import java.math.BigInteger;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class Dictionnaire {

    private HTNaive list;

*/\*\**

*\* A: Crée une liste sous-jacente de HTNaive avec m en taille.*

*\* @param m*

*\*/*

    public Dictionnaire(int m){

        this.list = new HTNaive(m);

    }

*/\*\**

*\* A: Constructeur de la classe*

*\* @param filename*

*\* @param m*

*\*/*

    public Dictionnaire(String filename, int m){

        this.list = new HTNaive(Dictionnaire.calculeListeInt(filename), m);

    }

*/\*\**

*\* A: Constructeur de la classe*

*\*/*

    public Dictionnaire(String filename, double f){

        this.list = new HTNaive(Dictionnaire.calculeListeInt(filename), f);

    }

*/\*\**

*\* A/R: Convertie un String en int avec les valeurs ascii.*

*\* @param s*

*\* @return*

*\*/*

    private static BigInteger stringToBigInteger(String s){

        BigInteger resultat = new BigInteger("0");

        BigInteger multiplieur = new BigInteger("1");

        for(int i=s.length()-1; 0<=i; i--){

            resultat = resultat.add(BigInteger.valueOf((int) s.charAt(i)).multiply(multiplieur));

            multiplieur = multiplieur.multiply(BigInteger.valueOf(256));

        }

        return resultat;

    }

*/\*\**

*\* A/R: Ajoute un mot convertie en ASCII*

*\* @param s*

*\* @return*

*\*/*

    public boolean ajout(String s){

        boolean resultat = false;

        BigInteger val = Dictionnaire.stringToBigInteger(s);

        if(this.list.ajout(val) == true){

            resultat = true;

        }

        return resultat;

    }

*/\*\**

*\* A/R: Retourne vrai si et seulement si s est dans le dictionnaire*

*\* @param s*

*\* @return*

*\*/*

    public boolean contient(String s){

        return this.list.contient(stringToBigInteger(s));

    }

*/\*\**

*\* A/R: Retourne le nombre d'éléments sotckés dans la table.*

*\* @return*

*\*/*

    public int getCardinal(){

        return this.list.getCardinal();

    }

*/\*\**

*\* A/R: Retourne la taille de la liste la plus longue.*

*\* @return*

*\*/*

    public int getMaxSize(){

        return this.list.getMaxSize();

    }

*/\*\**

*\* A/R: Retourne le nombre de listes vides ou non de la table.*

*\* @return*

*\*/*

    public int getNbListes(){

        return this.list.getNbListes();

    }

*/\*\**

*\* R: Retourne un String qui affiche tous les éléments du tableau de hash.*

*\*/*

    public String toString(){

        return this.list.toString();

    }

*/\*\**

*\* A/R: Retourne un string qui affiche le nom d'éléments présent dans chaque liste.*

*\* @return*

*\*/*

    public String toStringV2(){

        return this.list.toStringV2();

    }

*/\*\**

*\**

*\* @return*

*\*/*

    public int lectureMotsTexte(String filename){

*//on suppose que fichier.txt est un fichier dans le meme dossier*

*//que les .java*

        File f = new File(filename);

        ListeBigI res = new ListeBigI();

        Scanner sc;

*//un scanner est un objet permettant de "scanner" (parcourir)*

*//une entrée (clavier, ou une chaîne, ou un File, etc)*

        try {

            sc = new Scanner(f);

*//ici on construit le scanner avec comme entrée f*

*//cette construction peut échouer (si par exemple fichier.txt n’existe pas)*

        }catch(FileNotFoundException e){

*//si la construction échoue, on passe ici*

            System.out.println(("problème d’accès au fichier " + e.getMessage()));

            return 0;

        }

        sc.useDelimiter(" |\\n|,|;|:|\\.|!|\\?|-");

*//on définit les délimiteurs comme le caractère ’\n’, le caractère ’,’ etc...*

*//cela définit maintenant la notion de "morceau" comme une suite*

*//de caractères entre deux délimiteurs*

        int nbmots = 0;

        while (sc.hasNext()) { *//sc.hasNext() renvoie vrai ssi*

*//il reste encore un morceau à découvrir dans f*

            String mot = sc.next(); *//sc.next() renvoie le prochain morceau*

            nbmots ++;

        }

        return nbmots;

    }

    public static ListeBigI calculeListeInt(String fileName){

        File f = new File(fileName);

        ListeBigI res = new ListeBigI();

        Scanner sc;

        try {

            sc = new Scanner(f);

        } catch (FileNotFoundException e) {

            System.out.println(("problème d’accès au fichier " + e.getMessage()));

            return null;

        }

        sc.useDelimiter(" |\\n|,|;|:|\\.|!|\\?|-");

        String mot;

        while (sc.hasNext()) {

            mot = sc.next();

            res.ajoutTete(Dictionnaire.stringToBigInteger(mot));

        }

        sc.close();

        return res;

    }

    public long getTotalTimeh() {

        return this.list.getTotalTimeh();

    }

    public long getTotalTimeContient() {

        return this.list.getTotalTimeContient();

    }

}