import java.util.\*;

class Bingo{

// Opprettar 2 mengder med 75 bingokuler i kvar.

// Testar om ei spesiell bingokule er i den eine mengda,

// og om dei to mengdene er nøyaktig like (noko dei bør vera)

public static void main(String[] a){

final int ANTAL\_BALLAR=75;

TabellMengde2 <Bingokule>miMengde1 = new TabellMengde2<Bingokule>();

TabellMengde2 <Bingokule>miMengde2 = new TabellMengde2<Bingokule>();

Bingokule kule1 = null;

Bingokule kule2 = null;

for(int i=1; i <= ANTAL\_BALLAR; i++){

kule1 = new Bingokule(i);

kule2 = new Bingokule(ANTAL\_BALLAR+1-i);

miMengde1.leggTil(kule1);

miMengde2.leggTil(kule2);

}

System.out.println("\nAntal kuler totalt: " + miMengde1.antal());

System.out.println();

kule1 = new Bingokule(10);

if(miMengde1.inneheld(kule1))

System.out.println("kule 1 funne i mengde 1");

if(miMengde1.erLik(miMengde2))

System.out.println("Like mengder");

}

}

class Bingokule{

// Til å representera 'bingo-verdiar' B10, N43, ... osv

//

private char bokstav;

private int tal;

public Bingokule(int verdi){

// Konstruktør med parameter mellom 1 og 75

//

tal = verdi;

if(verdi<=15)

bokstav = 'B';

else if (verdi<=30)

bokstav = 'I';

else if (verdi<=45)

bokstav = 'N';

else if (verdi<=60)

bokstav = 'G';

else

bokstav = 'O';

}

public boolean equals(Object k2){

// Til å testa om to bingokuler har same verdi

//

Bingokule b2 = (Bingokule)k2; // Nødvendig typekonvertering

return (tal== b2.tal && bokstav== b2.bokstav);

}

public String toString(){

// String-representasjon av bingokule (til utskrift)

return bokstav + " " + tal;

}

}

class TabellMengde2<T> implements MengdeADT<T>{

// ADT-en Mengde implementert som tabell (og nesten alle metodar med)

//

private static Random tilf = new Random();

private final int START\_KAPASITET = 100;

private int antal;

private T[] tab;

public TabellMengde2(){

antal = 0;

tab = (T[])(new Object[START\_KAPASITET]);

}

public TabellMengde2(int start){

antal = 0;

tab = (T[])(new Object[start]);

}

public int antal(){

return this.antal;

}

public boolean erTom(){

return (antal==0);

}

public void leggTil(T element){

if(!inneheld(element)){

if (antal==tab.length){

utvidKapasitet();

}

tab[antal] = element;

antal++;

}

}

private void utvidKapasitet(){

T[] hjelpetabell = (T[])(new Object[2\*tab.length]);

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

hjelpetabell[i] = tab[i];

}

tab = hjelpetabell;

}

public T fjernTilfeldig(){

T svar = null;

if (antal>0){

int indeks = tilf.nextInt(antal);

svar = tab[indeks];

tab[indeks] = tab[antal-1];

antal--;

}

return svar;

}

public T fjern(T element){

// Søkjer etter og fjernar element. Retur med null ved ikkje-funn

//

int pos = -1;

T svar = null;

if(!erTom()){

for(int i=0; (i<antal)&&(pos==-1); i++){

if(tab[i].equals(element))

pos = i;

}

if(pos!=-1){

svar = tab[pos];

tab[pos] = tab[antal-1];

tab[antal-1] = null;

antal--;

}

}

return svar;

}

public MengdeADT<T> union(MengdeADT<T> m2){

TabellMengde2<T> begge = new TabellMengde2<T>();

for (int i = 0; i<antal; i++)

begge.leggTil(tab[i]);

Iterator<T> teljar = m2.oppramsar();

while (teljar.hasNext())

begge.leggTil(teljar.next());

return begge;

}

public boolean inneheld(T element){

int pos = -1;

for(int i = 0; (i<antal)&&(pos==-1); i++)

if(tab[i].equals(element))

pos = i;

return(pos!=-1);

}

public boolean erLik(MengdeADT m2){

boolean likeMengder = true;

T element;

if(antal()==m2.antal()){

Iterator<T> teljar = m2.oppramsar();

while (likeMengder && teljar.hasNext()){

element = teljar.next();

if(!this.inneheld(element)){

likeMengder = false;

}//if

}//while

}//if

else{

likeMengder = false;

}

return likeMengder;

}

public void leggTilAlle(MengdeADT m2){

Iterator<T> teljar = m2.oppramsar();

while (teljar.hasNext())

leggTil(teljar.next());

}

public Iterator oppramsar(){

return new TabellIterator(tab, antal);

}

public String toString(){

String resultat = "";

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

resultat = resultat + tab[i].toString() + "\n";

}

return resultat;

}

}