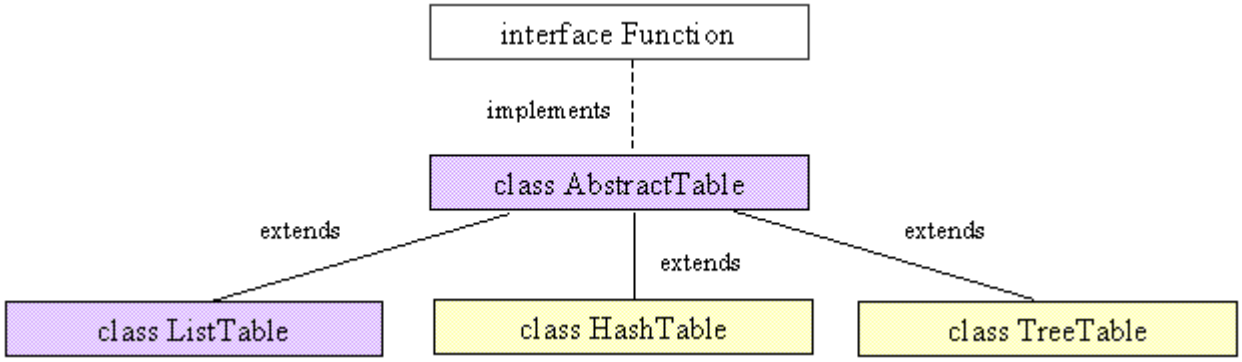
Дисциплина: *СД и ООП* **Тема:** *Таблица*
Специалност: *КН 1 курс, 3 и 4 групи* **Дата**: *19.05.2010 г.*

Задача: Дадена е йерархията от интерфейси и класове:



Интерфейсът **Function** представя операции на *таблица на функция* със стойности на променливата (наричани понататък *ключове*), от тип **Object** и функционални стойности (наричани понататък *стойности*) от тип **Object**:

```
public interface Function {
        public boolean containsKey(Object key);
        //Проверява дали в текущата таблица се съдържа ключът key
        public boolean containsValue(Object value);
        //Проверява дали в текущата таблица се съдържа стойността value
        public Object get(Object key);
        //Връща стойността на функцията за ключа key от текущата таблица
        public Object put(Object key,Object value);
        //Ако има ред с ключ key в текущата таблица, съответната стойност се променя на value.
        //В противен случай се включва нов ред (key,value) в текущата таблица
        public Object remove(Object key);
        //Изключва ред с ключ key от текущата таблица и връща съответната стойност
        public boolean isEmpty();
        //Проверява дали текущата таблица е празна
        public boolean equals(Object obj);
        //Проверява дали текущата таблица и таблицата obj представят една и съща функция
        public int size();
        //Връща броя на редовете на текущата таблица
        public java.util.Iterator keys();
        //Обхожда ключовете на текущата таблица
        public java.util.Iterator values();
        //Обхожда стойностите на текущата таблица
        public java.util.Set keySet();
        //Връща множество от ключовете на текущата таблица
        public java.util.Collection entryValues();
        //Връща колекция от стойностите на текущата таблица
        public java.util.Set entrySet();
        //Връща множество от двойките от вида (ключ,стойност) на текущата таблица
}
```

Таблица не може да съдържа обекти-ключове и стойности, равни на **null**. Всеки метод с параметри key или value активира изключение **NullPointerException**, ако key=null или value=null.

Класът **Entry** представя ред на таблица – двойка от вида (ключ, стойност):

```
public class Entry {
        //Data
        private Object key;
        private Object value;

        //Constructor
```

```
        public Entry(Object key) throws NullPointerException {
                if(key == null) throw new NullPointerException();
                this.key = key;
        }

        //Methods
        public Object getKey() { return key; }
        public Object getValue() { return value; }
        public Object setValue(Object newValue) throws NullPointerException {
                if(newValue == null) throw new NullPointerException();
                Object old = value;
                value = newValue;
                return old;
        }

        public boolean equals(Object obj) {
                Entry t = (Entry)obj;
                return key.equals(t.getKey()) && value.equals(t.getValue());
        }

        public String toString() { return "(" + key + "," + value + ")"; }
}
```

1.Да се реализира клас **AbstractTable**, съгласно спецификацията:

```
public abstract class AbstractTable implements Function {
        //Абстрактни методи
        public abstract Object get(Object key);
        public abstract Object put(Object key,Object value);
        public abstract Object remove(Object key);
        public abstract java.util.Iterator keys();
        public abstract java.util.Iterator values();

        //Реализация на методи на интерфейса Function
        public boolean containsKey(Object key) {…}
        public boolean containsValue(Object value) {…}
        public boolean isEmpty() {…}
        public boolean equals(Object obj) {…}
        public int size() {…}
        public java.util.Set keySet() {…}
        public java.util.Collection entryValues() {…}
        public java.util.Set entrySet() {…}

        //Предефиниране
        public String toString() {…}
}
```

2.Да се реализира клас **ListTable**, съгласно спецификацията:

```
public class ListTable extends AbstractTable implements Function {
        //Представяне на таблица
        private java.util.LinkedList list;

        //Конструктор
        public ListTable() { list = new java.util.LinkedList(); }

        //Метод за достъп
        public java.util.LinkedList getList() { return list; }

        //Реализация на абстрактните методи на класа AbstractTable
        //Наследяване на методите на интерфейса Function, реализирани в класа AbstractTable
}
```

3.Да се реализира клас **HashTable**, съгласно спецификацията:

```
public class HashTable extends AbstractTable implements Function {
        //Представяне на таблица
        private java.util.Hashtable hashtable;

        //Конструктор
        public HashTable() { hashtable = new java.util.Hashtable(); }

        //Реализация на абстрактните методи на класа AbstractTable
        //Наследяване на методите на интерфейса Function, реализирани в класа AbstractTable
}
```

При реализацията на класа **HashTable** се използва родова хеш-таблица **Hashtable<K,V>** - виж http://java.sun.com/javase/6/docs/api/.

4.Да се реализира клас **TreeTable**, съгласно спецификацията:

```
public class TreeTable extends AbstractTable implements Function {
        //Представяне на таблица
        private BinaryTree tree;

        //Конструктор
        public TreeTable(BinaryTree arg) { tree = arg; }

        //Реализация на абстрактните методи на класа AbstractTable
        //Наследяване на методите на интерфейса Function, реализирани в класа AbstractTable
}
```

5.*Итератори.* Да се реализира клас **TableValuesIterator** за обхождане на стойностите на текущата таблица. Класът **TableKeysIterator** представя итератор за обхождане на ключовете на текущата таблица:

```
public class TableKeysIterator implements java.util.Iterator {
        //Data
        java.util.Iterator it, itOld;

        //Constructor
        public TableKeysIterator(java.util.Iterator arg) { it = itOld = arg; }

        //Iterator methods
        public boolean hasNext() { return it.hasNext(); }
        public Object next() { return ((Entry)(it.next())).getKey(); }
        public void remove() { it.remove(); }
        public void reset() { it = itOld; }
}
```

При реализацията да се използва таблицата за основните интерфейси и тяхната реализация - виж http://java.sun.com/docs/books/tutorial:

General-purpose Implementations

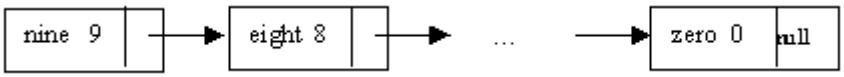
Interfaces	Implementations				
	Hash table	Resizable array	Tree	Linked list	Hash table + Linked list
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Queue					
Map	HashMap		TreeMap		LinkedHashMap

***Пример**:* Нека table е обект на клас, реализиращ интерфейса **Function**. Следващата таблица съдържа операции за включване в table, които се изпълняват в реда от таблицата. Хешкодът w.hashCode() на името w = $c_0c_1…c_{n-1}$ (обект на класа **String**), се определя по следния начин:

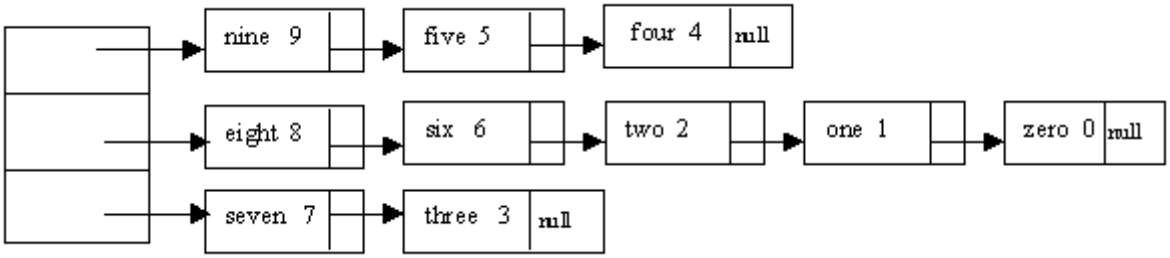
$$c_0*31^{(n-1)} + c_1*31^{(n-2)} + ... + c_{n-1}$$

Име на цифра на английски език	Цифра	Операция	Сума от номерата на буквите на името	Сума%3	Хешкод на името	Хешкод%3
Zero	0	table.put("zero",new Integer(0))	448	1	3735208	1
One	1	table.put("one",new Integer(1))	322	1	110182	1
Two	2	table.put("two",new Integer(2))	346	1	115276	1
Three	3	table.put("three",new Integer(3))	536	2	110339486	2
Four	4	table.put("four",new Integer(4))	444	0	3149094	0
Five	5	table.put("five",new Integer(5))	426	0	3143346	0
Six	6	table.put("six",new Integer(6))	340	1	113890	1
Seven	7	table.put("seven",new Integer(7))	545	2	109330445	2
Eight	8	table.put("eight",new Integer(8))	529	1	96505999	1
Nine	9	table.put("nine",new Integer(9))	426	0	3381426	0

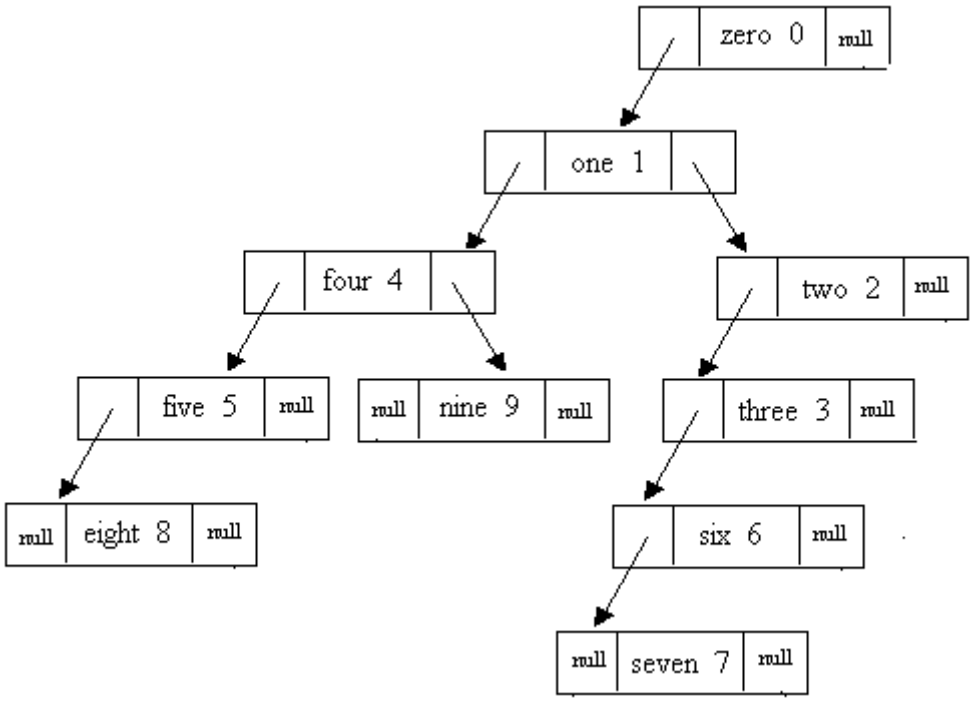
1.table = new ListTable() – представяне чрез линеен едносвързан списък



2.table = new HashTable() – представяне чрез масив от три линейни едносвързани списъка



3.table = new TreeTable(new BinarySearchTree(new EntryComparator()) – представяне чрез двоично дърво на търсене



Тестване: Програмата **TestTable:**

1.Чете низове от вида: дума@цяло_число от файла testTableData.txt и създава таблица от двойки (дума,цяло_число)

Function table = new ListTable()

Извежда елементите на таблицата на стандартния изход.

2.Чете низове от вида: дума@цяло_число от файла testTableData.txt и създава таблица от двойки (дума,цяло_число)

Function table = new HashTable()

Извежда елементите на таблицата на стандартния изход.

3.Чете низове от вида: дума@цяло_число от файла testTableData.txt и създава таблица от двойки (дума,цяло_число)

Function table = new TreeTable(new BinarySearchTree(new EntryComparator()))

Обхожда дървото по четирите начина и записва резултата във файла testTreeTableResult.txt.

4.Обхожда дървото от т.3, като използва итератора

java.util.Iterator it = new TableValuesIterator(tree.iterator());

и извежда резултата на стандартния изход.