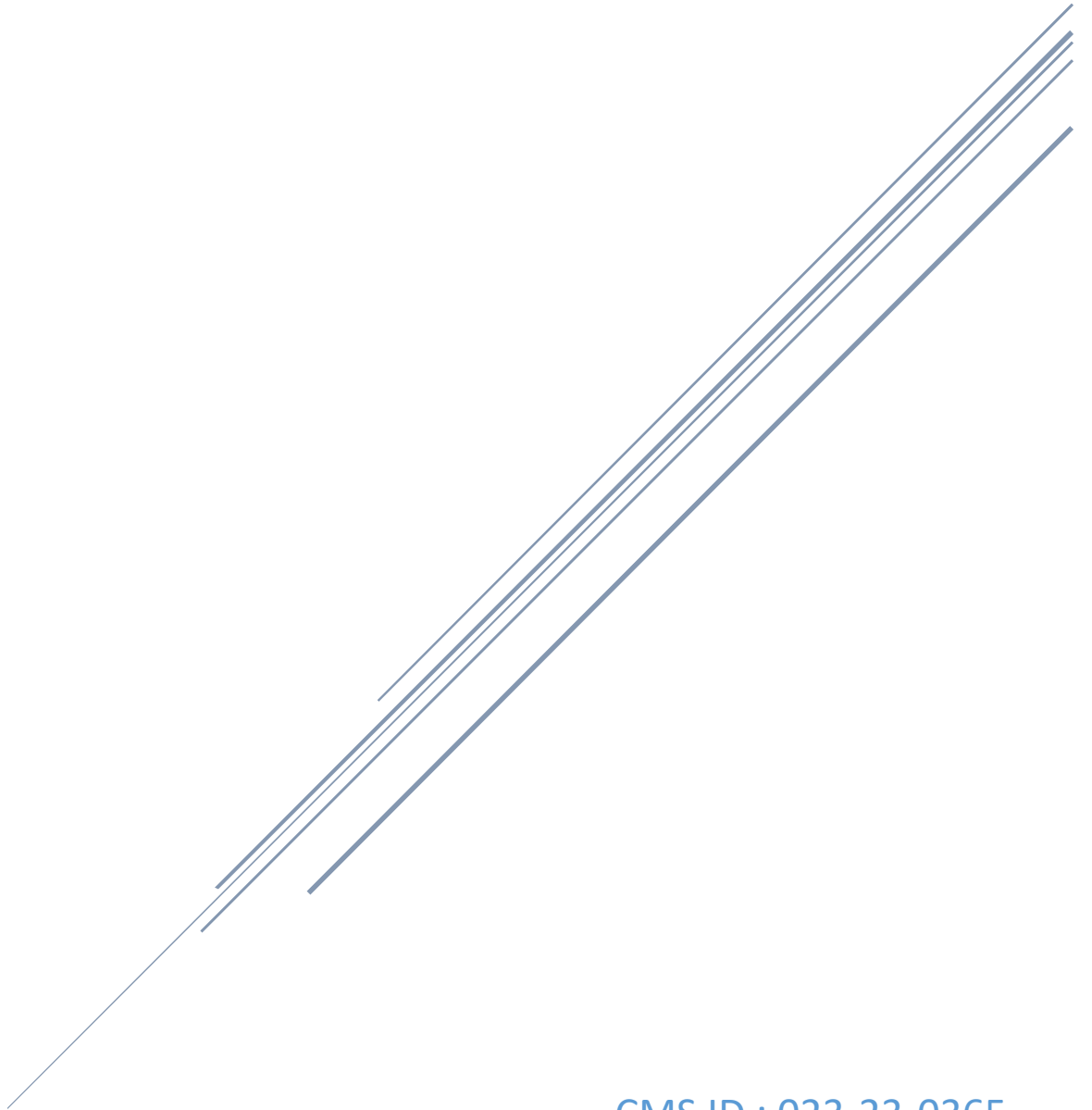


ASSIGNMENT 02 EXERCISE

FAHEEM AKBAR



CMS ID : 023-23-0365

DSA ASSIGNMENT 02 || Submitted to ma'am Marina Gul

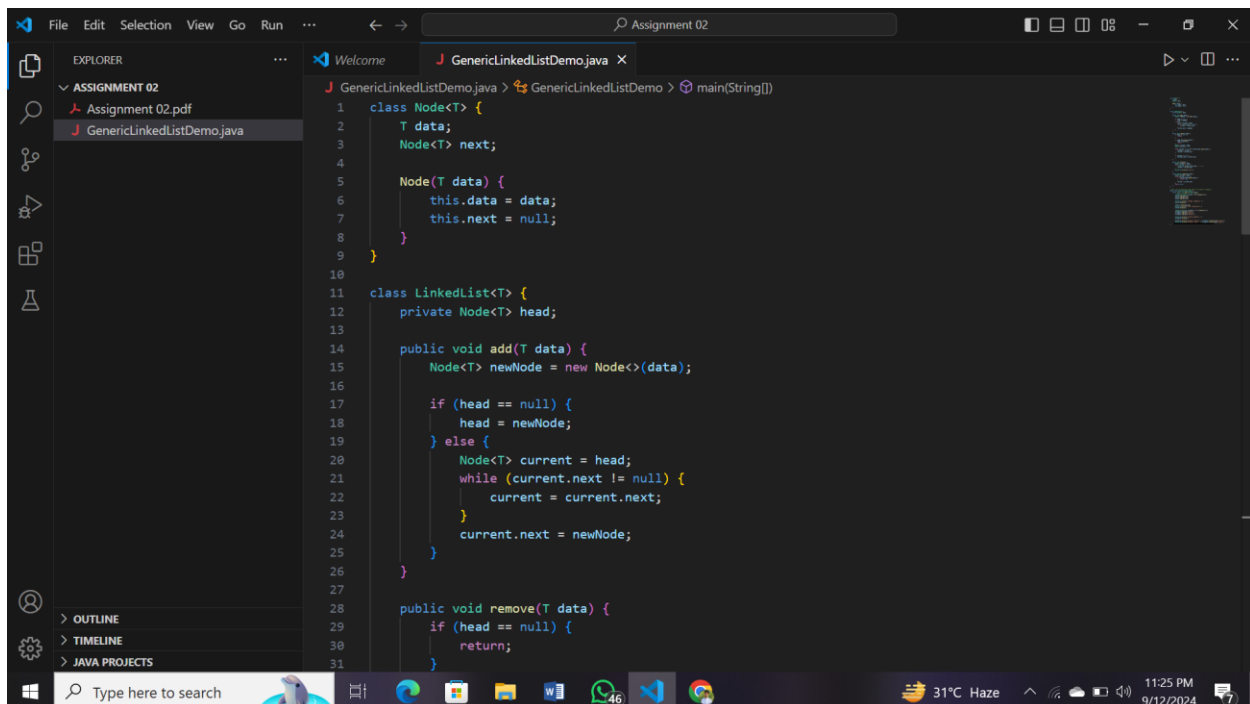
Assignment: 02

Generic Linked List Generics means parameterized types. The idea is to allow type (Integer, String, ... etc., and user-defined types) to be a parameter to methods, classes, and interfaces. Using Generics, it is possible to create classes that work with different data types. An entity such as class, interface, or method that operates on a parameterized type is a generic entity. we use <> to specify parameter types in generic class creation.

To create objects of a generic class, we use the following syntax.

Solution: To implement a generic linked list in Java, we need to create a LinkedList class that can store elements of any type. This means we'll make use of Java Generics to create a linked list that can be parameterized by any data type (e.g., Integer, String, CustomClass, etc.).

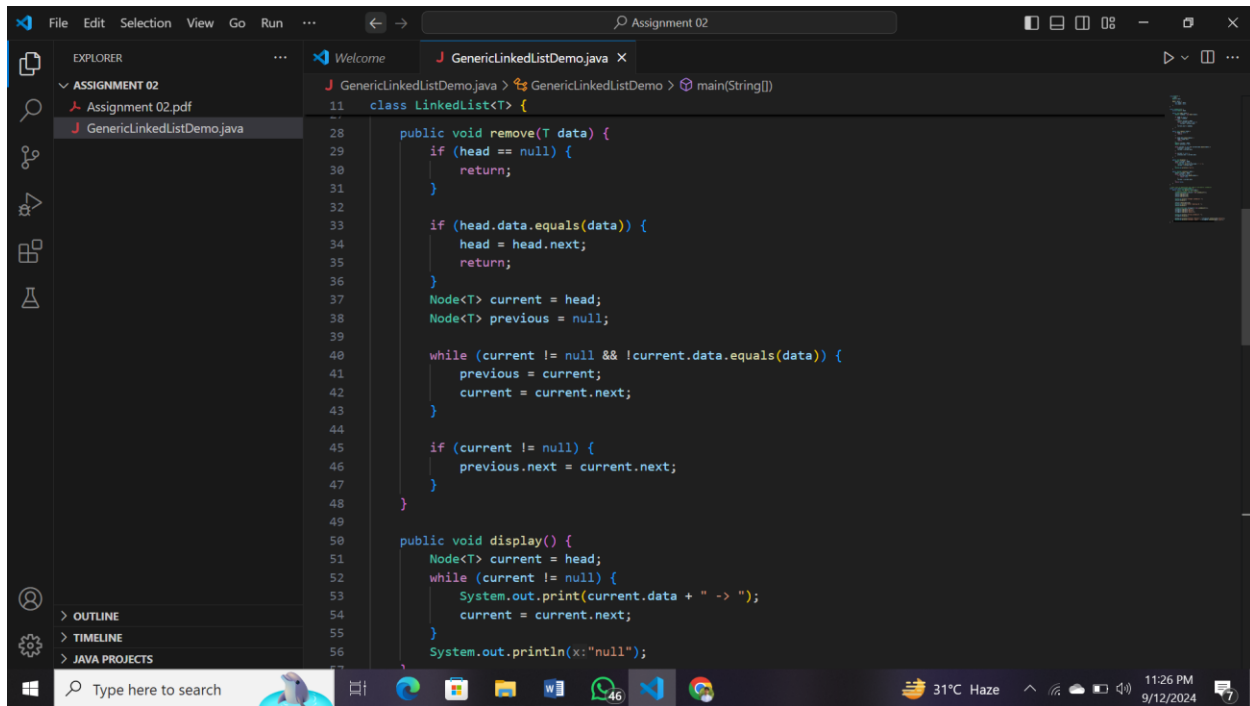
Below is the implementation of a simple generic linked list in Java:



The screenshot shows a code editor with the following Java code:

```
1 class Node<T> {
2     T data;
3     Node<T> next;
4
5     Node(T data) {
6         this.data = data;
7         this.next = null;
8     }
9 }
10
11 class LinkedList<T> {
12     private Node<T> head;
13
14     public void add(T data) {
15         Node<T> newNode = new Node<>(data);
16
17         if (head == null) {
18             head = newNode;
19         } else {
20             Node<T> current = head;
21             while (current.next != null) {
22                 current = current.next;
23             }
24             current.next = newNode;
25         }
26     }
27
28     public void remove(T data) {
29         if (head == null) {
30             return;
31         }
32     }
33 }
```

The IDE interface includes a sidebar with 'EXPLORER' showing 'ASSIGNMENT 02' and 'GenericLinkedListDemo.java'. The bottom status bar shows '31°C Haze' and '11:25 PM 9/12/2024'.



```
11 class LinkedList<T> {  
50     public void display() {  
51         Node<T> current = head;  
52         while (current != null) {  
53             System.out.print(current.data + " -> ");  
54             current = current.next;  
55         }  
56         System.out.println(x:"null");  
57     }  
58  
59     public boolean contains(T data) {  
60         Node<T> current = head;  
61         while (current != null) {  
62             if (current.data.equals(data)) {  
63                 return true;  
64             }  
65             current = current.next;  
66         }  
67         return false;  
68     }  
69 }  
70  
71 // Main class to demonstrate the usage of the generic LinkedList  
72 public class GenericLinkedListDemo {  
73     public static void main(String[] args) {  
74         // Create a linked list for integers  
75         LinkedList<Integer> intList = new LinkedList<>();  
76         intList.add(data:10);  
77         intList.add(data:20);  
78         intList.add(data:30);  
79     }  
80 }
```

```
71 // Main class to demonstrate the usage of the generic LinkedList  
72 public class GenericLinkedListDemo {  
73     public static void main(String[] args) {  
74         // Create a linked list for integers  
75         LinkedList<Integer> intList = new LinkedList<>();  
76         intList.add(data:10);  
77         intList.add(data:20);  
78         intList.add(data:30);  
79         System.out.print(s:"Integer LinkedList: ");  
80         intList.display();  
81         intList.remove(data:20);  
82         System.out.print(s:"After removing 20: ");  
83         intList.display();  
84         LinkedList<String> stringList = new LinkedList<>();  
85         stringList.add(data:"Hello");  
86         stringList.add(data:"World");  
87         stringList.add(data:"Generic");  
88         System.out.print(s:"String LinkedList: ");  
89         stringList.display();  
90         System.out.println("Contains 'World'? " + stringList.contains(data:"World"));  
91         System.out.println("Contains 'Java'? " + stringList.contains(data:"Java"));  
92     }  
93 }
```

OUTPUT:

