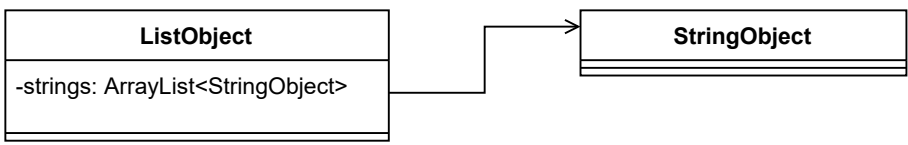**A List data structure is implemented with a String array which can contain a series of String objects.**

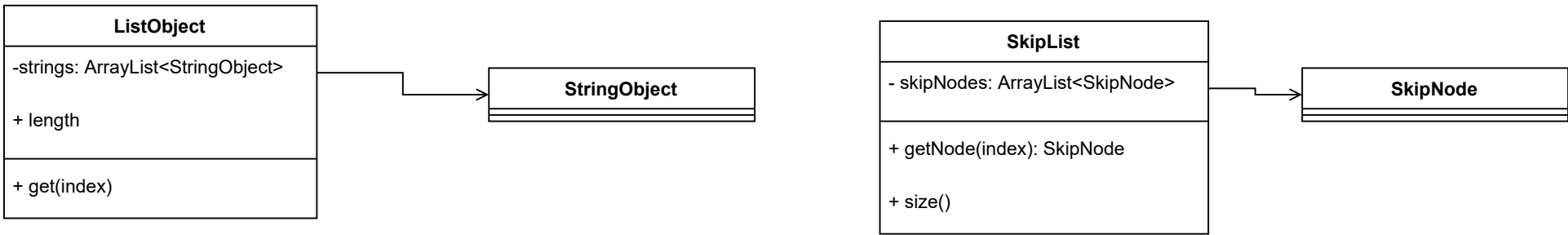| ListObject |
|---|
| -strings: ArrayList<StringObject> |

→ | StringObject |

We can access List by calling the get() method with an index, and know how many Strings inside the List with a public attribute: length.
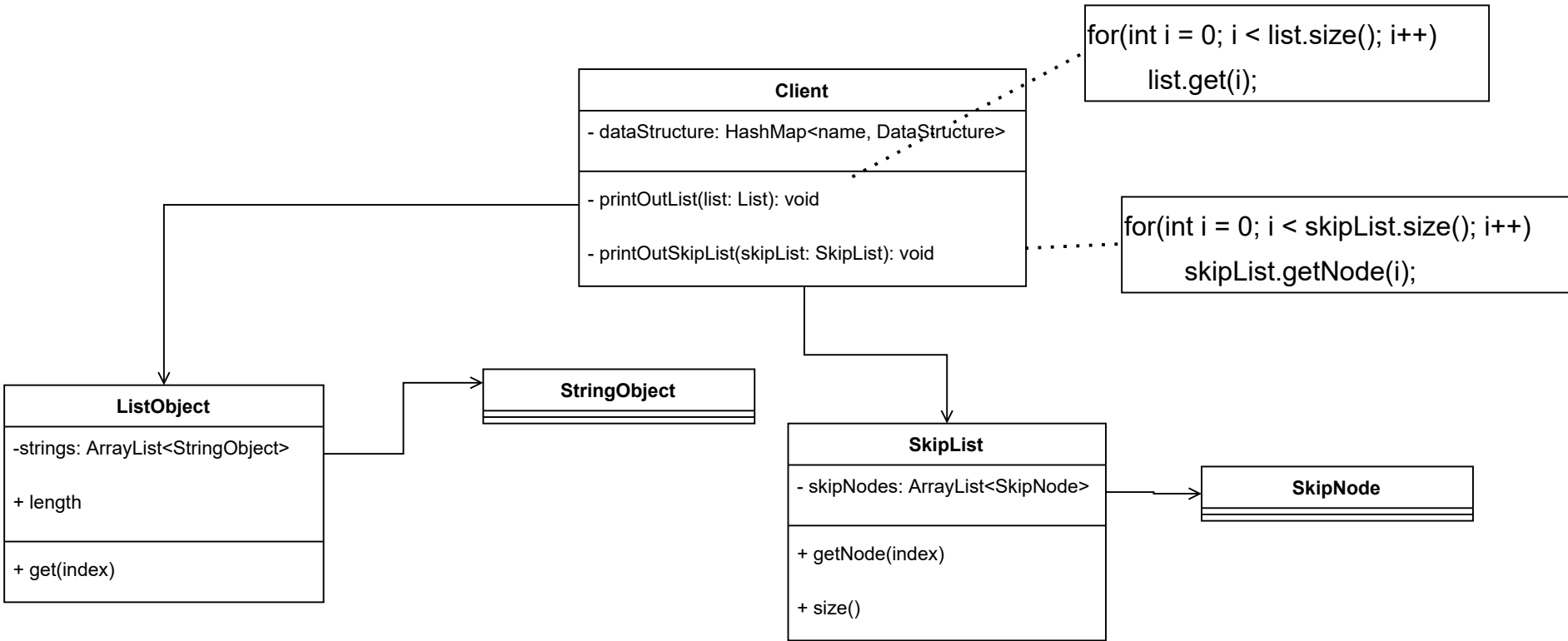
ListObject
-strings: ArrayList<StringObject>
+ length
+ get(index): String

→ | StringObject |
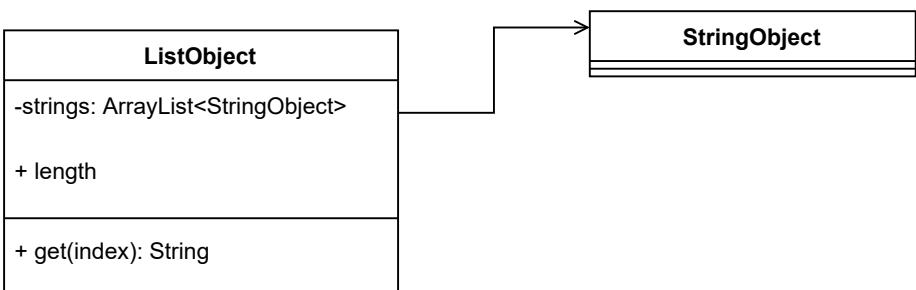
Furthermore, another data structure called SkipList which consists of a series of SkipNodes.

ListObject
-strings: ArrayList<StringObject>
+ length
+ get(index)

→ | StringObject |

SkipList
- skipNodes: ArrayList<SkipNode>
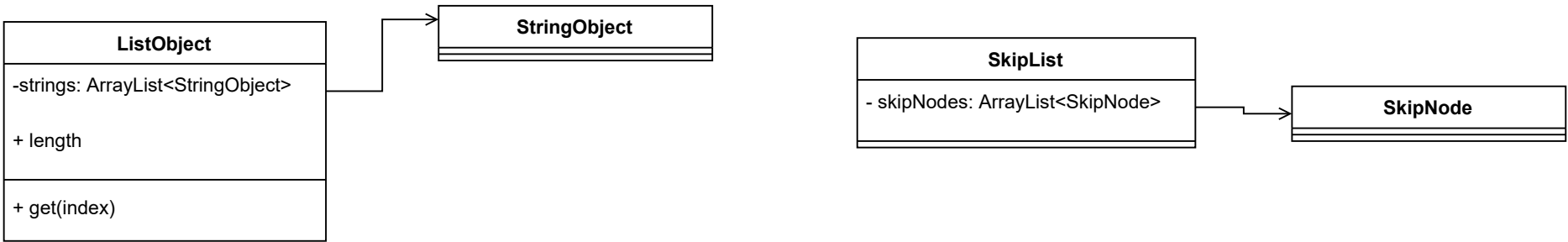
→ | SkipNode |

Each SkipNode can be accessed by invoking the getNode() method in SkipList with an index. And we have the idea about the size of SkipList with its size() method.

ListObject
-strings: ArrayList<StringObject>
+ length
+ get(index)

→ | StringObject |

SkipList
- skipNodes: ArrayList<SkipNode>
+ getNode(index): SkipNode
+ size()

→ | SkipNode |

Now we have to traverse both List and SkipList to print out those object items in the two different data structures for some purpose.

Client
- dataStructure: HashMap<name, DataStructure>
- printOutList(list: List): void
- printOutSkipList(skipList: SkipList): void

```
for(int i = 0; i < list.size(); i++)
    list.get(i);
```

```
for(int i = 0; i < skipList.size(); i++)
    skipList.getNode(i);
```

ListObject
-strings: ArrayList<StringObject>
+ length
+ get(index)

→ | StringObject |

SkipList
- skipNodes: ArrayList<SkipNode>
+ getNode(index)
+ size()

→ | SkipNode |

Refactor

Client
- dataStructure: HashMap<name, DataStructure>
- printOutList(dataStructure: DataStructure): void

```
private void PrintOutList(String inStructureName){
    DataStructure curStructure = dataStructure.get(inStructureName);
    curStructure.printOutList();
}
```

<<Interface>> Iterator
+ hasNext(): boolean
+ next(): Content

<<Interface>> DataStructure
+ get(): Element
+ createIterator(): Iterator
+ addContent()

SkipListIterator
- list: SkipList
+ hasNext()
+ next()

ListIterator
- list: List
+ hasNext()
+ next()

ListObject
- strings: ArrayList<StringObject>
+ length
+ get(index)
+ createIterator(): ListIterator
+ addContent(String content)
+ printOutList(dataStructure: DataStructure): void

SkipList
- skipNodes: ArrayList<SkipNode>
+ get(index)
+ size()
+ createIterator(): SkipListIterator
+ addContent()
+ printOutList(dataStructure: DataStructure): void

<<create>>

<<create>>

<<Interface>> Content

StringObject

SkipNode