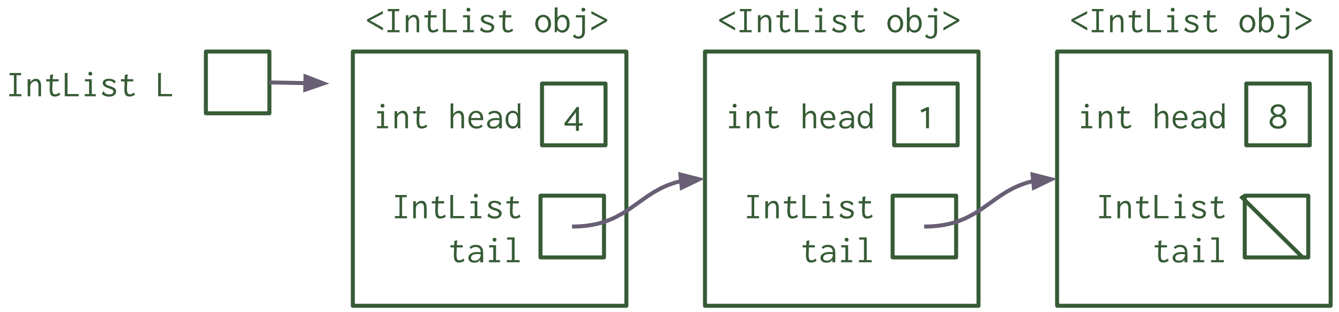
IntDList

Lab 3

Review: Linked Lists

A linked list is a data structure that consists of individual links that each have two fields: *head* which holds a value and *tail* which stores a pointer to the next link. Each link is an object, e.g. an IntList object.

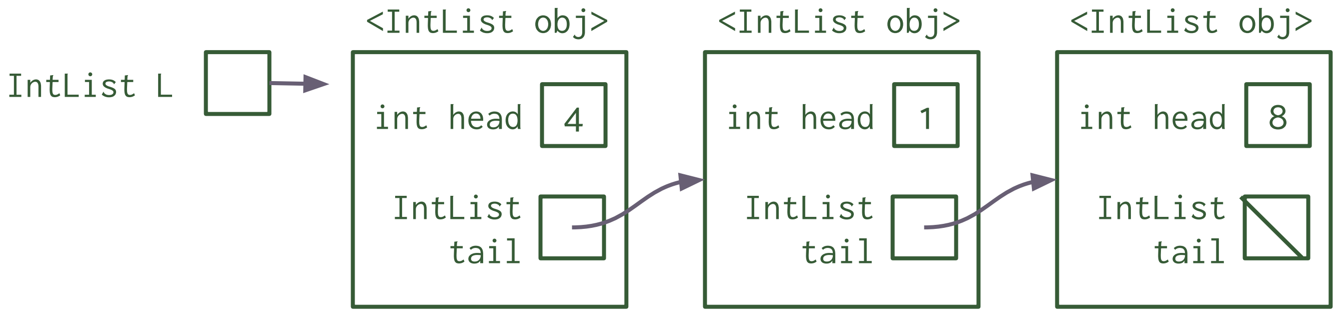
```
IntList L = IntList.list(4, 1, 8);
```

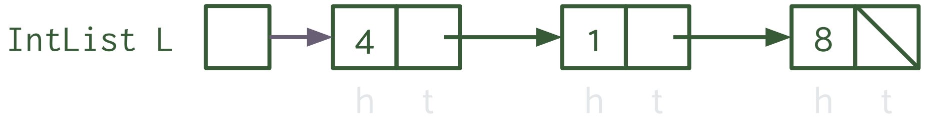


Review: Linked Lists

A linked list is a data structure that consists of individual links that each have two fields: *head* which holds a value and *tail* which stores a pointer to the next link. Each link is an object, e.g. an IntList object.

```
IntList L = IntList.list(4, 1, 8);
```





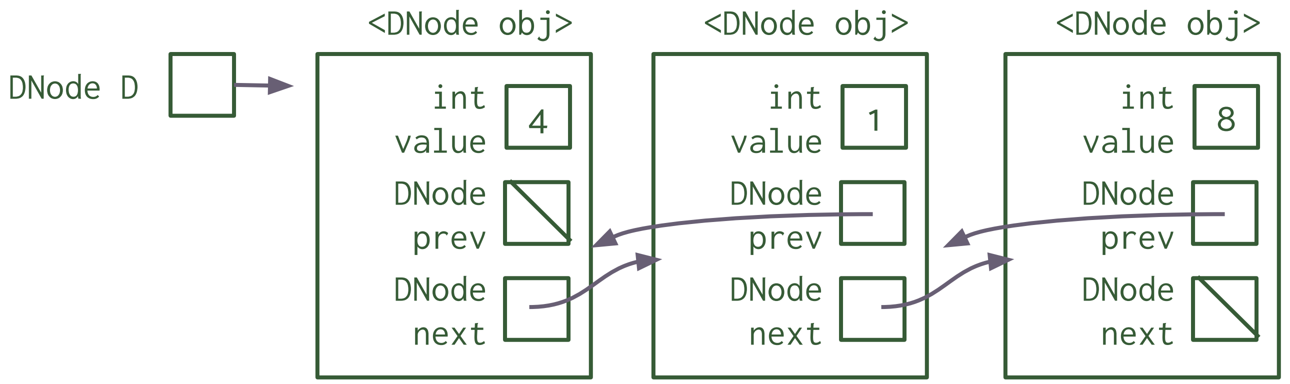
shorthand version of drawing an IntList

Doubly Linked Lists

Doubly Linked Lists

A doubly linked list is a data structure that consists of individual links that each have three fields: *value* which holds a value, *prev* which stores a pointer to the previous link, and *next* which stores a pointer to the next link. Each link is an object, e.g. an DNode object.

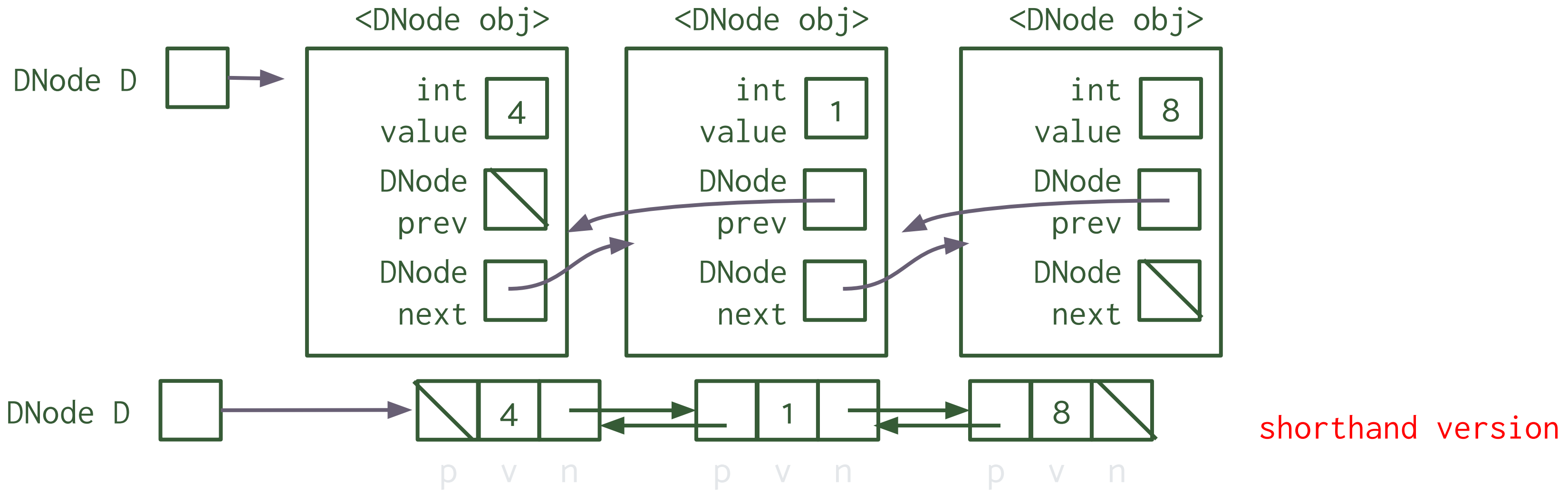
```
// THIS METHOD DOES NOT ACTUALLY EXIST - we will see more later
DNode D = DNode.list(4, 1, 8);
```



Doubly Linked Lists

A doubly linked list is a data structure that consists of individual links that each have three fields: *value* which holds a value, *prev* which stores a pointer to the previous link, and *next* which stores a pointer to the next link. Each link is an object, e.g. an DNode object.

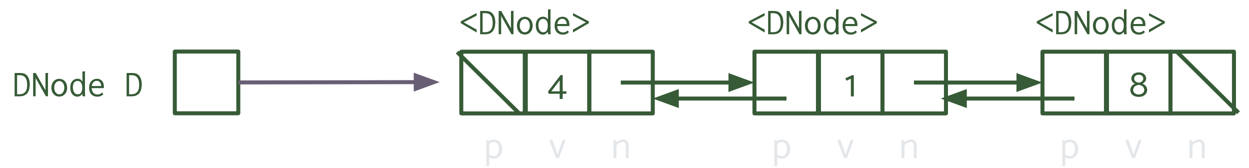
```
// THIS METHOD DOES NOT ACTUALLY EXIST - we will see more later
DNode D = DNode.list(4, 1, 8);
```



Doubly Linked Lists

A doubly linked list is a data structure that consists of individual links that each have three fields: *value* which holds a value, *prev* which stores a pointer to the previous link, and *next* which stores a pointer to the next link. Each link is an object, e.g. an DNode object.

```
// THIS METHOD DOES NOT ACTUALLY EXIST - we will see more later
DNode D = DNode.list(4, 1, 8);
```

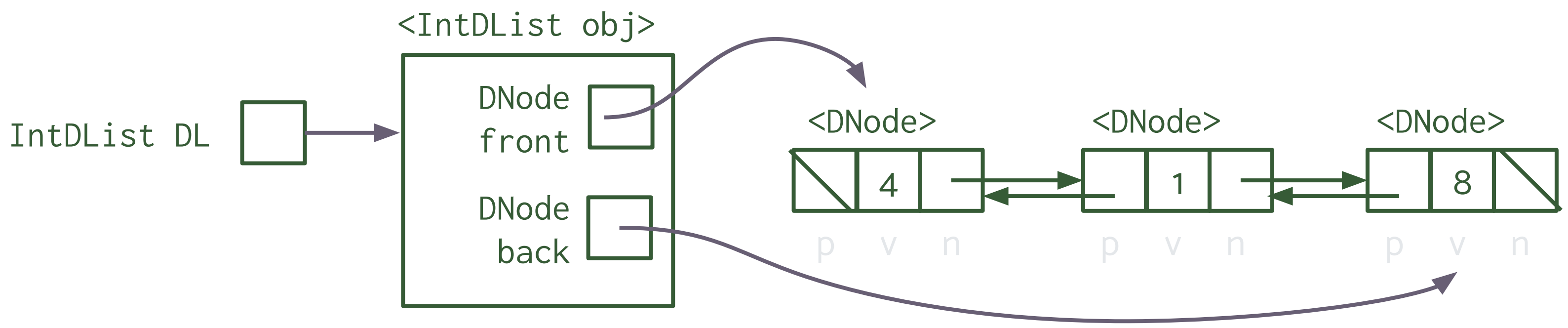


We will be using this shorthand version of DNode for the rest of the slides.

Doubly Linked Lists: IntDList

The IntDList class is another class that *wraps* around a doubly linked list of DNodes. An IntDList object has two fields: *front* which stores a pointer to the front DNode and *back* which stores a pointer to the back DNode.

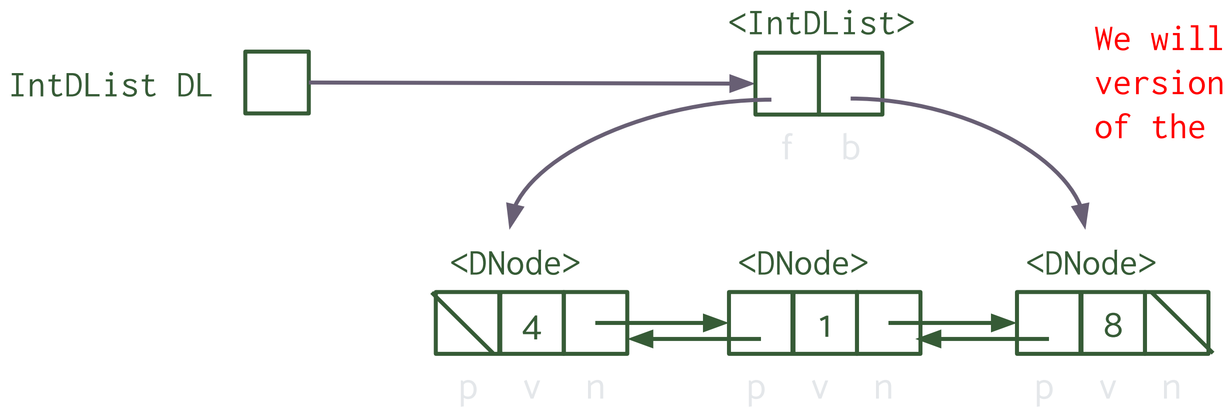
```
IntDList DL = new IntDList(4, 1, 8);
```



Doubly Linked Lists: IntDList

The IntDList class is another class that *wraps* around a doubly linked list of DNodes. An IntDList object has two fields: *front* which stores a pointer to the front DNode and *back* which stores a pointer to the back DNode.

```
IntDList DL = new IntDList(4, 1, 8);
```



We will be using this shorthand version of IntDList for the rest of the slides.

IntDList: insertFront

IntDList: insertFront

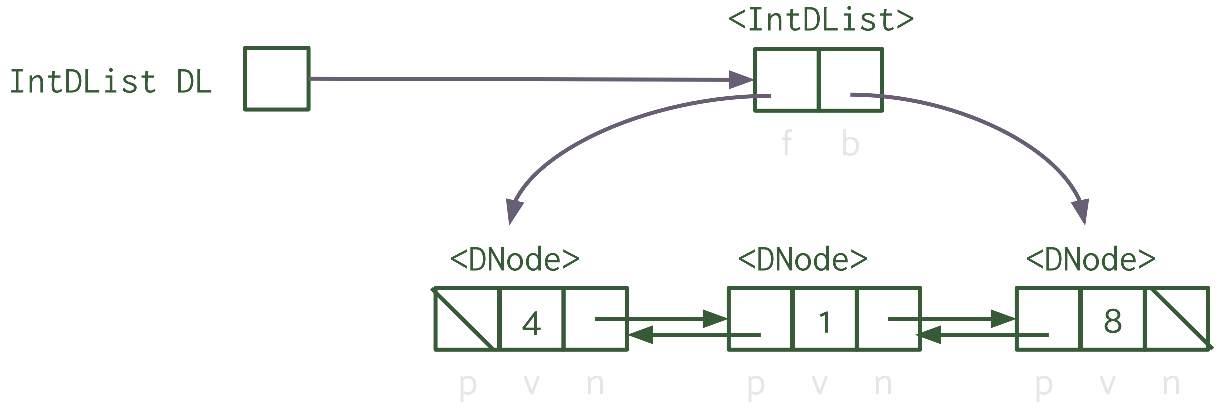
Insert a value to the beginning of your IntDList.

```
IntDList DL = new IntDList(4, 1, 8);
DL.insertFront(7); // how do we do this?
```

IntDList: insertFront

Insert a value to the beginning of your IntDList.

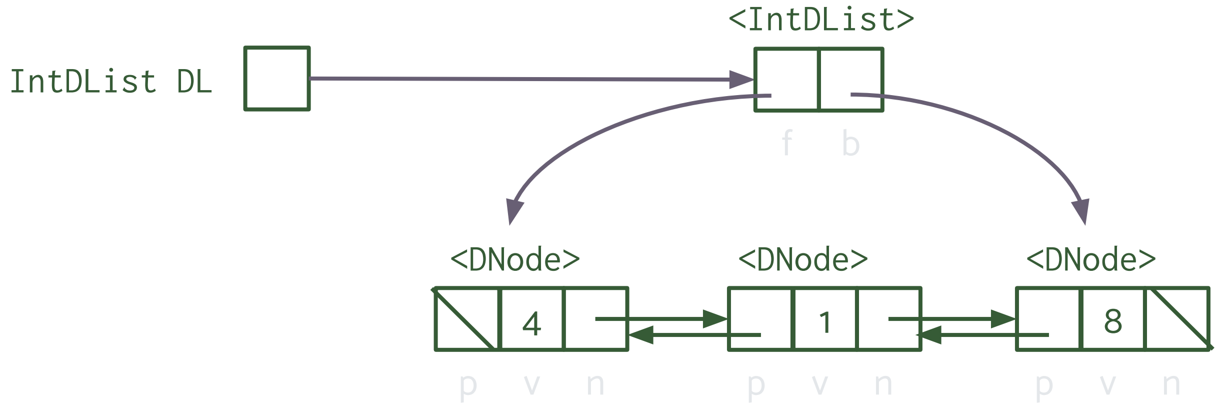
```
IntDList DL = new IntDList(4, 1, 8);
DL.insertFront(7);
```



IntDList: insertFront

Insert a value to the beginning of your IntDList.

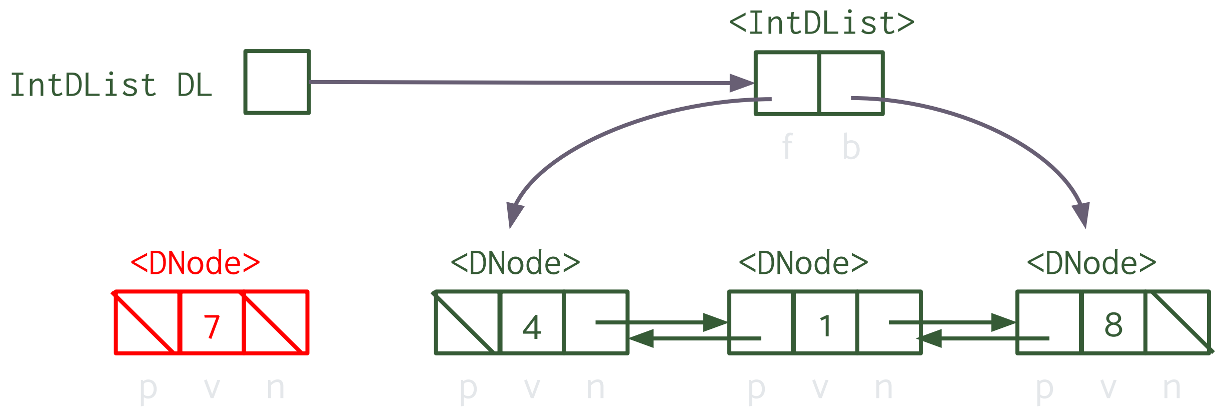
```
IntDList DL = new IntDList(4, 1, 8);
DL.insertFront(7);
```



IntDList: insertFront

Insert a value to the beginning of your IntDList.

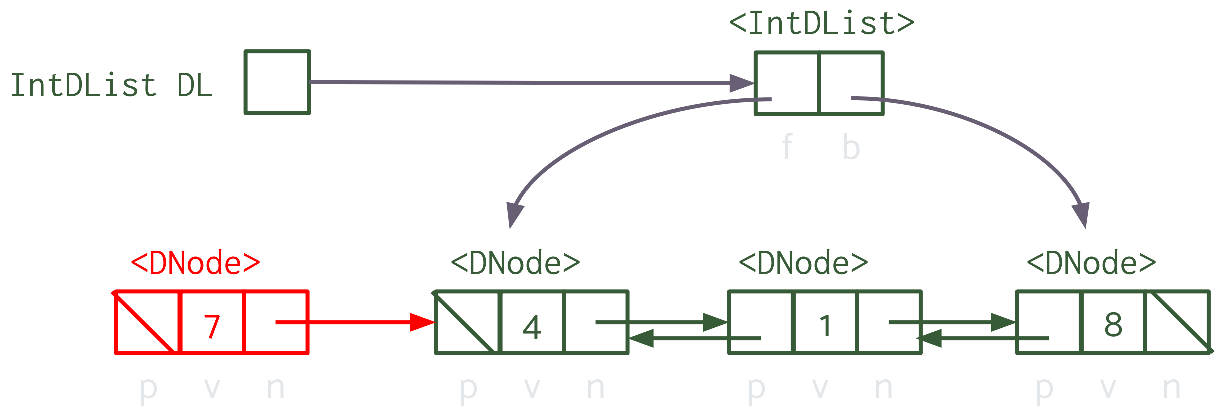
```
IntDList DL = new IntDList(4, 1, 8);
DL.insertFront(7);
```



IntDList: insertFront

Insert a value to the beginning of your IntDList.

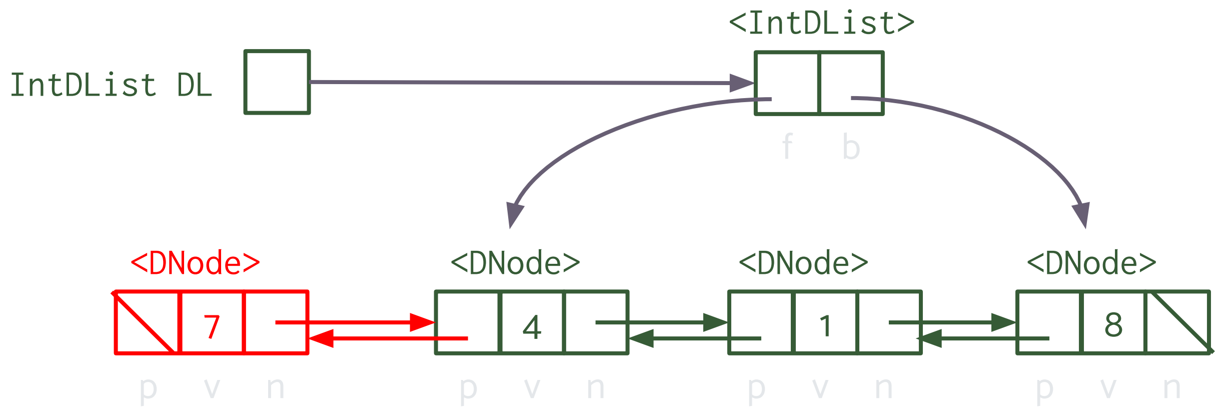
```
IntDList DL = new IntDList(4, 1, 8);
DL.insertFront(7);
```



IntDList: insertFront

Insert a value to the beginning of your IntDList.

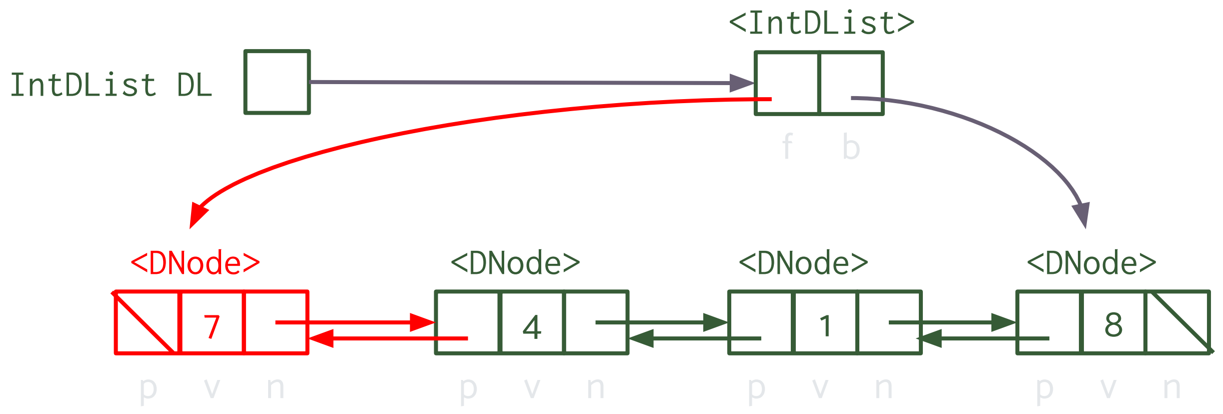
```
IntDList DL = new IntDList(4, 1, 8);
DL.insertFront(7);
```



IntDList: insertFront

Insert a value to the beginning of your IntDList.

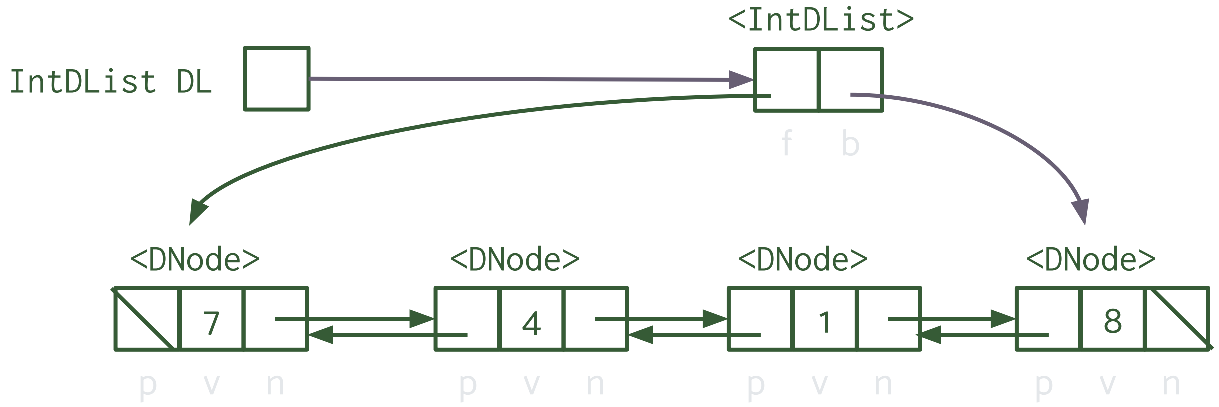
```
IntDList DL = new IntDList(4, 1, 8);
DL.insertFront(7);
```



IntDList: insertFront

Insert a value to the beginning of your IntDList.

```
IntDList DL = new IntDList(4, 1, 8);
DL.insertFront(7);
```



Edge case: what if this is the first element in the IntDList?

IntDList: deleteFront

IntDList: deleteFront

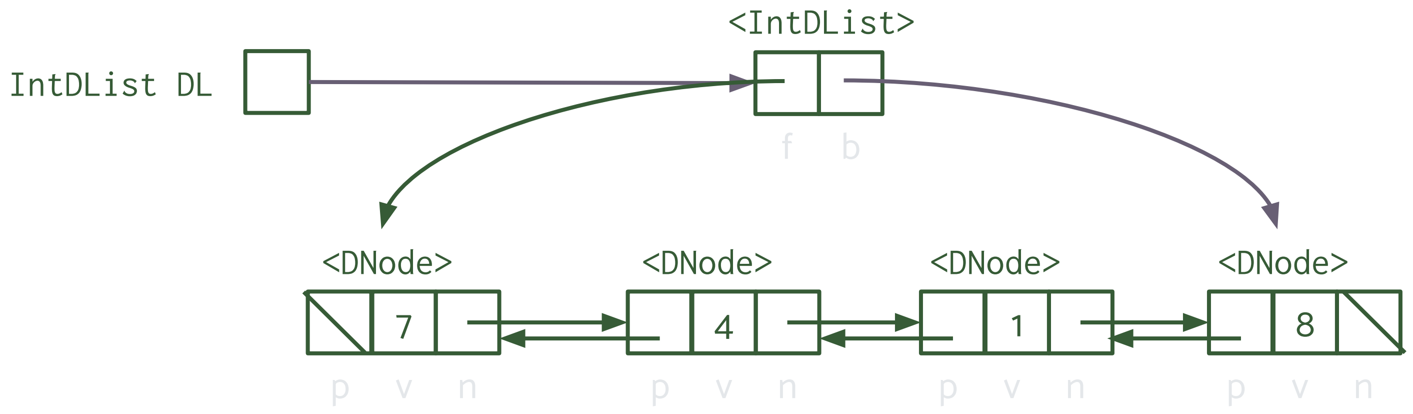
Delete a value to the beginning of your IntDList.

```
IntDList DL = new IntDList(7, 4, 1, 8);
int x = DL.deleteFront(); // how do we do this?
```

IntDList: deleteFront

Delete a value to the beginning of your IntDList.

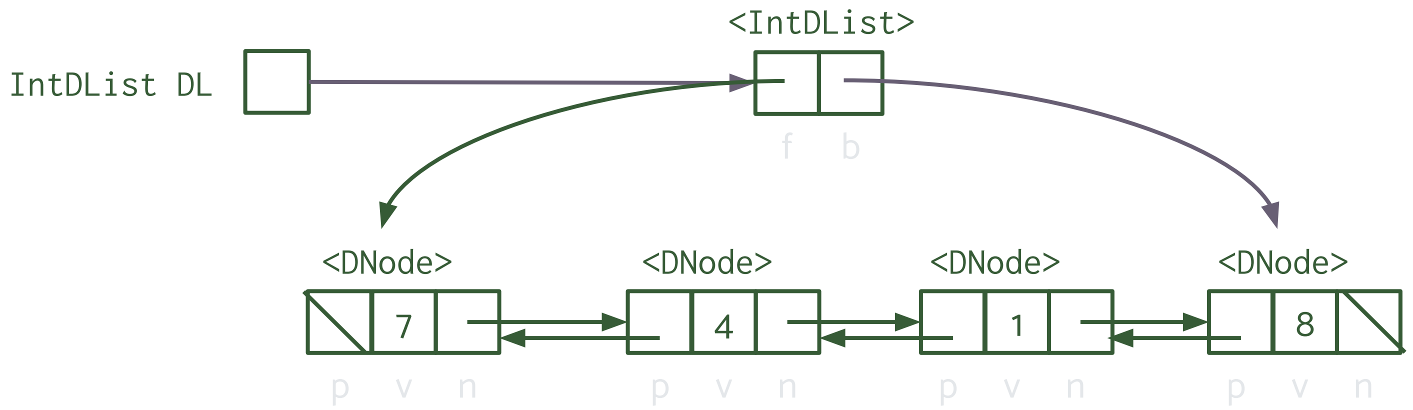
```
IntDList DL = new IntDList(7, 4, 1, 8);
int x = DL.deleteFront();
```



IntDList: deleteFront

Delete a value to the beginning of your IntDList.

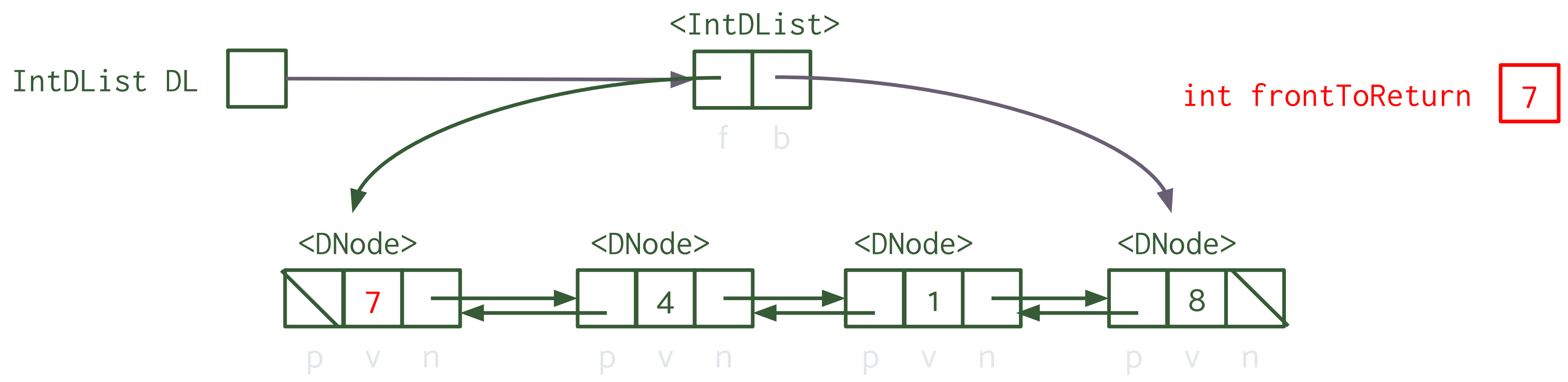
```
IntDList DL = new IntDList(7, 4, 1, 8);
int x = DL.deleteFront();
```



IntDList: deleteFront

Delete a value to the beginning of your IntDList.

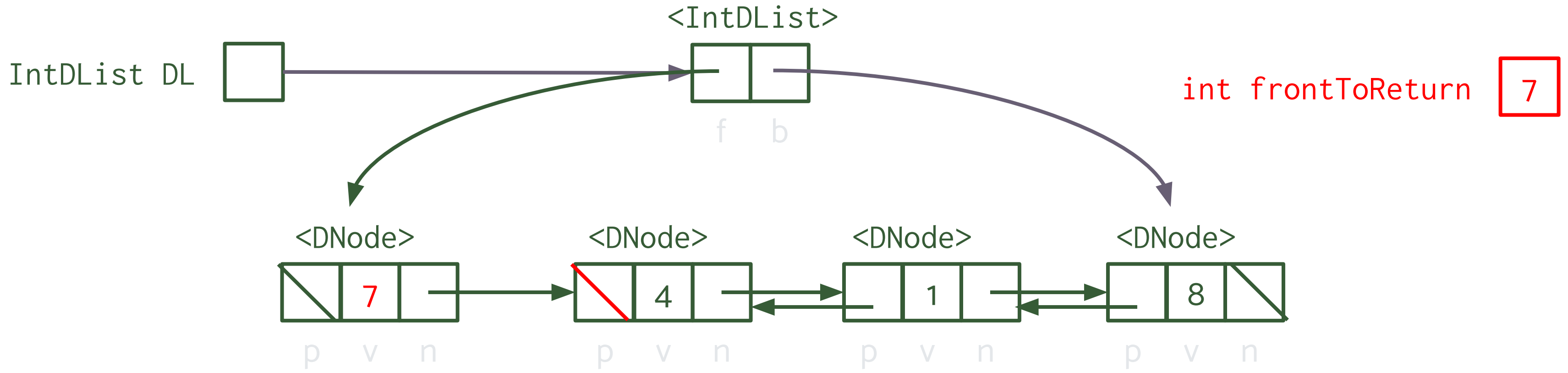
```
IntDList DL = new IntDList(7, 4, 1, 8);
int x = DL.deleteFront();
```



IntDList: deleteFront

Delete a value to the beginning of your IntDList.

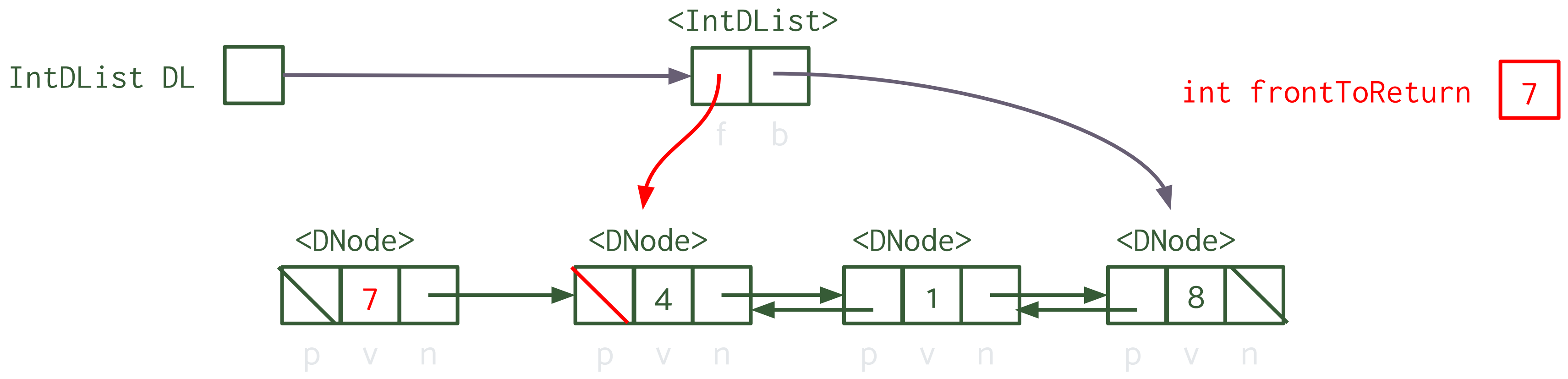
```
IntDList DL = new IntDList(7, 4, 1, 8);
int x = DL.deleteFront();
```



IntDList: deleteFront

Delete a value to the beginning of your IntDList.

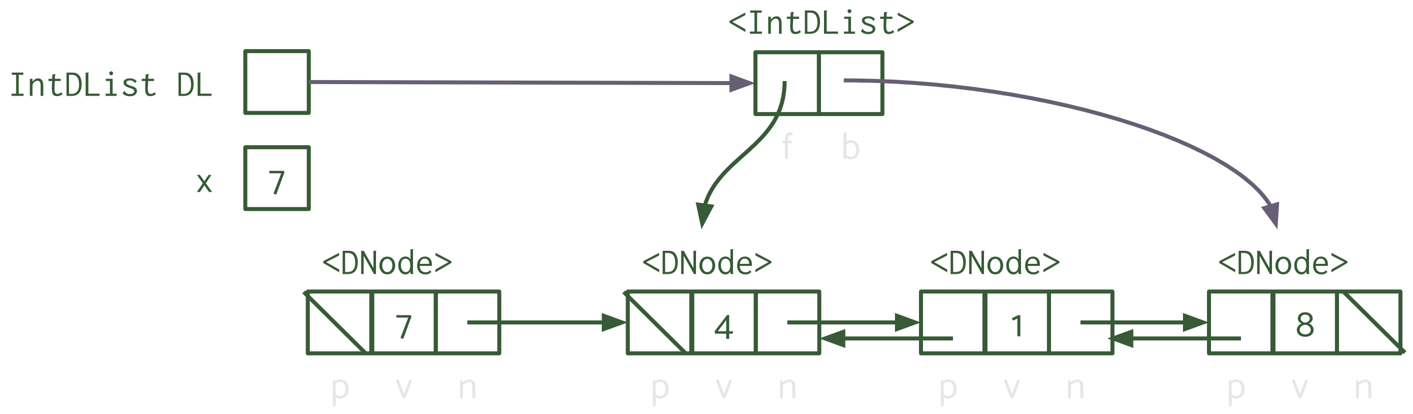
```
IntDList DL = new IntDList(7, 4, 1, 8);
int x = DL.deleteFront();
```



IntDList: deleteFront

Delete a value to the beginning of your IntDList.

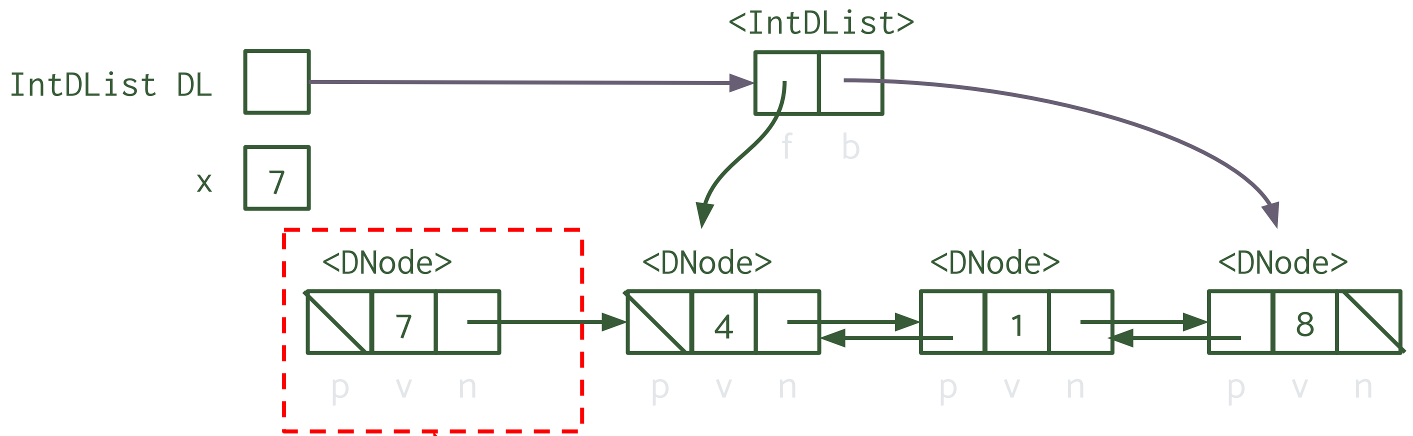
```
IntDList DL = new IntDList(7, 4, 1, 8);
int x = DL.deleteFront();
```



IntDList: deleteFront

Delete a value to the beginning of your IntDList.

```
IntDList DL = new IntDList(7, 4, 1, 8);
int x = DL.deleteFront();
```

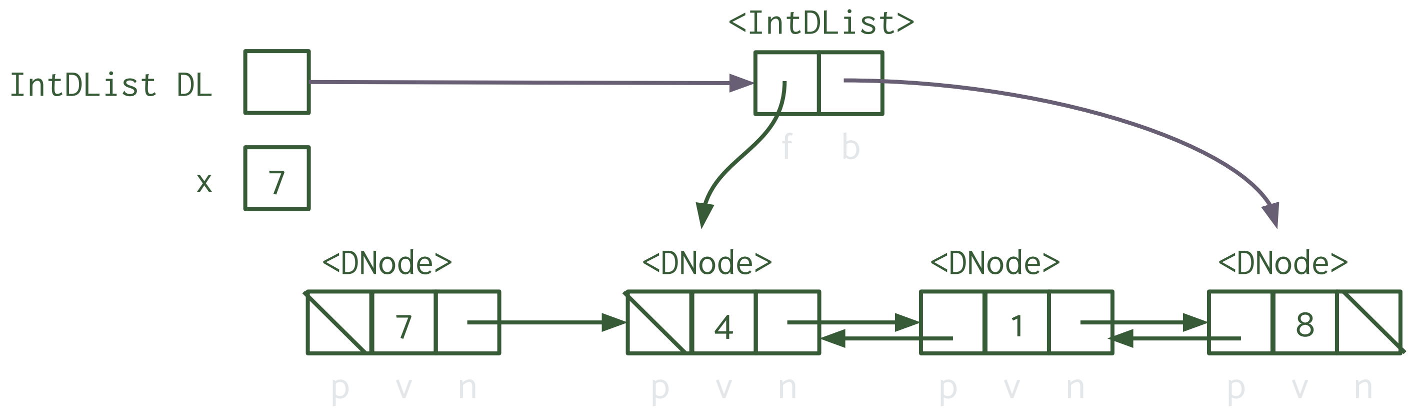


Don't need to worry about this! It is no longer referenced so it will be automatically garbage collected.

IntDList: deleteFront

Delete a value to the beginning of your IntDList.

```
IntDList DL = new IntDList(7, 4, 1, 8);
int x = DL.deleteFront();
```



Edge case: what if we are removing the only element of the IntDList?