package td8;

public class EE {

    private int[] ensTab;

    private int cardinal;

    public EE(int[] ensTab, int cardinal) {

        this.ensTab = ensTab;

        this.cardinal = cardinal;

    }

    public int[] getEnsTab() {

        return this.ensTab;

    }

    public void setEnsTab(int[] ensTab) {

        this.ensTab = ensTab;

    }

    public int getCardinal() {

        return this.cardinal;

    }

    public void setCardinal(int cardinal) {

        this.cardinal = cardinal;

    }

    public EE ensTab(int[] ensTab) {

        setEnsTab(ensTab);

        return this;

    }

    public EE cardinal(int cardinal) {

        setCardinal(cardinal);

        return this;

    }

    public String getEnsTabString() {

        String tab = "{";

        for (int i = 0; i < this.cardinal - 1; i++) {

            tab = tab + this.ensTab[i] + " , ";

        }

        tab = tab + this.ensTab[this.cardinal - 1] + "}";

        return tab;

    }

    public EE(int max) {

        this.ensTab = new int[max];

        this.cardinal = 0;

    }

    public EE(int max, int[] tab) {

        this.ensTab = new int[max];

        for (int i = 0; i < max; i++) {

            this.ensTab[0] = tab[0];

        }

        this.cardinal = max;

    }

    public EE(EE ensemble) {

        this(ensemble.cardinal, ensemble.ensTab);

    }

    public EE(String ensemble) {

    }

    private int contientPratique(int n) {

        int i = 0;

        if(estVide()){

            return -1;

        }

        while ( i < this.cardinal-1 && ! ( this.ensTab[i] == n ) ) {

            i++;

        }

        if (this.ensTab[i] == n) {

            return i;

        } else

            return -1;

    }

    public boolean contient(int n){

        boolean c = !(this.contientPratique(n) == -1);

        return c;

    }

    private void ajoutPratique(int n){

        if ( ! deborde() && ! ( this.contient(n) ) ){

            this.ensTab[this.cardinal] = n;

            this.cardinal++;

        }

    }

    public int retraitPratique(int n){

        int a = 0;

        if ( 0 <= n && n <= this.cardinal){

            this.cardinal--;

            a = this.ensTab[n] ;

            for (int i = n; i < this.cardinal ; i++ ){

                this.ensTab[i] = this.ensTab[i+1];

            }

        }

        return a;

    }

    public boolean estVide(){

        return (this.cardinal <= 0);

    }

    public boolean deborde(){

        return (this.cardinal >= this.ensTab.length);

    }

    public boolean retraitElt(int n){

        boolean a = false;

        if(contient(n)){

            retraitPratique(contientPratique(n));

            a = true;

        }

        return a;

    }

    public int ajoutElt(int n){

        int res = -1;

        if (contient(n)){

            res = 0;

        }else if(deborde()){

        }else{

            ajoutPratique(n);

            res = 1;

        }

        return res;

    }

    public boolean estInclus(EE ens){

        boolean a = true;

        if (this.cardinal <= ens.cardinal){

            for (int i = 0; i < this.cardinal ;  i++ ){

                a = a && ens.contient(this.ensTab[i]);

            }

            return a;

        }else{ return false ;}

    }

    public boolean estEgal(EE ens){

        boolean a = true;

        if (this.cardinal == ens.cardinal){

            for (int i = 0; i < ens.cardinal ;  i++ ){

                a = a && ens.contient(this.ensTab[i]);

            }

        }else {a = false;}

        return a;

    }

    public boolean estDisjoint(EE ens){

        boolean a = true;

        for (int i = 0; i < this.cardinal - 1;  i++ ){

            a = a && ! (ens.contient(this.ensTab[i]));

        }

        return a;

    }

    public EE intersection(EE ens){

        int i\_inter = 0 ;

        EE inter = new EE (this.ensTab.length);

        for (int i = 0; i < this.cardinal && i < ens.cardinal ; i++){

            if (ens.contient(this.ensTab[i])){

                inter.ensTab[i\_inter] = this.ensTab[i];

                i\_inter++;

                inter.cardinal++;

            }

        }

        return inter;

    }

    public EE reunion(EE ens){

        int i\_inter = 0 ;

        EE inter = new EE (this.ensTab.length + ens.ensTab.length);

        for (int i = 0; i < this.cardinal;i++){

            inter.ensTab[i\_inter] = this.ensTab[i];

            i\_inter++;

            inter.cardinal++;

        }

        for (int i = 0; i < ens.cardinal;i++){

            if ( ! ( inter.contient(ens.ensTab[i]) ) ){

                inter.ensTab[i\_inter] = ens.ensTab[i];

                i\_inter++;

                inter.cardinal++;

            }

        }

        return inter;

    }

    public EE differecence(EE ens){

        EE inter = new EE (0);

        inter = this;

        for ( int i = 0 ; i < ens.cardinal; i++){

            if(inter.contient(ens.ensTab[i])){

                inter.retraitElt(ens.ensTab[i]);

            }

        }

        return inter;

    }

    public int retraitEltAleatoirement() {

*// Pré-requis : ensemble this est non vide*

*// Résultat/action : enlève un élément de this (aléatoirement) et le renvoie*

        int i = Java.Ut.randomMinMax (0, this.cardinal - 1);

        int select = retraitPratique(i);

        return select;

     }

     public int selectionEltAleatoirement() {

*// Pré-requis : ensemble this est non vide*

*// Résultat : un élément quelconque de this choisi aléatoirement*

        int i = Java.Ut.randomMinMax (0, this.cardinal - 1);

        return this.ensTab[i];

   }

   public int selectionElt() {

*// Pré-requis : ensemble this est non vide*

*// Résultat : un élément quelconque de this (le dernier, par commodité)*

       return this.ensTab[this.cardinal - 1];

   }

}