4学习情况表

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 杜金瑞 | **学号** | 2020905073 |
| **学院** | 信息工程学院 | **专业** | 计算机科学与技术 |

|  |
| --- |
| 学习情况简述 |
| C:\Users\29043\Documents\Tencent Files\2904326062\Image\C2C\812C3E5E1929EB67D6453577BFFF221E.jpg |
| C:\Users\29043\Documents\Tencent Files\2904326062\Image\C2C\70E77B13411AAEE85677508D255C0551.jpg |
| C:\Users\29043\Documents\Tencent Files\2904326062\Image\C2C\136E6B75E429ADCCF41B06065E2AE6E2.jpg |
| 本周练习过的代码（例） |

import sun.reflect.generics.tree.Tree;  
  
import java.util.HashMap;  
import java.util.TreeMap;  
public class HashTable <K,V>{  
 private final int[] capacity={  
53,97,193,389,769,1543,3079,6151,12289,24593,  
 49157,98317,196613,393241,786433,1572869,3145739,6291469,  
 12582917,25165843,50331653,100663319,201326611,402653189,805306457,1610612741};  
 private static final int *upperTol*=10;  
 private static final int *lowerTol*=2;//不能更改  
 private static int *capacityIndex*=0;  
 public TreeMap<K,V>[] hashtable;  
 private int M; //地址数  
 private int size;  
  
 public HashTable(){  
 this.M=capacity[*capacityIndex*];  
 size=0;  
 hashtable=new TreeMap[M];  
 for(int i=0;i<M;i++){  
 hashtable[i]=new TreeMap<>();  
 }  
  
 }  
  
  
  
 private int hash(K key){  
 return (key.hashCode()&0x7fffffff)%M;  
 }  
  
 public int getSize(){  
 return size;  
 }  
  
 public void add(K key,V value){  
 TreeMap<K,V> map=hashtable[hash(key)];  
 if(map.containsKey(key))  
 map.put(key, value);  
 else {  
 map.put(key, value);  
 size++;  
  
 if(size>=*upperTol*\*M&&*capacityIndex*+1<capacity.length)  
 *capacityIndex*++;  
 resize(capacity[*capacityIndex*]);  
 }  
 }  
  
 public V remove(K key){  
 TreeMap<K,V> map=hashtable[hash(key)];  
 V ret=null;  
 if(map.containsKey(key)) {  
  
 ret = map.remove(key);  
 size--;  
  
 if(size<*lowerTol*\*2&&*capacityIndex*-1>=0)  
 *capacityIndex*--;  
 resize(capacity[*capacityIndex*]/2);  
 }  
 return ret;  
 }  
  
 public void set(K key,V value){  
 TreeMap<K,V> map=hashtable[hash(key)];  
 if(!map.containsKey(key))  
 throw new IllegalArgumentException(key+"don't exist!");  
  
 map.put(key,value);  
 }  
  
 public boolean contains(K key){  
 return hashtable[hash(key)].containsKey(key);  
 }  
  
 public V get(K key){  
 return hashtable[hash(key)].get(key);  
 }  
  
 public void resize(int newM){  
 TreeMap<K,V> [] newHashTable=new TreeMap[newM];  
 for(int i=0;i<newM;i++)  
 newHashTable[i]=new TreeMap<>();  
  
 int oldM=M;  
 this.M=newM;  
 for(int i=0;i<oldM;i++){  
 TreeMap<K,V> map=hashtable[i];  
 for(K key:map.keySet()){  
 newHashTable[hash(key)].put(key,map.get(key));  
 }  
 }  
 this.hashtable=newHashTable;  
 }  
  
}

import java.util.HashMap;  
import java.util.HashSet;  
  
public class Main {  
 public static void main(String[] args) {  
 int a=42;  
 System.*out*.println(((Integer)a).hashCode());  
 int b=-42;  
 System.*out*.println(((Integer)b).hashCode());  
 double c=3.1415926;  
 System.*out*.println(((Double)c).hashCode());  
 String d="imooc";  
 System.*out*.println(d.hashCode());  
 student stu=new student(3,2,"bobo","Liu");  
 System.*out*.println(stu.hashCode());  
  
 HashSet<student> set=new HashSet<>();  
 set.add(stu);  
 HashMap<student,Integer> scores=new HashMap<>();  
 scores.put(stu,100);  
 }  
}

public class Solution {  
 public int firstUniqChar(String s){  
 int[] freq=new int[26];  
  
 for(int i=0;i<s.length();i++){  
 freq[s.charAt(i)-'a']++;  
 }  
 for(int i=0;i<s.length();i++){  
 if(freq[s.charAt(i)-'a']==1)  
 return i;  
 }  
 return -1;  
 }  
}

public class student {  
 int gra;  
 int cla;  
 String firstName;  
 String LastName;  
  
 student(int gra,int cla,String firstName,String LastName){  
 this.gra=gra;  
 this.cla=cla;  
 this.firstName=firstName;  
 this.LastName=LastName;  
 }  
  
 @Override  
 public int hashCode(){  
 int B=31;  
 int hash=0;  
  
 hash=hash\*B+gra;  
 hash=hash\*B+cla;  
 hash=hash\*B+firstName.toLowerCase().hashCode();  
 hash=hash\*B+LastName.toLowerCase().hashCode();  
  
 return hash;  
 }  
  
 @Override  
 public boolean equals(Object o){  
 if(this==o)  
 return true;  
 if(o==null)  
 return false;  
 if(getClass()!=o.getClass())  
 return false;  
 student another=(student)o;  
 return this.gra== another.gra&&  
 this.cla==another.cla&&  
 this.firstName.toLowerCase().equals(another.firstName.toLowerCase())&&  
 this.LastName.toLowerCase().equals(another.LastName.toLowerCase());  
 }  
}