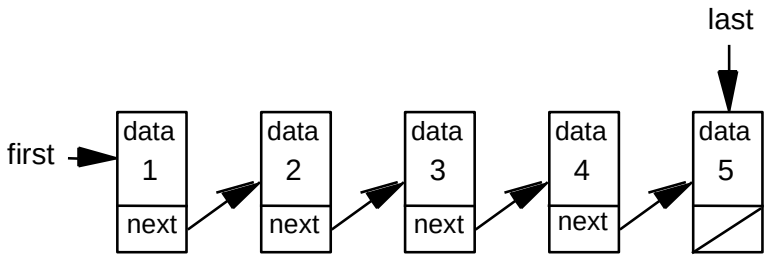
List1.java

Phase 1: Make sure you can complete this phase, with correct output as shown at the bottom of this phase, then complete Phase 2 before submission

Background:

1. The student outline contains a program that stored a linked list of five nodes, but in reverse order. The objective of this lab exercise is to create a linked list with the nodes assembled in order as they are generated. If a loop is used to create the values 1-5, the resulting list should look like this:



   The next value to be inserted would go at the right end of the list. To avoid traversing the entire list each time, two references will provide access at the front (first = left end) and the end (last = right end) of the list.

2. Your lab work can begin by getting ideas from the example program *ListDemo.java*, but your driver will be named *List1.java* and provide the actions described in the "Assignment" section below this one. *SinglyLinkedList.java* will be modified and *ListNode.java* will be used (with NO modification) as provided to support *SinglyLinkedList.java*

Make the following modifications to the *SinglyLinkedList.java* class:

3. An additional instance variable, `last`, should be added to the `SinglyLinkedList` class (found in *SinglyLinkedList.java*) to supply direct access to the last node in the list. The constructor and any other appropriate methods should also be modified to initialize the variable.

4. Create an `addLast` method to create and add a `ListNode` (found in *ListNode.java*) to the end of the list. The method will have to deal with several cases this time, an empty list case and a general case. A two case solution is required because the links are different for each case. The following code/pseudocode is provided:

```
void addLast(Integer value)
// Takes in value, creates a new node, adds the new node
// at the end of the list.
{
  if an empty list then
    set both first and last to reference the newly constructed node
  else
    construct a new node at the end of the list
}
```

5. Add a `getLast` method to return the last element of the list.

6. Add a `size` method that returns a count of the number of nodes in the list

7. Add a `printList` method which begins at the first `ListNode` and traverses the list, printing the values as each `ListNode` is "visited"

8. Make sure to adjust the code in the other routines as needed.

Assignment, phase 1:

1. Write/modify 2 classes: the modification of *SinglyLinkedList.java* (as described above) and *List1.java*, the driver (as described below). Make no modification to the *ListNode.java* file.

2. Using the guidelines in the Background section above, code a linked list that stores the 20 integers from 1-20 in ascending order.

3. After the list is created, call `getLast` display the contents of the last node to the screen.

4. Call `printList` to traverse the list and print out the 20 numbers in one line on the screen. Note that the implementation for this method should not depend on either `size` or `last`.

5. Use the `size` method to display the number of nodes in the list.

Instructions:

The run output will consist of the value of the first node in the list, the value of the last node in the list, 20 numbers and a count of how many nodes are in the list. An example run output is shown below:

```
Phase 1:
First Element: 1
Last Element: 20
SinglyLinkedList: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Number of Nodes: 20
```

Phase 2: After you have completed Phase 1, complete the following tasks

`SinglyLinkedList.java` – modify to add the methods described below

Modify this class so that two additional methods are part of it. Add external `ListNode` references as appropriate, keeping them as local as possible. This means that it is NOT appropriate to add extra instance variables when the external `ListNode` is only needed in a specific method.

`void insertInOrder(int num)`
inserts a `ListNode` such that the value of the data member `value` is the smallest at the beginning of the list and the largest at the end of the list (creates an ordered list, small to big). I suggest you handle 4 cases inside `insertInOrder`:

empty list
lowest value – place at beginning
the value is between 2 existing ListNodes
the largest element – place at end

`int remove(int num)`
removes the first instance of the integer passed as a parameter and returns its value without any other changes in the linked list. I suggest you handle the 3 cases inside remove:

empty list
an item not in the list at all
correct deletion/hook-ups of remaining `ListNodes`

`List1.java` – add to the driver to do the tasks described below. It is expected that you will keep the Phase 1 portion, then create a separate list for this phase.

Use your modified `SinglyLinkedList` class to create an ordered list using the following data input. You may use an input file, an array or prompt the user in order to fill your list. You can do the same when deleting values.

Tasks:
Make a second `SinglyLinkedList` object.
Create an ordered list with input given in this order: 3 5 15 1 3 9 -2 8
Print the list, starting at the head (beginning) of the list.
Delete the values 9, -2 and 6 (there isn't a 6!)
Print the list again from the head.

Submission:

Submit `List1.java` and your `SinglyLinkedList.java`. I will use the unmodified, posted code for `ListNode.java` to run your programs