CSP2348/CSP5243 Data Structures

Solutions to Tutorial 11: - Abstract Data Types (ADTs) (3): Sets and Maps

---

Tasks:

Complete the following.

Tasks:

Complete the following.

Task 1: Set theoretical operations

Let A = {1, 2, 3, 4, 5}, B = {4, 5, 6, 7}, and C = {2, 3, 4}, attempt the following set operations:

(1) A ∪ B
(2) A ∩ B
(3) A – B
(4) B – A
(5) C – A
(6) the relationship between A and C.

Answers:

A ∪ B = {1, 2, 3, 4, 5, 6, 7}
A ∩ B = {4, 5}
A – B = {1, 2, 3}
B – A = {6, 7}
C – A = {}
A ⊇ C

Task 2: Test the Java implementation of the `TreeSet Class` given in WS1101

(Download the Java code from Blackboard)

a. Notice the invocation of `add()` `first()`, `last()`, `headSet()`, `subSet()`, and `tailSet()` methods provided by the `TreeSet Class`;
b. Execute this program and analyse the results corresponding to individual method invocations in the program.

Executed results

[Australia, Canada, China, Egypt, Greece, India, Oman, Peru, Togo, USA]
countries.first() = Australia
countries.last() = USA
countries.headSet(Australia) = []
countries.subSet(Australia,Canada) = [Australia]
countries.tailSet(Canada) = [Canada, China, Egypt, Greece, India, Oman, Peru, Togo, USA]

Task 3: Test the Java implementation of the `HashSet Class` given in WS1102

(Download the Java code from Blackboard)

a. Notice the invocation of `add()`, `contains()`, and `remove()` methods provided by the `HashSet Class`;
b. Execute this program and analyse the results corresponding to individual method invocations in the program.

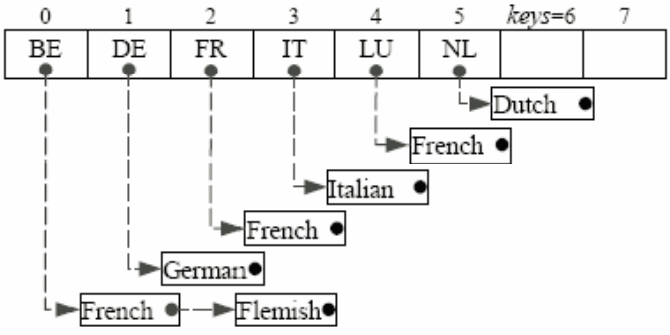
Executed results

```
7 elements: [China, USA, Peru, New Zealand, India, Australia, Canada]
set.contains(Australia) = true
countries.remove("Australia") = true
6 elements: [China, USA, Peru, New Zealand, India, Canada]
set.contains(Australia) = false
6 elements: [China, USA, Peru, New Zealand, India, Canada]
set.contains(Fiji) = false
countries.remove("Fiji") = false
countries.add("Fiji") = true
7 elements: [China, USA, Fiji, Peru, New Zealand, India, Canada]
set.contains(Fiji) = true
countries.add("Fiji") = false
7 elements: [China, USA, Fiji, Peru, New Zealand, India, Canada]
set.contains(Fiji) = true
```

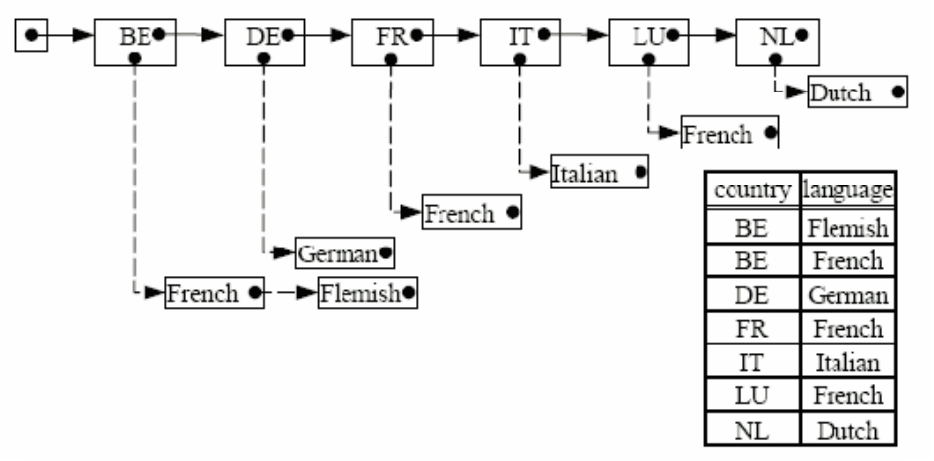
Task 4: A multi-map is similar to a map, except there may be several entries with the same key. How would you represent a multi-map without storing multiple copies of the same key? Use the table given below to illustrate your representations

Country	*Language*
BE	Flemish
BE	French
DE	German
FR	French
IT	Italian
LU	French
NL	Dutch

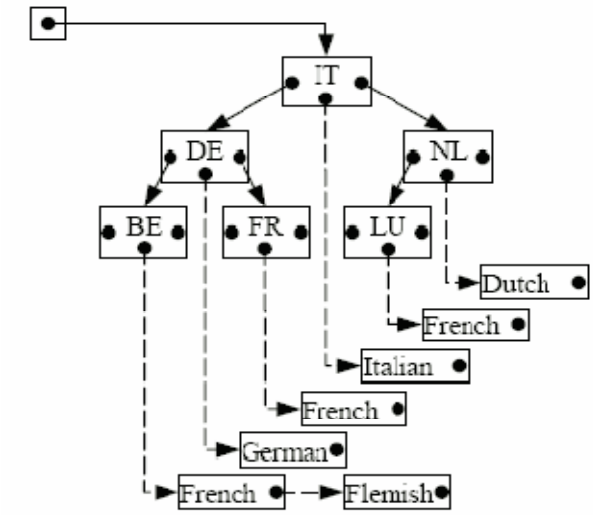
We can represent a multi-map in the same manner as an ordinary map, except that each key is associated with a *set* of values. The following figures illustrate sorted array, sorted SLL, and BST representations of a multi-map. In each case the set of values associated with a particular key is represented by an unsorted SLL, which is adequate if it can be assumed that multiple entries with the same key will be relatively uncommon.



country	language
BE	Flemish
BE	French
DE	German
FR	French
IT	Italian
LU	French
NL	Dutch



country	language
BE	Flemish
BE	French
DE	German
FR	French
IT	Italian
LU	French
NL	Dutch



country	language
BE	Flemish
BE	French
DE	German
FR	French
IT	Italian
LU	French
NL	Dutch

Task 5: Test the Java implementation of the `HashMap Class` given in WS1103

(Download the Java code from Blackboard)

a. Notice how to build a dictionary using `HashMap Class`
b. Notice the order of the words in this dictionary created using `HashMap Class`.

Executed results

```
map={Tag=day, Rat=advice, counsel, Ohr=ear, Ast=gate, Ost=east, Mal=mark,
signal, Uhr=clock, Zug=procession, train, Wal=whale, Eis=ice, Hof=court,
yard, farm, Lob=praise, Tor=gate, Rad=wheel, Hut=hat, Mut=courage}
map.keySet()=[Tag, Rat, Ohr, Ast, Ost, Mal, Uhr, Zug, Wal, Eis, Hof, Lob,
Tor, Rad, Hut, Mut]
map.size()=16
```

Task 6: Test the Java implementation of the `TreeMap Class` given in WS1004

(Download the Java code from Blackboard)

a. Notice how to build a dictionary using `TreeMap Class`;
b. Notice the order of the words in this dictionary created using `TreeMap Class`;
c. Discuss the difference between the two dictionaries implemented using different classes.

Executed results

```
Ast=gate
Eis=ice
Hof=court, yard, farm
Hut=hat
Lob=praise
Mal=mark, signal
Mut=courage
Ohr=ear
Ost=east
Rad=wheel
Rat=advice, counsel
Tag=day
Tor=gate
Uhr=clock
Wal=whale
Zug=procession, train
```