Nombre y Apellidos: **Iván José Alba García**

---

idealista i+d ha desarrollado un sencillo driver para un drone (vehículo aéreo no tripulado). Con este dispositivo se desea sobrevolar un conjunto de urbanizaciones (definidas por un identificador) con el objetivo de conocer cuántos chalets tienen piscina.

El equipo de cartografía indicará al equipo de desarrollo de idealista i+d unas coordenadas y un rango a sobrevolar (número entero que representa un área con un tamaño determinado), pero al drone sólo se le podrá proporcionar una colección de urbanizaciones a visitar.

...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	id urb_1	id urb_2	id urb_3	id urb 4	id urb 5	...	...
...	...	id urb 6	id urb_7	id urb_8	id urb_9	id urb_{10}	...	...
...	...	id urb_{11}	id urb_{12}	ORIGEN (id urb_{13})	id urb_{14}	id urb_{15}	...	...
...	...	id urb_{16}	id urb_{17}	id urb_{18}	id urb_{19}	id urb_{20}	...	...
...	...	id urb_{21}	id urb_{22}	id urb_{23}	id urb_{24}	id urb_{25}	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

NOTA: el área azul representa las urbanizaciones encerradas en el rango 1, el área verde claro representa las urbanizaciones encerradas en el rango 2 y así sucesivamente.

idealista i+d ya ha programado dos funciones del driver:

- ***obtenerIdentificadorUrbanización***: dadas unas coordenadas, devuelve el identificador de la urbanización en la que están encerradas dichas coordenadas.
- ***obtenerAdyacente***: dado un identificador y la dirección de adyacencia devuelve el identificador de la urbanización adyacente.

obtenerIdentificadorUrbanización(coordenadaX, coordenadaY) = identificadorUrbanización

obtenerAdyacente(identificadorUrbanizaciónOrigen, dirección) = identificadorAdyacente

las direcciones de adyacencia posibles son: ARRIBA, ABAJO, DERECHA e IZQUIERDA

ejemplo de uso de las funciones:

obtenerIdentificadorUrbanización(38.56889, 40.511107) = id urbanización$_{13}$ (la marcada como "ORIGEN" en color verde)

obtenerAdyacente(id urbanización$_{13}$, ARRIBA) = id urbanización$_{8}$
obtenerAdyacente(id urbanización$_{13}$, ABAJO) = id urbanización$_{18}$
obtenerAdyacente(id urbanización$_{13}$, DERECHA) = id urbanización$_{14}$
obtenerAdyacente(id urbanización$_{13}$, IZQUIERDA) = id urbanización$_{12}$

Te necesitamos para que programes (en el lenguaje que desees) una función que dadas unas coordenadas y un rango, nos devuelva el identificador de todos las urbanizaciones que deberá visitar el drone.

obtenerUrbanizaciónes(*38.56889, 40.511107, 1*) = [*id urbanización$_{7,}$ id urbanización$_{8,}$ id urbanización$_{9,}$ id urbanización$_{12,}$ id urbanización$_{13,}$ id urbanización$_{14,}$ id urbanización$_{17,}$ id urbanización$_{18.}$ id urbanización$_{19}$]*

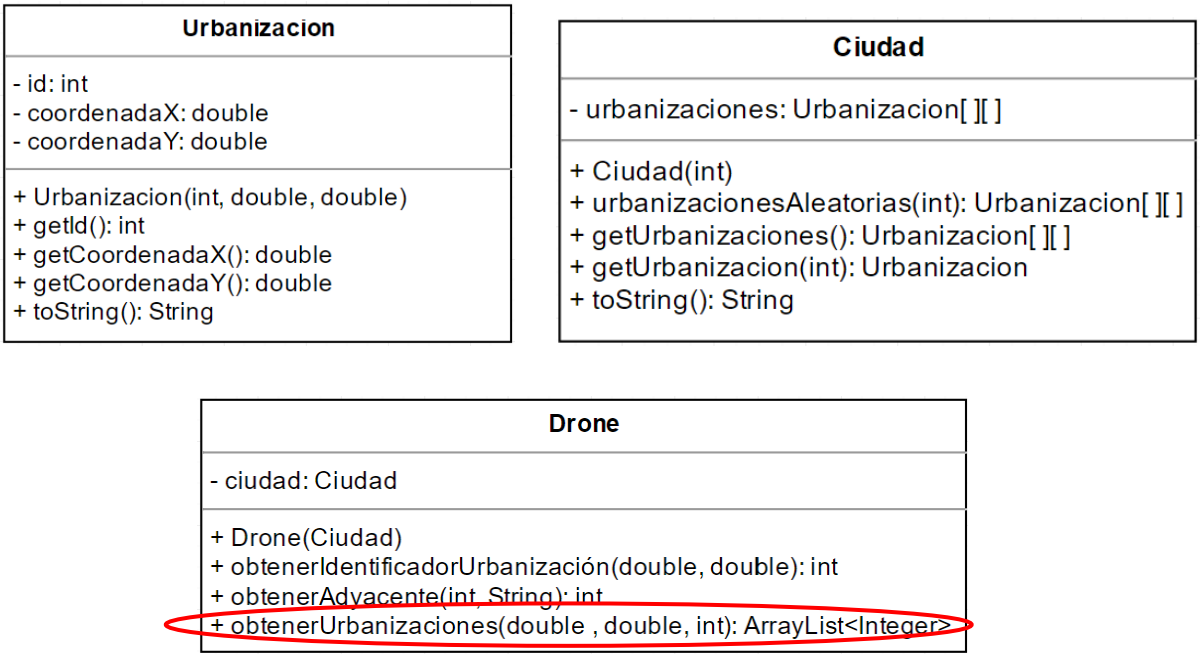
IVÁN JOSÉ ALBA GARCÍA - Prueba Drone Idealista

Para la realización del ejercicio he utilizado el lenguaje de programación orientado a objetos Java y el entorno de desarrollo Eclipse.

El objetivo de esta actividad es la programación de una función que dadas unas coordenadas y un rango, nos devuelva el identificador de todos las urbanizaciones que deberá visitar el drone.

Para poder utilizar la función y mostrar que todo funciona correctamente, se ha creado 3 clases: Urbanizacion, Ciudad y Drone. Aparte se ha creado una clase llamada "Principal" que es donde se van a realizar las pruebas.

Los atributos y métodos de estas clases se pueden observar en los siguientes diagramas de clases:



La clase Urbanizacion posee tres atributos (id, coordenadaX y coordenadaY), un constructor con todos sus atributos, los correspondientes métodos getters y el método toString().

La clase Ciudad posee como atributo una matriz bidimensional que corresponderá a todas las urbanizaciones que vamos a tener. El constructor recibe un número entero correspondiente al número de filas y columnas que tendrá la matriz. Con el número introducido por parámetro se rellenará la matriz bidimensional de forma que la primera urbanización tendrá índice 1 y coordenadas (0,0) y así sucesivamente. Por ejemplo, si introducimos el número 5, la matriz resultante es la siguiente:

1(0.0,0.0)	2(0.0,1.0)	3(0.0,2.0)	4(0.0,3.0)	5(0.0,4.0)
6(1.0,0.0)	7(1.0,1.0)	8(1.0,2.0)	9(1.0,3.0)	10(1.0,4.0)
11(2.0,0.0)	12(2.0,1.0)	13(2.0,2.0)	14(2.0,3.0)	15(2.0,4.0)
16(3.0,0.0)	17(3.0,1.0)	18(3.0,2.0)	19(3.0,3.0)	20(3.0,4.0)
21(4.0,0.0)	22(4.0,1.0)	23(4.0,2.0)	24(4.0,3.0)	25(4.0,4.0)

Por último, la clase Drone tiene como atributo un objeto de la clase Ciudad (creada anteriormente) y los tres métodos siguientes:

- **obtenerIdentificadorUrbanización**: dadas unas coordenadas, devuelve el identificador de la urbanización en la que están encerradas dichas coordenadas.

- **obtenerAdyacente**: dado un identificador y la dirección de adyacencia devuelve el identificador de la urbanización adyacente.
- **obtenerUrbanizaciones**: dadas unas coordenadas y un rango, nos devuelve el identificador de todas las urbanizaciones que deberá visitar el drone.

Función obtenerUrbanizaciones(coordenadaX, coordenadaY, rango)

```java
public ArrayList<Integer> obtenerUrbanizaciones(double x, double y, int rango){
    //Lista de números enteros donde se almacenarán los ids de las distintas urbanizaciones
    //que el drone tendrá que recorrer.
    ArrayList<Integer> ids = new ArrayList<>();
    // id de la urbanización que se obtiene a través de las coordenadas introducidas por parámetro.
    int idUrbanizacion = this.obtenerIdentificadorUrbanización(x,y);

    //si la id Urbanización obtenida es -1, significará que no existe urbanización con tal coordenadas,
    //por lo que se mandará un mensaje indicativo
    if(idUrbanizacion>0){
        System.out.println("Las coordenadas ("+x+","+y+") corresponden a la "
                + "urbanización con id="+idUrbanizacion);

        //Nº de filas y columnas que posee la matriz de urbanizaciones
        int n = this.ciudad.getUrbanizaciones().length;

        //Las siguientes líneas de código sirven para calcular el número de veces que podemos desplazar
        //el drone desde la id obtenida hacia la derecha, izquierda, arriba y abajo, dependiendo del
        //rango indicado. El número de desplazamientos se almacenarán en las variables:
        //desplazamientosDerecha, desplazamientosIzquierda, desplazamientosArriba y desplazamientosAbaj

        //Desplazamientos hacia la derecha
        int suma = n;
        while(idUrbanizacion>suma){
            suma += n;
        }

        int hastaExtremoDerecho = suma - idUrbanizacion;
        int hastaExtremoIzquierdo = idUrbanizacion - (suma-n+1);

        int desplazamientosDerecha = rango;
        if(hastaExtremoDerecho <= rango){
            desplazamientosDerecha=hastaExtremoDerecho;
        }
        //Desplazamientos hacia la izquierda
        int desplazamientosIzquierda = rango;
        if(hastaExtremoIzquierdo <= rango){
            desplazamientosIzquierda=hastaExtremoIzquierdo;//
        }
        //Desplazamientos hacia abajo
        suma = n;
        while(idUrbanizacion>suma){
            suma += n;
        }
        int hastaExtremoAbajo = (n*n -suma)/n;
        int hastaExtremoArriba = (suma-n)/n;

        int desplazamientosAbajo = rango;
        if(hastaExtremoAbajo <= rango){
            desplazamientosAbajo=hastaExtremoAbajo;
        }

        //Desplazamientos hacia arriba
        int desplazamientosArriba = rango;
        if(hastaExtremoArriba <= rango){
            desplazamientosArriba=hastaExtremoArriba;
        }

        /*
         * Para obtener la lista de ids se ha optado por realizar el siguiente procedimiento:
         * En un primer momento colocamos el drone en la esquina superior izquierda.
         * Para ello hacemos uso de la función "obtenerAdyacente(int,String)" anteriormente creada,
         * tantas veces como los desplazamientos anteriormente calculados indiquen. Una vez en ese
         * punto, recorremos la fila mediante un for tantas veces como indique la suma entre los
         * desplazamientos derechos e izquierdos. En cada vuelta añadimos la id de la urbanización
         * en la que se encuentra el drone y seguidamente lo desplazamos hacia la derecha.Una vez
         * acabe el bucle for, desplazamos el drone hacia la fila de abajo y al extremo izquierdo,
         * para volver a recorrer la fila de nuevo. Así sucesivamente, mientras lo indique las
         * variables desplazamientos arriba y abajo.
         */
```

```
123             */
124
125             //Situamos el drone en la esquina superior izquierda deseada.
126             int i = 0;
127             int id=idUrbanizacion;
128             for(i=0; i<desplazamientosIzquierda; i++){
129                 id = obtenerAdyacente(id, "izquierda");
130             }
131
132             for(i=0; i<desplazamientosArriba; i++){
133                 id = obtenerAdyacente(id, "arriba");
134             }
135
136             //recorremos arriba abajo las filas
137             int idInicioIzquierda=id;
138             for(i=0; i<desplazamientosAbajo+desplazamientosArriba+1; i++){
139                 //recorremos de izquierda a derecha
140                 for(int j=0; j<desplazamientosDerecha+desplazamientosIzquierda+1; j++){
141                     ids.add(id); //añadimos las ids en la lista.
142                     id=obtenerAdyacente(id, "derecha");
143                 }
144                 //situamos el drone en la fila de abajo y en el extremo izquierdo
145                 id = idInicioIzquierda;
146                 id=obtenerAdyacente(id, "abajo");
147                 idInicioIzquierda = id;
148             }
149         }else{
150             //si la urbanización no existe, se devuelve el siguiente mensaje por la consola.
151             System.err.println("No existe urbanización con coordenadas ("+x+","+y+")");
152         }
153         return ids;
154     }
```

El procedimiento de dicha función es el siguiente:

En un primer momento se calcula cual es la posición origen del drone. Para eso utilizamos las coordenadas recibidas por parámetro y mediante la función “obtenerIdentificadorUrbanización” calculamos la id correspondiente a dichas coordenadas.

En un segundo lugar calculamos cuantos movimientos puede hacer el drone en cada dirección, dependiendo del rango indicado.

Sabiendo esto, ya podemos colocar el drone en la esquina superior izquierda del área que hay que recorrer. Para ello hacemos uso de la función "obtenerAdyacente(int,String)" anteriormente creada, tantas veces como los desplazamientos anteriormente calculados indiquen.
Una vez en ese punto, recorremos la fila mediante un bucle “for” tantas veces como indique la suma entre los desplazamientos derechos e izquierdos. En cada vuelta añadimos la id de la urbanización en la que se encuentra el drone y seguidamente lo desplazamos hacia la derecha.
Una vez acabe el bucle “for”, desplazamos el drone hacia la fila de abajo y al extremo izquierdo, para volver a recorrer la fila de nuevo. Así sucesivamente, mientras lo indique las variables desplazamientos arriba y abajo.

De esta forma obtenemos la lista de urbanizaciones que el drone tendrá que recorrer.

Para comprobar que todo funciona he realizado una simulación, donde el usuario introduce el número de filas y columnas, las coordenadas X e Y, y el rango.

```java
package principal;

import java.util.Scanner;

import clases.Ciudad;
import clases.Drone;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Introduce nº de filas y columnas de urbanizaciones: ");
        int n = scanner.nextInt(); //Introducimos el número por teclado

        Ciudad ciudad = new Ciudad(n); //Creamos la ciudad con el número filas y columnas introducido
        System.out.println(ciudad);

        Drone drone = new Drone(ciudad);

        //Introducimos coordenada X
        System.out.print("Introduce coordenada x: ");
        double coordenadaX = scanner.nextDouble();

        //Introducimos coordenada Y
        System.out.print("Introduce coordenada y: ");
        double coordenadaY = scanner.nextDouble();

        //Introducimos rango
        System.out.print("Introduce rango: ");
        int rango = scanner.nextInt();

        System.out.println("Lista de urbanizaciones que el Drone tendra que visitar:\n"
        +drone.obtenerUrbanizaciones(coordenadaX,coordenadaY,rango));

        scanner.close();
    }

}
```

El resultado sale en consola tal y como se observa en la siguiente imagen:

```
<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (11 feb. 2017 19:15:26)
Introduce nº de filas y columnas de urbanizaciones: 5
        1(0.0,0.0)      2(0.0,1.0)      3(0.0,2.0)      4(0.0,3.0)      5(0.0,4.0)
        6(1.0,0.0)      7(1.0,1.0)      8(1.0,2.0)      9(1.0,3.0)      10(1.0,4.0)
        11(2.0,0.0)     12(2.0,1.0)     13(2.0,2.0)     14(2.0,3.0)     15(2.0,4.0)
        16(3.0,0.0)     17(3.0,1.0)     18(3.0,2.0)     19(3.0,3.0)     20(3.0,4.0)
        21(4.0,0.0)     22(4.0,1.0)     23(4.0,2.0)     24(4.0,3.0)     25(4.0,4.0)

Introduce coordenada x: 2
Introduce coordenada y: 2
Introduce rango: 1
Las coordenadas (2.0,2.0) corresponden a la urbanización con id=13
Lista de urbanizaciones que el Drone tendra que visitar:
[7, 8, 9, 12, 13, 14, 17, 18, 19]
```