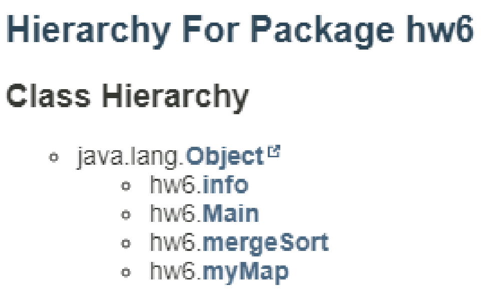
GTU Department of Computer Engineering
CSE 222 - Homework 6 - Report

Doğukan Taştan 1901042627



Hierarchy For Package hw6

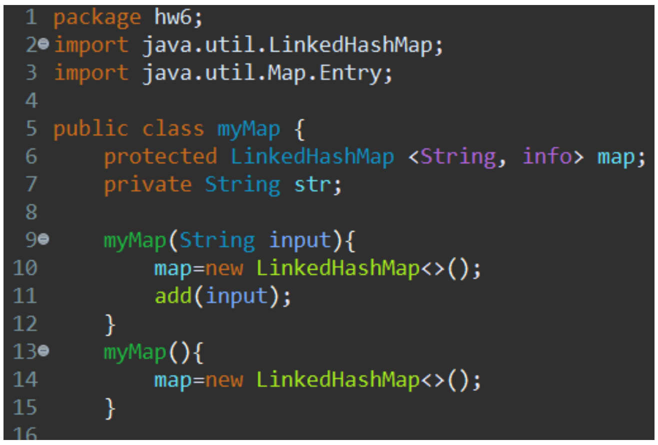
Class Hierarchy

- java.lang.Object
  - hw6.info
  - hw6.Main
  - hw6.mergeSort
  - hw6.myMap

compilation method:

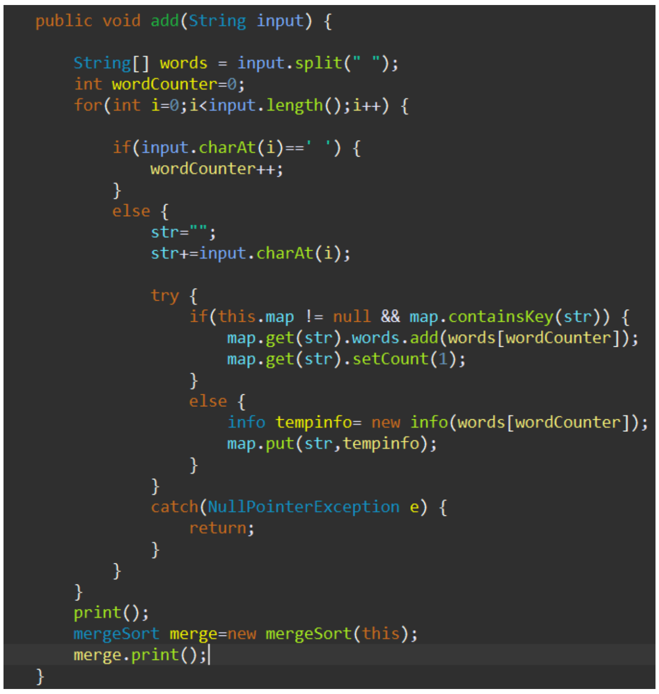
```
javac *.java
cd ..
java hw6.Main
```

myMap Class



```java
package hw6;
import java.util.LinkedHashMap;
import java.util.Map.Entry;

public class myMap {
    protected LinkedHashMap <String, info> map;
    private String str;

    myMap(String input){
        map=new LinkedHashMap<>();
        add(input);
    }
    myMap(){
        map=new LinkedHashMap<>();
    }
```

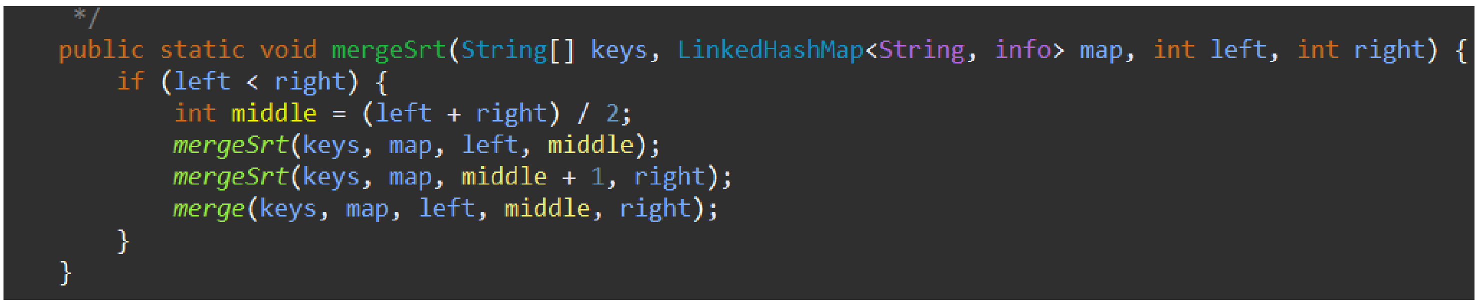
In the constructor, the map object is initialized and the add function is triggered. There is also an empty constructor for some uses.



```java
public void add(String input) {

    String[] words = input.split(" ");
    int wordCounter=0;
    for(int i=0;i<input.length();i++) {

        if(input.charAt(i)==' ') {
            wordCounter++;
        }
        else {
            str="";
            str+=input.charAt(i);

            try {
                if(this.map != null && map.containsKey(str)) {
                    map.get(str).words.add(words[wordCounter]);
                    map.get(str).setCount(1);
                }
                else {
                    info tempinfo= new info(words[wordCounter]);
                    map.put(str,tempinfo);
                }
            }
            catch(NullPointerException e) {
                return;
            }
        }
    }
    print();
    mergeSort merge=new mergeSort(this);
    merge.print();
}
```

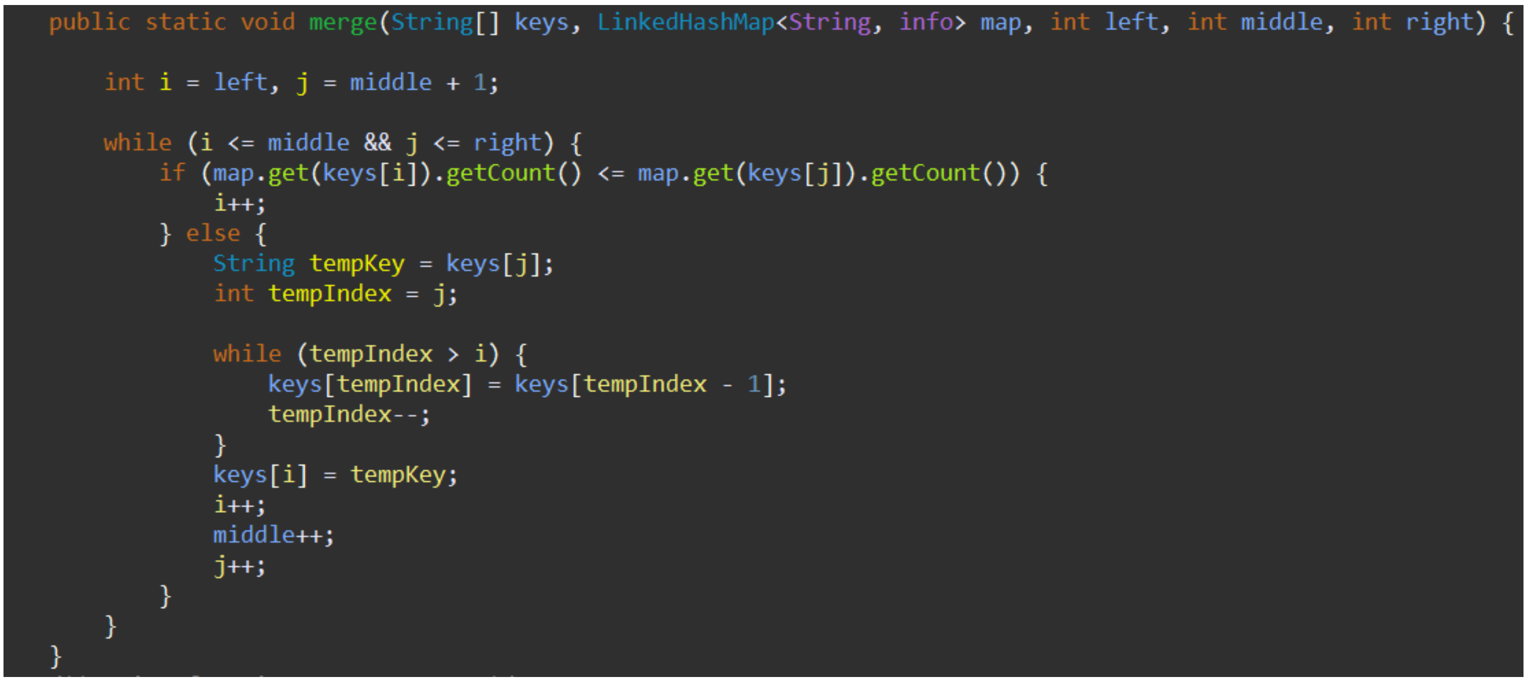
The append function first separates the incoming input into words. Then it starts looking at all the char one by one. If there is a space, it moves on to the next word. If there is such a char in the map before, it increases the counter and adds the word, but if there is no such char before, it creates it. At the bottom, merge sort object is called and merge operations are started.

mergeSort class



```
    */
   public static void mergeSrt(String[] keys, LinkedHashMap<String, info> map, int left, int right) {
       if (left < right) {
           int middle = (left + right) / 2;
           mergeSrt(keys, map, left, middle);
           mergeSrt(keys, map, middle + 1, right);
           merge(keys, map, left, middle, right);
       }
   }
```

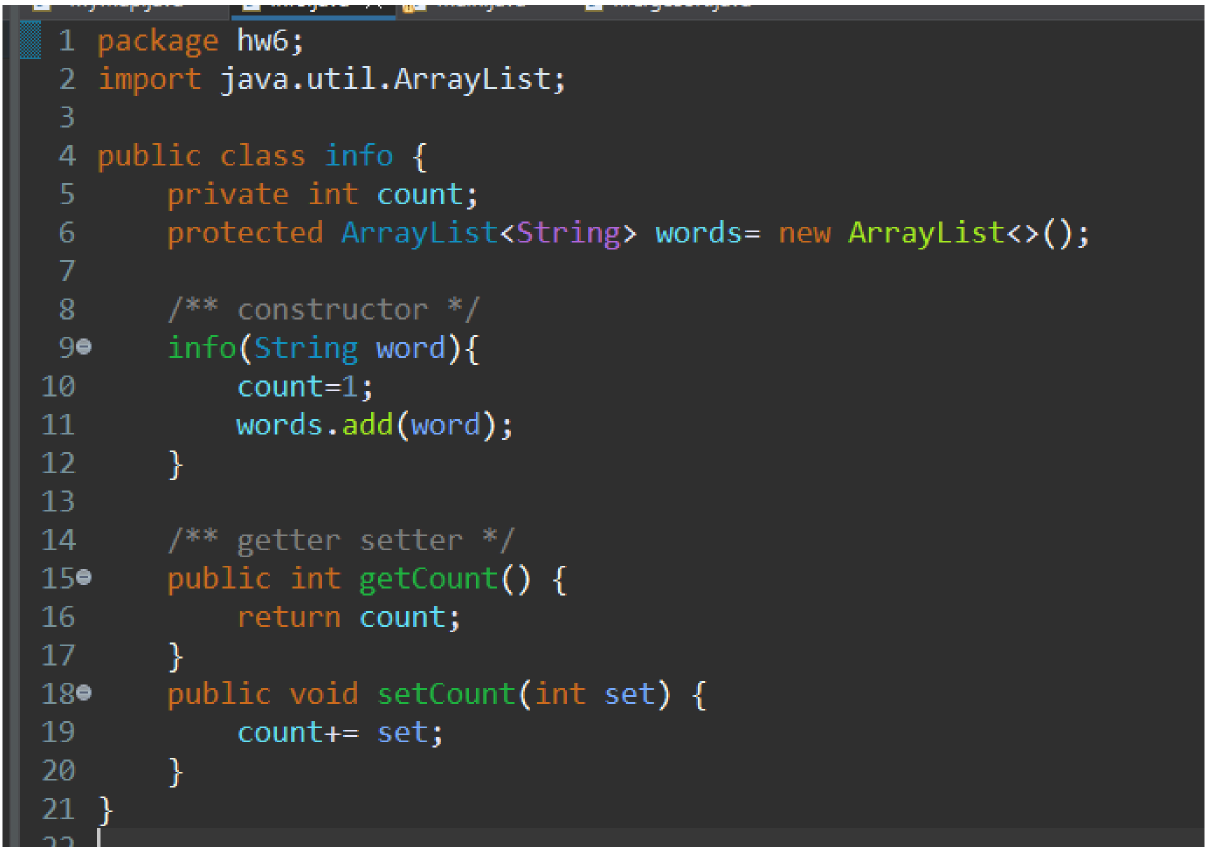
mergeSort(): This function is the main function that applies the merge sort algorithm on the array. It splits the array into two parts and applies merge sort again on both parts. Finally, it merges both parts with the merge() function.



```
public static void merge(String[] keys, LinkedHashMap<String, info> map, int left, int middle, int right) {

    int i = left, j = middle + 1;

    while (i <= middle && j <= right) {
        if (map.get(keys[i]).getCount() <= map.get(keys[j]).getCount()) {
            i++;
        } else {
            String tempKey = keys[j];
            int tempIndex = j;

            while (tempIndex > i) {
                keys[tempIndex] = keys[tempIndex - 1];
                tempIndex--;
            }
            keys[i] = tempKey;
            i++;
            middle++;
            j++;
        }
    }
}
```

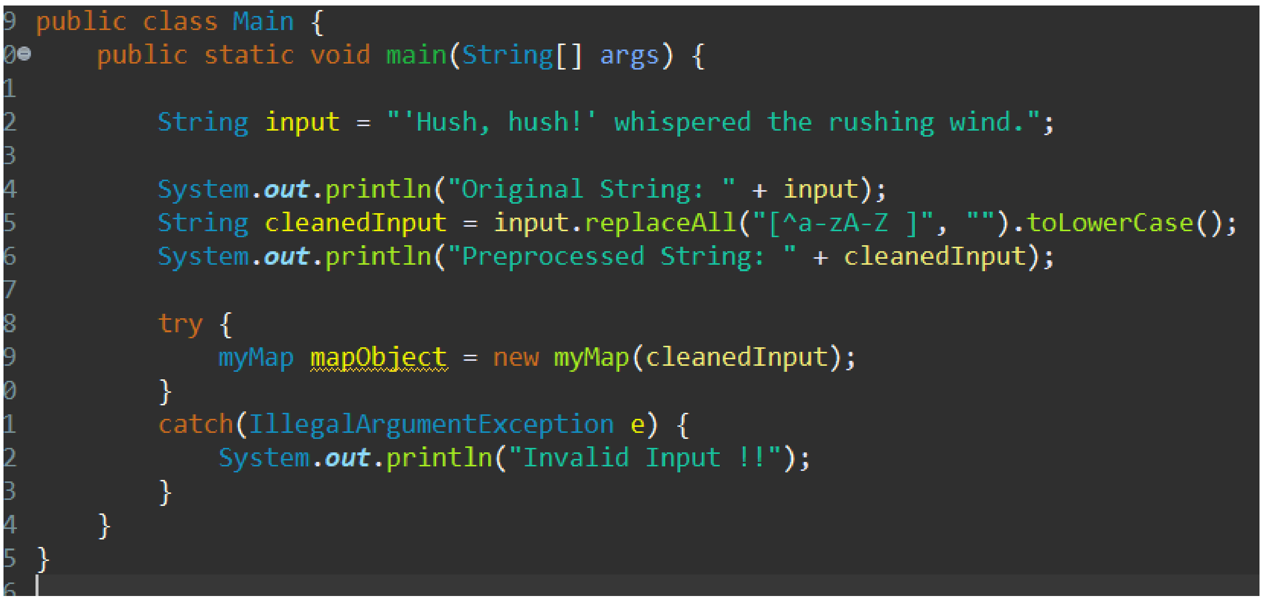
We define the initial indices of the left and right subarrays by defining the variables i and j. i represents the left subarray and j represents the right subarray.
By iterating over the left array (i) and the right array (j), we compare the elements in both subarrays.
If the count value of the element in the left subarray is less than or equal to the count value of the element in the right subarray, we increment i because the element in the left subarray is already in the correct order.
If the count of the element in the left subarray is greater than the count of the element in the right subarray, we move the element in the right subarray to the appropriate position:We define two temporary variables, tempKey and tempIndex, to store the value and index of the element in the right sub-array.
By iterating tempIndex backwards from the beginning to i, we move the elements in the keys array one step to the right. Finally, by assigning the value tempKey to keys[i], we place the element in the right subarray in the appropriate position.
Once we are done placing the elements in the correct position, we continue processing the next elements in the left and right sub-arrays by incrementing i, middle and j by 1 each.
When all elements have been compared and sorted, the merge() function completes and the keys array becomes sorted.In this method, we do the sorting directly on the keys array without the need for additional arrays.

info Class



```java
package hw6;
import java.util.ArrayList;

public class info {
    private int count;
    protected ArrayList<String> words= new ArrayList<>();

    /** constructor */
    info(String word){
        count=1;
        words.add(word);
    }

    /** getter setter */
    public int getCount() {
        return count;
    }
    public void setCount(int set) {
        count+= set;
    }
}
```

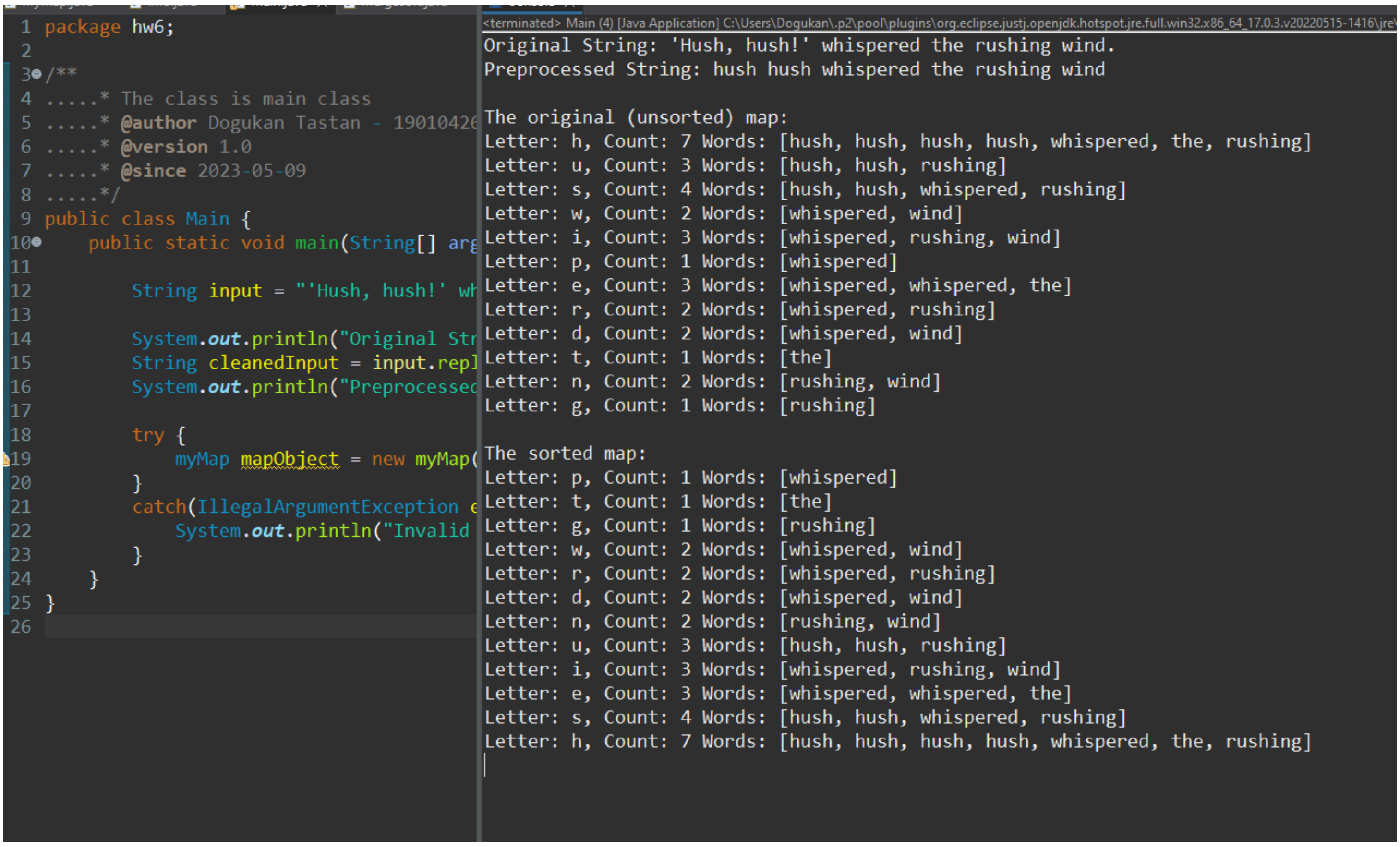
Main Class



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

        String input = "'Hush, hush!' whispered the rushing wind.";

        System.out.println("Original String: " + input);
        String cleanedInput = input.replaceAll("[^a-zA-Z ]", "").toLowerCase();
        System.out.println("Preprocessed String: " + cleanedInput);

        try {
            myMap mapObject = new myMap(cleanedInput);
        }
        catch(IllegalArgumentException e) {
            System.out.println("Invalid Input !!");
        }
    }
}
```

The received input is extracted with replaceAll command, converted to lowercase letters and sent to myMap object as cleanedInput.

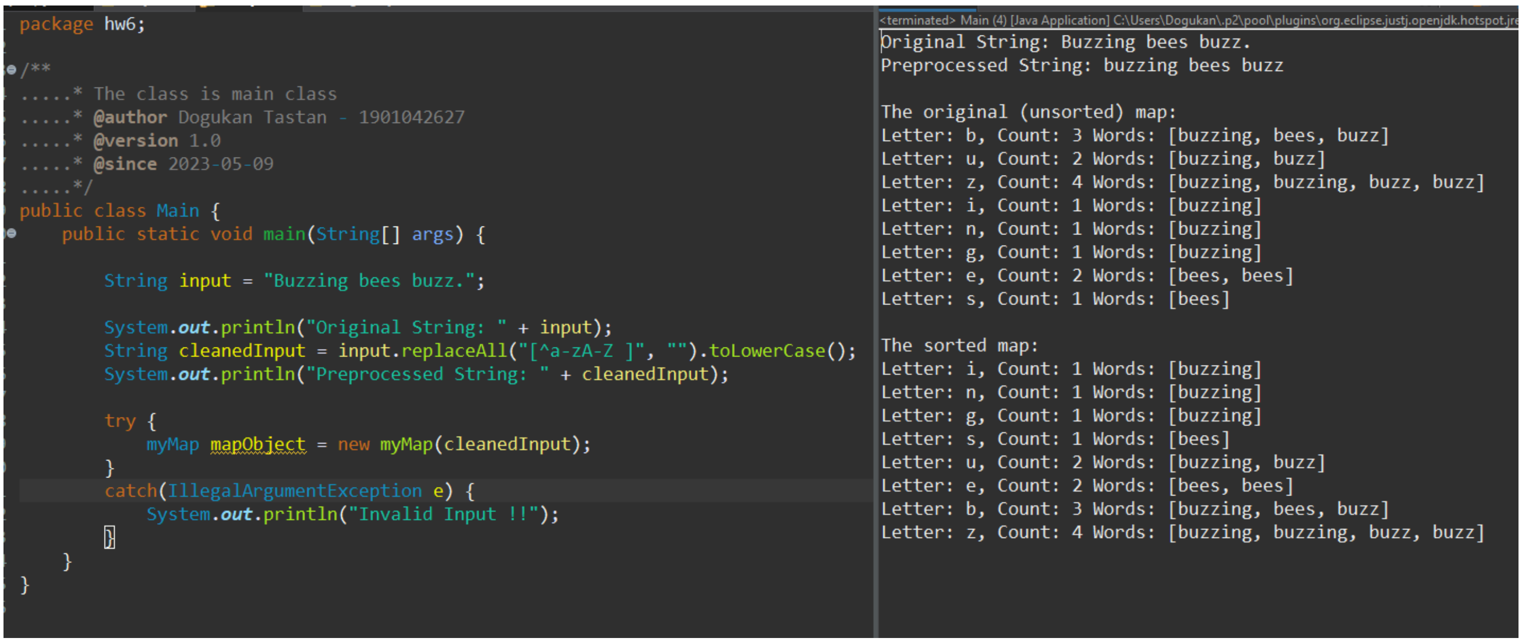
Outputs



```
package hw6;

/**
.....* The class is main class
.....* @author Dogukan Tastan - 19010426
.....* @version 1.0
.....* @since 2023-05-09
.....*/
public class Main {
    public static void main(String[] arg

        String input = "'Hush, hush!' wh

        System.out.println("Original Str
        String cleanedInput = input.repl
        System.out.println("Preprocessed

        try {
            myMap mapObject = new myMap(
        }
        catch(IllegalArgumentException e
            System.out.println("Invalid
        }
    }
}
```

<terminated> Main (4) [Java Application] C:\Users\Dogukan\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20220515-1416\jre\

```
Original String: 'Hush, hush!' whispered the rushing wind.
Preprocessed String: hush hush whispered the rushing wind

The original (unsorted) map:
Letter: h, Count: 7 Words: [hush, hush, hush, hush, whispered, the, rushing]
Letter: u, Count: 3 Words: [hush, hush, rushing]
Letter: s, Count: 4 Words: [hush, hush, whispered, rushing]
Letter: w, Count: 2 Words: [whispered, wind]
Letter: i, Count: 3 Words: [whispered, rushing, wind]
Letter: p, Count: 1 Words: [whispered]
Letter: e, Count: 3 Words: [whispered, whispered, the]
Letter: r, Count: 2 Words: [whispered, rushing]
Letter: d, Count: 2 Words: [whispered, wind]
Letter: t, Count: 1 Words: [the]
Letter: n, Count: 2 Words: [rushing, wind]
Letter: g, Count: 1 Words: [rushing]

The sorted map:
Letter: p, Count: 1 Words: [whispered]
Letter: t, Count: 1 Words: [the]
Letter: g, Count: 1 Words: [rushing]
Letter: w, Count: 2 Words: [whispered, wind]
Letter: r, Count: 2 Words: [whispered, rushing]
Letter: d, Count: 2 Words: [whispered, wind]
Letter: n, Count: 2 Words: [rushing, wind]
Letter: u, Count: 3 Words: [hush, hush, rushing]
Letter: i, Count: 3 Words: [whispered, rushing, wind]
Letter: e, Count: 3 Words: [whispered, whispered, the]
Letter: s, Count: 4 Words: [hush, hush, whispered, rushing]
Letter: h, Count: 7 Words: [hush, hush, hush, hush, whispered, the, rushing]
```



```
package hw6;

/**
.....* The class is main class
.....* @author Dogukan Tastan - 1901042627
.....* @version 1.0
.....* @since 2023-05-09
.....*/
public class Main {
    public static void main(String[] args) {

        String input = "Buzzing bees buzz.";

        System.out.println("Original String: " + input);
        String cleanedInput = input.replaceAll("[^a-zA-Z ]", "").toLowerCase();
        System.out.println("Preprocessed String: " + cleanedInput);

        try {
            myMap mapObject = new myMap(cleanedInput);
        }
        catch(IllegalArgumentException e) {
            System.out.println("Invalid Input !!");
        }
    }
}
```

<terminated> Main (4) [Java Application] C:\Users\Dogukan\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre

```
Original String: Buzzing bees buzz.
Preprocessed String: buzzing bees buzz

The original (unsorted) map:
Letter: b, Count: 3 Words: [buzzing, bees, buzz]
Letter: u, Count: 2 Words: [buzzing, buzz]
Letter: z, Count: 4 Words: [buzzing, buzzing, buzz, buzz]
Letter: i, Count: 1 Words: [buzzing]
Letter: n, Count: 1 Words: [buzzing]
Letter: g, Count: 1 Words: [buzzing]
Letter: e, Count: 2 Words: [bees, bees]
Letter: s, Count: 1 Words: [bees]

The sorted map:
Letter: i, Count: 1 Words: [buzzing]
Letter: n, Count: 1 Words: [buzzing]
Letter: g, Count: 1 Words: [buzzing]
Letter: s, Count: 1 Words: [bees]
Letter: u, Count: 2 Words: [buzzing, buzz]
Letter: e, Count: 2 Words: [bees, bees]
Letter: b, Count: 3 Words: [buzzing, bees, buzz]
Letter: z, Count: 4 Words: [buzzing, buzzing, buzz, buzz]
```