

```

1. import java.util.LinkedList;

public class Stack<T> {

    private LinkedList<T> list = new LinkedList<>();

    public void push(T item) {

        list.addFirst(item);
    }

    public T pop() {

        if (isEmpty()) {

            System.out.println("Stack is empty. Cannot pop.");

            return null;

        }

        return list.removeFirst();
    }

    public T peek() {

        if (isEmpty()) {

            System.out.println("Stack is empty. Cannot peek.");

            return null;

        }

        return list.getFirst();
    }

    public boolean isEmpty() {

        return list.isEmpty();
    }

    public int size() {

        return list.size();
    }

    public static void main(String[] args) {

        Stack<Integer> stack = new Stack<>();

        stack.push(200);

        stack.push(100);
    }
}

```

```

stack.push(600);

stack.push(300);

stack.push(500);

System.out.println("Top element: " + stack.peek());

System.out.println("Stack size: " + stack.size());

System.out.println("Popped element: " + stack.pop());

System.out.println("Stack size after pop: " + stack.size());

stack.pop();

stack.pop();

System.out.println("Top element after popping all elements: " + stack.peek());

}

}

```

The screenshot displays the Programiz Online Java Compiler interface. The code editor on the left contains a Java program for a stack implemented using a linked list. The program includes methods for pushing elements, popping elements, peeking at the top element, checking if the stack is empty, and getting the stack size. The output window on the right shows the results of the program execution, which are: Top element: 500, Stack size: 5, Popped element: 500, Stack size after pop: 4, Top element after popping all elements: 100. The output also indicates that the code execution was successful.

```

Main.java
1 import java.util.LinkedList;
2 public class Stack<T> {
3     private LinkedList<T> list = new LinkedList<>();
4     public void push(T item) {
5         list.addFirst(item);
6     }
7     public T pop() {
8         if (isEmpty()) {
9             System.out.println("Stack is empty. Cannot pop.");
10            return null;
11        }
12        return list.removeFirst();
13    }
14    public T peek() {
15        if (isEmpty()) {
16            System.out.println("Stack is empty. Cannot peek.");
17            return null;
18        }
19        return list.getFirst();
20    }
21    public boolean isEmpty() {
22        return list.isEmpty();
23    }
24    public int size() {
25        return list.size();

```

```

Output
java -cp /tmp/v0350X19U7/Stack
Top element: 500
Stack size: 5
Popped element: 500
Stack size after pop: 4
Top element after popping all elements: 100

=== Code Execution Successful ===

```

```
2. import java.util.LinkedList;
```

```
public class LinkedListQueue{
```

```
    public static void main(String[] args) {
```

```
        LinkedList<Integer> queue = new LinkedList<>();
```

```
        queue.addLast(100);
```

```
        queue.addLast(200);
```

```
        queue.addLast(300);
```

```
        queue.addLast(400);
```

```
        queue.addLast(500);
```

```
        System.out.println("Queue: " + queue);
```

```
        System.out.println("Dequeued: " + queue.removeFirst());
```

```
        System.out.println("Queue after all elements removed: " + queue);
```

```
    }
```

```
}
```

The screenshot shows a web browser window with the URL `programiz.com/java-programming/online-compiler/`. The page features the Programiz logo and a banner for learning programming. The main content area is divided into two panels: 'Main.java' on the left and 'Output' on the right. The 'Main.java' panel contains the following code:

```
1. import java.util.LinkedList;
2. public class LinkedListQueue{
3.     public static void main(String[] args) {
4.         LinkedList<Integer> queue = new LinkedList<>();
5.         queue.addLast(100);
6.         queue.addLast(200);
7.         queue.addLast(300);
8.         queue.addLast(400);
9.         queue.addLast(500);
10.        System.out.println("Queue: " + queue);
11.
12.        System.out.println("Dequeued: " + queue.removeFirst());
13.        System.out.println("Queue after all elements removed: " + queue);
14.    }
15. }
```

The 'Output' panel shows the results of the code execution:

```
java -cp /tmp/uZqewE1zoL/LinkedListQueue
Queue: [1, 2, 3, 4, 5]
Dequeued: 1
Queue after all elements removed: [2, 3, 4, 5]

=== Code Execution Successful ===
```

The bottom of the screenshot shows a Windows taskbar with the date and time set to 14:00 on 05-08-2024.

```
3. import java.util.HashMap;

import java.util.Map;

public class hashmap{

    public static void main(String[] args) {

        HashMap<String, String> map = new HashMap<>();

        map.put("Apple", "Green");

        map.put("Banana", "Yellow");

        map.put("Cherry", "Red");

        map.put("Date", "Brown");

        System.out.println("Initial HashMap: " + map);

        String value1 = map.get(1);

        String value2 = map.get(2);

        System.out.println("Value for key 1: " + value1);

        System.out.println("Value for key 2: " + value2);


        boolean hasKey3 = map.containsKey(3);

        boolean hasKey5 = map.containsKey(5);

        System.out.println("HashMap contains key 3: " + hasKey3);

        System.out.println("HashMap contains key 5: " + hasKey5);


        boolean hasValueRed = map.containsValue("Red");

        boolean hasValueBrown = map.containsValue("Brown");

        System.out.println("HashMap contains value 'Cherry': " + hasValueRed);

        System.out.println("HashMap contains value 'Grape': " + hasValueBrown);


        map.remove("Cherry");

        System.out.println("HashMap after removing key 2: " + map);
```

```
map.put("Apple", "Crimson Red");
```

```
System.out.println("HashMap after updating key 3: " + map);
```

```
System.out.println("Iterating over HashMap:");
```

```
for (Map.Entry<String, String> entry : map.entrySet()) {
```

```
    System.out.println("Key: " + entry.getKey() + ", Value: " + entry.getValue());
```

```
}
```

```
int size = map.size();
```

```
System.out.println("Size of HashMap: " + size);
```

```
map.clear();
```

```
System.out.println("HashMap after clearing: " + map);
```

```
System.out.println("Size after clearing: " + map.size());
```

```
}
```

```
}
```

The screenshot shows a web browser window with the URL `programiz.com/java-programming/online-compiler/`. The page features a banner for "Programiz" with the text "LOOKING TO LEARN PROGRAMMING? Start your programming journey with Programiz AT NO COST." and a "Programiz PRO" button. Below the banner, there is a code editor with a file named "Main.java" and an "Output" window. The code in "Main.java" is as follows:

```
1 import java.util.HashMap;
2 import java.util.Map;
3 public class hashmap{
4     public static void main(String[] args) {
5         HashMap<String, String> map = new HashMap<>();
6         map.put("Apple", "Green");
7         map.put("Banana", "Yellow");
8         map.put("Cherry", "Red");
9         map.put("Date", "Brown");
10        System.out.println("Initial HashMap: " + map);
11        String value1 = map.get(1);
12        String value2 = map.get(2);
13        System.out.println("Value for key 1: " + value1);
14        System.out.println("Value for key 2: " + value2);
15        boolean hasKey3 = map.containsKey(3);
16        boolean hasKey5 = map.containsKey(5);
17        System.out.println("HashMap contains key 3: " + hasKey3);
18        System.out.println("HashMap contains key 5: " + hasKey5);
19        boolean hasValueRed = map.containsValue("Red");
20        boolean hasValueBrown = map.containsValue("Brown");
21        System.out.println("HashMap contains value 'Cherry': " + hasValueRed);
22        System.out.println("HashMap contains value 'Grape': " + hasValueBrown);
23        map.remove("Cherry");
24        System.out.println("HashMap after removing key 2: " + map);
25        map.put("Apple", "Crimson Red");
```

The "Output" window displays the following results:

```
java -cp /tmp/2y1zYw8Cw/hashmap
Initial HashMap: {Apple=Green, Cherry=Red, Date=Brown, Banana=Yellow}
Value for key 1: null
Value for key 2: null
HashMap contains key 3: false
HashMap contains key 5: false
HashMap contains value 'Cherry': true
HashMap contains value 'Grape': true
HashMap after removing key 2: {Apple=Green, Date=Brown, Banana=Yellow}
HashMap after updating key 3: {Apple=Crimson Red, Date=Brown, Banana=Yellow}
Iterating over HashMap:
Key: Apple, Value: Crimson Red
Key: Date, Value: Brown
Key: Banana, Value: Yellow
Size of HashMap: 3
HashMap after clearing: {}
Size after clearing: 0

=== Code Execution Successful ===
```

```

4. import java.util.ArrayList;

import java.util.Collections;

class Student implements Comparable<Student> {

    int rollNo;

    String name;

    int age;

    Student(int rollNo, String name,int age) {

        this.rollNo = rollNo;

        this.name = name;

        this.age=age;

    }

    public int compareTo(Student other) {

        if (this.rollNo > other.rollNo) {

            return -1;

        } else if (this.rollNo < other.rollNo) {

            return 1;

        } else {

            return 0;

        }

    }

    public String toString() {

        return "Student Details {rollNo=" + rollNo + ", name=" + name + ",age="

        + age+ "}";

    }

}

public class Main {

    public static void main(String[] args) {

        ArrayList<Student> students = new ArrayList<>();

        students.add(new Student(3, "Prajiith",19));

        students.add(new Student(1, "Sandy",20));

```

```
students.add(new Student(2, "Prashanth",17));
```

```
students.add(new Student(6, "Mano",16));
```

```
students.add(new Student(4, "Deenesh",19));
```

```
students.add(new Student(5, "Ganesh",20));
```

```
Collections.sort(students);
```

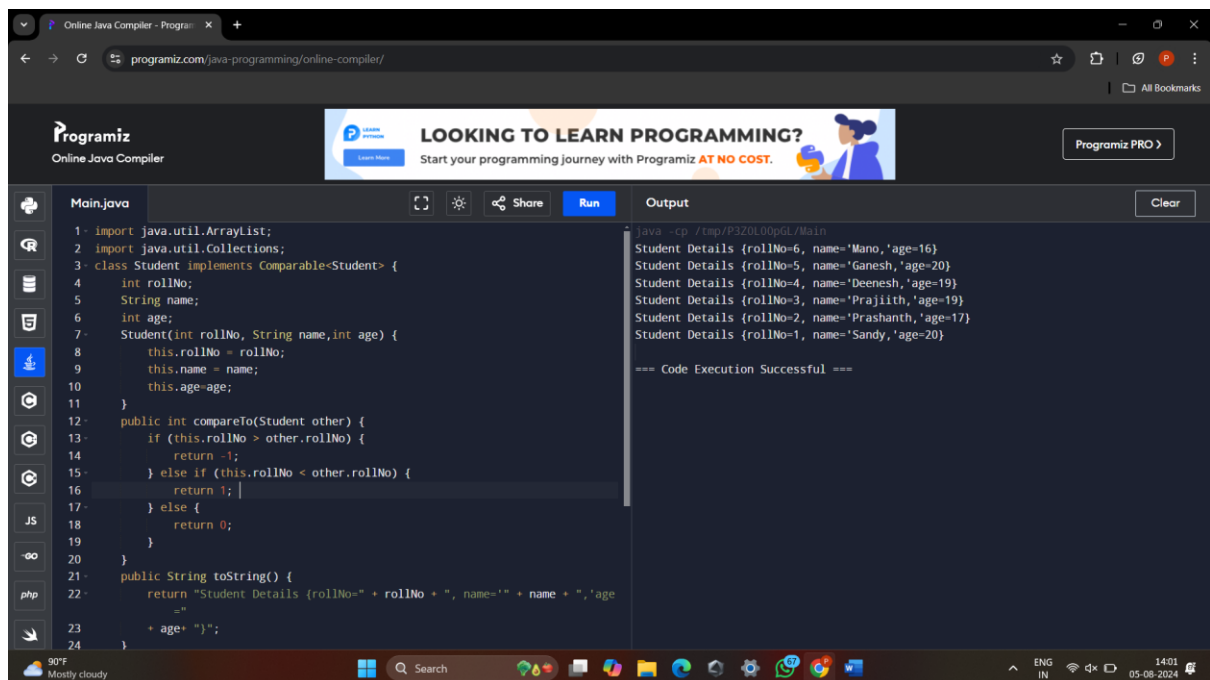
```
for (Student student : students) {
```

```
    System.out.println(student);
```

```
}
```

```
}
```

```
}
```



The screenshot displays the Programiz Online Java Compiler interface. The code editor on the left contains the following Java code:

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 class Student implements Comparable<Student> {
4     int rollNo;
5     String name;
6     int age;
7     Student(int rollNo, String name,int age) {
8         this.rollNo = rollNo;
9         this.name = name;
10        this.age=age;
11    }
12    public int compareTo(Student other) {
13        if (this.rollNo > other.rollNo) {
14            return -1;
15        } else if (this.rollNo < other.rollNo) {
16            return 1;
17        } else {
18            return 0;
19        }
20    }
21    public String toString() {
22        return "Student Details {rollNo=" + rollNo + ", name=" + name + ",age="
23        + age+ " }";
24    }
25 }
```

The output window on the right shows the execution results:

```
java -cp /tmp/P320L00pGL/Main
Student Details {rollNo=6, name='Mano','age=16}
Student Details {rollNo=5, name='Ganesh','age=20}
Student Details {rollNo=4, name='Deenesh','age=19}
Student Details {rollNo=3, name='Prajiith','age=19}
Student Details {rollNo=2, name='Prashanth','age=17}
Student Details {rollNo=1, name='Sandy','age=20}

=== Code Execution Successful ===
```