Class work (05.08.24)

SANTHOSH M

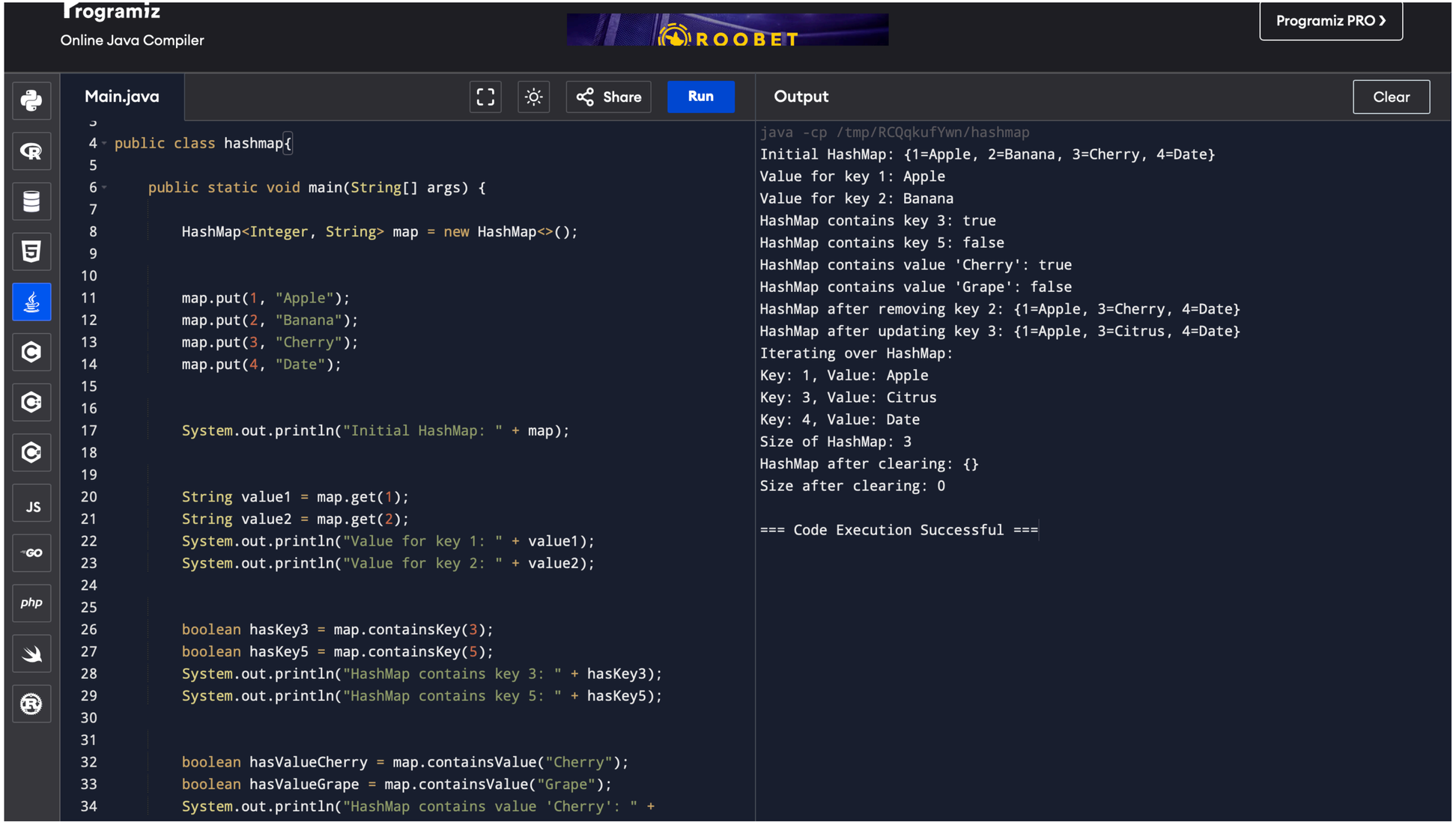
Main.java

Share | Run

```java
import java.util.LinkedList;

class Stack {
    private LinkedList<Integer> list;

    public Stack() {
        list = new LinkedList<>();
    }

    public void push(int value) {
        list.addLast(value);
    }

    public int pop() {
        if (list.isEmpty()) {
            throw new IllegalStateException("Stack is empty");
        }
        return list.removeLast();
    }

    public int peek() {
        if (list.isEmpty()) {
            throw new IllegalStateException("Stack is empty");
        }
        return list.getLast();
    }

    public boolean isEmpty() {
        return list.isEmpty();
    }
```

Output | Clear

```
java -cp /tmp/rLF3lAnroO/llstack
Stack size: 3
Top element: 30
Popped element: 30
Stack size after pop: 2

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


Programiz
Online Java Compiler
ROOBET
Programiz PRO ›
Main.java
Share
Run
Output
Clear
JS
go
php
4 public class hashmap{
5
6 public static void main(String[] args) {
7
8 HashMap<Integer, String> map = new HashMap<>();
9
10
11 map.put(1, "Apple");
12 map.put(2, "Banana");
13 map.put(3, "Cherry");
14 map.put(4, "Date");
15
16
17 System.out.println("Initial HashMap: " + map);
18
19
20 String value1 = map.get(1);
21 String value2 = map.get(2);
22 System.out.println("Value for key 1: " + value1);
23 System.out.println("Value for key 2: " + value2);
24
25
26 boolean hasKey3 = map.containsKey(3);
27 boolean hasKey5 = map.containsKey(5);
28 System.out.println("HashMap contains key 3: " + hasKey3);
29 System.out.println("HashMap contains key 5: " + hasKey5);
30
31
32 boolean hasValueCherry = map.containsValue("Cherry");
33 boolean hasValueGrape = map.containsValue("Grape");
34 System.out.println("HashMap contains value 'Cherry': " +
java -cp /tmp/RCQqkufYwn/hashmap
Initial HashMap: {1=Apple, 2=Banana, 3=Cherry, 4=Date}
Value for key 1: Apple
Value for key 2: Banana
HashMap contains key 3: true
HashMap contains key 5: false
HashMap contains value 'Cherry': true
HashMap contains value 'Grape': false
HashMap after removing key 2: {1=Apple, 3=Cherry, 4=Date}
HashMap after updating key 3: {1=Apple, 3=Citrus, 4=Date}
Iterating over HashMap:
Key: 1, Value: Apple
Key: 3, Value: Citrus
Key: 4, Value: Date
Size of HashMap: 3
HashMap after clearing: {}
Size after clearing: 0
=== Code Execution Successful ===

Main.java

Share | Run

```java
import java.util.LinkedList;

public class Queue<T> {
    private LinkedList<T> queue = new LinkedList<>();

    public void enqueue(T element) {
        queue.addLast(element);
    }

    public T dequeue() {
        if (isEmpty()) {
            throw new IllegalStateException("Queue is empty");
        }
        return queue.removeFirst();
    }

    public T peek() {
        if (isEmpty()) {
            throw new IllegalStateException("Queue is empty");
        }
        return queue.getFirst();
    }

    public boolean isEmpty() {
        return queue.isEmpty();
    }

    public int size() {
        return queue.size();
    }

```

Output | Clear

```
java -cp /tmp/iJf39tgQdu/Queue
Queue: [10, 20, 30]
Dequeued: 10
Peek: 20
Queue size: 2
Is queue empty? false

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

Main.java

Share | Run

```java
import java.util.ArrayList;
import java.util.Collections;
class Student implements Comparable<Student> {
    int rollNo;
    String name;

    Student(int rollNo, String name) {
        this.rollNo = rollNo;
        this.name = name;
    }
    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{rollNo=" + rollNo + ", name='" + name + "'}";
    }
}

public class Main {
    public static void main(String[] args) {
        ArrayList<Student> students = new ArrayList<>();
        students.add(new Student(3, "Alice"));
        students.add(new Student(1, "Bob"));
        students.add(new Student(2, "Charlie"));

        Collections.sort(students);
```

Output

Clear

```
java -cp /tmp/6vU5uqNNF2/Main
Student{rollNo=1, name='Bob'}
Student{rollNo=2, name='Charlie'}
Student{rollNo=3, name='Alice'}

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