Container Classes

Maps, Sorted Maps

COMP2603
Object Oriented Programming 1

Week 9, Lecture 2

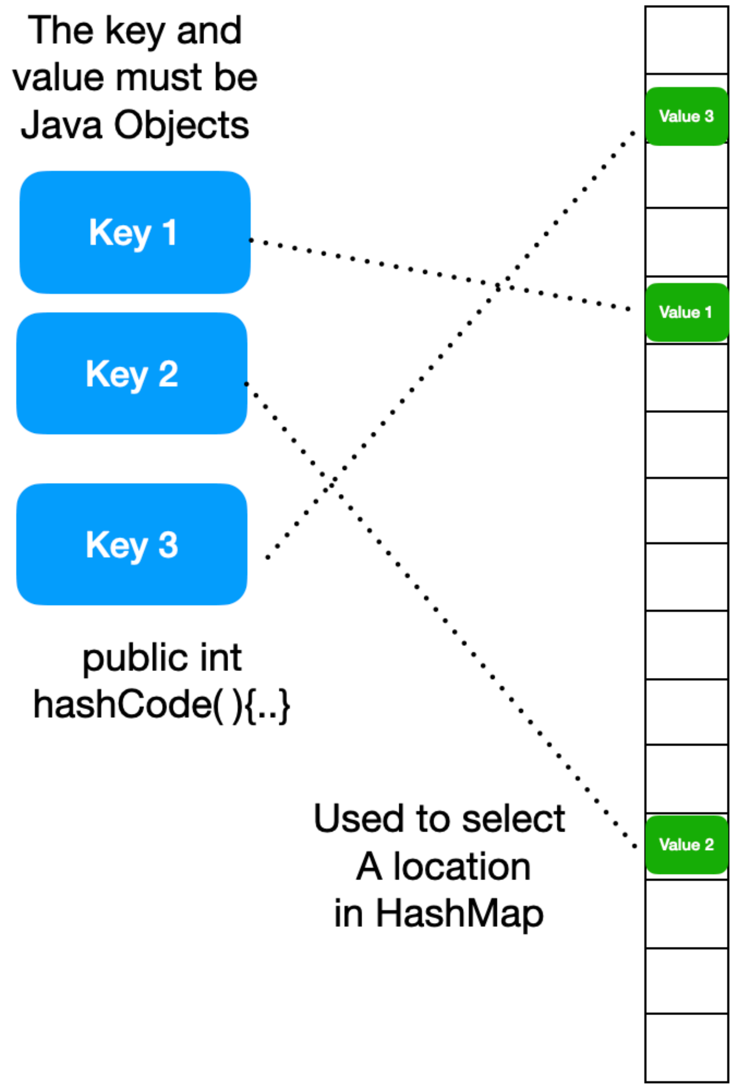
Outline

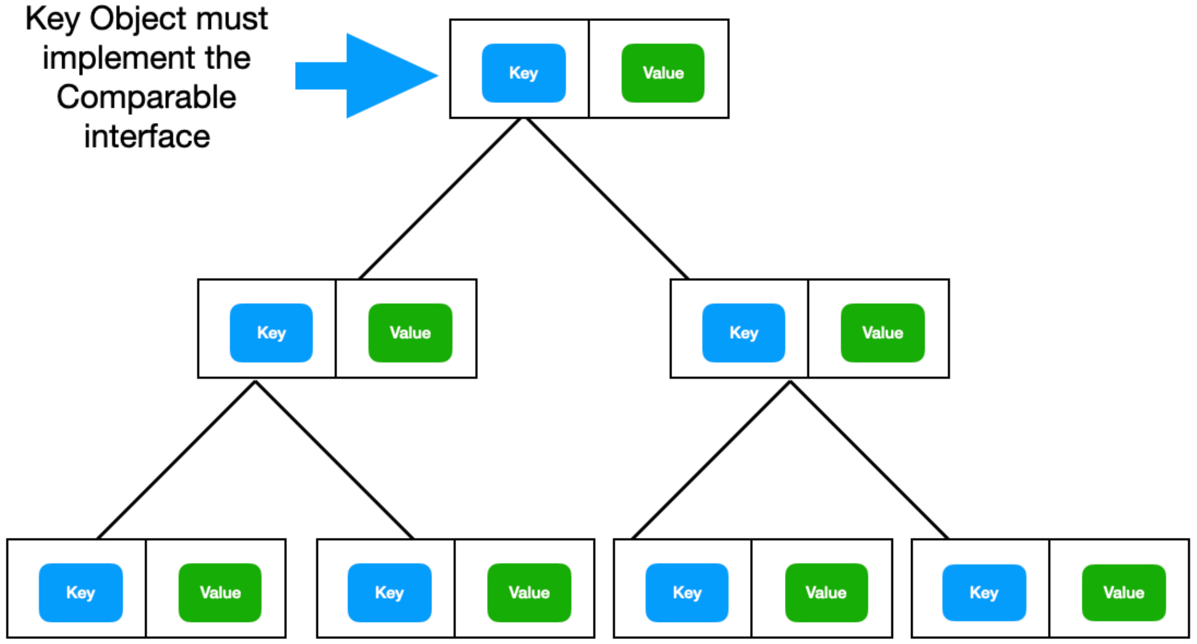
- Java Map Interface
  - Map
    - HashMap
    - TreeMap
  - Finding Objects
  - Removing Objects

HashMap

TreeMap

{ Key, Value }


The key and value must be Java Objects
Key 1
Key 2
Key 3
public int hashCode( ){..}
Used to select A location in HashMap
Value 3
Value 1
Value 2



public int compareTo(Object obj){..} needs to be written in the Key class

public boolean equals(Object obj){..} needs to be overridden and consistent with hashCode( )

public boolean equals(Object obj){..} needs to be overridden and consistent with hashCode( )

The Map Interface

	Method	Description	
Create	V **put**(K key, V value)	Creates a key/value mapping in the Map. If the key already exists in the Map, put( ) replaces the	
Read	V **get** (Object key)	Returns the value object associated with the specified key or null if there is no mapping for the	
Update	boolean **replace**(K key, V oldValue, V newValue)	Replaces the entry for the specified key only if currently mapped to the specified value.	* overloaded
Delete	boolean **remove**(Object key, Object value)	Removes the entry for the specified key only if it is currently mapped to the specified value.	* overloaded

Traverse	Collection<V> values( )	Returns a Collection of all the value objects in the Map
Traverse	Set<E> keySet( )	Returns a Set of all the key objects in the Map

The Map Interface

	Method	Description
Create	V **put**(K key, V value)	Creates a key/value mapping in the Map. If the key already exists in the Map, put( ) replaces the value
Read	V **get** (Object key)	Returns the value object associated with the specified key or null if there is no mapping for the key
Delete	boolean **remove**(Object key, Object value)	Removes the entry for the specified key only if it is currently mapped to the specified value.
	boolean containsKey(Object key)	Returns true if the Map contains a mapping for the specified key, and false otherwise
	Collection<V> values( )	Returns a Collection of all the value objects in the Map
	Set<E> keySet( )	Returns a Set of all the key objects in the Map

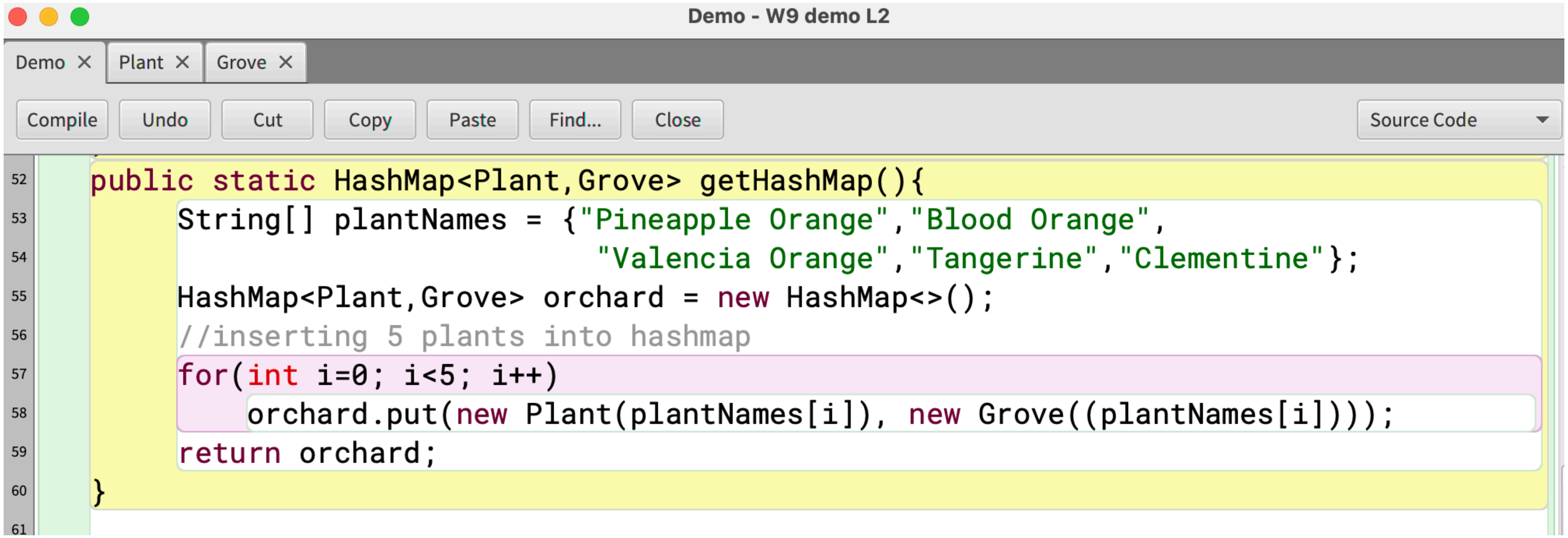
Find a value

In order to find a value mapped to a key:

(1) Check if the key is contained in the map

(2) If found then use the key to get the value mapped in the collection and then return the value.

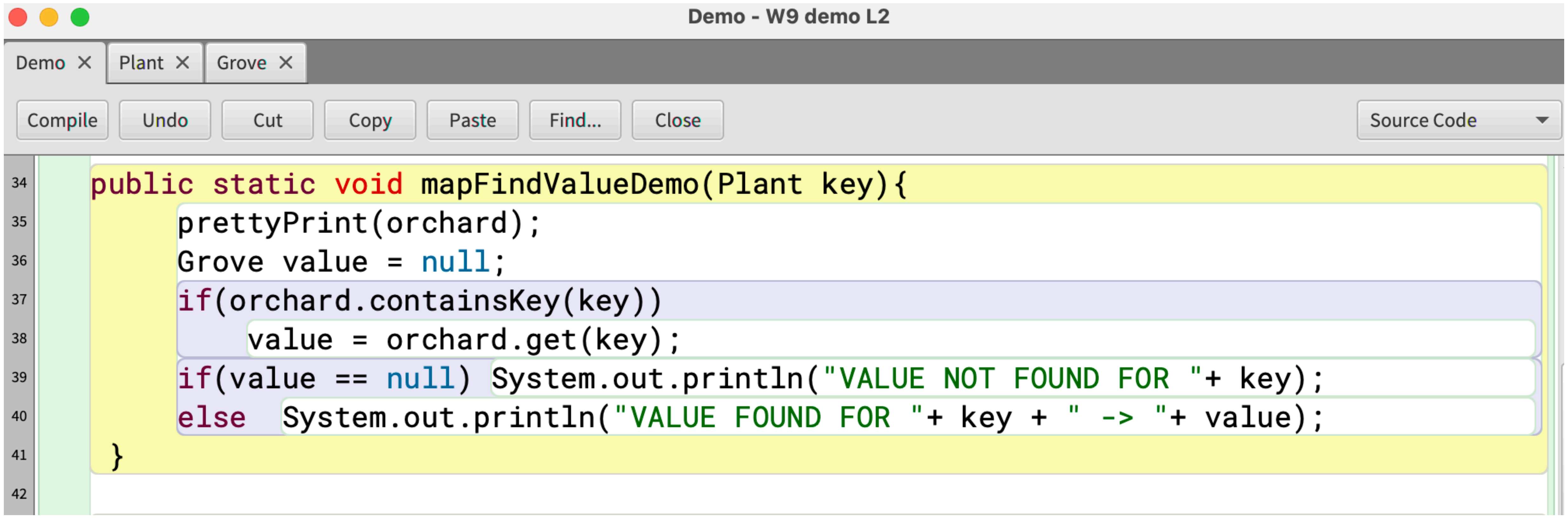
Example 1- HashMap



```java
public static HashMap<Plant,Grove> getHashMap(){
     String[] plantNames = {"Pineapple Orange","Blood Orange",
                            "Valencia Orange","Tangerine","Clementine"};
     HashMap<Plant,Grove> orchard = new HashMap<>();
     //inserting 5 plants into hashmap
     for(int i=0; i<5; i++)
         orchard.put(new Plant(plantNames[i]), new Grove((plantNames[i])));
     return orchard;
}
```

Suppose there is a method that loads a HashMap with Plant objects in our Demo class

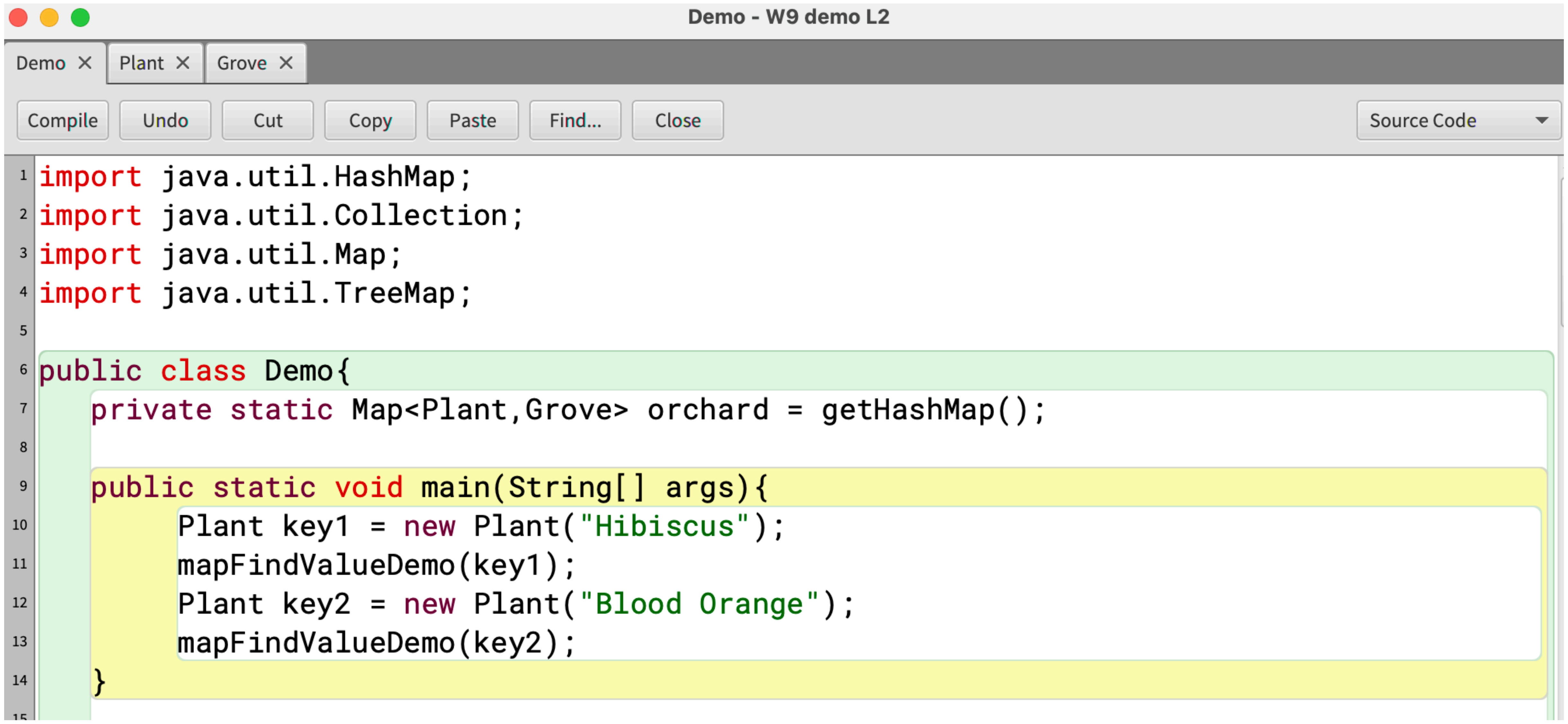
Example 1- HashMap



```
public static void mapFindValueDemo(Plant key){
        prettyPrint(orchard);
        Grove value = null;
        if(orchard.containsKey(key))
            value = orchard.get(key);
        if(value == null) System.out.println("VALUE NOT FOUND FOR "+ key);
        else  System.out.println("VALUE FOUND FOR "+ key + " -> "+ value);
 }
```

This method locates a value given a key

Example 1- HashMap



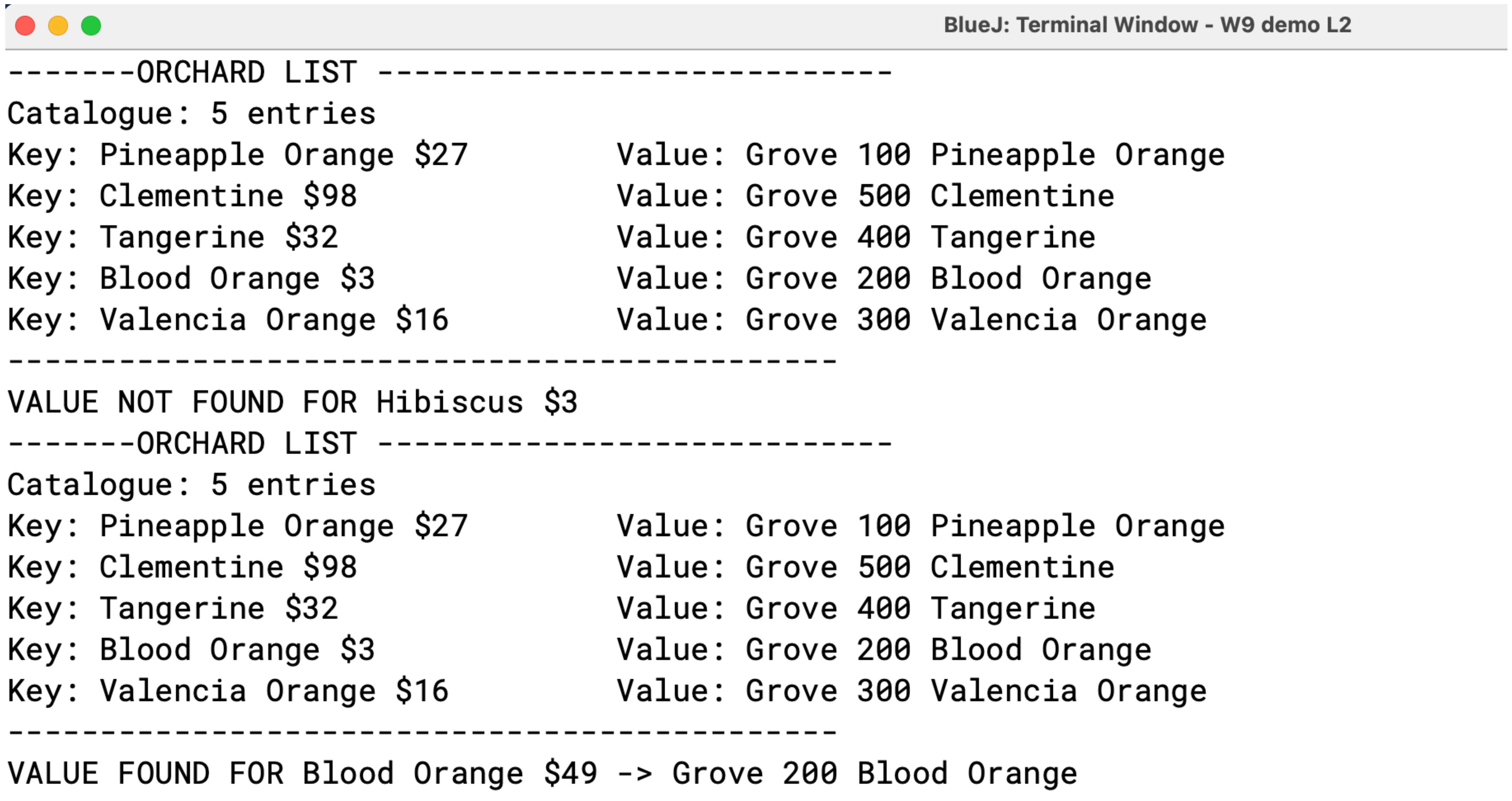
```java
import java.util.HashMap;
import java.util.Collection;
import java.util.Map;
import java.util.TreeMap;

public class Demo{
    private static Map<Plant,Grove> orchard = getHashMap();

    public static void main(String[] args){
        Plant key1 = new Plant("Hibiscus");
        mapFindValueDemo(key1);
        Plant key2 = new Plant("Blood Orange");
        mapFindValueDemo(key2);
    }
```

Supplying two keys

Example 1- HashMap

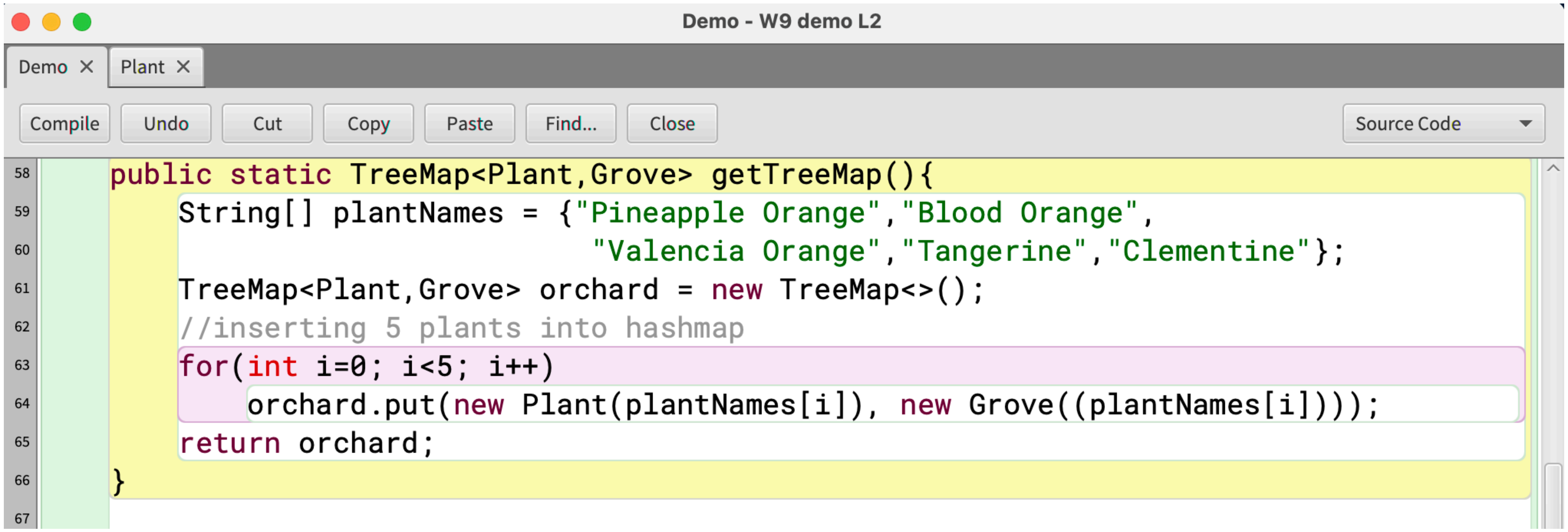

BlueJ: Terminal Window - W9 demo L2

```
-------ORCHARD LIST ----------------------------
Catalogue: 5 entries
Key: Pineapple Orange $27        Value: Grove 100 Pineapple Orange
Key: Clementine $98              Value: Grove 500 Clementine
Key: Tangerine $32               Value: Grove 400 Tangerine
Key: Blood Orange $3             Value: Grove 200 Blood Orange
Key: Valencia Orange $16         Value: Grove 300 Valencia Orange
---------------------------------------------
VALUE NOT FOUND FOR Hibiscus $3
-------ORCHARD LIST ----------------------------
Catalogue: 5 entries
Key: Pineapple Orange $27        Value: Grove 100 Pineapple Orange
Key: Clementine $98              Value: Grove 500 Clementine
Key: Tangerine $32               Value: Grove 400 Tangerine
Key: Blood Orange $3             Value: Grove 200 Blood Orange
Key: Valencia Orange $16         Value: Grove 300 Valencia Orange
---------------------------------------------
VALUE FOUND FOR Blood Orange $49 -> Grove 200 Blood Orange
```

To get a value, a key must be supplied. The first key was not found, so no value. The second key one was found so the value was returned.

Observe that the second key has a different price compared to the one stored in the HashMap but it still was used successfully to locate the value (Grove 700). This happens because plant name alone is used for Plant equality.

Example 2- TreeMap

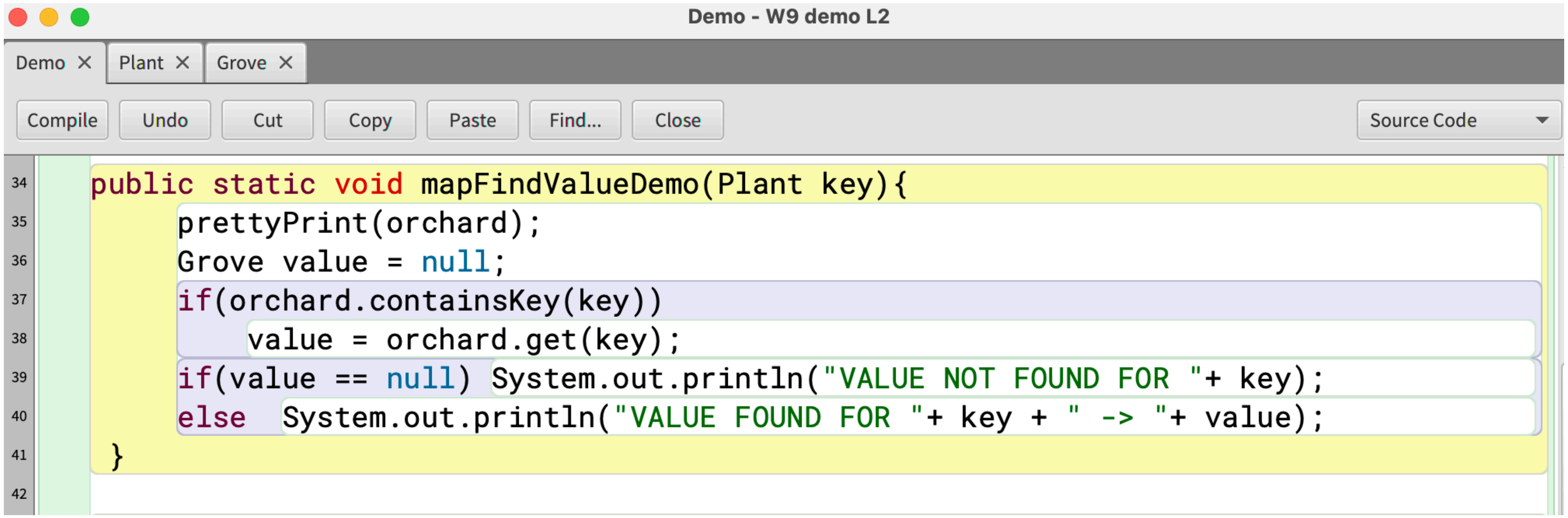


```java
public static TreeMap<Plant,Grove> getTreeMap(){
    String[] plantNames = {"Pineapple Orange","Blood Orange",
                           "Valencia Orange","Tangerine","Clementine"};
    TreeMap<Plant,Grove> orchard = new TreeMap<>();
    //inserting 5 plants into hashmap
    for(int i=0; i<5; i++)
        orchard.put(new Plant(plantNames[i]), new Grove((plantNames[i])));
    return orchard;
}
```

Suppose there is a method that loads a TreeMap with Plant objects in our Demo class.

The Plant class is Comparable (refer to end of Week 9 Lecture 1 slide deck)

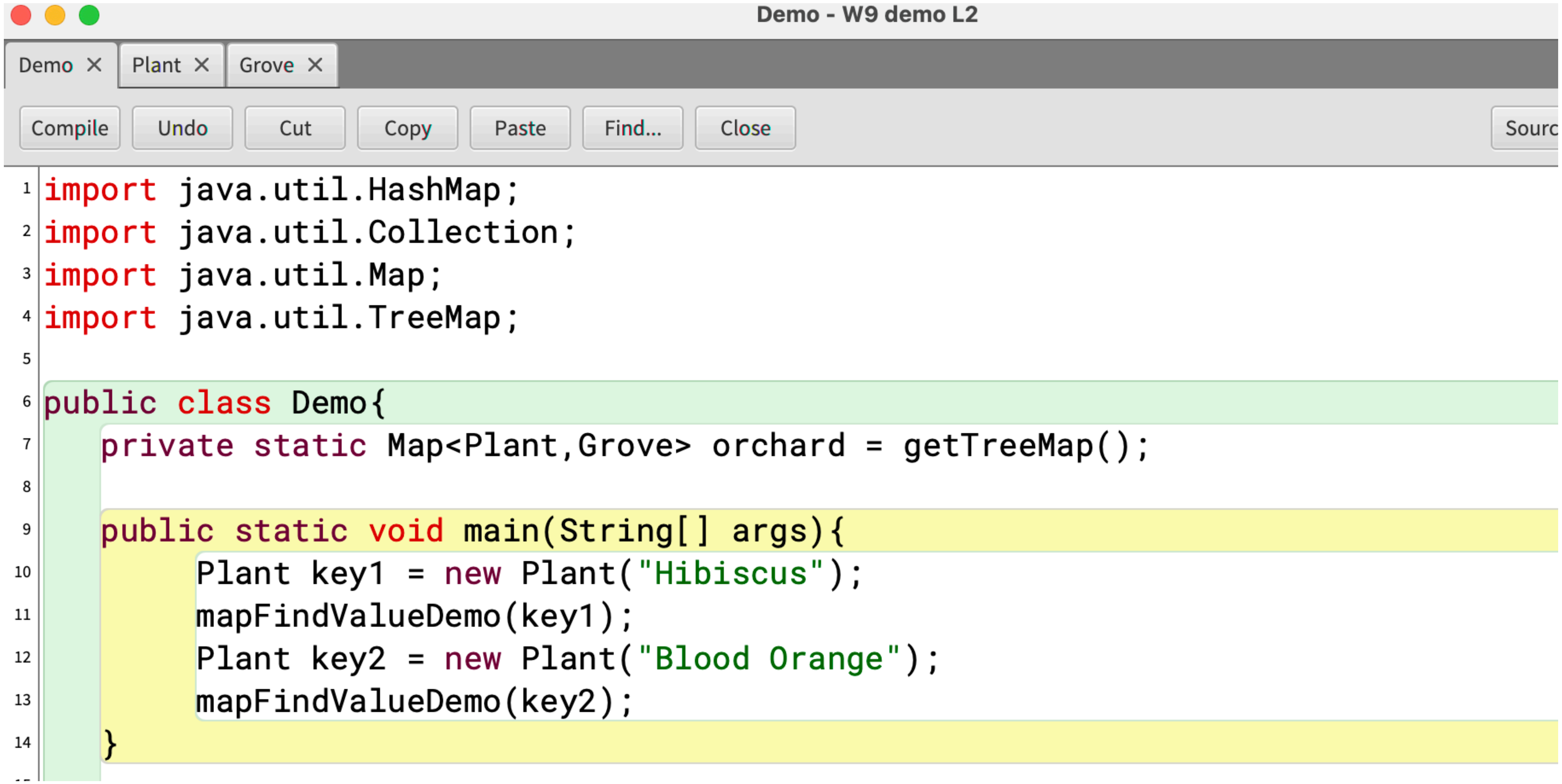
Example 2- TreeMap



```java
public static void mapFindValueDemo(Plant key){
        prettyPrint(orchard);
        Grove value = null;
        if(orchard.containsKey(key))
            value = orchard.get(key);
        if(value == null) System.out.println("VALUE NOT FOUND FOR "+ key);
        else  System.out.println("VALUE FOUND FOR "+ key + " -> "+ value);
  }
```

This method locates a value given a key (exactly the same as in Example 1)

Example 2- TreeMap



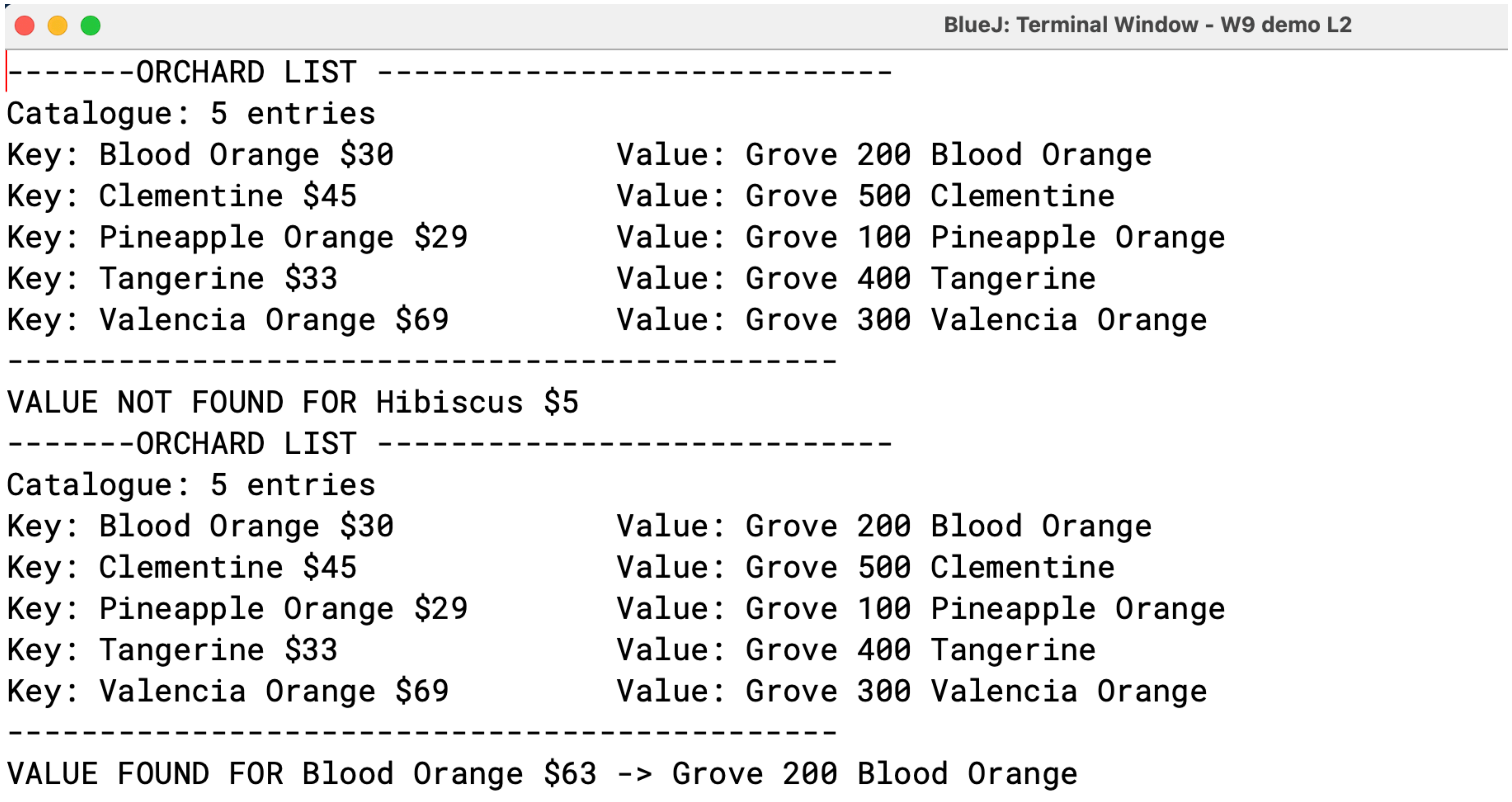
```java
import java.util.HashMap;
import java.util.Collection;
import java.util.Map;
import java.util.TreeMap;

public class Demo{
    private static Map<Plant,Grove> orchard = getTreeMap();

    public static void main(String[] args){
        Plant key1 = new Plant("Hibiscus");
        mapFindValueDemo(key1);
        Plant key2 = new Plant("Blood Orange");
        mapFindValueDemo(key2);
    }
```

Supplying two keys (as before) but with the TreeMap

Example 2- TreeMap


BlueJ: Terminal Window - W9 demo L2

```
-------ORCHARD LIST ----------------------------
Catalogue: 5 entries
Key: Blood Orange $30            Value: Grove 200 Blood Orange
Key: Clementine $45              Value: Grove 500 Clementine
Key: Pineapple Orange $29        Value: Grove 100 Pineapple Orange
Key: Tangerine $33               Value: Grove 400 Tangerine
Key: Valencia Orange $69         Value: Grove 300 Valencia Orange
---------------------------------------------
VALUE NOT FOUND FOR Hibiscus $5
-------ORCHARD LIST ----------------------------
Catalogue: 5 entries
Key: Blood Orange $30            Value: Grove 200 Blood Orange
Key: Clementine $45              Value: Grove 500 Clementine
Key: Pineapple Orange $29        Value: Grove 100 Pineapple Orange
Key: Tangerine $33               Value: Grove 400 Tangerine
Key: Valencia Orange $69         Value: Grove 300 Valencia Orange
---------------------------------------------
VALUE FOUND FOR Blood Orange $63 -> Grove 200 Blood Orange
```

Consistent results as before. One key not found so no value, the other key found and the value returned. Note the sorted data.

Find a key

In order to find a value mapped to a key:

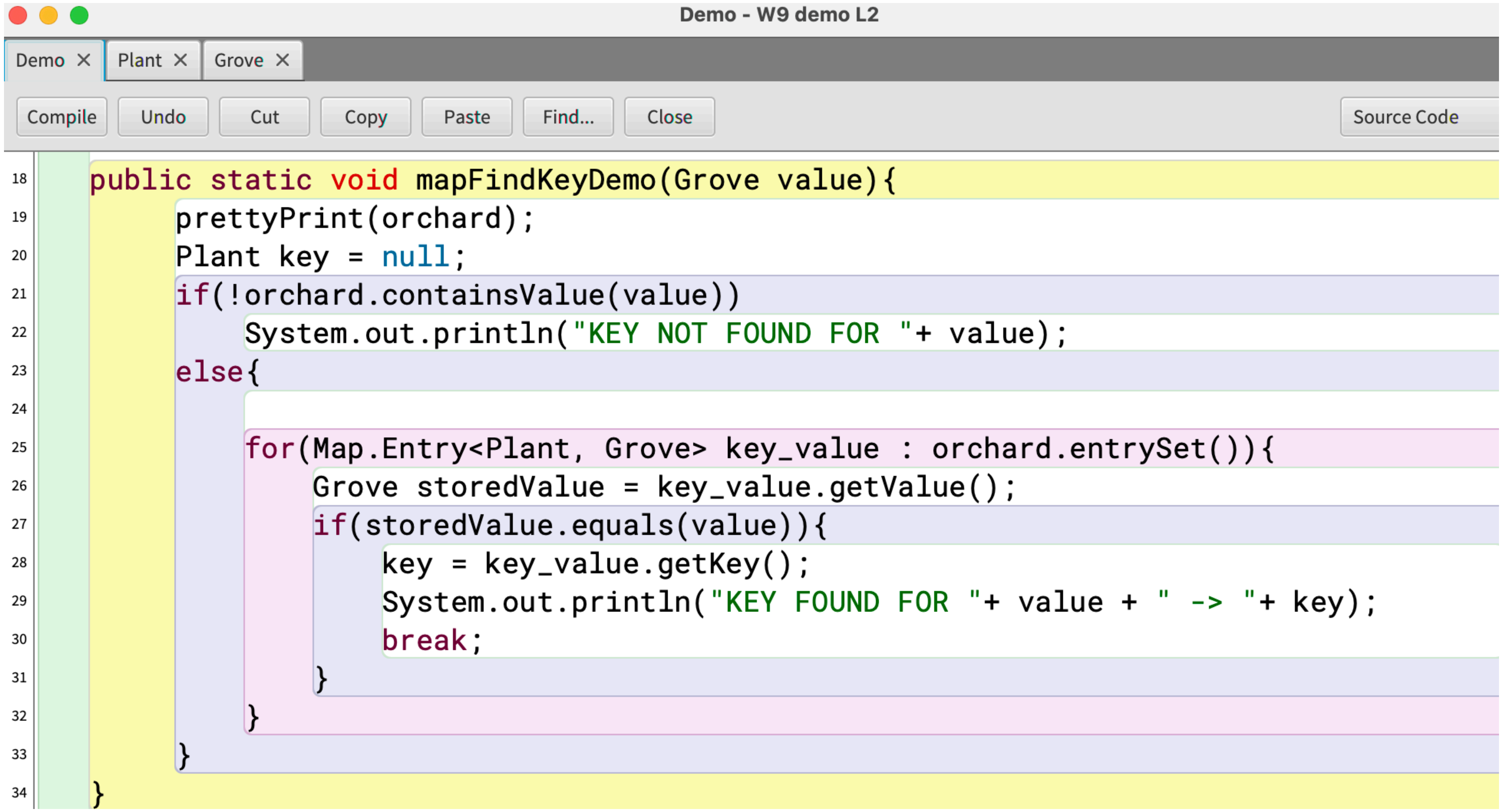
(1) Check if the value is contained in the map

(2) Retrieve the Set of key-value pairs from the map using entrySet( )

(3) Check each value against the supplied value

(4) If found then the key mapped to the value in the collection is retrieved from the key-value Entry.

Example 3- HashMap

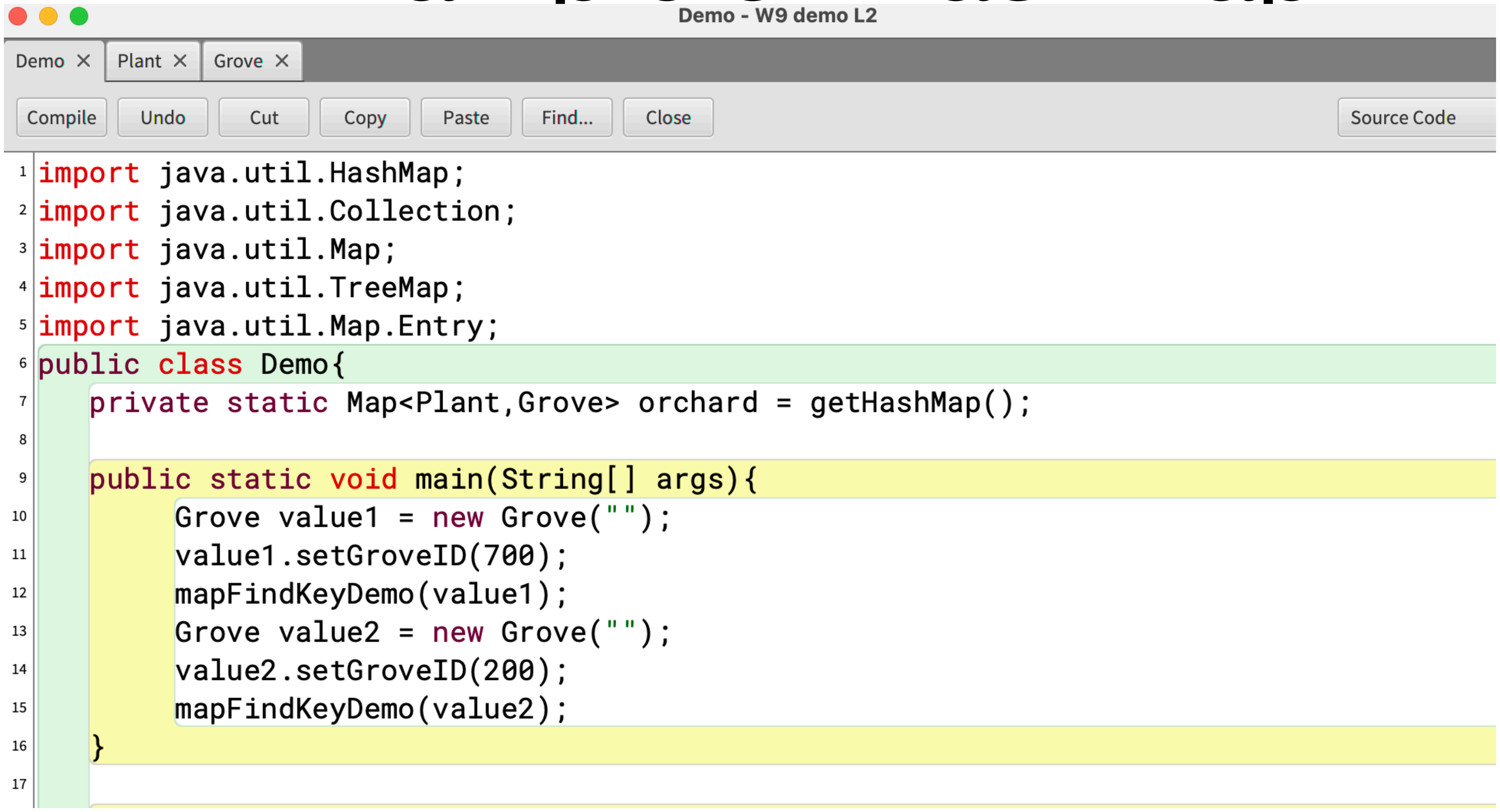


```java
public static void mapFindKeyDemo(Grove value){
    prettyPrint(orchard);
    Plant key = null;
    if(!orchard.containsValue(value))
        System.out.println("KEY NOT FOUND FOR "+ value);
    else{

        for(Map.Entry<Plant, Grove> key_value : orchard.entrySet()){
            Grove storedValue = key_value.getValue();
            if(storedValue.equals(value)){
                key = key_value.getKey();
                System.out.println("KEY FOUND FOR "+ value + " -> "+ key);
                break;
            }
        }
    }
}
```

This method locates a key given a value

Example 3- HashMap



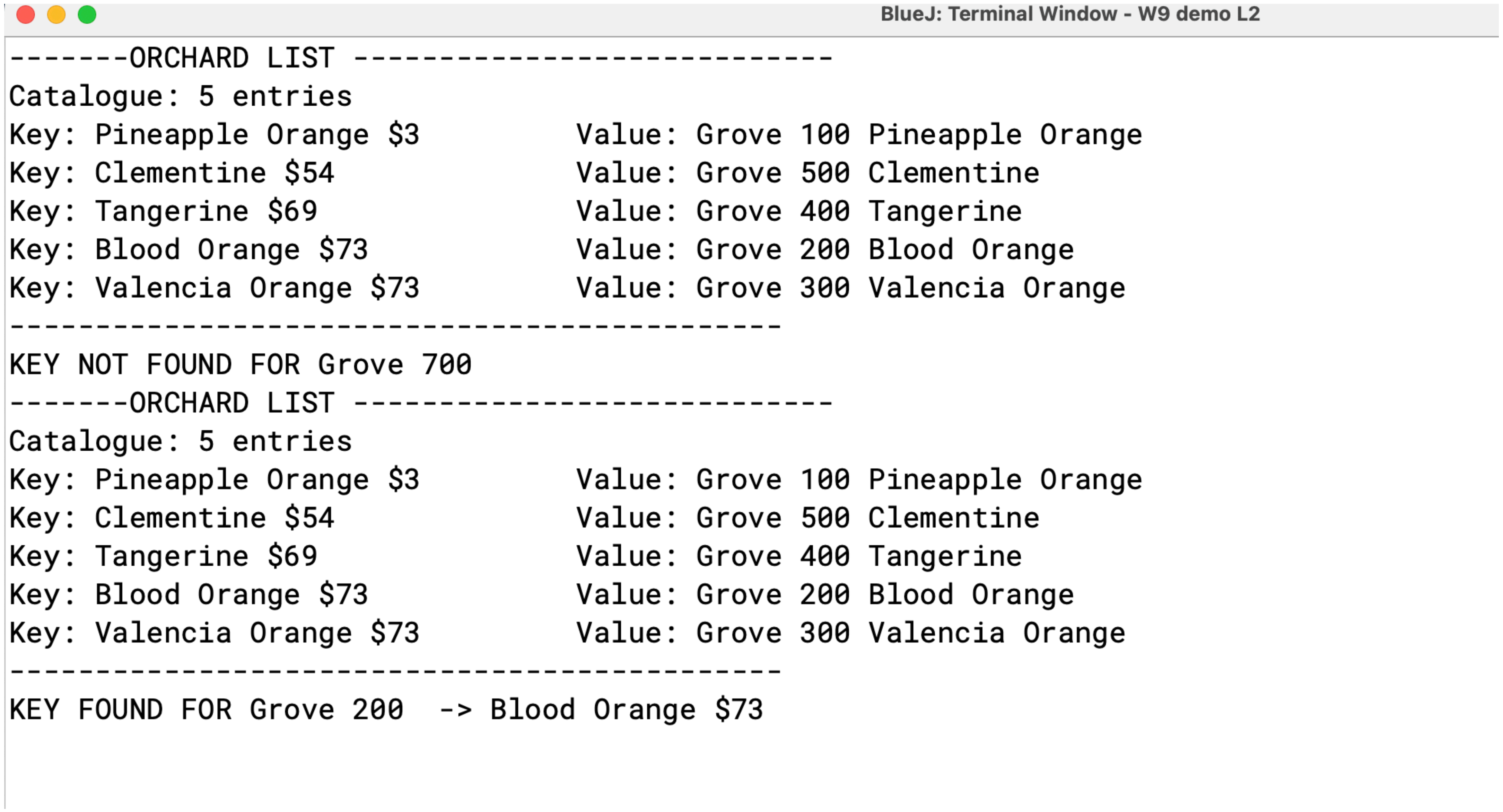
```
import java.util.HashMap;
import java.util.Collection;
import java.util.Map;
import java.util.TreeMap;
import java.util.Map.Entry;
public class Demo{
    private static Map<Plant,Grove> orchard = getHashMap();

    public static void main(String[] args){
        Grove value1 = new Grove("");
        value1.setGroveID(700);
        mapFindKeyDemo(value1);
        Grove value2 = new Grove("");
        value2.setGroveID(200);
        mapFindKeyDemo(value2);
    }

```

Using the method on Slide #7 to load the HashMap

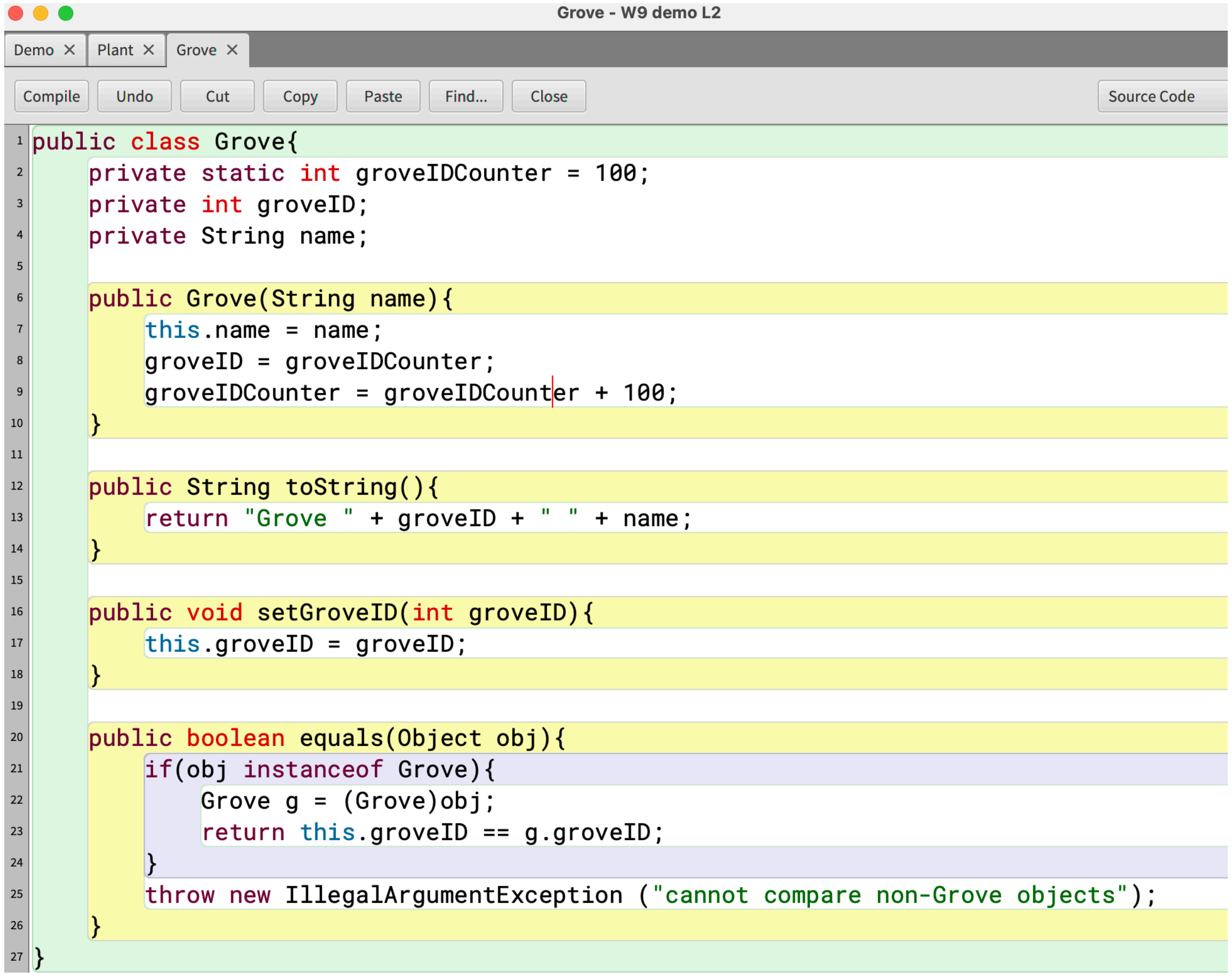
Example 3- HashMap



```
-------ORCHARD LIST ---------------------------
Catalogue: 5 entries
Key: Pineapple Orange $3         Value: Grove 100 Pineapple Orange
Key: Clementine $54              Value: Grove 500 Clementine
Key: Tangerine $69               Value: Grove 400 Tangerine
Key: Blood Orange $73            Value: Grove 200 Blood Orange
Key: Valencia Orange $73         Value: Grove 300 Valencia Orange
---------------------------------------------
KEY NOT FOUND FOR Grove 700
-------ORCHARD LIST ---------------------------
Catalogue: 5 entries
Key: Pineapple Orange $3         Value: Grove 100 Pineapple Orange
Key: Clementine $54              Value: Grove 500 Clementine
Key: Tangerine $69               Value: Grove 400 Tangerine
Key: Blood Orange $73            Value: Grove 200 Blood Orange
Key: Valencia Orange $73         Value: Grove 300 Valencia Orange
---------------------------------------------
KEY FOUND FOR Grove 200  -> Blood Orange $73
```

One key not found, one was found

Example 3- HashMap



```java
public class Grove{
    private static int groveIDCounter = 100;
    private int groveID;
    private String name;

    public Grove(String name){
        this.name = name;
        groveID = groveIDCounter;
        groveIDCounter = groveIDCounter + 100;
    }

    public String toString(){
        return "Grove " + groveID + " " + name;
    }

    public void setGroveID(int groveID){
        this.groveID = groveID;
    }

    public boolean equals(Object obj){
        if(obj instanceof Grove){
            Grove g = (Grove)obj;
            return this.groveID == g.groveID;
        }
        throw new IllegalArgumentException ("cannot compare non-Grove objects");
    }
}
```

The Grove class has an equals( ) and a mutator for ID that allow us to create Grove objects as ‘queries’

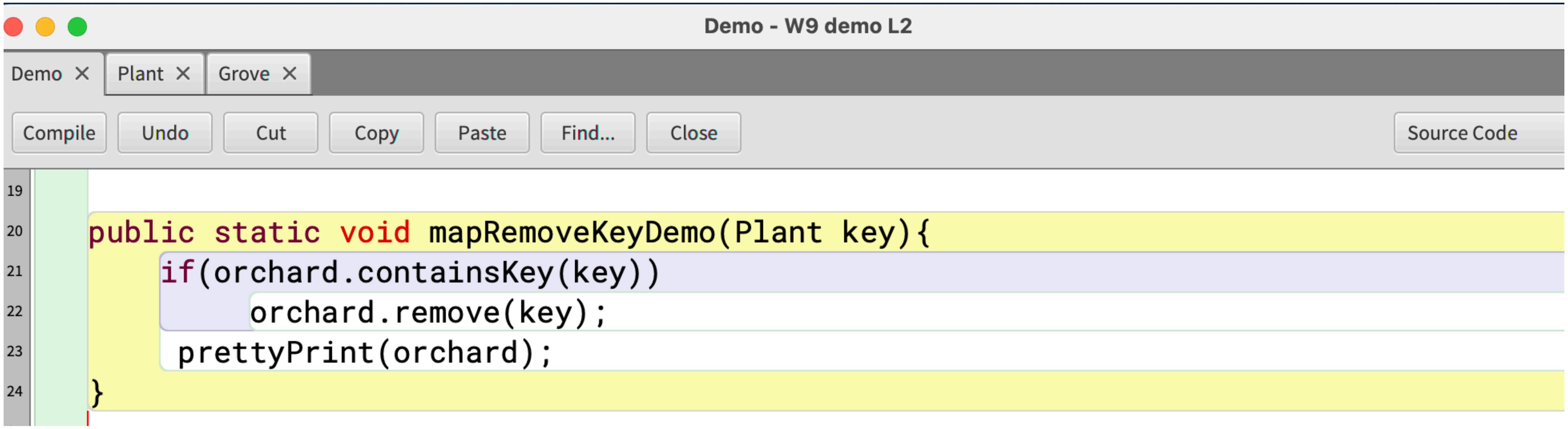
Remove a key

In order to remove a key:

(1) Check that the map contains the key

(2) Remove the key

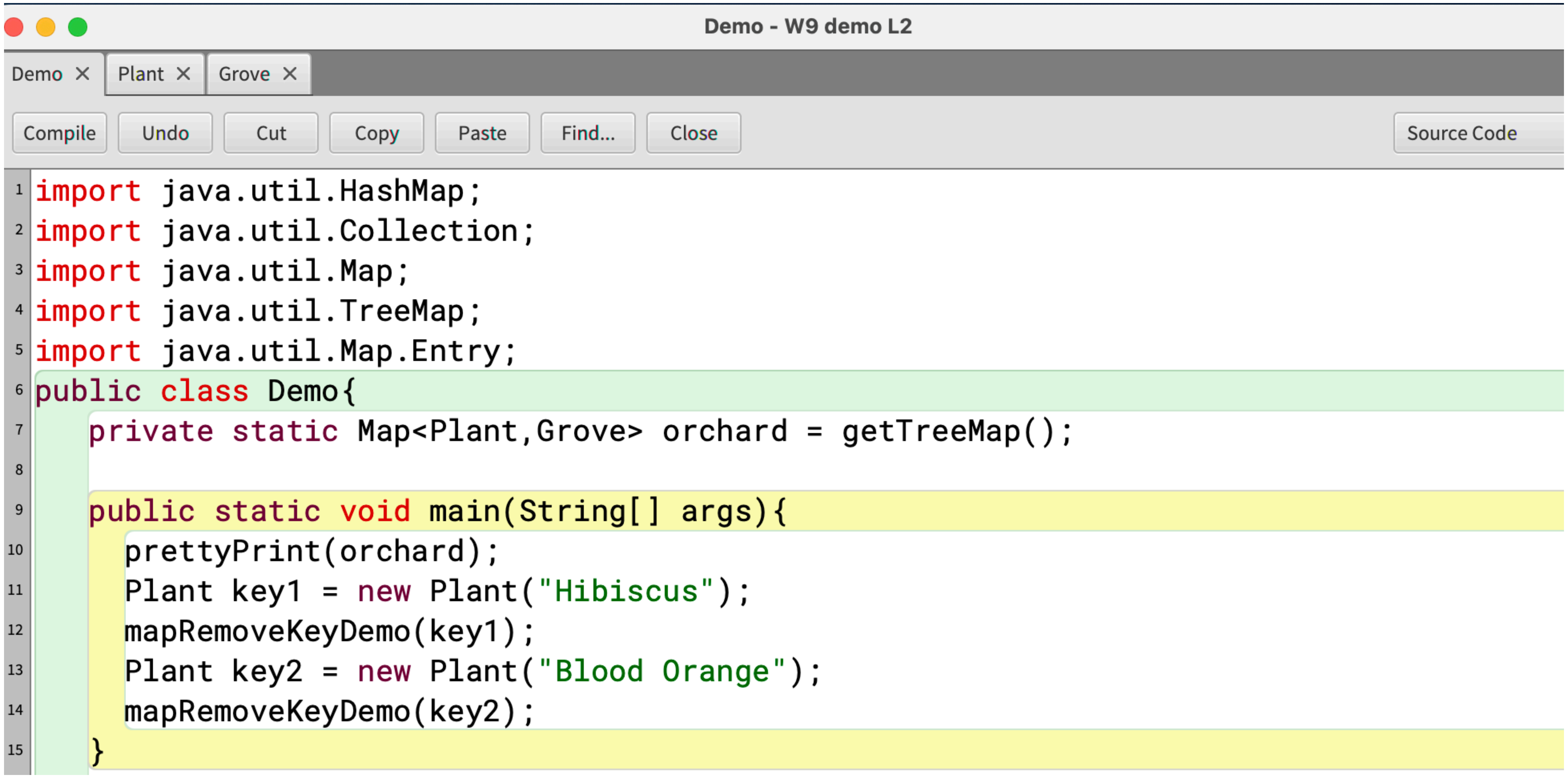
Example 4- TreeMap



```java

public static void mapRemoveKeyDemo(Plant key){
    if(orchard.containsKey(key))
        orchard.remove(key);
    prettyPrint(orchard);
}
```

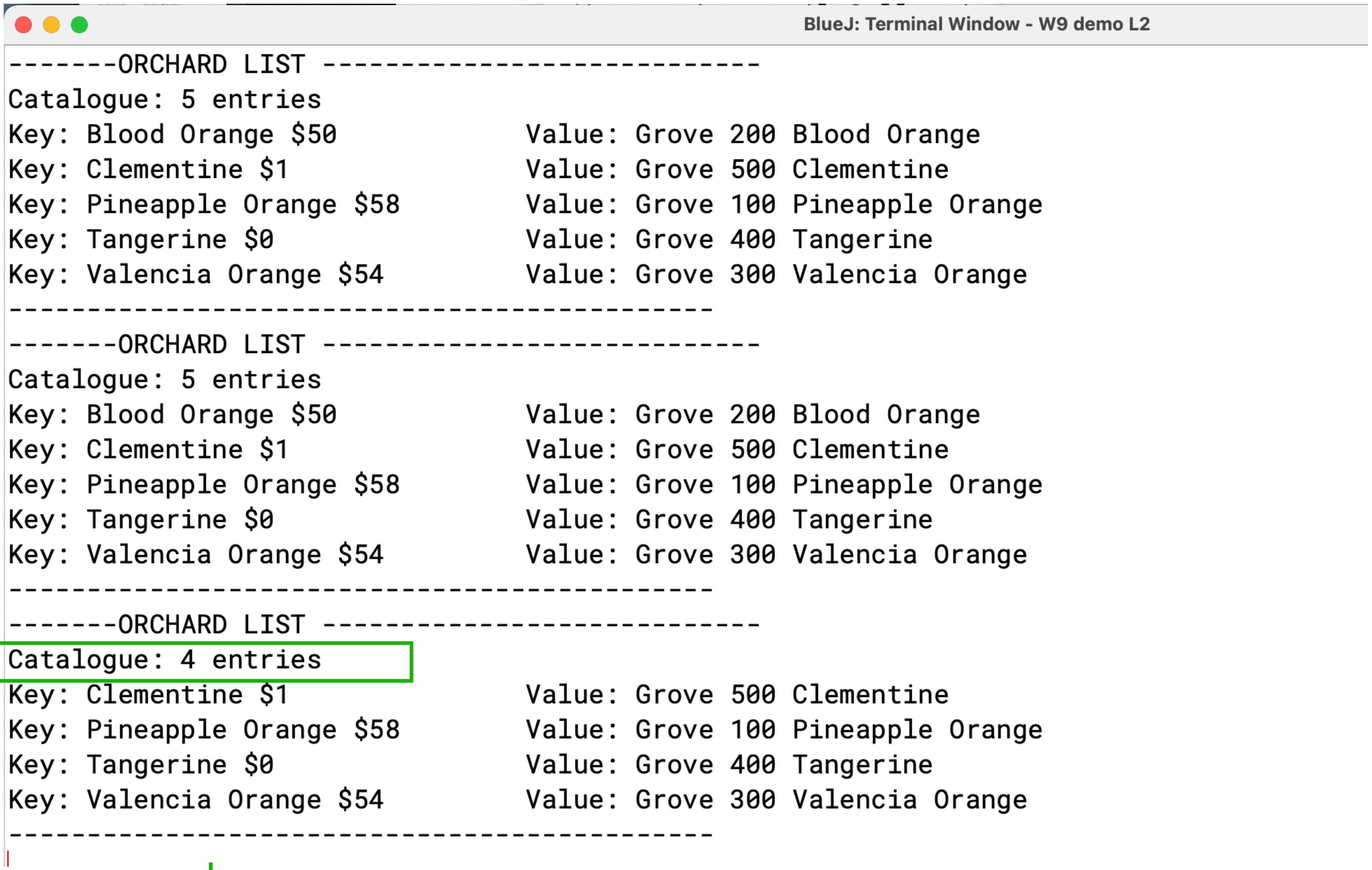
Example 4- TreeMap



```java
import java.util.HashMap;
import java.util.Collection;
import java.util.Map;
import java.util.TreeMap;
import java.util.Map.Entry;
public class Demo{
    private static Map<Plant,Grove> orchard = getTreeMap();

    public static void main(String[] args){
        prettyPrint(orchard);
        Plant key1 = new Plant("Hibiscus");
        mapRemoveKeyDemo(key1);
        Plant key2 = new Plant("Blood Orange");
        mapRemoveKeyDemo(key2);
    }
```

Example 4- TreeMap



```
-------ORCHARD LIST ---------------------------
Catalogue: 5 entries
Key: Blood Orange $50           Value: Grove 200 Blood Orange
Key: Clementine $1              Value: Grove 500 Clementine
Key: Pineapple Orange $58       Value: Grove 100 Pineapple Orange
Key: Tangerine $0               Value: Grove 400 Tangerine
Key: Valencia Orange $54        Value: Grove 300 Valencia Orange
--------------------------------------------
-------ORCHARD LIST ---------------------------
Catalogue: 5 entries
Key: Blood Orange $50           Value: Grove 200 Blood Orange
Key: Clementine $1              Value: Grove 500 Clementine
Key: Pineapple Orange $58       Value: Grove 100 Pineapple Orange
Key: Tangerine $0               Value: Grove 400 Tangerine
Key: Valencia Orange $54        Value: Grove 300 Valencia Orange
--------------------------------------------
-------ORCHARD LIST ---------------------------
Catalogue: 4 entries
Key: Clementine $1              Value: Grove 500 Clementine
Key: Pineapple Orange $58       Value: Grove 100 Pineapple Orange
Key: Tangerine $0               Value: Grove 400 Tangerine
Key: Valencia Orange $54        Value: Grove 300 Valencia Orange
--------------------------------------------
```

Key removed

Remove a value

In order to remove a value:

(1) Check that the value is stored in the map

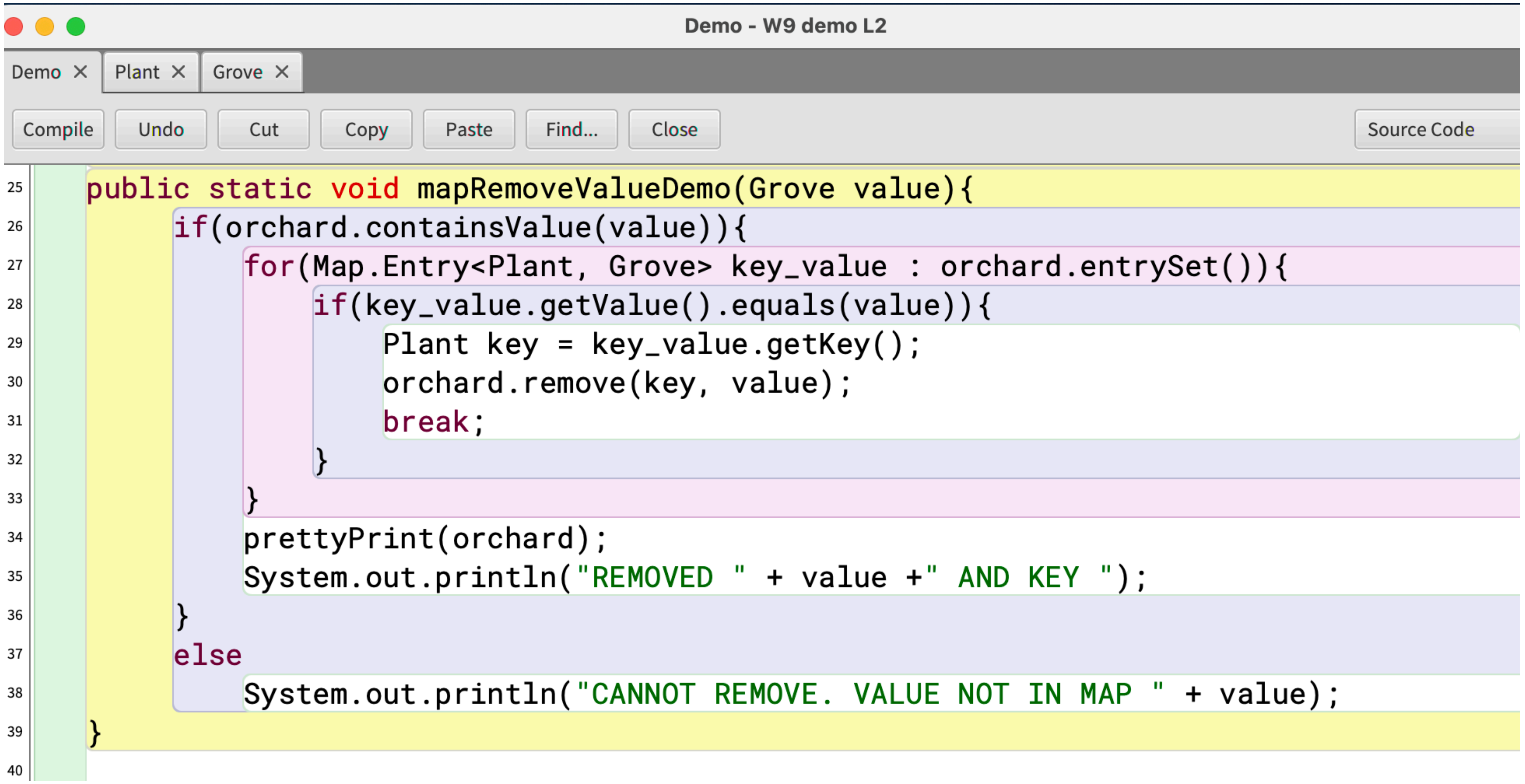
(2) Retrieve the set of key-value pairs

(3) Locate the value in the pairs that matches the one to be removed

(4) Retrieve the key for that value from the pair

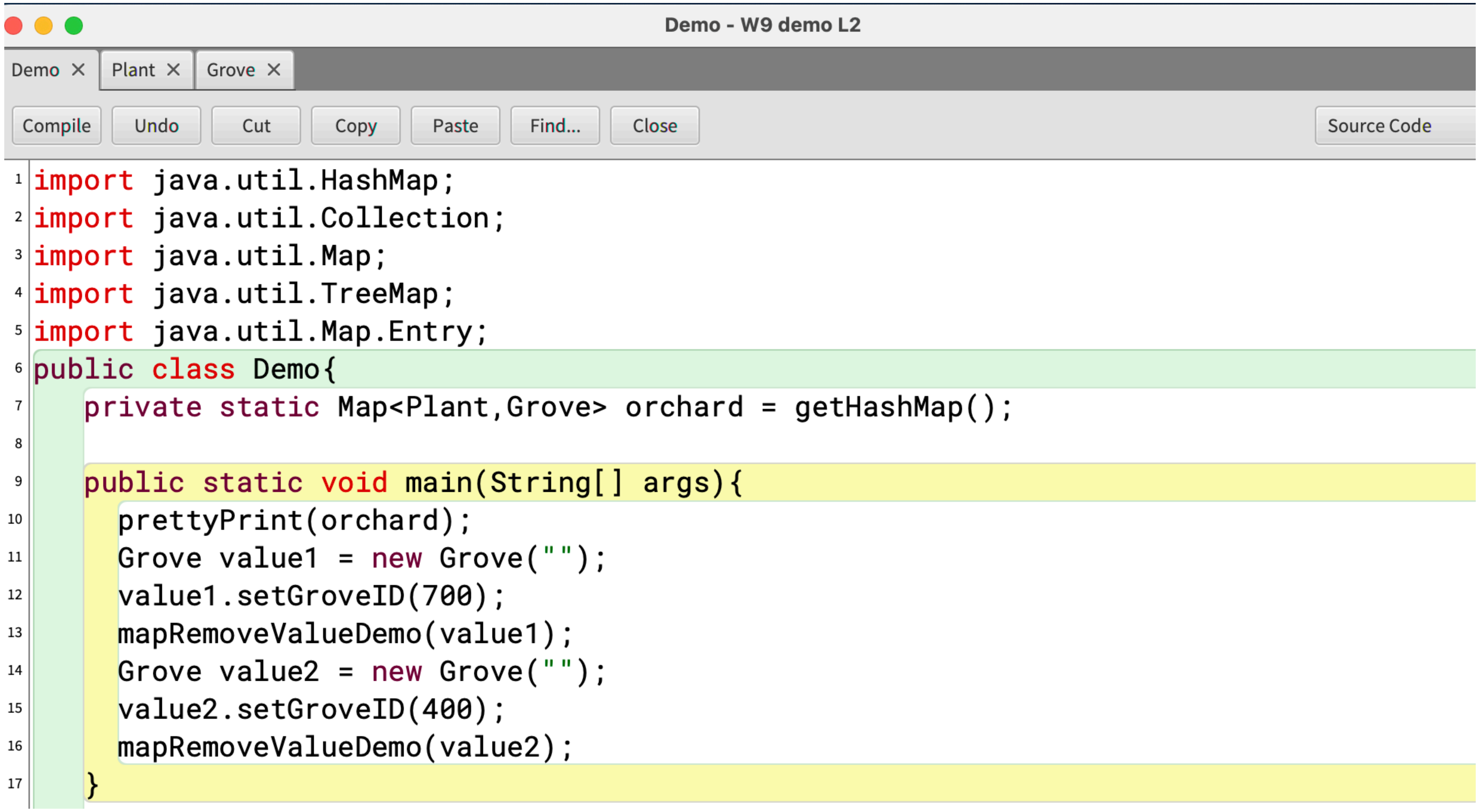
(5) Remove the key and value from the map

Example 5- HashMap



```
public static void mapRemoveValueDemo(Grove value){
    if(orchard.containsValue(value)){
        for(Map.Entry<Plant, Grove> key_value : orchard.entrySet()){
            if(key_value.getValue().equals(value)){
                Plant key = key_value.getKey();
                orchard.remove(key, value);
                break;
            }
        }
        prettyPrint(orchard);
        System.out.println("REMOVED " + value +" AND KEY ");
    }
    else
        System.out.println("CANNOT REMOVE. VALUE NOT IN MAP " + value);
}
```

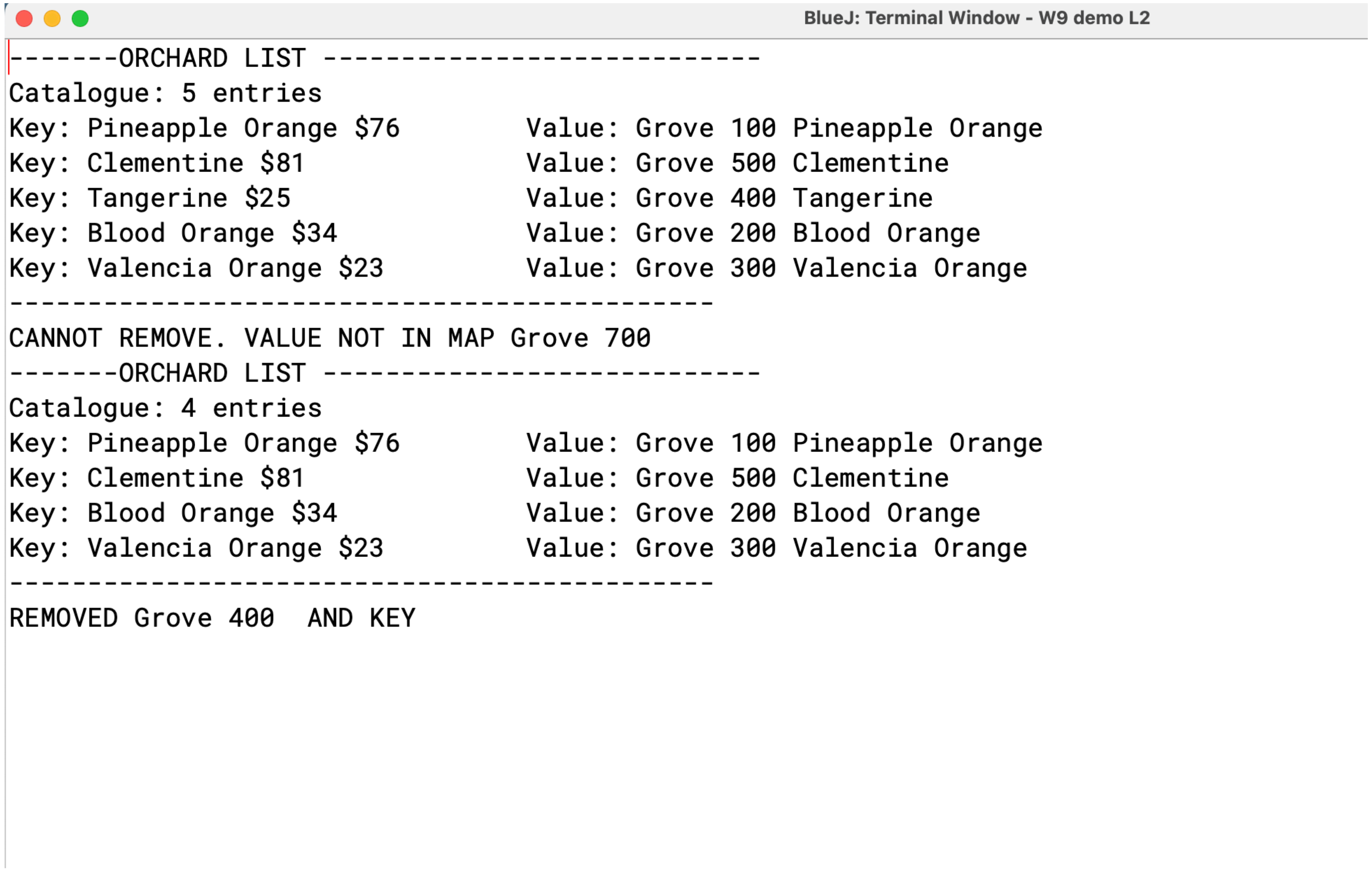
Example 5- HashMap



```java
import java.util.HashMap;
import java.util.Collection;
import java.util.Map;
import java.util.TreeMap;
import java.util.Map.Entry;
public class Demo{
    private static Map<Plant,Grove> orchard = getHashMap();

    public static void main(String[] args){
        prettyPrint(orchard);
        Grove value1 = new Grove("");
        value1.setGroveID(700);
        mapRemoveValueDemo(value1);
        Grove value2 = new Grove("");
        value2.setGroveID(400);
        mapRemoveValueDemo(value2);
    }
```

Example 5- HashMap



```
-------ORCHARD LIST ----------------------------
Catalogue: 5 entries
Key: Pineapple Orange $76         Value: Grove 100 Pineapple Orange
Key: Clementine $81               Value: Grove 500 Clementine
Key: Tangerine $25                Value: Grove 400 Tangerine
Key: Blood Orange $34             Value: Grove 200 Blood Orange
Key: Valencia Orange $23          Value: Grove 300 Valencia Orange
---------------------------------------------
CANNOT REMOVE. VALUE NOT IN MAP Grove 700
-------ORCHARD LIST ----------------------------
Catalogue: 4 entries
Key: Pineapple Orange $76         Value: Grove 100 Pineapple Orange
Key: Clementine $81               Value: Grove 500 Clementine
Key: Blood Orange $34             Value: Grove 200 Blood Orange
Key: Valencia Orange $23          Value: Grove 300 Valencia Orange
---------------------------------------------
REMOVED Grove 400  AND KEY
```

Summary

Today you learned about:

- Concrete Maps: HashMap, TreeMap
    - Retrieval of keys and values
    - Removal of keys and values