Clone a linked list with next and random pointer | Set 2

We have already discussed 2 different ways to clone a linked list. In this post, one more simple method to clone a linked list is discussed.

The idea is to use Hashing. Below is algorithm.
1. Traverse the original linked list and make a copy in terms of data.
2. Make a hash map of key value pair with original linked list node and copied linked list node.
3. Traverse the original linked list again and using the hash map adjust the next and random reference of cloned linked list nodes.

A Java based approach of the above idea is below.

```
// Java program to clone a linked list with random pointers
import java.util.HashMap;
import java.util.Map;

// Linked List Node class
class Node
{
    int data;//Node data
    Node next, random;//Next and random reference

    //Node constructor
    public Node(int data)
    {
        this.data = data;
        this.next = this.random = null;
    }
}

// linked list class
class LinkedList
{
    Node head;//Linked list head reference

    // Linked list constructor
    public LinkedList(Node head)
    {
        this.head = head;
    }

    // push method to put data always at the head
    // in the linked list.
    public void push(int data)
    {
        Node node = new Node(data);
        node.next = this.head;
        this.head = node;
    }

    // Method to print the list.
    void print()
    {
        Node temp = head;
        while (temp != null)
        {
            Node random = temp.random;
            int randomData = (random != null)? random.data: -1;
            System.out.println("Data = " + temp.data +
                               ", Random data = "+ randomData);
            temp = temp.next;
        }
    }

    // Actual clone method which returns head
    // reference of cloned linked list.
```

```java
    // reference of cloned linked list.
    public LinkedList clone()
    {
        // Initialize two references, one with original
        // list's head.
        Node origCurr = this.head, cloneCurr = null;

        // Hash map which contains node to node mapping of
        // original and clone linked list.
        Map<Node, Node> map = new HashMap<Node, Node>();

        // Traverse the original list and make a copy of that
        // in the clone linked list.
        while (origCurr != null)
        {
            cloneCurr = new Node(origCurr.data);
            map.put(origCurr, cloneCurr);
            origCurr = origCurr.next;
        }

        // Adjusting the original list reference again.
        origCurr = this.head;

        // Traversal of original list again to adjust the next
        // and random references of clone list using hash map.
        while (origCurr != null)
        {
            cloneCurr = map.get(origCurr);
            cloneCurr.next = map.get(origCurr.next);
            cloneCurr.random = map.get(origCurr.random);
            origCurr = origCurr.next;
        }

        //return the head reference of the clone list.
        return new LinkedList(map.get(this.head));
    }
}

// Driver Class
class Main
{
    // Main method.
    public static void main(String[] args)
    {
        // Pushing data in the linked list.
        LinkedList list = new LinkedList(new Node(5));
        list.push(4);
        list.push(3);
        list.push(2);
        list.push(1);

        // Setting up random references.
        list.head.random = list.head.next.next;
        list.head.next.random =
            list.head.next.next.next;
        list.head.next.next.random =
            list.head.next.next.next.next;
        list.head.next.next.next.random =
            list.head.next.next.next.next.next;
        list.head.next.next.next.next.random =
            list.head.next;

        // Making a clone of the original linked list.
        LinkedList clone = list.clone();

        // Print the original and cloned linked list.
        System.out.println("Original linked list");
        list.print();
        System.out.println("\nCloned linked list");
        clone.print();
    }
}
```

Output:

```
Original linked list
Data = 1, Random data = 3
Data = 2, Random data = 4
Data = 3, Random data = 5
Data = 4, Random data = -1
Data = 5, Random data = 2

Cloned linked list
Data = 1, Random data = 3
Data = 2, Random data = 4
Data = 3, Random data = 5
Data = 4, Random data = -1
Data = 5, Random data = 2
```

Time complexity : O(n)
Auxiliary space : O(n)