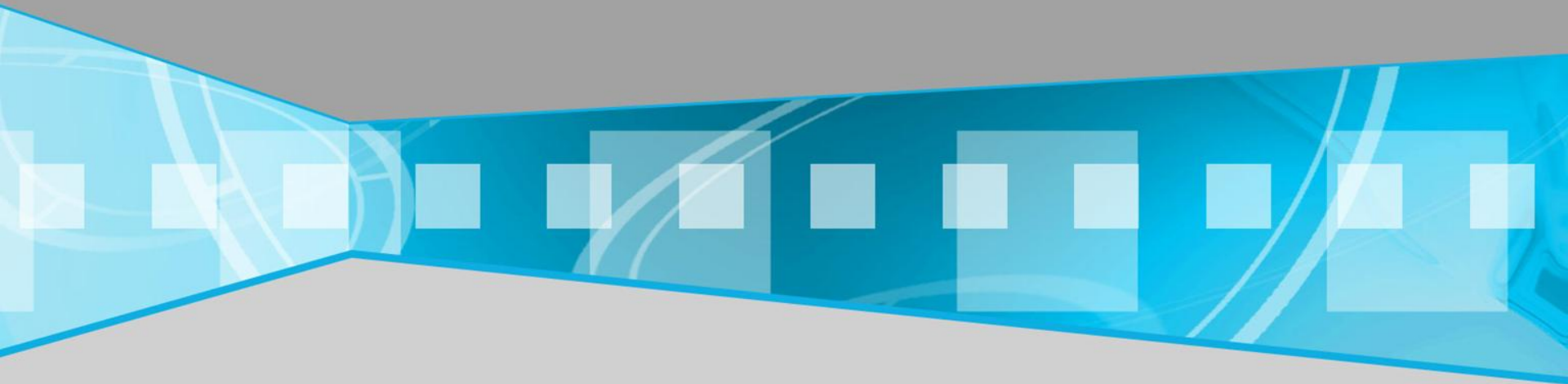



MAP



DEFINITION

- A set is a collection that lets you quickly find an existing element.
- A map stores key/value pairs. You can find a value if you provide the key.
- For example, you may store a table of employee records, where the keys are the employee IDs and the values are Employee objects.



USER NAME	PASSWORD
myAccount	1234544
yourAccount	1234544

MAP

AbstractMap

HashMap

TreeMap

METHODS

- `Int size()`
- `Boolean containsKey(Object key)`
- `Object get(Object key)`
- `Object put(Object key, Object value)`
- `Object remove(Object key)`
- `Void putAll(Map map)`
- `Void clear()`
- `Set keySet()`
- `Collection values()`
- `Set EntrySet();`

HASHMAP

- A hash map hashes the keys.
- As with sets, hashing is a bit faster, and it is the preferred choice if you don't need to visit the keys in sorted order.

FAMILIAR WITH HASH MAP - 1

```
public static void main(String[] args) {  
    HashMap av = new HashMap();  
    av.put("love", "tinh yeu");  
    av.put("hate", "ghet");  
    av.put("sad", "buon");  
    av.put("apple", "tao");  
    av.put("sad", "chan");  
    System.out.println("HashMap: "+av.toString());  
    System.out.println(av.keySet());  
    ...  
}
```

Example 2

Tạo một lớp Employee

```
class Employee{ /**Constructs an employee with $0  
salary. @param n the employee name*/  
private String name;  
private double salary;  
public Employee(String n)  
{ name = n;  
  salary = 0;}  
public String toString(){  
return "[name=" + name + ", salary=" + salary + "];"  
}
```


Example 2 (cont)

```
public class MapTest
{
    public static void main(String[] args)
    {
        Map<String, Employee> staff = new HashMap<String,
            Employee>();
        staff.put("144-25-5464", new Employee("Amy Lee"));
        staff.put("567-24-2546", new Employee("Harry Hacker"));
        staff.put("157-62-7935", new Employee("Gary Cooper"));
        staff.put("456-62-5527", new Employee("Francesca Cruz"));

        // print all entries

        System.out.println(staff);

        // remove an entry

        staff.remove("567-24-2546");
```

```
// replace an entry

        staff.put("456-62-5527", new Employee("Francesca
            Miller"));

        // look up a value

        System.out.println(staff.get("157-62-7935"));

        // iterate through all entries

        for (Map.Entry<String, Employee> entry : staff.entrySet())
        {
            String key = entry.getKey();
            Employee value = entry.getValue();
            System.out.println("key=" + key + ", value=" + value);
        }
    }
}
```


Excercise

Concordance(bảng tần số): là một danh sách của các từ xuất hiện trong một tài liệu văn bản với số dòng trên đó có các từ xuất hiện.

Concordance hữu dụng để phân tích các tài liệu nhằm tìm ra tần số xuất hiện các từ trong văn bản.

Vd: trong chuỗi “Love me love my dog” thì concordance sẽ đọc ra được bảng bên cạnh.

→ Viết chương trình mô tả Concordance.

key	value
love	2
me	1
my	1
dog	1

Solution 1

```
/** A program that counts words in a document, printing the most frequent. */
public class WordCount {
    public static void main(String[] args) {
        Map<String,Integer> freq = new ChainHashMap<>(); // or any concrete map
        // scan input for words, using all nonletters as delimiters
        Scanner doc = new Scanner(System.in).useDelimiter("[^a-zA-Z]+");
        while (doc.hasNext()) {
            String word = doc.next().toLowerCase(); // convert next word to lowercase
            Integer count = freq.get(word); // get the previous count for this word
            if (count == null)
                count = 0; // if not in map, previous count is zero
            freq.put(word, 1 + count); // (re)assign new count for this word
        }
        int maxCount = 0;
        String maxWord = "no word";
        for (Entry<String,Integer> ent : freq.entrySet()) // find max-count word
            if (ent.getValue() > maxCount) {
                maxWord = ent.getKey();
                maxCount = ent.getValue();
            }
        System.out.print("The most frequent word is " + maxWord);
        System.out.println(" with " + maxCount + " occurrences.");
    }
}
```

Solution 2

Tạo 1 file .txt, lưu 1 văn bản.

Dùng đoạn chương trình load file .txt đã học trong bài ArrayList để đọc dữ liệu thành từng dòng.

Một **HashMap concordance** = **new HashMap();**

Để đọc từng từ trong 1 dòng ta dùng StringTokenizer.

```
StringTokenizer parser = new StringTokenizer(line, " ,.:;-!()?");
```

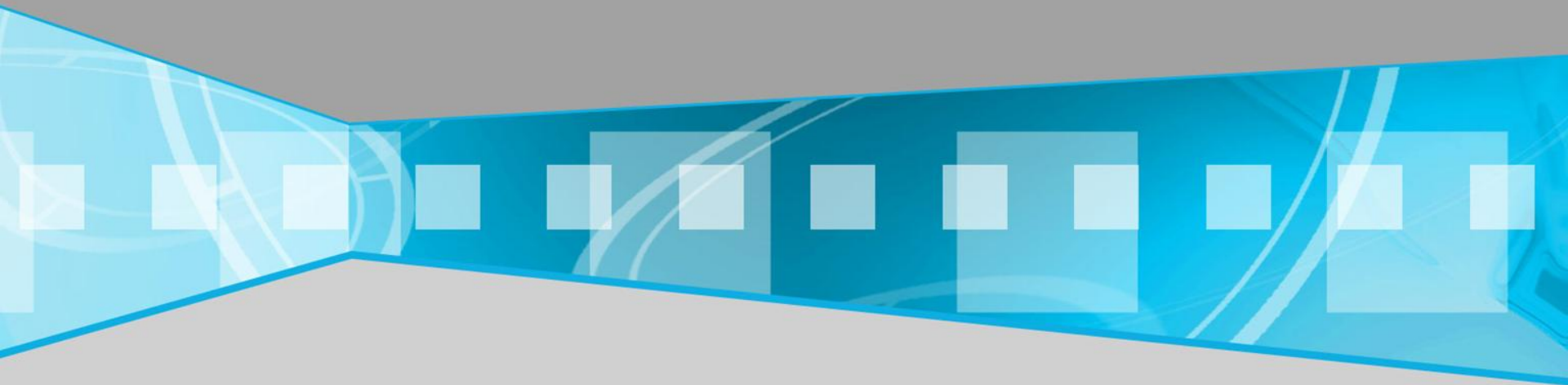
```
While(parser.hasMoreTokens()){
```

```
String word = parser.nextTokens();
```

```
// Nếu word đã có trong concordance thì value được cộng thêm 1,  
//chưa có thì thêm vào concordance
```

```
}
```

TreeMap



Definition

a tree map uses a total ordering on the keys to organize them in a search tree.

Example: mtd2000 dictionary store orderly words and their meaning.

key	value
angry	Giận dữ
jealous	Ghen
happy	Vui vẻ
sad	Buồn

Excercise

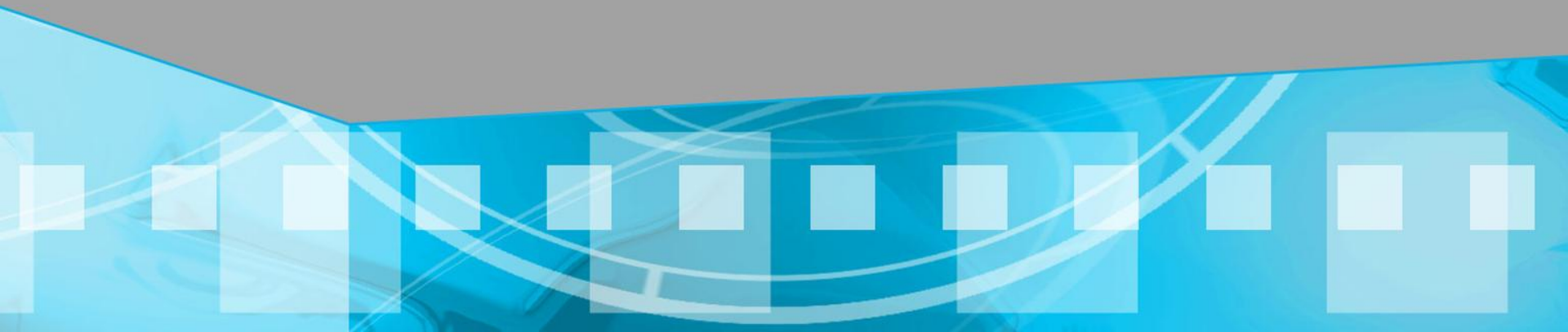
Xây dựng một từ điển có khả năng chứa đựng các từ và nghĩa của nó một cách có thứ tự.

Có các chức năng như cập nhật nghĩa của từ khi cung cấp cho một từ (key).

Có khả năng tìm kiếm từ Anh sang Việt.

Có khả năng tìm kiếm từ Việt sang Anh.

***Có khả năng tìm kiếm gần đúng. VD: nhập vào từ app thì hệ thống trả ra hàng loạt từ có bắt đầu là app.*



Challenge ?

Viết một trình biên dịch nhận vào một văn bản tiếng Việt hoặc tiếng Anh sử dụng từ điển tự tạo để dịch từ Anh sang Việt hoặc từ Việt sang Anh.

