**Дисциплина**: *СД и ООП* **Тема:** *Таблица* - продължение
**Специалност**: *КН 1 курс, 3 и 4 групи* **Седмица**: *26.05.2010 г.*

***Задача:*** Интерфейсът **Dictionary** представя операции на *речник* - таблица с ключове от тип **String** и стойности от тип **Object**:

```
public interface Dictionary {
	public boolean containsKey(String key);
	//Проверява дали в текущия речник се съдържа ключът key
	public boolean containsValue(Object value);
	//Проверява дали в текущия речник се съдържа стойността value
	public Object get(String key);
	//Връща стойността, асоциирана с ключа key на текущия речник
	public Object put(String key,Object value);
	//Ако има двойка с ключ key в текущия речник, променя стойността в нея на value.
	//В противен случай включва двойка (key,value) в текущия речник
	public Object remove(String key);
	//Изключва двойка с ключ key от текущия речник и връща съответната стойност
	public boolean isEmpty();
	//Проверява дали текущия речник е празен
	public boolean equals(Object obj);
	//Проверява дали текущия речник съвпада с речника obj
	public int size();
	//Връща броя на редовете в текущия речник
	public java.util.Iterator keys();
	//Обхожда ключовете на текущия речник в азбучен ред
	public java.util.Iterator values();
	//Обхожда стойностите на текущия речник
}
```

Класът **EntryComparator** представя обект за сравняване на два обекта на класа **Entry**:

```
public class EntryComparator implements java.util.Comparator {
	public int compare(Object a,Object b) {
		Object key1 = ((Entry)a).getKey();
		Object key2 = ((Entry)b).getKey();
		return ((java.lang.Comparable)key1).compareTo(key2);
	}
}
```

Класът **SortedLinkedList<E>** е реализация на сортиран списък:

```
public class SortedLinkedList<E> extends java.util.LinkedList<E> {
	//Methods inherited from class LinkedList<E>

	//New method
	public void addInSortedList(java.util.Comparator<E> c,E x) {
		if(super.isEmpty() || c.compare(x,super.get(0)) <= 0) super.add(0,x);
		else {
			int i;
			for(i = 0; i < super.size(); i++) {
				if(c.compare(x,super.get(i)) <= 0) {
					super.add(i,x);
					break;
				}
			}
			if(i == super.size()) super.add(x);
		}
	}
}
```

1.Да се реализира клас **DictionryImp** за представяне на речник, съгласно спецификацията:

```
public class DictionaryImp implements Dictionary {
        //Представяне на речник
        private Function table;

        //Конструктор
        public DictionaryImp(Function table) { this.table = table; }

        //Реализация на методите на интерфейса Dictionary
}
```

При реализацията на метода keys() да се използва класът **SortedLinkedList<E>.**

2.Да се илюстрира използването на класа **DictionaryImp** за създаване и извеждане на следните речници:

2.1. *Речник* $D(L_1,L_2)=\{(a_1,b_1),(a_2,b_2),\ldots,(a_n,b_n)\}$ *за превод от език* $L_1$ *на език* $L_2$, където:

- $a_1,a_2,\ldots,a_n$ са думи на езика $L_1$
- $b_1,b_2,\ldots,b_n$ са съответните им думи на езика $L_2$

2.2. *Честотен речник* $D(T)=\{(\omega_1,n_1/p),\ldots,(\omega_k,n_k/p)\}$ *на текст* $T$, където:

- $\omega_1,\ldots,\omega_k$ са различните думи в текста
- $n_1$-брой срещания на думата $\omega_1$ в текста, $n_2$-брой срещания на думата $\omega_2$ в текста, ..., $n_k$-брой срещания на думата $\omega_k$ в текста
- p е общият брой думи в текста

***Тестване:*** Програмата **TestDictionary:**

1.Чете низове от вида: дума1@дума2 от файла testDictData.txt и създава речник от двойки (дума1,дума2)

***Dictionary dict1 = new DictionaryImp(new ListTable())***

Извежда елементите на речника на стандартния изход.

2.Чете текст от файла testFreqDictData.txt и създава честотен речник на текста, състоящ се от двойки (дума,честота_на_срещане)

***FrequencyDictionaryImp dict2 = new FrequencyDictionaryImp(new ListTable())***

Извежда елементите на честотния речник на стандартния изход.