

# Dynamic Data Structures—Linked List Editor

## Introduction

DDS—Linked List provides a `Node` data type, with data fields `payload` and `next`.

```
class Node{
    String payload;
    Node next;
}
```

You may create `Node` references and instantiate `Node` objects. They are graphically represented on the screen. Links from a `Node` reference to a `Node` object are represented as edges. The link attaches to a dot in the reference. Null references are represented by a reference dot with no attached link.

In this example, the `Node` reference `list` has a link to the `Node` object containing the payload 123. The `next` field in the `Node` object is null.

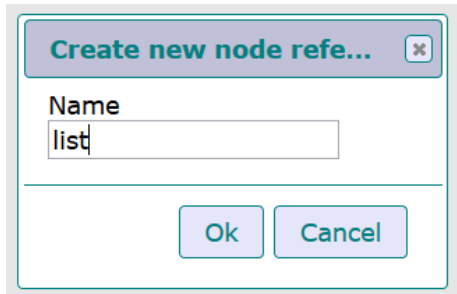


You can left click on a `Node` reference or a `Node` object to select it. They can be dragged with the mouse to reposition them. Selected references and objects have action buttons.

Action	Semantic meaning
Click <input type="button" value="Add Node Ref"/>	<code>Node list;</code>
Select a Node Reference, click <input type="button" value="= new( )"/>	<code>list = new Node(String);</code>
Select a Node Reference, click <input type="button" value="= .next"/>	<code>list = list.next;</code>
Select a Node Reference, click <input type="button" value="= null"/>	<code>list = null;</code>
Select a Node object, click <input type="button" value="= new( )"/>	<code>node.next = new Node();</code>
Select a Node object, click <input type="button" value="= .next"/>	<code>node.next = list.next.next;</code>
Select a Node object, click <input type="button" value="= null"/>	<code>node.next = null;</code>
Mouse down on a reference source dot, drag the reference, release on a destination reference dot	<code>destination = source;</code>

## Tutorial

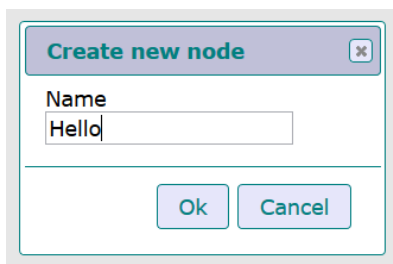
Load or refresh DDS. Click on the **Add Node Ref** button. Enter “list” in the dialog.



The Node reference named `list` will be created and highlighted.



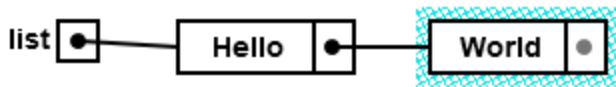
Click **= new()** to create a new Node object referenced by `list`. This is the semantic action for `list = new Node("Hello"); ...` Enter the value “Hello” in the dialog box.



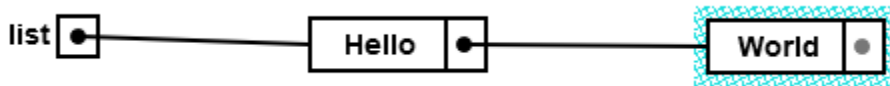
A new Node object is created.



Click **= new()** again to create another Node object that will be referenced by `list.next`. Enter the value “World” in the dialog box. A new Node object is appended to your list.



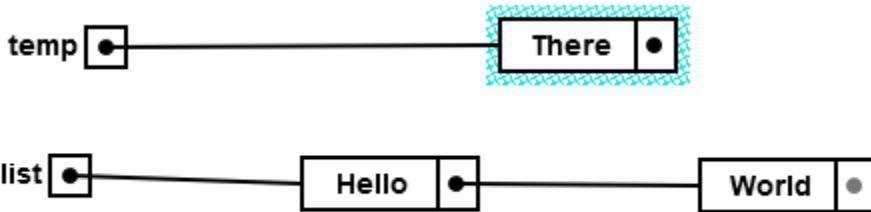
Drag the `Node` objects to space them out.



Now we want to create a new `Node` object that we can link into the list. Here is the code for the desired actions.

```
Node temp;
temp = new Node("There");
```

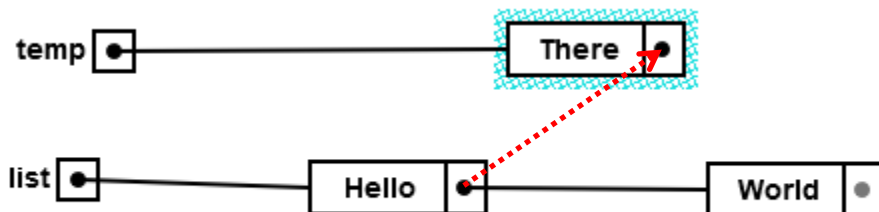
Click `Add Node Ref` to create the `Node` reference, then click `New` to instantiate the new `Node`. You will need to drag things around to make it look nice.



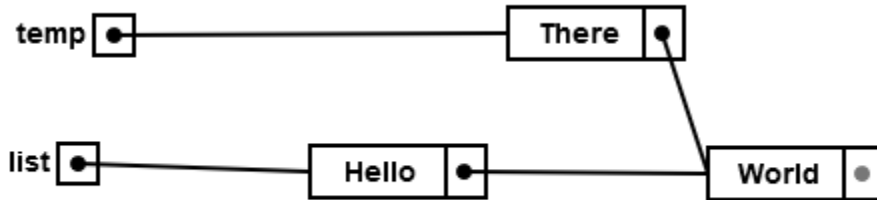
Now we want to do two assignments to link the "There" `Node` object into the list.

```
temp.next = list.next;
list.next = temp;
```

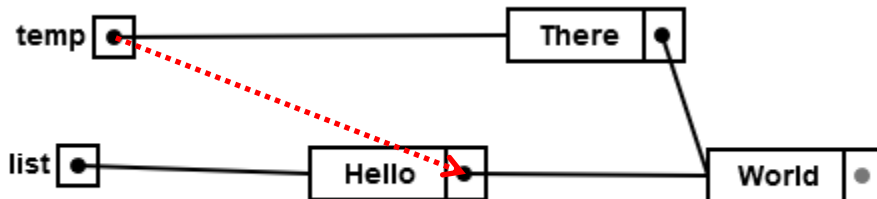
Assignment is done by dragging the source dot of a link. If you carefully mouse down on the dot in the "Hello" `Node` object, you can drag the value of that link and release it on the reference dot in the "There" `Node` object.



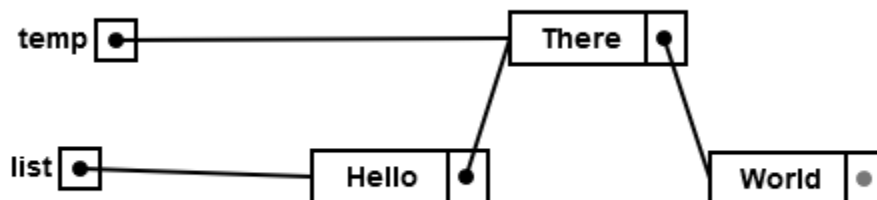
A copy of the link will drag. Should the actual Node object drag, you did not mouse down on the reference dot. The result of the first assignment should look like this.



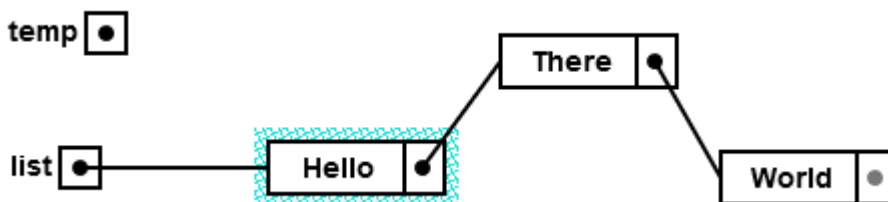
Now do the same thing to drag the value of the temp reference to the next Node reference in the "There" Node object.



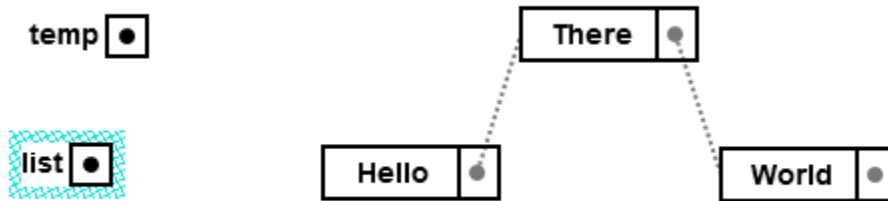
This should be the result. Note the next reference in the "Hello" Node object has changed.



Finally, click to select the temp reference, then click `= null`.



For your final task, click on the `list Node` reference to select it, then click `= null`.



Congratulations. You have created garbage. But fear not, all is not lost. On the right side of the menu bar there is a garbage button. Be considerate and give it a click to clean up after yourself.



## Other Considerations

The `= next` button is typically used with a `Node` reference to walk a list. Repeated applications represent this loop.

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

The editor as configured does not allow actions that would require an equivalent code expression involving more than two `next` references from a `Node` reference, such as `list.next.next`. You will see these links as gray dotted lines. Selecting a `Node` object that would require such a path to access it will not display any action buttons.

## Limitations and Planned Enhancements

This is a prototype. There is no undo facility. It will be added. Work needs to be done on layout issues of new `Node` references and objects. To facilitate its usefulness for student usage, a block language has been prototyped to execute code on linked lists, and a record and play animation feature is being prototyped.

## DDS Information.

DDS—Linked List was designed and implemented by Dr. Robert A. Ravenscroft, Jr. at Rhode Island College in Providence, RI. You are free to use the program for academic purposes. The source code is currently not available.

*Copyright 2017, Dr. Robert A. Ravenscroft, Jr.*