Земи

Вие сте председател на поземлена комисия. Данните, които предстои да обработите, представляват претенции на собственици върху земни парцели: правоъгълници, чиито страни са разположени изток-запад и север-юг. На Вашата карта те се характеризират с по две двойки цели неотрицателни числа, не по-големи от 1000 – координати на северозападния ъгъл и координати на югоизточния (север, запад), (юг, изток). Всеки кандидат-собственик е записан със собствено и фамилно име и претендира за един парцел.

Осигурете въвеждането на данните от стандартния вход. Изведете списък на кандидат-собствениците, подредени по намаляваща площ, върху която имат претенции, а тези, чиито претенции са с равни площи – по азбучен ред на фамилните имена. Списъкът трябва да има следния вид: номер по ред (дясно подравнен в 3 символа, водещи интервали), точка, интервал, име и фамилия, двоеточие, интервал, площ на парцела. Изведете списък на собствениците, чиито претенции не конфликтират с други и могат да бъдат удовлетворени веднага, в следния вид: номер по ред (дясно подравнен в 3 символа, водещи интервали), точка, интервал, съкращение на името (до първата гласна след първата буква), точка, интервал, фамилно име, интервал, координати на северозападния ъгъл на парцела, заградени в скоби и разделени със запетая и интервал, интервал, аналогичен запис на координатите на югоизточния ъгъл. Списъкът да е подреден по координатите на северозападния ъгъл, като парцел с по-голяма северна координата е по-напред, а при еднакви северни координати по-напред е този, който има по-малка западна координата. Изведете накрая и размера на цялата площ, върху която има претенции.

ПРИМЕРНИ ДАННИ: Нека имате четирима претенденти: Иван Тонев за парцел (0, 200), (200, 0); Митьо Данев за парцел (20,100), (500,0), Кольо Минев за парцел (300, 120), (320, 110) и Исмаил Димов за парцел (200, 200), (700, 120). Тогава изходите са:

Списък 1:

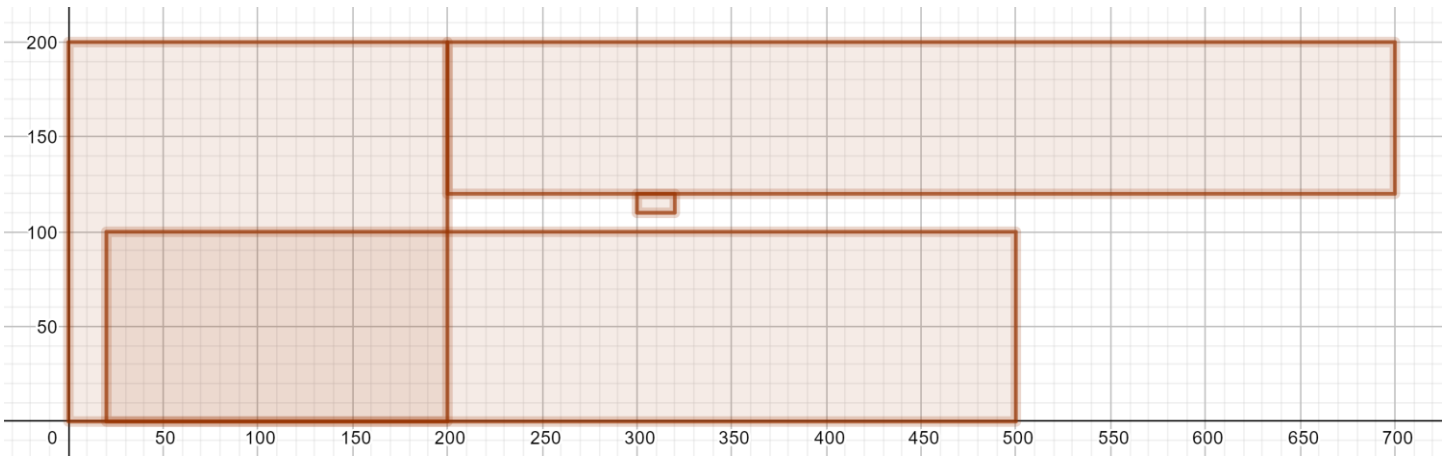
```
  1. Митьо Данев: 48000
  2. Исмаил Димов: 40000
  3. Иван Тонев: 40000
  4. Кольо Минев: 200
```

Списък 2:

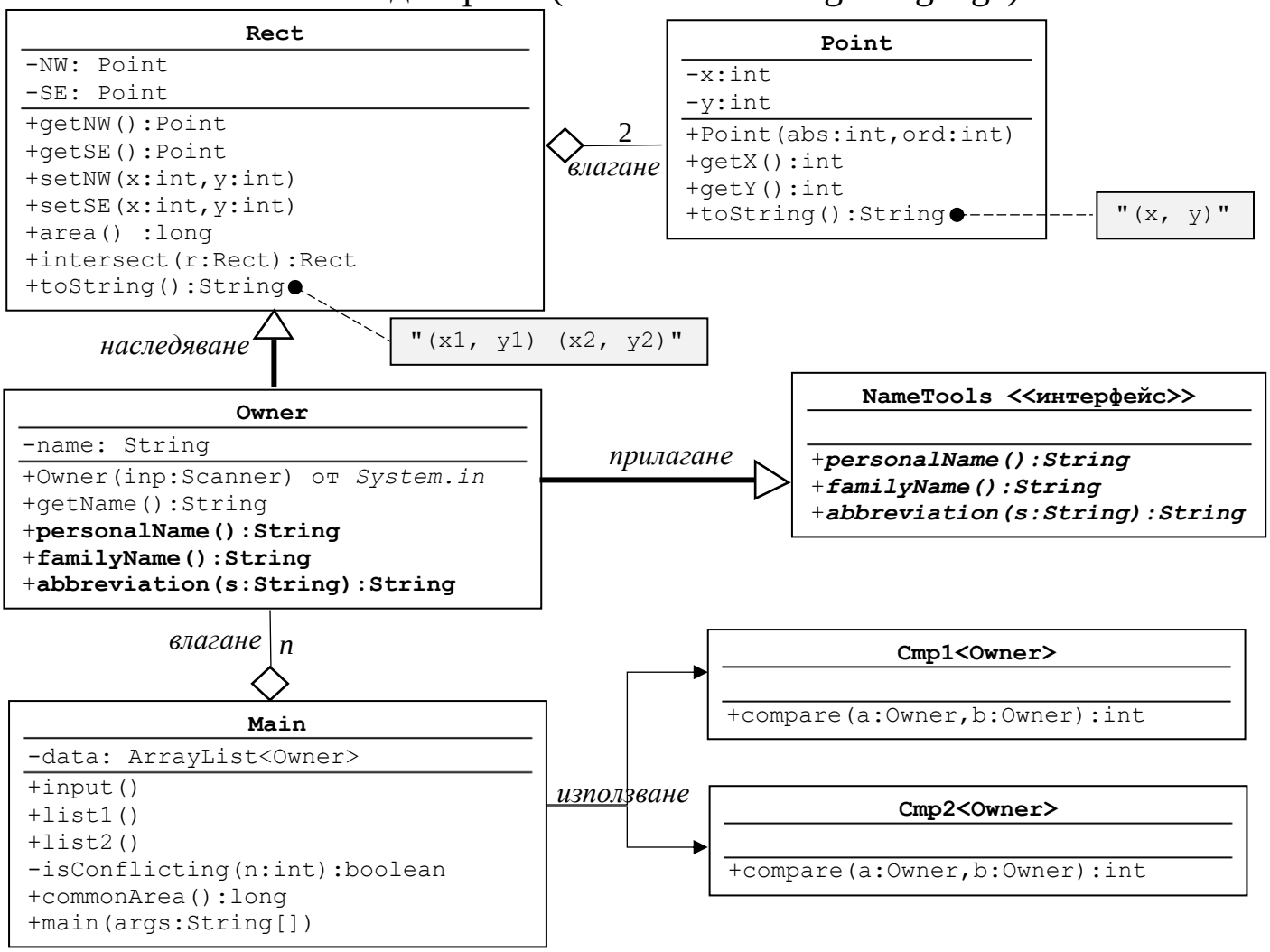
```
  1. Исм. Димов (200, 200) (700, 120)
  2. К. Минев (300, 120) (320, 110)
```

Обща претендирана площ:

```
110200
```



UML-диаграма (**U**nified **M**odeling **L**anguage)



- ✓ наследяване: *extending* (inheriting)
- ✓ влагане: *composing* (embedding)
- ✓ прилагане: *implementing*
- ✓ използване: *importing* (using)

Реализация

```
public class Point {
 //Клас "Точка"
 //Свойства: целочислени координати
 private int x,y;
 //Контсруктор по коорсинати
 public Point (int a,int b){
  x=a;
  y=b;
 }
 //Гетъри
 public int getX(){
  return x;
 }
 public int getY(){
  return y;
 }
 //Низов образ
 @Override
 public String toString(){
  return "("+x+", "+y+")";
 }
}
```

```java
public class Rect {
 //Клас "Правоъгълник"
 private Point NW,SE; //Вложени обекти от тип "Точка" -
                      // северозападен и югоизточен ъгъл
 //Гетъри
 public Point getNW(){
  return NW;
 }
 public Point getSE(){
  return SE;
 }
 //Сетъри
 public void setNW(int x,int y){
  NW=new Point(x,y);
 }
 public void setSE(int x,int y){
  SE=new Point(x,y);
 }
 long area(){//Площ
  return (long)(SE.getX()-NW.getX())*(NW.getY()-SE.getY());
 }
 //Правоъгълник на пресичане (null, ако this и r не се пресичат)
 public Rect intecsect(Rect r){
   //Кога не се пресичат:
   if (SE.getX()<=r.NW.getX()||NW.getX()>=r.SE.getX()||
       SE.getY()>=r.NW.getY()||NW.getY()<=r.SE.getY()) return null;
   Rect d=new Rect();//Резултатен правоъгълник
   //Определяне на северозападния и югоизточния ъгъл на резултата
   d.setNW(Integer.max(NW.getX(), r.NW.getX()),Integer.min(NW.getY(),r.NW.getY()));
   d.setSE(Integer.min(NW.getX(), r.NW.getX()),Integer.max(NW.getY(),r.NW.getY()));
   return d;
 }
 @Override
 public String toString(){
  return NW.toString()+" "+SE.toString();
 }
}
```

```java
import java.util.Scanner;
public class Owner extends Rect implements NameTools {
 //Клас "Собственик", наследник на "Правоъгълник",
 // реализиращ инструменти за работа с имена (NameTools)
 private String name;//Ново свойство: име
 //Конструктор от потока inp, stdInput=true, ако
 // inp е върху System.in, stdInput=false ако inp е върху файл
 public Owner(Scanner inp, boolean stdInput){
  //Подсаказки - само ако е от стандартния вход
  if (stdInput) System.out.print("Име и презиме: ");
  name=inp.nextLine(); //Предполагаме, че входният буфер е чист!
  int x,y;
  if (stdInput)System.out.println("Северозападен ъгъл:");
  if (stdInput)System.out.print("X: ");
  x=inp.nextInt();
  if (stdInput)System.out.print("Y: ");
  y=inp.nextInt();
  setNW(x,y);
  if (stdInput)System.out.println("Югоизточен ъгъл:");
  if (stdInput)System.out.print("X: ");
  x=inp.nextInt();
  if (stdInput)System.out.print("Y: ");
  y=inp.nextInt();
  setSE(x,y);
  inp.nextLine();//Оставяме входния буфер чист!
 }
 //Гетър за име
 public String getName(){
  return name;
 }
 //Инструменти за работа с имена
 @Override
 public String personalName() {//Лично име - от name до интервал
```

```
  return name.substring(0,name.indexOf(' '));
 }
 @Override
 public String familyName() {//Фамилно име - от name, всичко след интервал
  return name.substring(name.indexOf(' ')+1,name.length());
 }
}
```

```
public interface NameTools {
 //Инструменти за работа с имена
 public String personalName();//Лично име
 public String familyName();  //Фамилно има
 //По принцип така се извършва съкращаване на низ,
 // но класовете, които прилагат този интерфейс,
 // могат да променят това правило: default
 default String abbreviation(String s){//Съкращение на име(на)
  final class Vowel{
   boolean isVowel(char c){
    String bVowels="аеиоуъюя";//българските гласни като малки букви
    bVowels+=bVowels.toUpperCase();//и като главни
    return bVowels.indexOf(c)>=0;//Гласна е, ако е измежду тях
   }
  };
  final Vowel v=new Vowel();
  //Търсим съгласна, следвана от гласна
  for (int i=0;i<s.length()-2;i++)
   if (!v.isVowel(s.charAt(i))&&v.isVowel(s.charAt(i+1)))
   {if (s.charAt(i)!='ь') return s.substring(0,i+1)+".";
    return s.substring(0,i)+"."; //"Гьоре" да става "Г."
   }
  return s;//Някои имена не се съкращават: Ая, Ан, Иво, Анка
 }
}
```

```
import java.util.Comparator;
public class Cmp1 implements Comparator<Owner> {
//Наредба намаляващо по площ, а при еднакви площи -
// в растящ азбучен ред на фамилното име
 public int compare(Owner a,Owner b){
  if (a.area()>b.area()) return -1;//Правилна подредба по първи критерий
  if (a.area()<b.area()) return 1; //Неправилна подредба по първи критерий
  //При равенство - подредба по втори критерий
  return a.familyName().compareTo(b.familyName());
 }
}
```

```
import java.util.Comparator;
public class Cmp2 implements Comparator<Owner> {
 //Наредба по северозападен ъгъл
 public int compare(Owner a,Owner b){
  int t=a.getNW().getY()-b.getNW().getY();//Разлика между ординатите
                                          // (може, защото типът е int)
  if (t!=0) return -t;//Намаляващо, ако не са равни
  //При равенство - растящо по абсцисите
  return a.getNW().getX()-b.getNW().getX();
 }
}
```

```
import java.util.Scanner;
import java.util.ArrayList;
import java.io.File;
public class Main {
 static final boolean stdInput=false; //Дали входът е от стандартния вход
 private static ArrayList<Owner> data=new ArrayList<Owner>();
 //Дали n-тият собственик конфликтира с някого
 private static boolean isConflicting(int n){
  for (int i=0;i<data.size();i++) if (i!=n) {
   if (data.get(i).intecsect(data.get(n))!=null) return true;
  }
  return false;
 }
 public static void input(){
  Scanner inp=null;
```

```
  File f=null;
  if (stdInput) inp=new Scanner(System.in);
  else f= new File("Data.txt");
  try{
   if (!stdInput) inp=new Scanner(f);
   //Подсказки само ако входът е System.in
   if (stdInput) System.out.print("Брой собственици: ");
   int n=inp.nextInt();
   inp.nextLine();//Зачистване на входния буфер!
   for (int i=0;i<n;i++){
    if (stdInput) System.out.println("Данни за собственик № "+(i+1));
    Owner owner=new Owner(inp,stdInput);//Създаване на собственик от поток inp
    data.add(owner);//Добавяне към динамичния масив
   }
   inp.close();
  }catch(Exception e){
        System.out.println("File not found");
        System.exit(0);//Изход от процеса
  }
 }
 static void list1(){//Първи списък
  data.sort(new Cmp1());//Сортиране по критериите
  for (int i=0;i<data.size();i++){//Извеждане на списъка
   if (i<10) System.out.print(' ');
   if (i<100) System.out.print(' ');
   System.out.println((i+1)+". "+data.get(i).getName()+": "+data.get(i).area());
  }
 }
 static void list2(){//Втори списък
  data.sort(new Cmp2());//Сортиране по критериите
  for (int i=0,j=1;i<data.size();i++) if (!isConflicting(i)){//Филтриране и
                                                            //извеждане на списъка
   if (i<10) System.out.print(' ');
   if (i<100) System.out.print(' ');
   System.out.println(j+++". "+
                      data.get(i).abbreviation(data.get(i).personalName())+" "+
                          data.get(i).familyName()+" "+data.get(i).toString());
  }
 }
 static long commonArea(){//Обща претендирана площ
  long s=0;//Суматор
  int minx=1000,miny=1000,maxx=0,maxy=0;//Обхващащ правоъгълник
  for (Owner owner:data){
      if (owner.getNW().getX()<minx)minx=owner.getNW().getX();
      if (owner.getNW().getY()>maxy)maxy=owner.getNW().getY();
      if (owner.getSE().getX()>maxx)maxx=owner.getSE().getX();
      if (owner.getSE().getY()<miny)miny=owner.getSE().getY();
  }
  for (int x=minx;x<maxx;x++)
   for (int y=maxy;y>miny;y--){
      Rect r=new Rect();//Създаване на единично квадратче
      r.setNW(x, y);
      r.setSE(x+1, y-1);
      for (int i=0;i<data.size();i++) //Проверка за принадлежност
       if (r.intecsect(data.get(i))!=null){//Ако принадлежи на някой парцел...
        s++;                               //...добавяме към площта,...
        break;                             //...но само веднъж.
       }
   }
  return s;
 }
 public static void main(String[] args) {
 //повикване на метод за всяка подзадача
  input();
  list1();
  list2();
  System.out.println(commonArea());
 }
}
```