ESKİŞEHİR TECHNICAL UNIVERSITY

ALGORITHMS AND COMLEXITY

EEM480

HOMEWORK4

HALİL İBRAHİM ÖZTÜRK

14467071268

GOAL

The goal of this homework is , document indexing, which enables to speed up the content search of documents. Like all search engines do, all documents in the internet is indexed and inserted to a database. Thus, whenever you search for a document including word(s), search engines can bring the documents which contain the word you are looking for in a fraction of milliseconds. Obviously, they do not perform the actual search operation in that moment, i.e. when you search for the word. Instead, they are performing the search operation when they are indexing the documents. Thus they already know which documents include which words or phrases and the time consuming search operation is shifted to the offline stage.

Indexing can be done in several ways. In real life, search engines use huge matrices which keep the relationships between the documents and the words. In this assignment, we kept the information within a hash table. In this project we realized a hash table using open addressing method in order to solve collusion. Here it is preferred double hashing in order to distribute clusters evenly on database. For each word also keep the frequency variable in order to track the number of occurrences of a word in text.

HASH TABLE

The Hashtable class implements a hash table, which maps keys to values. Any non-null object can be used as a key or as a value. To successfully store and retrieve objects from a hashtable, the objects used as keys must implement the hashCode method and the equals method.

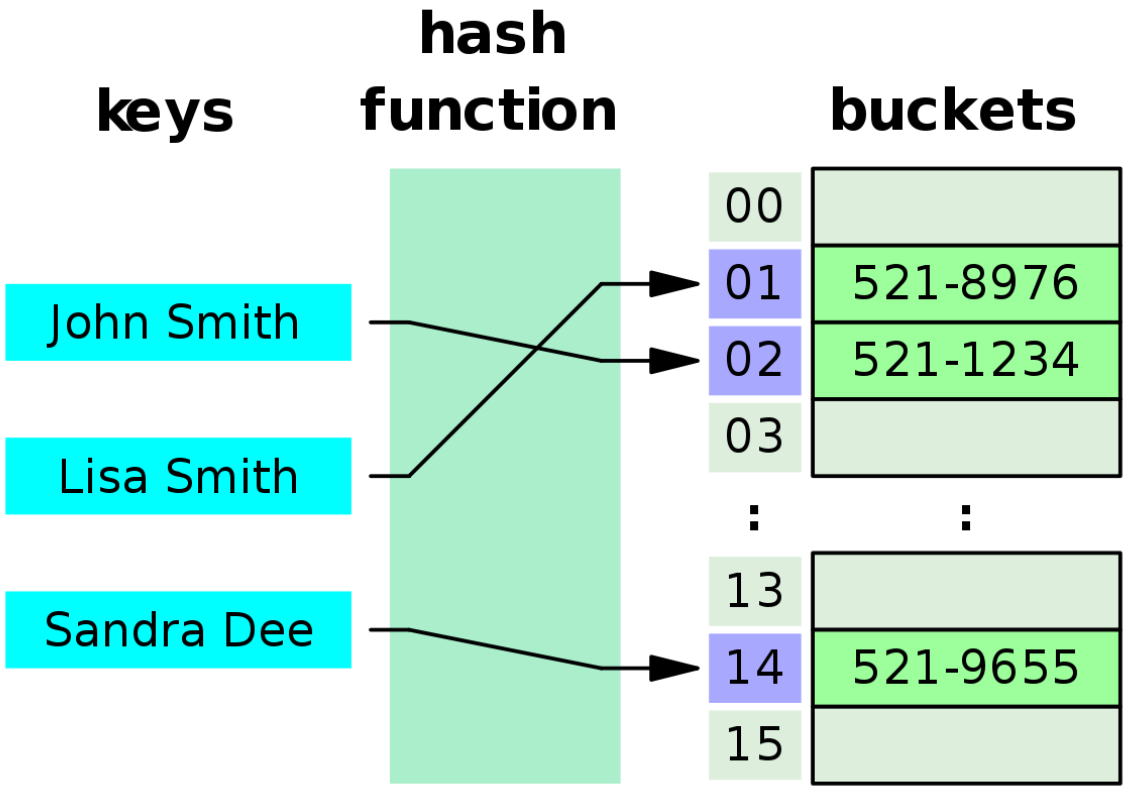
Features of Hashtable

-It is similar to HashMap, but is synchronized.

-Hashtable stores key/value pair in hash table.

-In Hashtable we specify an object that is used as a key, and the value we want to associate to that key. The key is then hashed, and the resulting hash code is used as the index at which the value is stored within the table.

-HashMap doesn't provide any Enumeration, while Hashtable provides not fail-fast Enumeration.



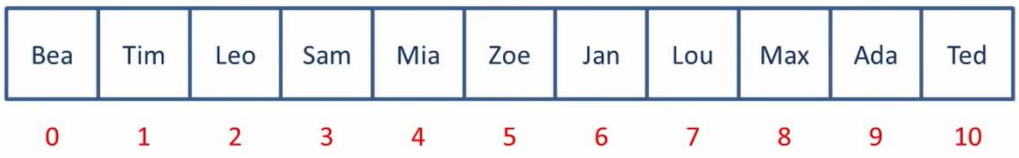
How I Obtain Key From Word

A hash function takes an input as a key, which is associated with a datum or record and used to identify it to the data storage and retrieval application. The keys may be fixed length, like an integer, or variable length, like a name. In some cases, the key is the datum itself. Calculation applied to a key to transform it into an address.
For numeric keys, divide the key by the number of available addresses, n, and take remainder.

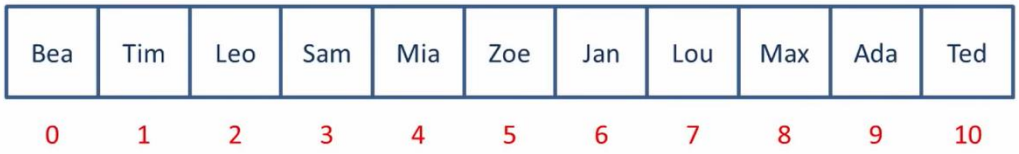
Address=key Mod n

For alphanumeric keys, divide the sum of ASCII codes in a key by the number of available addresses, n , and take the remainder. I used sum of ASCII codes of word as a key to determine all words and I reach all words as different values of key .

Index number = *sum ASCII codes* Mod *size of array*

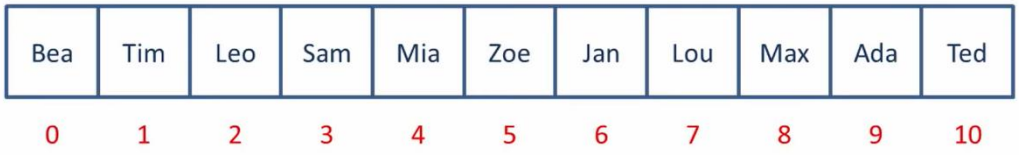


Find Ada Ada = (65 + 100 + 97) = 262



Find Ada 262 Mod 11 = 9

myData = Array(9)



Mia	M	77	i	105	a	97	279	4
Tim	T	84	i	105	m	109	298	1
Bea	B	66	e	101	a	97	264	0
Zoe	Z	90	o	111	e	101	302	5
Jan	J	74	a	97	n	110	281	6
Ada	A	65	d	100	a	97	262	9
Leo	L	76	e	101	o	111	288	2
Sam	S	83	a	97	m	109	289	3
Lou	L	76	o	111	u	117	304	7
Max	M	77	a	97	x	120	294	8
Ted	T	84	e	101	d	100	285	10

Bea	Tim	Leo	Sam	Mia	Zoe	Jan	Lou	Max	Ada	Ted
0	1	2	3	4	5	6	7	8	9	10

ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

DOUBLE HASHING

Double hashing is a collision resolving technique in Open Addressed Hash tables. Double hashing uses the idea of applying a second hash function to key when a collision occurs.

Double hashing can be done using :
(hash1(key) + i * hash2(key)) % TABLE_SIZE
Here hash1() and hash2() are hash functions and TABLE_SIZE is size of hash table. (We repeat by increasing i when collision occurs)

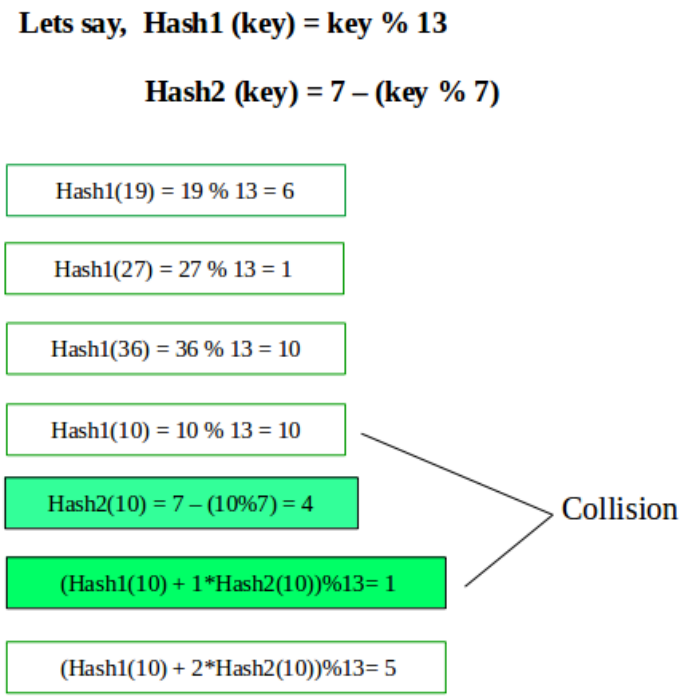
First hash function is typically hash1(key) = key % TABLE_SIZE

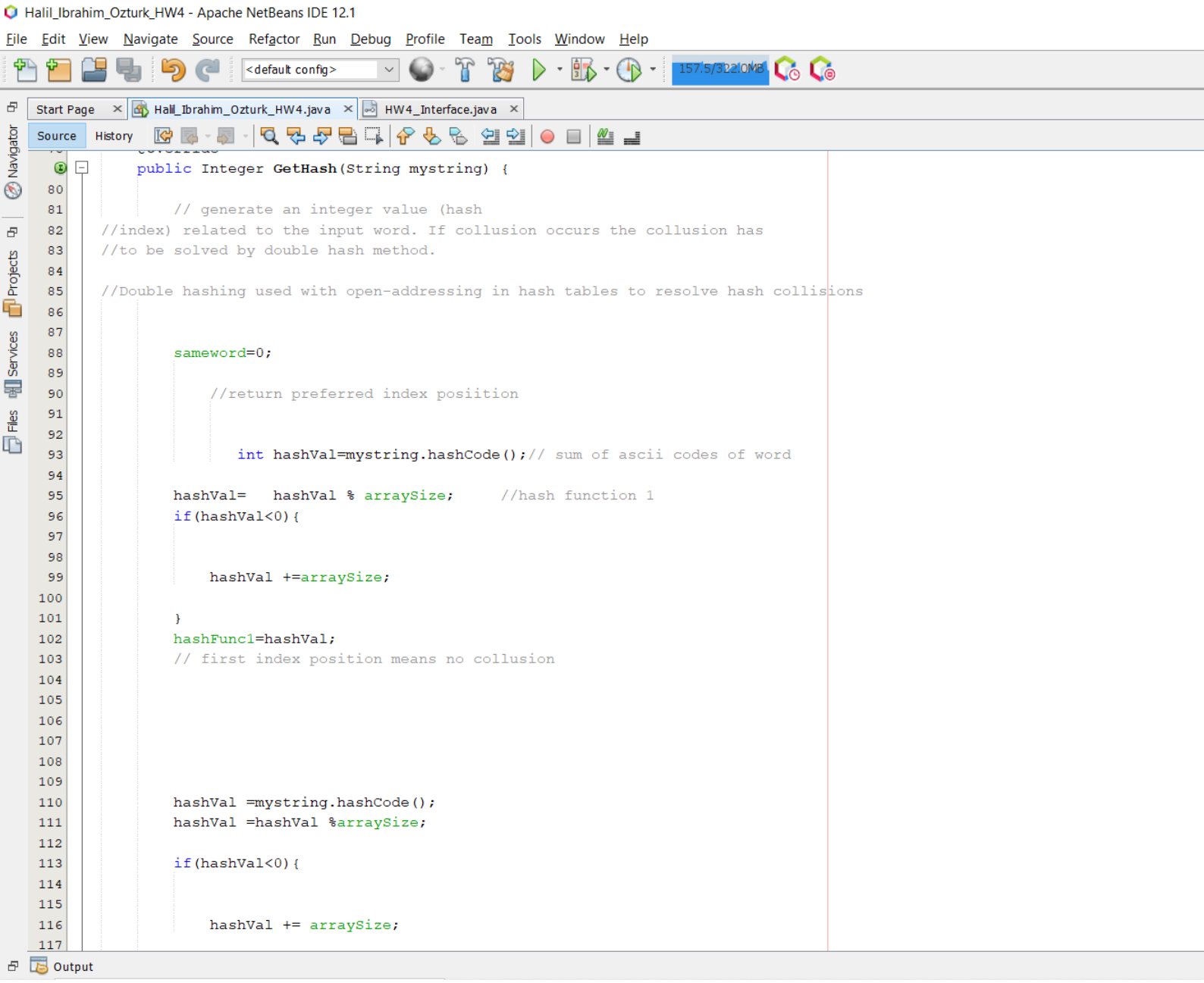
A popular second hash function is : hash2(key) = PRIME – (key % PRIME) where PRIME is a prime smaller than the TABLE_SIZE.

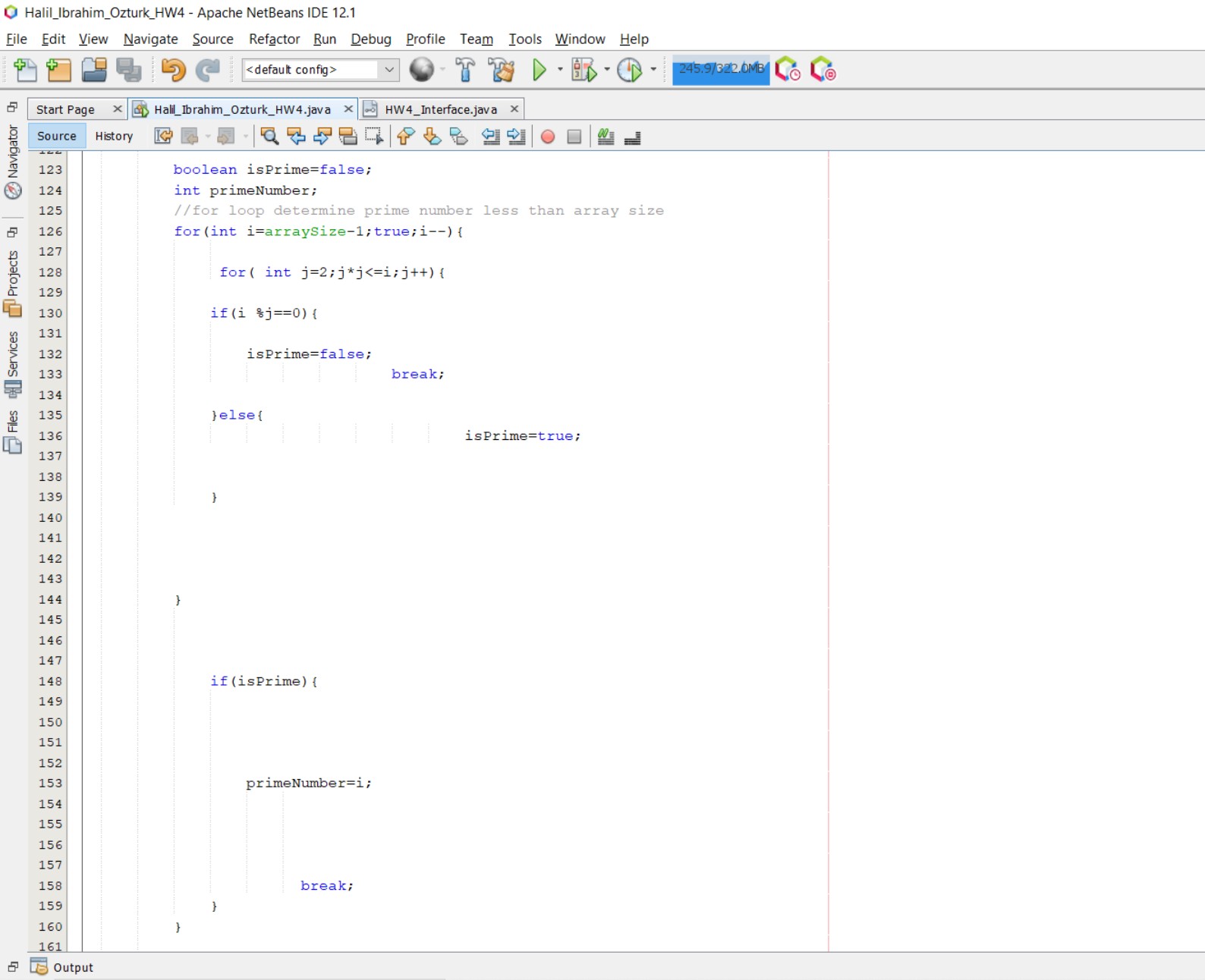
A good second Hash function is:

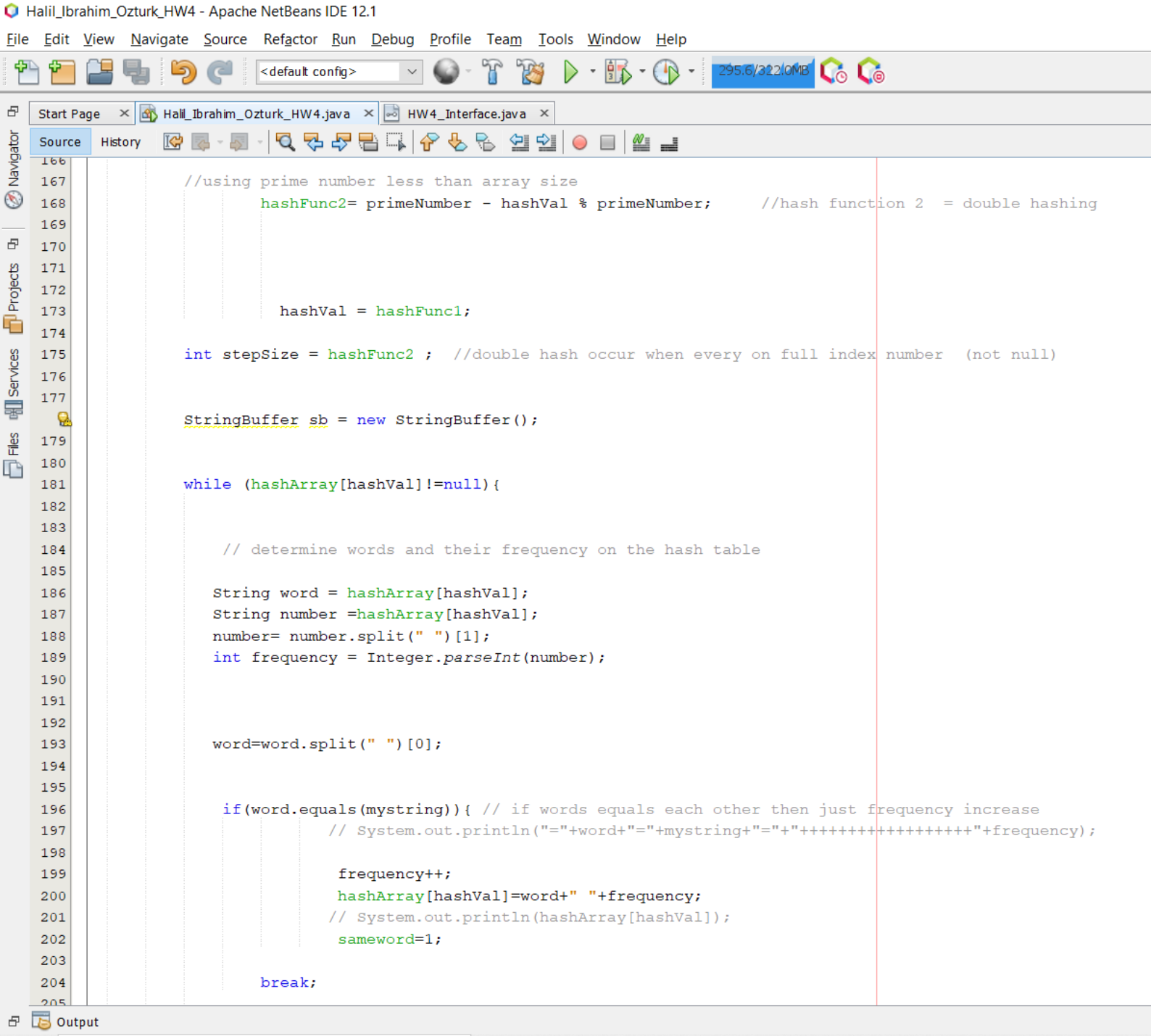
-It must never evaluate to zero

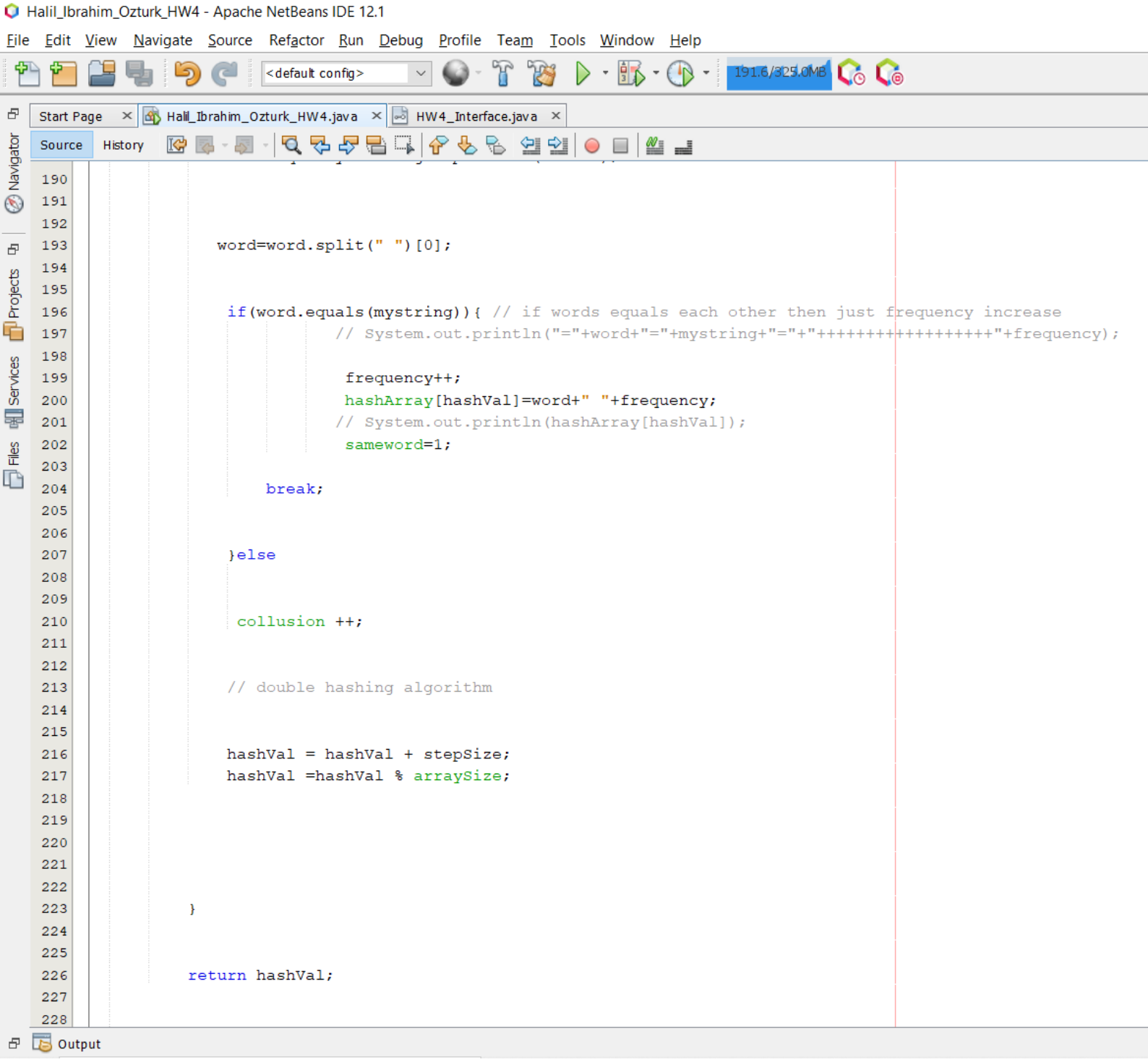
-Must make sure that all cells can be probed




Halil_Ibrahim_Ozturk_HW4 - Apache NetBeans IDE 12.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
157.5/322.0MB
Start Page
Halil_Ibrahim_Ozturk_HW4.java
HW4_Interface.java
Source History
Navigator
Projects
Services
Files
public Integer GetHash(String mystring) {
// generate an integer value (hash
//index) related to the input word. If collusion occurs the collusion has
//to be solved by double hash method.
//Double hashing used with open-addressing in hash tables to resolve hash collisions
sameword=0;
//return preferred index posiition
int hashVal=mystring.hashCode();// sum of ascii codes of word
hashVal= hashVal % arraySize; //hash function 1
if(hashVal<0){
hashVal +=arraySize;
}
hashFunc1=hashVal;
// first index position means no collusion
hashVal =mystring.hashCode();
hashVal =hashVal %arraySize;
if(hashVal<0){
hashVal += arraySize;
Output


Halil_Ibrahim_Ozturk_HW4 - Apache NetBeans IDE 12.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
245.9/322.0MB
Start Page
Halil_Ibrahim_Ozturk_HW4.java
HW4_Interface.java
Source History
Navigator
Projects
Services
Files
boolean isPrime=false;
int primeNumber;
//for loop determine prime number less than array size
for(int i=arraySize-1;true;i--){
for(int j=2;j*j<=i;j++){
if(i %j==0){
isPrime=false;
break;
}else{
isPrime=true;
}
}
if(isPrime){
primeNumber=i;
break;
}
}
Output


Halil_Ibrahim_Ozturk_HW4 - Apache NetBeans IDE 12.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
295.6/322.0MB
Start Page
Halil_Ibrahim_Ozturk_HW4.java
HW4_Interface.java
Source History
Navigator
Projects
Services
Files
//using prime number less than array size
hashFunc2= primeNumber - hashVal % primeNumber; //hash function 2 = double hashing
hashVal = hashFunc1;
int stepSize = hashFunc2 ; //double hash occur when every on full index number (not null)
StringBuffer sb = new StringBuffer();
while (hashArray[hashVal]!=null){
// determine words and their frequency on the hash table
String word = hashArray[hashVal];
String number =hashArray[hashVal];
number= number.split(" ")[1];
int frequency = Integer.parseInt(number);
word=word.split(" ")[0];
if(word.equals(mystring)){ // if words equals each other then just frequency increase
// System.out.println("="+word+"="+mystring+"="+"+++++++++++++++++++"+frequency);
frequency++;
hashArray[hashVal]=word+" "+frequency;
// System.out.println(hashArray[hashVal]);
sameword=1;
break;
Output


Halil_Ibrahim_Ozturk_HW4 - Apache NetBeans IDE 12.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
191.6/325.0MB
Start Page
Halil_Ibrahim_Ozturk_HW4.java
HW4_Interface.java
Source History
Navigator
Projects
Services
Files
word=word.split(" ")[0];
if(word.equals(mystring)){ // if words equals each other then just frequency increase
// System.out.println("="+word+"="+mystring+"="+"++++++++++++++++++"+frequency);
frequency++;
hashArray[hashVal]=word+" "+frequency;
// System.out.println(hashArray[hashVal]);
sameword=1;
break;
}else
collusion ++;
// double hashing algorithm
hashVal = hashVal + stepSize;
hashVal =hashVal % arraySize;
}
return hashVal;
Output

HASH STRUCTURE

Index	
0	unintelligible 1
1	
2	except 3
3	
4	
5	is 13
6	

.

.

n-4	
n-3	was 3
n-2	
n-1	
n	house-front 1

This hash structure is used to trace a text file. The program got a path of a text file written in English. The program traced each word and keep the number of occurrences of each word. All punctuation marks removed. (Ex. The boy, who has green hair is walking down the street. “boy” and “street” has to be isolated from comma or dot)

INTERFACE

```
public interface HW3_Interface {
    Integer GetHash(String mystring);
    void ReadFileandGenerateHash(String filename, int size);
    void DisplayResult(String Outputfile);
    void DisplayResult();
    void DisplayResultOrdered(String Outputfile);
    int showFrequency(String myword);
    String showMaxRepeatedWord();
    boolean checkWord(String myword);
    float TestEfficiency();
}
```

Halil_Ibrahim_Ozturk_HW4 CLASS FUNCTIONS AND OUTPUTS

```
/**
 *
 * @author halilibrahim
 */
public class Halil_Ibrahim_Ozturk_HW4 implements HW4_Interface {

    String [] hashArray;

    int arraySize;
    int size =0; //counter for number of elements in hash table


    int collusion =0;       //number of cullusion occur when creating hash structure
    int hashFunc1;      //double hash functions
    int hashFunc2 ;

    int sameword; //if there is a same word when it place on index number then frequency of word
```

```
        public static void main(String[] args) {

            // TODO code application logic here

        Halil_Ibrahim_Ozturk_HW4 table = new
Halil_Ibrahim_Ozturk_HW4(); //create a new object of hash table of
Halil_Ibrahim_Ozturk_HW4 class


        table.ReadFileandGenerateHash("file.txt",5000); //Create the
//open address hash structure with the size given by the user. The file
which
//contains a very long text will be parsed and during the parsing hash
table must
//be modified by the words.
            table.DisplayResult("wordlist.txt");// All the words in the text
and
```

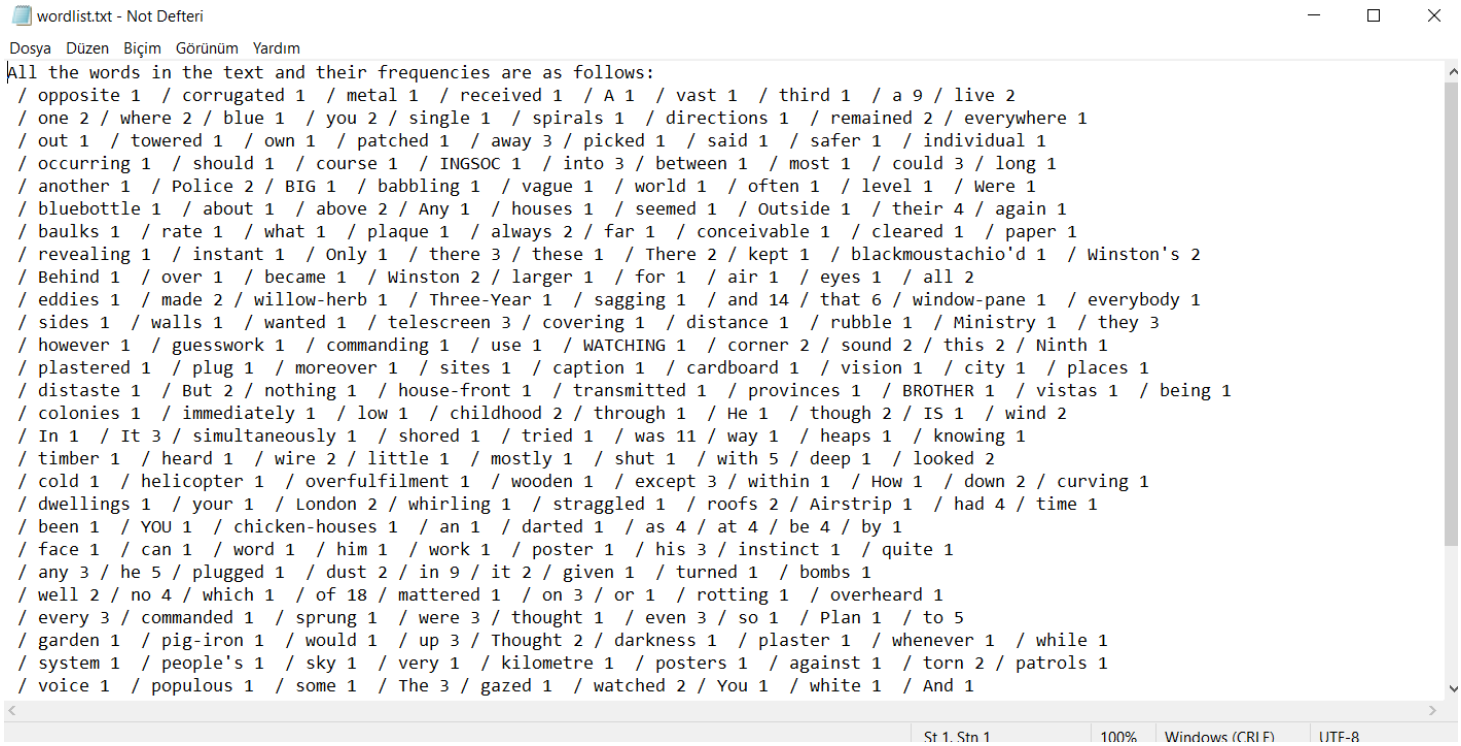
```
//their frequency has to be displayed in a text file.
        table.DisplayResult();// All the words in the text and their frequency
//has to be displayed on the screen.
        table.DisplayResultOrdered("orderedwordlist.txt");// All the words and in the
//text and their frequency has to be displayed in a text file in an ordered
//fashion. The most repeated words will be listed at the beginning and the least
//repeated words at the end
        System.out.println(table.showMaxRepeatedWord());// The most repeated word has to be
//returned.
         System.out.println("The status of the word you searched for is found in the text : ("+table.checkWord("the")+")");
         // Checks whether myword is found in the text.
        System.out.println("The frequency of the word you questioned : ("+table.showFrequency("as") + ") (If it is -1 , there is no word you questioned.) ");
//          The frequency of myword in the text
//file will be given. If there is no myword in the text -1 must be returned.
        System.out.println("There are " + table.TestEfficiency() +" collusion occured.");
// Returns the number of collusions during parsing the file.
   }
```

```java
    @Override
    public Integer GetHash(String mystring) {

        // generate an integer value (hash
//index) related to the input word. If collusion occurs the collusion has
//to be solved by double hash method.

//Double hashing used with open-addressing in hash tables to resolve hash collisions


    @Override
    public void ReadFileandGenerateHash(String filename, int size) {

        // Create the
//open address hash structure with the size given by the user. The file which
//contains a very long text will be parsed and during the parsing hash table modified by the words
```

Output - Halil_Ibrahim_Ozturk_HW4 (run)

```
Hash table size given 5000is not a prime
Hash table size changed to 5003
inserted word :Outside
inserted word :even
inserted word :through
inserted word :the
inserted word :shut
inserted word :window-pane
same word :the
inserted word :world
inserted word :looked
inserted word :cold
inserted word :Down
inserted word :in
same word :the
inserted word :street
inserted word :little
inserted word :eddies
inserted word :of
inserted word :wind
inserted word :were
inserted word :whirling
inserted word :dust
inserted word :and
inserted word :torn
inserted word :paper
inserted word :into
inserted word :spirals
same word :and
inserted word :though
same word :the
inserted word :sun
```

```java
    @Override
    public void DisplayResult(String Outputfile) {
        // All the words in the text and
//their frequency has to be displayed in a text file.
```



```
    @Override
    public void DisplayResult() {

// All the words in the text and their frequency
//has to be displayed on the screen.
```

```java
    @Override
    public void DisplayResultOrdered(String Outputfile) {

        // All the words and in the
//text and their frequency has to be displayed in a text file in an ordered
//fashion. The most repeated words will be listed at the beginning and the least
//repeated words at the end
```

orderedwordlist.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
The words in the text are in descending order according to their frequency:
<--the (37)-->
<--of (18)-->
<--and (14)-->
<--was (11)-->
<--a (9)-->
<--in (9)-->
<--that (6)-->
<--with (5)-->
<--he (5)-->
<--to (5)-->
<--their (4)-->
<--had (4)-->
<--as (4)-->
<--at (4)-->
<--be (4)-->
<--no (4)-->
<--away (3)-->
<--into (3)-->
<--could (3)-->
<--there (3)-->
<--telescreen (3)-->
<--they (3)-->
<--It (3)-->
<--except (3)-->
<--his (3)-->
<--any (3)-->
<--on (3)-->
```

St 1, Stn 1 | 100% | Windows (CRLF) | UTF-8

```java
    public int showFrequency(String myword) {

//The frequency of myword in the text
//file will be given. If there is no myword in the text -1 must be returned.
            int     frequencyofword;
```

Output - Halil_Ibrahim_Ozturk_HW4 (run)

```
Successfully wrote to the file.
 ' the '  is the most repeated word in the text.
The status of the word you searched for is found in the text : (true)
The frequency of the word you questioned : (5) (If it is -1 , there is no word you questioned.)
There are 46.0 collusion occured.
BUILD SUCCESSFUL (total time: 2 minutes 57 seconds)
```

```
    public String showMaxRepeatedWord() {
//          The most repeated word has to be
//returned.
```

```
Output - Halil_Ibrahim_Ozturk_HW4 (run)
Successfully wrote to the file.
 ' the '  is the most repeated word in the text.
The status of the word you searched for is found in the text : (true)
The frequency of the word you questioned : (5) (If it is -1 , there is no word you questioned.)
There are 46.0 collusion occured.
BUILD SUCCESSFUL (total time: 2 minutes 57 seconds)
```

```
    public boolean checkWord(String myword) {
// Checks whether myword is found in the text.
        boolean isfound=false;
        int hashVal = GetHash(myword);

        //int stepSize = hashFunc2 (myword);
```

```
Output - Halil_Ibrahim_Ozturk_HW4 (run)
Successfully wrote to the file.
 ' the '  is the most repeated word in the text.
The status of the word you searched for is found in the text : (true)
The frequency of the word you questioned : (5) (If it is -1 , there is no word you questioned.)
There are 46.0 collusion occured.
BUILD SUCCESSFUL (total time: 2 minutes 57 seconds)
```

```
public float TestEfficiency() {
/          Returns the number of collusions during parsing the file.
    return collusion;
    // collusion already calculated on GetHash method
```

```
Output - Halil_Ibrahim_Ozturk_HW4 (run)
Successfully wrote to the file.
 ' the '  is the most repeated word in the text.
The status of the word you searched for is found in the text : (true)
The frequency of the word you questioned : (5) (If it is -1 , there is no word you questioned.
There are 46.0 collusion occured.
BUILD SUCCESSFUL (total time: 2 minutes 57 seconds)
```

EXAMPLE TEXT IN TXT FILE

Outside, even through the shut window-pane, the world looked cold. Down in the street little
eddies of wind were whirling dust and torn paper into spirals, and though the sun was shining
and the sky a harsh blue, there seemed to be no colour in anything, except the posters that were
plastered everywhere. The blackmoustachio'd face gazed down from every commanding corner.
There was one on the house-front immediately opposite. BIG BROTHER IS WATCHING YOU, the
caption said, while the dark eyes looked deep into Winston's own. Down at streetlevel another
poster, torn at one corner, flapped fitfully in the wind, alternately covering and uncovering the
single word INGSOC. In the far distance a helicopter skimmed down between the roofs, hovered
for an instant like a bluebottle, and darted away again with a curving flight. It was the police
patrol, snooping into people's windows. The patrols did not matter, however. Only the Thought
Police mattered.
Behind Winston's back the voice from the telescreen was still babbling away about pig-iron and
the overfulfilment of the Ninth Three-Year Plan. The telescreen received and transmitted
simultaneously. Any sound that Winston made, above the level of a very low whisper, would be
picked up by it, moreover, so long as he remained within the field of vision which the metal
plaque commanded, he could be seen as well as heard. There was of course no way of knowing
whether you were being watched at any given moment. How often, or on what system, the
Thought Police plugged in on any individual wire was guesswork. It was even conceivable that
they watched everybody all the time. But at any rate they could plug in your wire whenever they
wanted to. You had to live -- did live, from habit that became instinct -- in the assumption that
every sound you made was overheard, and, except in darkness, every movement scrutinized.
Winston kept his back turned to the telescreen. It was safer, though, as he well knew, even a
back can be revealing. A kilometre away the Ministry of Truth, his place of work, towered vast
and white above the grimy landscape. This, he thought with a sort of vague distaste -- this was

London, chief city of Airstrip One, itself the third most populous of the provinces of Oceania. He tried to squeeze out some childhood memory that should tell him whether London had always been quite like this. Were there always these vistas of rotting nineteenth-century houses, their sides shored up with baulks of timber, their windows patched with cardboard and their roofs with corrugated iron, their crazy garden walls sagging in all directions? And the bombed sites where the plaster dust swirled in the air and the willow-herb straggled over the heaps of rubble; and the places where the bombs had cleared a larger patch and there had sprung up sordid colonies of wooden dwellings like chicken-houses? But it was no use, he could not remember: nothing remained of his childhood except a series of bright-lit tableaux occurring against no background and mostly unintelligible.