HW4 REPORT

OSMAN BAHADIR ERBEK

```
public class HW4_Hash {


	public Integer GetHash(String mystring) {
		 int sum=0;
		String myword= mystring;
		char[] dizi=myword.toCharArray();

		for(int i=0;i<myword.length();i++) {

		  char x =dizi[i];
		  int ascii = x;
		  sum += ascii;

		}
		  return sum;
	}
```

Fonksiyon içerisine giren string kelime önce boyutu kadar bir char değişkenine dönüştürülür. Bi for döngüsü içerisinde kelimenin tüm harfleri ascii değerine dönüştürülür ve bu değerler toplanır. Bu sebeple her kelimenin “yüksek ihtimalle” kendine ait bir integer değeri olmuş olur. Bu değerler hashtable’a kelimeleri yerleştirmek için kullanılacaktır.

```
	public void ReadFileandGenerateHash(String filename, int size) throws
IOException {

		  int tablesize=size;

		  Map<Integer,String> map1=new TreeMap<>();

		File f1=new File(filename);
		  String[] arr=null;
		  FileReader fr = new FileReader(f1);
		  BufferedReader br = new BufferedReader(fr);
		  String mytext;
		  String  mystring=null;
```

```
                while((mytext=br.readLine())!=null)
                {
                   mytext = mytext.toLowerCase();
                   mytext = mytext.replace("-", "");
                   mytext = mytext.replace(":", "");
                   mytext = mytext.replace(",", "");
                   mytext = mytext.replace("'", "");
                   mytext = mytext.replace("ı", "i");
                   mytext = mytext.replace("-", "");
                   mytext = mytext.replace(";", "");
                   mytext = mytext.replace("?", "");
                   mytext = mytext.replace("--", "");
                   mytext = mytext.replace(".", "");
                    arr=mytext.split(" ");


                    for(int i=0;i<arr.length;i++){
               mystring= arr[i];
               int mode= GetHash(mystring)%tablesize;
               map1.put(mode, arr[i]);
                                }
                    }
                //System.out.println(map1.entrySet());
    }
```

Bu fonksiyon kelimeleri, ascii değerlerinin hastable size'ına bölümünden kalan numaralarına yerleştirir. Maalesef bölümünden kalanları aynı olan kelimeleri birbirine bağlayamadım. Linkedlist ile bunu yapmaya çalışsamda ilk iki değerden sonrakileri birleştiremedim. Bu sebeple bu fonksiyon hashtable'da üst üste binen kelimelerden sadece en son okunanı gösterir. Algoritmayı anlatmam gerekirse;

Text dosyasındaki metini bufferreader ile satır satır okudum. Okunan her satırda satır eğer boş değilse, öncelikle metinde istemediğimiz karakterleri null karaktere çevirdim. Sonra her satırı bir array içerisinde tuttum, for döngüsü içerisinde her kelimeyi gethash metoduna gönderdim ve bana ascii değerini return etti, gelen bu ascii değerini tablesize'a bölerek bu bölümden kalan değer ile hashtable'da kaçıncı satıra yazılacağı belirlendi. Sonrasında her kelime mode(bölümden kalan) değeri ile hastable'a yerleştirildi.

```
void DisplayResult(String Outputfile) throws IOException {

            Map<String,Integer> map1=new TreeMap<>();

        File f1=new File("hw4text.txt");
           String[] arr=null;
           FileReader fr = new FileReader(f1);
           BufferedReader br = new BufferedReader(fr);
           String mytext;
           FileWriter fw = new FileWriter(Outputfile);
           while((mytext=br.readLine())!=null)
           {
              mytext = mytext.toLowerCase();
              mytext = mytext.replace("-", "");
              mytext = mytext.replace(":", "");
              mytext = mytext.replace(",", "");
```

```
                mytext = mytext.replace("'", "");
                mytext = mytext.replace("ı", "i");
                mytext = mytext.replace("-", "");
                mytext = mytext.replace(";", "");
                mytext = mytext.replace("?", "");
                mytext = mytext.replace("--", "");
                mytext = mytext.replace(".", "");
                 arr=mytext.split(" ");


                 for(int i=0;i<arr.length;i++)
                 {

                     if(map1.containsKey(arr[i]))
                     {
                         map1.put(arr[i], map1.get(arr[i])+1);
                     }
                     else
                     {
                         map1.put(arr[i],1);
                     }
                 }

                 }

                 for(Map.Entry<String,Integer> map:map1.entrySet())
                 {
                     fw.write(map.getKey()+ " - "+map.getValue());
                     fw.write("\n");
            }
        fw.close();

    }
```

Yine benzer yöntemi kullandığım bu metotta okunan her satır for döngüsü içerisinde, eğer hastable kelimeyi içerisiyorsa değeri bir artırılarak yeniden konumlandırılır “Value” aslında bize o kelimenin kaç kez tekrar ettiğini gösterir, eğer içermiyosa değeri 1 olacak şekilde table’a yerleştirilir. ContainsKey komutu ile arrayin o anki değerinin hashmapim içerisinde olup olmadığını kontrol edebiliyorum. Daha sonra hashmapi bir foreach kullanarak key ve value değerlerini sırası ile yazdırdım. Size’ı belirli olan hashtable’da collision’ları birleştiremedğim için bu yöntemi kullandım. Burada hashmap size’ı her zaman kelime sayısı kadar olacaktır.

```
void DisplayResultOrdered(String Outputfile) throws IOException {

                    List<Entry<String, Integer>> sorted = new
ArrayList<>(map1.entrySet());
                        Collections.sort(sorted, new Comparator<Entry<String,
Integer>>() {

                            public int compare(Entry<String, Integer> o1,
Entry<String, Integer> o2) {
                            int comp = Integer.compare(o2.getValue(),
o1.getValue());
                            if (comp != 0) {
                                return comp;
                            }
                            return o1.getKey().compareTo(o2.getKey());
                            }
                        }
                        );
```

Bu fonksiyon DisplayResult'ın çok benzeri olduğu için aynı kodları tekrar buraya eklemedim. Farklı olan tek şey bir comparator fonksiyonu. Bu kod parçasını internetten alıp biraz modifiye ederek kullandım. Bir linkedlist yardımıyla hashmap'in tüm değerlerinin value'ları karşılaştırılır. Büyük olan her zaman diğeriyle yer değişecek şekilde çalışır. Bu sayede tüm kelimelerin value değerlerin büyük küçüğe sıralanmış olur.

```
    String showMaxRepeatedWord() throws IOException {

              String mytext, word = "";
            int count = 0, maxCount = 0;
            ArrayList<String> words = new ArrayList<String>();

            FileReader fr = new FileReader("hw4text.txt");
            BufferedReader br = new BufferedReader(fr);
            String string[];

            while((mytext = br.readLine()) != null) {

                  mytext = mytext.toLowerCase();
                   mytext = mytext.replace("-", "");
                   mytext = mytext.replace(":", "");
                   mytext = mytext.replace(",", "");
                   mytext = mytext.replace("'", "");
                   mytext = mytext.replace("ı", "i");
                   mytext = mytext.replace("-", "");
                   mytext = mytext.replace(";", "");
                   mytext = mytext.replace("?", "");
                   mytext = mytext.replace("--", "");
                   mytext = mytext.replace(".", "");
                  string=mytext.split(" ");

                for(String s : string){
                    words.add(s);
                }
            }
```

```
            for(int i = 0; i < words.size(); i++){
            count = 1;
               for(int j = i+1; j < words.size(); j++){

                if(words.get(i).equals(words.get(j))){
                    count++;
                }
            }

            if(count > maxCount){


                maxCount = count;
                word = words.get(i);
            }
        }
          System.out.println(word);
        return word;
    }
```

Burada yine aynı şekilde text dosyasını satır satır taradım. Her satır için noktalama düzeltmelerini yaptıktan sonra kelimeleri words arraylistemin içerisine attım. 2 for döngüsü kullanarak her bir kelimeyi ikinci for vasıtasıyla aynı satır içinde tarattım. Iki array değeri aynı olduğu anda countu 1 artırdım. Her kelime için bu işlem yapılırken eğer kelimenin count'ı maxcount değerinden büyükse yeni kelime maxcount değerine sahip olan kelime oldu. 2 for döngüsüde bittikten sonra countu en yuksek olan "word" değişkeni yazdırıldı.

```
void checkWord(String myword) throws IOException {

            Map<String,Integer> map1=new TreeMap<>();

       String text = "";


            File f1=new File("hw4text.txt");
               String[] arr=null;
             FileReader fr = new FileReader(f1);
               BufferedReader br = new BufferedReader(fr);
               String mytext;

               int count=0;
             while((mytext=br.readLine())!=null)      {


                int c=0;
               for(int i=0;i<arr.length;i++)
                {
              map1.put(arr[i], c);
              c++;
                }
```

```
            }
        for(Map.Entry<String,Integer> map:map1.entrySet())
    {

        if(map1.containsKey(myword) ) {
       System.out.println(myword + " is  found "  );
       break;


        }
        else {
             System.out.println(myword+" is not found in the text");
             break;
        }

}
```

Yine aynı işlemleri yaparak satırlar tek tek okundu. Satırdaki kelimeler “map” hashmapinin içerisine atıldı. Tekrar bir for döngüsü kullanarak kullanıcı tarafından girilen “myword” değişkeni hashmap içerisinde arandı. Kelime bulunduğunda o kelimeyi yazdırdım. If komutunun sonuna break; ekledim çünkü aynı kelime metinde mevcutsa aynı kelimeyi her bulduğunda “myword is found” yazdıracağı için bundan kaçındım.