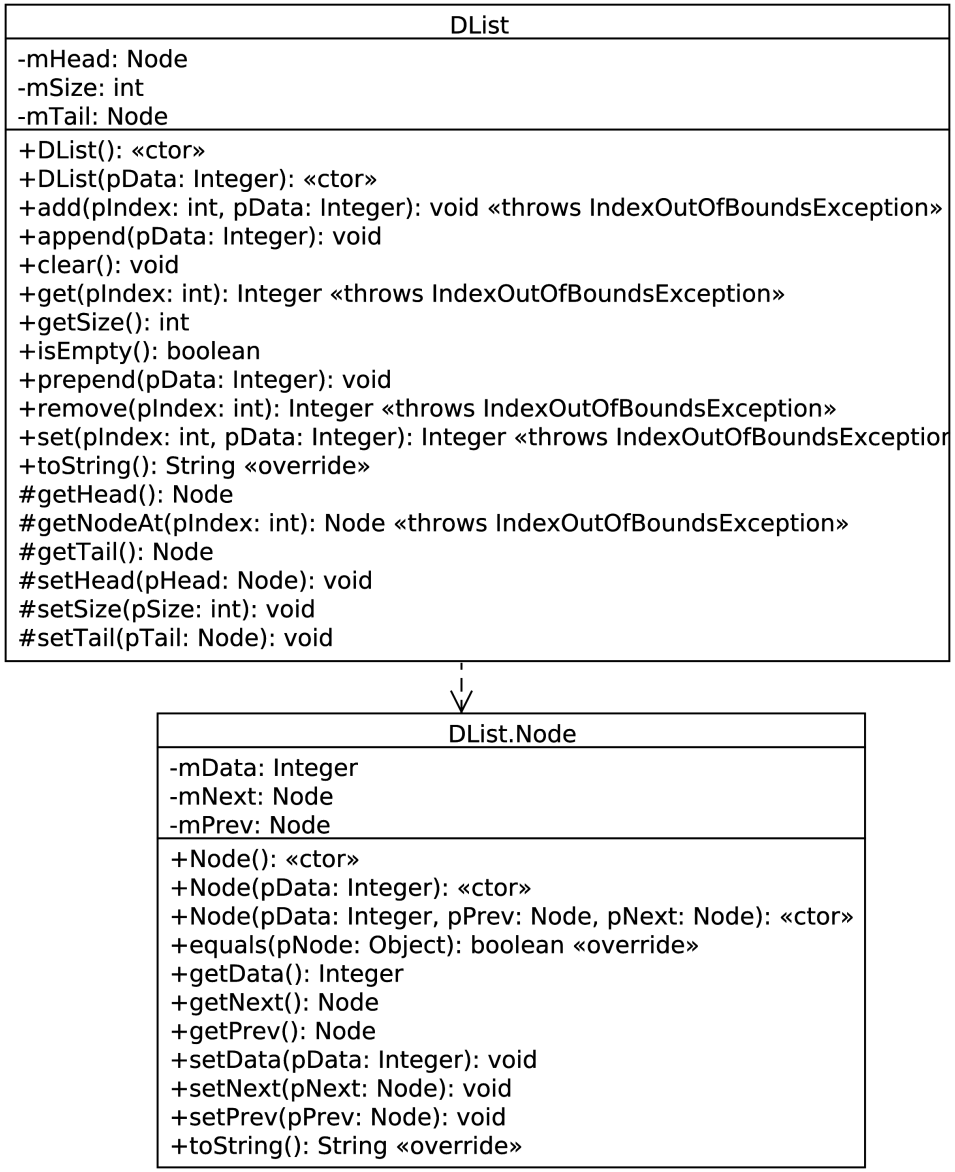
4. Linked Lists :: Implementation :: DList Class :: UML Class Diagram

Here is the *DList* class UML class diagram:



4. Linked Lists :: Implementation :: DList Class :: Accessor/Mutator Methods

Note that the *DList* class depends on the *DList.Node* class. The *DList* class contains three instance variables:

1. *mHead* is a reference to the first node (called the head) of the list.
2. *mTail* is a reference to the last node (called the tail) of the list.
3. *mSize* is an integer which stores the size of the list. An empty list has size 0.

The implementation of the following accessor/mutator methods is straightforward, so we will not discuss their implementation:

```
+getSize(): int
+isEmpty(): boolean
#getHead(): Node
#getTail(): Node
#setHead(pHead: Node): void
#setSize(pSize: int): void
#setTail(pTail: Node): void
```

isEmpty() is not strictly necessary since it simply returns **getSize() == 0** but it is a convenient method to have.

Note that the accessor/mutator methods that manipulate the *mHead* and *mTail* instance variables are protected so they cannot be called on a *DList* object from other classes of the application. We made them protected rather than private so subclasses of *DList* may call them. Also, *setSize*() is protected because it would be inappropriate for other classes to change the *mSize* instance variable of a *DList*, and again, it is protected so subclasses of *DList* may call it.

4. Linked Lists :: Implementation :: DList Class :: Constructors

The *DList* class implements two constructors:

```
// Creates an empty DList. For an empty DList, mHead = null, mTail = null, and
// mSize = 0.
public DList() {
  setHead(null);
  setTail(null);
  setSize(0);
}

// Creates a new DList with one element containing pData.
public DList(Integer pData) {
  // Create a new Node storing pData. Make the mPrev and mNext references null.
  Node newNode = new Node(pData);

  // Make the mHead reference refer to the new node.
  setHead(newNode);

  // Make the mTail reference refer to the new node.
  setTail(newNode);

  // The size of the list is now 1.
  setSize(1);
}
```