Ai\_7实现KNN算法

KNN算法：求出被测试例子和之前确定例子的距离，然后按照距离排序，选取最小的前K个，这K个里面占百分比最多的类型就是被测试例子的类型

代码实现：

**package** KNN;

**public** **class** PointBean {

**int** x;

**int** y;

**public** **int** getX() {

**return** x;

}

**public** **void** setX(**int** x) {

**this**.x = x;

}

**public** **int** getY() {

**return** y;

}

**public** **void** setY(**int** y) {

**this**.y = y;

}

**public** PointBean(**int** x, **int** y) {

**super**();

**this**.x = x;

**this**.y = y;

}

**public** PointBean() {

**super**();

}

@Override

**public** String toString() {

**return** "PointBean [x=" + x + ", y=" + y + "]";

}

}

**package** KNN;

**import** java.util.ArrayList;

**public** **class** KnnMain {

**public** **double** getPointLength(ArrayList<PointBean> list,PointBean bb){

**int** b\_x=bb.getX();

**int** b\_y=bb.getY();

**double** temp=(b\_x -list.get(0).getX())\*(b\_x -list.get(0).getX())+

(b\_y -list.get(0).getY())\*(b\_y -list.get(0).getY());

// 找出最小的距离

**for**(**int** i=1;i<list.size();i++){

**if**(temp<((b\_x -list.get(i).getX())\*(b\_x -list.get(i).getX())+

(b\_y -list.get(i).getY())\*(b\_y -list.get(i).getY()))){

temp=(b\_x -list.get(i).getX())\*(b\_x -list.get(i).getX())+

(b\_y -list.get(i).getY())\*(b\_y -list.get(i).getY());

}

}

**return** Math.*sqrt*(temp);

}

**public** **void** getContent(ArrayList<PointBean> list1,ArrayList<PointBean> list2,

ArrayList<PointBean> list3,PointBean bb){

**double** A=getPointLength(list1,bb);

**double** B=getPointLength(list2,bb);

**double** C=getPointLength(list3,bb);

//做出比较

**if**(A>B){

**if**(B>C){

System.***out***.println("这个点:"+bb.getX()+" , "+bb.getY()+" " +"属于C");

}**else** {

System.***out***.println("这个点:"+bb.getX()+" , "+bb.getY()+" " +"属于B");

}

}**else** {

**if**(A>C){

System.***out***.println("这个点:"+bb.getX()+" , "+bb.getY()+" " +"属于C");

}**else** {

System.***out***.println("这个点:"+bb.getX()+" , "+bb.getY()+" " +"属于A");

}

}

}

}

**package** KNN;

**import** java.util.ArrayList;

/\*

\* 主函数 KNN

\*/

**public** **class** TestJava {

**static** ArrayList< PointBean> *listA*;

**static** ArrayList< PointBean> *listB*;

**static** ArrayList< PointBean> *listC*;

**static** ArrayList< PointBean> *listD*;

**public** **static** **void** main(String[] args) {

//创建Arraylist

*listA*=**new** ArrayList<PointBean>();

*listB*=**new** ArrayList<PointBean>();

*listC*=**new** ArrayList<PointBean>();

*listD*=**new** ArrayList<PointBean>();

//写入数据

*setDate*();

*getTestResult*();

}

/\*\*

\* 得到结果

\*/

**private** **static** **void** getTestResult() {

//创建对象

KnnMain km=**new** KnnMain();

**for**(**int** i=0;i<*listD*.size();i++){

km.getContent(*listA*, *listB*, *listC*, *listD*.get(i));

}

}

/\*\*

\* 写入数据

\*/

**private** **static** **void** setDate() {

//A的坐标点

**int** A\_x[]={1,1,2,2,1};

**int** A\_y[]={0,1,1,0,2};

//B的坐标点

**int** B\_x[]={2,3,3,3,4};

**int** B\_y[]={4,4,3,2,3};

//C的坐标点

**int** C\_x[]={4,5,5,6,6};

**int** C\_y[]={1,2,0,2,1};

// 测试数据

//B的坐标点

**int** D\_x[]={3,3,3,0,5};

**int** D\_y[]={0,1,5,0,1};

//

PointBean bA;

**for**(**int** i=0;i<5;i++){

bA=**new** PointBean(A\_x[i], A\_y[i]);

*listA*.add(bA);

}

//

PointBean bB ;

**for**(**int** i=0;i<5;i++){

bB=**new** PointBean(B\_x[i], B\_y[i]);

*listB*.add(bB);

}

//

PointBean bC ;

**for**(**int** i=0;i<5;i++){

bC=**new** PointBean(C\_x[i], C\_y[i]);

*listC*.add(bC);

}

//

PointBean bD ;

**for**(**int** i=0;i<5;i++){

bD=**new** PointBean(D\_x[i], D\_y[i]);

*listD*.add(bD);

}

}

}

实现结果：