OBJECT ORIENTED PROGRAMMING USING

Java

Let's explore technology
together to live in the future

Checkout more on
https://github.com/Sy-hash-collab

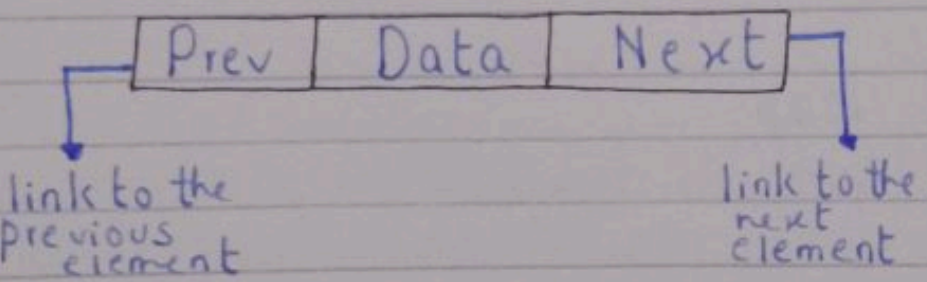
Sy-hash-collab

Linked List in Java :

Linked List is part of Collection framework present in java.util package. This class is an implementation of LinkedList data str. where the elements are not stored in contigous locations and every element is a separate object with a data part and adress part.

The elements are linked using pointers and addresss. | Each element is known as node.

The LinkedList class of the Java Collections framework provides the functionality of the linked list data structure (double linkedlist)



Each element in linked list is called node. It consist of 3 fields:

- Prev - stores an address of the previous element in the list. It is null for firsttime.

Next - stores an address of the next element in the list. It is null for last element

Data - stores actual data.

Creating a Java LinkedList:

```
LinkedList<Type> linkedList = new LinkedList
                                <Type>();
```

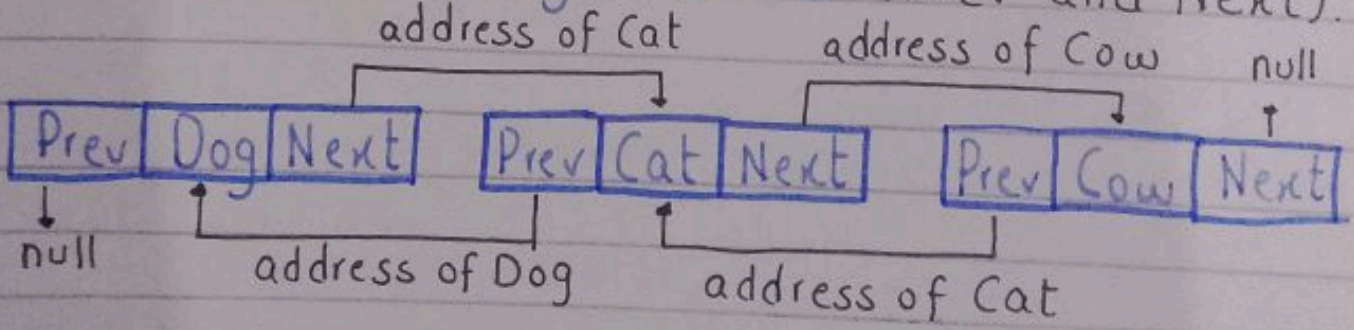
```
import java.util.LinkedList;
class Main {
public static void main (String [] args)
{ LinkedList <String> a = new LinkedList<String>(

  a.add ("Dog");
  a.add ("Cat");
  a.add ("Cow");
  System.out.println ("Linked List: "+ a);
}}
```

Linked List : [Dog, Cat, Cow]

Working of Java Linked List:

Elements in linked list are not stored in sequence. Instead they are scattered and connected through links (Prev and Next).



ArrayList vs LinkedList

The LinkedList class is a collection which can contain many objects of the same type just like ArrayList.

How Arraylist works?

The arraylist has a regular array inside it. When an element is added, it is placed into the array.
If the array is not big enough, a new, larger array is created to replace the old one and old one is removed.

How LinkedList works?

The LinkedList stores it items in
The list has a link to the first container and each container has a link to next container in the list.
To add an element to the list, the element is placed into a new container and that container is linked to one of other containers in the list.

When to use:

Use an ArrayList for storing and accessing dat and LinkedList to manipulate data.

LinkedList Methods:

addFirst() adds an item to beginning of list.

```
import java.util.LinkedList;
public class Main {
 public static void main (String[] args)
{
  LinkedList<String> cars = new LinkedList<String>();
  cars.add ("Volvo");
  cars.add ("Ford");

  cars.addFirst ("BMW");
  System.out.println (cars);
}}
```

[BMW, Volvo, Ford]

addLast() adds an item to last of list.

```
import java.util.LinkedList;
public class Main {
 public static void main (String [] args)
{
LinkedList<String> cars = new LinkedList<String>();

 cars.add ("Volvo");
 cars.add ("BMW");
 cars.add ("Ford");

 cars.addLast ("Mazda");

 System.out.println (cars);
}}
```

[Volvo, BMW, Ford, Mazda]

removeFirst() Remove an item From the beginning of the list.

```
import java.util.LinkedList;
public class Main {
 public static void main (String [] args)
{
LinkedList<String> cars = new LinkedList<String>();

 cars.add ("volvo");
 cars.add ("BMW");
 cars.add ("Ford");

 cars.removeFirst ();
 System.out.println (cars);
}}
```

[BMW , Ford]

removeLast() Remove an item from the end of the list.

```
import java.util.Linkedlist;
public class Main {
 public static void main (String [] args)
{
LinkedList<String> cars = new linkedlist<String>();

 cars.add ("Volvo");
 cars.add ("BMW");
 cars.add ("Ford");

 cars.removeLast ();
 System.out.println (cars);
}}
```

getFirst() Get the item at beginning of list

```
import java.util.LinkedList;
public class Main {
 public static void main (String [] args)
{ LinkedList cars = new LinkedList<String>();
  cars.add ("Volvo");
  cars.add ("BMW");
  cars.add ("Ford");

  cars.getFirst();

  System.out.println (cars.getFirst());
}}
```

Volvo

getLast() Get the item at the end of the list

```
import java.util.LinkedList;
public class Main {
 public static void main (String [] args)
{ LinkedList<String> cars = new LinkedList<String>();

  cars.add ("Volvo");
  cars.add ("BMW");
  cars.add ("Ford");

  System.out.println (cars.getLast());
}}
```

Ford

Performing various operations on Linkedlist:

Operation 1 : Adding Elements

In order to add an element to Arraylist, we use add() method.

add(Object) : This method is used to add an element at end of Linkedlist.

add(int index, Obj) : This method is used to add element at specific index.

```
import java.util.LinkedList;
public class Main{
 public static void main (String [] args)
{
LinkedList<String> o = new LinkedList<String>();
 o.add ("Geeks");
 o.add ("Geeks");
 o.add (1, "For");

 System.out.println (o);
}}
```

[Geeks, For, Geeks]

Operation 2 : Changing Elements

It can be done useng the set() method The Linkedlist is indexed so the element we wish to change is referrenced by index of the element. This method takes an index and updated element

```
import java.util.Linkedlist;
public class Main {
 public static void main (String [] args)
{ LinkedList<String> o = new Linkedlist<String
  o.add ("Geelcs");
  o.add ("Geelcs");
  o.add ("Geelcs");
  o.set (1, "For");

  System.out.println (o);
}}
```

Operation 3 : Removing Elements

In order to remove element from Linkedlist we use remove() method.

- **remove (Object):** This method is used to remove object from linkedli
- **remove (int index):** This method takes an integer which removes t element present at that inde

```
import java.util.Linkedlist;
public class Main {
 public static void main (String [] args)
{ LinkedList<String> o = new LinkedList<String>(
  o.add ("Geelcs");
  o.add ("Geelcs");
  o.add (1, "For");
```

```
System.out.println ("Initial Linked List: "+o);

o.remove (1);

System.out.println ("After index Removal: "+o);

o.remove ("Geeks");

System.out.println ("After Object Removal: "+o);

}}
```

Initial LinkedList [Geeks, for, Geeks]
After index Removal [Geeks, Geeks]
After object Removal [Geeks]

Operation 4: Iterating the LinkedList

```
import java.util.Linkedlist ;
public class Main {
public static void main (String [Jargs)
{ Linkedlist<String> o = new Linkedlist<String>();

o.add ("Geeks") ;
o.add ("For ");
o.add ("Geeks ");
for (int i=0 ; i<o.size() ; i++)
{ System.out.println (o.get (i));
}}
}}
```

Geeks for Geeks