Data Structures in Java

Agenda

1. Implementation of LinkedList in Java
2. Implementation of ArrayList in Java
3. Implementation of HashMap in Java

Objectives

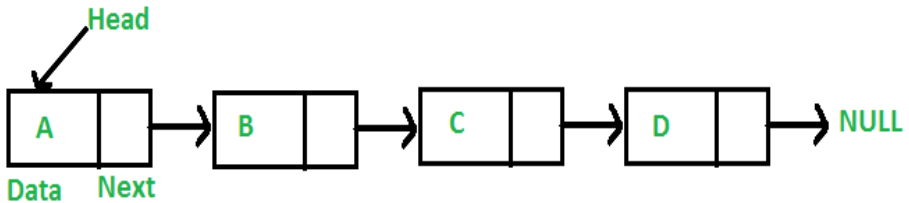
At the end of this session, you will be able to:

- Understand How to implement LinkedList, ArrayList and HashMap data structures in Java

LinkedList

LinkedList

- A linked list is a linear data structure, in which the elements are not stored in contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



- In simple words, a linked list consists of nodes where each node contains a data field and a reference (link) to the next node in the list.

LinkedList implementation in Java

LinkedList implementation in Java

- To do list:
  1. Define a Node class
  2. Define a LinkedList class
  3. Add the following functionality to a LinkedList class
     1. Is empty check and get size
     2. Insert a node at beginning
     3. Insert a node at end
     4. Insert a node at a given position
     5. Delete a node from a given position
     6. Display nodes
  4. Define a LinkedListTest class to check all the functionalities of LinkedList

1. Define a Node class

```
class Node
{
    protected int data;
    protected Node link;

    /*  Constructor for an empty node */
    public Node()
    {
        link = null;
        data = 0;
    }
    /*  Constructor with data */
    public Node(int d,Node n)
    {
        data = d;
        link = n;
    }
```

1. Define a Node class – getters & setters

```
    /*  Function to set link to next Node  */
    public void setLink(Node n)
    {
        link = n;
    }
    /*  Function to set data to current Node  */
    public void setData(int d)
    {
        data = d;
    }
    /*  Function to get link to next node  */
    public Node getLink()
    {
        return link;
    }
    /*  Function to get data from current Node  */
    public int getData()
    {
        return data;
    }
}
```

2. LinkedList class

```
class LinkedList
{
    protected Node start;
    protected Node end ;
    public int size ;

    /*  Constructor for empty LinkedList */
    public LinkedList()
    {
        start = null;
        end = null;
        size = 0;
    }
}
```

3.1 Is empty check and get size

```
/*  Function to check if list is empty  */
public boolean isEmpty()
{
    return start == null;
}


/*  Function to get size of list  */
public int getSize()
{
    return size;
}
```

3.2 Insert a node at beginning

```
/*  Function to insert a node at beginning  */

public void insertAtStart(int val)
    {
        Node nptr = new Node(val, null);
        size++ ;
        if(start == null)
        {
            start = nptr;
            end = start;
        }
        else
        {
            nptr.setLink(start);
            start = nptr;
        }
    }
```

3.3 Insert a node at end

```
/*  Function to insert a node at end  */

    public void insertAtEnd(int val)
    {
        Node nptr = new Node(val,null);
        size++ ;
        if(start == null)
        {
            start = nptr;
            end = start;
        }
        else
        {
            end.setLink(nptr);
            end = nptr;
        }
    }
```

3.4 Insert a node at position

```
/*  Function to insert a node at position  */

    public void insertAtPos(int val , int pos)
    {
        Node nptr = new Node(val, null);
        Node ptr = start;
        pos = pos - 1 ;
        for (int i = 1; i < size; i++)
        {
            if (i == pos)
            {
                Node tmp = ptr.getLink() ;
                ptr.setLink(nptr);
                nptr.setLink(tmp);
                break;
            }
            ptr = ptr.getLink();
        }
        size++ ;
    }
```

3.4 Delete a node at position

```
/*  Function to delete a node at position  */
    public void deleteAtPos(int pos)
    {
        if (pos == 1)
        {
            start = start.getLink();
            size--;
            return ;
        }
        if (pos == size)
        {
            Node s = start;
            Node t = start;
            while (s != end)
            {
                t = s;
                s = s.getLink();
            }
            end = t;
            end.setLink(null);
            size --;
            return;
        }
```

3.4 Delete a node at position

```
Node ptr = start;
        pos = pos - 1 ;
        for (int i = 1; i < size - 1; i++)
        {
            if (i == pos)
            {
                Node tmp = ptr.getLink();
                tmp = tmp.getLink();
                ptr.setLink(tmp);
                break;
            }
            ptr = ptr.getLink();
        }
        size-- ;
    }
```

3.5 Display nodes

```
/*  Function to display nodes  */
    public void display()
    {
        System.out.print("\nLinked List = ");
        if (size == 0)
        {
            System.out.print("empty\n");
            return;
        }
        if (start.getLink() == null)
        {
            System.out.println(start.getData() );
            return;
        }
        Node ptr = start;
        System.out.print(start.getData()+ "->");
        ptr = start.getLink();
        while (ptr.getLink() != null)
        {
            System.out.print(ptr.getData()+ "->");
            ptr = ptr.getLink();
        }
        System.out.print(ptr.getData()+ "\n");
    }
```

4. LinkedListTest class

```
/*  Class LinkedListTest  */
public class LinkedListTest
{
    public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);
        /* Creating object of class linkedList */
        LinkedList list = new LinkedList();
        System.out.println("Linked List Test\n");
        char ch;
        /*  Perform list operations  */
        do
        {
            System.out.println("\nLinked List Operations\n");
            System.out.println("1. insert at beginning");
            System.out.println("2. insert at end");
            System.out.println("3. insert at position");
            System.out.println("4. delete at position");
            System.out.println("5. check empty");
            System.out.println("6. get size");
            int choice = scan.nextInt();
```

4. LinkedListTest class

```
switch (choice)
            {
            case 1 :
                System.out.println("Enter integer element to insert");
                list.insertAtStart( scan.nextInt() );
                break;
            case 2 :
                System.out.println("Enter integer element to insert");
                list.insertAtEnd( scan.nextInt() );
                break;
            case 3 :
                System.out.println("Enter integer element to insert");
                int num = scan.nextInt() ;
                System.out.println("Enter position");
                int pos = scan.nextInt() ;
                if (pos <= 1 || pos > list.getSize() )
                    System.out.println("Invalid position\n");
                else
                    list.insertAtPos(num, pos);
                break;
```

4. LinkedListTest class

```
case 4 :
        System.out.println("Enter position");
        int p = scan.nextInt() ;
        if (p < 1 || p > list.getSize() )
            System.out.println("Invalid position\n");
        else
            list.deleteAtPos(p);
        break;
    case 5 :
        System.out.println("Empty status = "+ list.isEmpty());
        break;
    case 6 :
        System.out.println("Size = "+ list.getSize() +" \n");
        break;
     default :
        System.out.println("Wrong Entry \n ");
        break;
    }
```

4. LinkedListTest class

```
/*  Display List  */
            list.display();
            System.out.println("\nDo you want to continue (Type y or n) \n");
            ch = scan.next().charAt(0);
        } while (ch == 'Y'|| ch == 'y');
    }
}
```

Output – Adding the first node

Linked List Operations

1. insert at beginning
2. insert at end
3. insert at position
4. delete at position
5. check empty
6. get size

1

Enter integer element to insert

10

Linked List = 10

Do you want to continue (Type y or n)

Output – Adding the last node

1. insert at beginning
2. insert at end
3. insert at position
4. delete at position
5. check empty
6. get size

2

Enter integer element to insert

20

Linked List = 10->20

Do you want to continue (Type y or n)

Output – Adding node in the middle

1. insert at beginning
2. insert at end
3. insert at position
4. delete at position
5. check empty
6. get size

3

Enter integer element to insert

15

Enter position

2

Linked List = 10->15->20

Do you want to continue (Type y or n)

Output – Deleting node from the middle

1. insert at beginning
2. insert at end
3. insert at position
4. delete at position
5. check empty
6. get size

4

Enter position

2

Linked List = 10->20

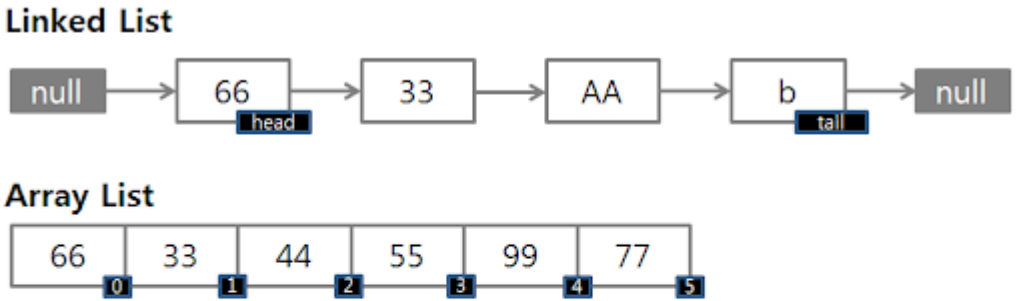
Do you want to continue (Type y or n)

ArrayList

ArrayList

- An array list is implemented with an array. When the array hits capacity, the Array List class will create a new array with double the capacity and copy all the elements over to the new array.

- LinkedList vs ArrayList:



ArrayList implementation in Java

ArrayList implementation in Java

- To do list:
  1. Define a MyArrayList class
  2. Add the following functionality to the MyArrayList class
     1. Add a object to array list
     2. Increase array size and reallocate memory
     3. Get an object from the array list
     4. Remove an object from the array list
  3. Define a main method in MyArrayList class to check all the functionalities of MyArrayList

1. Define a MyArrayList class

```
/* MyArrayList class */
public class MyArrayList {

private static final int SIZE_FACTOR=5;

private Object data[];

private int index;

private int size;

public MyArrayList(){
this.data=new Object[SIZE_FACTOR];
this.size=SIZE_FACTOR;
}
}
```

2.1 Add an object to array list

```
/* Add object to array list */
public void add(Object obj){
System.out.println("index:"+this.index+"size:"+this.size+"data
size:"+this.data.length);
if(this.index==this.size-1){
//we need to increase the size of data[]
increaseSizeAndReallocate();
}
data[this.index]=obj;
this.index++;
System.out.println("Array List : ");
for(int i=0; i<this.index; i++) {
   if(i != (this.index - 1 ))
System.out.print(data[i]+ " -> ");
   else
System.out.print(data[i]);
}
System.out.println();
}
```

2.2 Increase array size and reallocate memory

```
/* Increase array size and reallocate memory */
private void increaseSizeAndReallocate() {
this.size=this.size+SIZE_FACTOR;
Object newData[]=new Object[this.size];
for(int i=0; i<data.length;i++){
newData[i]=data[i];
}
this.data=newData;
System.out.println("***index:"+this.index+"size:"+this.size+"data
size:"+this.data.length);
}
```

2.3 Get an object from the array list

```
/* Get an object from the array list */
public Object get(int i) throws Exception{
if(i>this.index-1){
throw new Exception("ArrayIndexOutOfBound");
}
if(i<0){
throw new Exception("Negative Value");
}
return this.data[i];

}
```

2.4 Remove an object from the array list

```
/* Remove an object from the array list */
public void remove(int i) throws Exception{
if(i>this.index-1){
throw new Exception("ArrayIndexOutOfBound");
}
if(i<0){
throw new Exception("Negative Value");
}
System.out.println("Object getting removed:"+this.data[i]);
for(int x=i; x<this.data.length-1;x++){
data[x]=data[x+1];
}
this.index--;
}
```

3. Define a main method to test all the functionalities

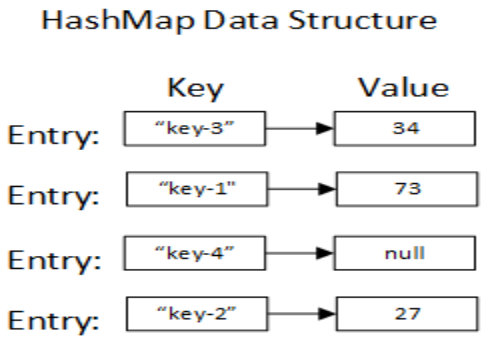
```
public static void main(String[] args) throws Exception {
MyArrayList mal = new MyArrayList();
mal.add("0");
mal.add("1");
mal.add("2");
mal.add("3");
mal.add("4");
mal.add("5");
mal.add("6");
mal.add("7");
mal.add("8");
mal.add("9");

mal.remove(5);
System.out.println(mal.get(7));
}
```

HashMap

HashMap

- A hash map is a data structure that is able to map certain keys to certain values. Hash Maps are much faster for retrieving data than arrays and linked lists. Hash Maps use an array in the background. Each element in the array is another data structure (usually a linked list or array list).



HashMap implementation in Java

HashMap implementation in Java

- To do list:
  1. Define a MyHashMap class along with a nested class Container
  2. Add the following functionality to the MyHashMap class
     1. Put key and value pair inside HashMap with the help of the Container class
     2. Get the value by using key
  3. Define a main method in MyHashMap class to check all the functionalities of MyHashMap

1. Define a MyHashMap class

```
public class MyHashMap {
class Container{
Object key;
Object value;
public void insert(Object k, Object v){
this.key=k;
this.value=v;
}
@Override
public String toString() {
return "{" + this.key +"="+this.value+ "}";
}
}
private Container c;
private List<Container> recordList;
public MyHashMap(){
this.recordList=new ArrayList<Container>();
}
}
```

2.1 Put key and value pair inside HashMap

```
/* Put key and value pair inside HashMap with the help Container class */
public void put(Object k, Object v){
this.c=new Container();
c.insert(k, v);
//check for the same key before adding
for(int i=0; i<recordList.size(); i++){
Container c1=recordList.get(i);
if(c1.key.equals(k)){
//remove the existing object
recordList.remove(i);
break;
}
}
recordList.add(c);
System.out.print("My Map : ");
for(int i=0; i<recordList.size(); i++)
System.out.print(recordList.get(i) + " ");
System.out.println();
}
```

2.2 Get the value by using key

```
/* Get the value by using key */
public Object get(Object k){
for(int i=0; i<this.recordList.size(); i++){
Container con = recordList.get(i);
if (k.toString()==con.key.toString()) {

return con.value;
}

}
return null;
}
```

3. Define a main method to test all the functionalities

```
public static void main(String[] args) {
MyHashMap hm = new MyHashMap();
hm.put("1", "10");
hm.put("2", "20");
hm.put("3", "30");
System.out.println("Key 3 value : " + hm.get("3"));
hm.put("3", "40");

System.out.println("Key 1 value : " + hm.get("1"));
System.out.println("Key 3 value : " + hm.get("3"));
System.out.println("Key 8 value : " + hm.get("8"));
}
```

Exercise

Exercise

- Implement Stack(Last In First Out) functionality in Java
- Implement Queue(First In First Out) functionality in Java
- Implement Circular Linked List in Java
- Implement Doubly Linked List in Java

Thank you