ALGORITMOS Y ESTRUCTURA DE DATOS 2

Documentación - Obligatorio

Daniel Friedmann
Nro. 144276

Marcelo Alexandre
Nro. 153012

Grupo M4A
Sebastián Grattarola

20/11/2017

Índice

Interfaz del Sistema: Pre y Post condiciones

//Precondiciones: Se debe pasar un entero valido mayor a cero
//Postcondiciones: Se inicializa el sistema con la cantidad de puntos seleccionada listo para ser usado

public Retorno inicializarSistema (int cantPuntos);

//Precondiciones: Debe existir un sistema generado
//Postcondiciones: Se ponen en null todas las variables del sistema y el garbage collector termina por deshacerse de todo

public Retorno destruirSistema();

//Precondiciones: El sistema debe estar inicializado, los parámetros que se envían al método deben ser strings validos
//Postcondiciones: El sistema valida que los datos sean correctos y que no exista un productor registrado ya con esa cedula, si todo es correcto se registra el productor y se devuelve un OK, en caso de surgir un error se devuelve el error correspondiente

public Retorno registrarProductor(String cedula, String nombre, String direccion, String email, String celular);

//Precondiciones: El sistema debe estar inicializado, los parámetros que se envían al método deben ser string valido para el nombre y doubles válidos para las coordenadas.
//Postcondiciones: Se registra en el grafo una ciudad en dichas coordenadas y se da un mensaje de retorno de OK, en caso de que ya exista una ciudad en esas coordenadas se da error

public Retorno registrarCiudad(String nombre, Double coordX, Double coordY);

//Precondiciones: El sistema debe estar inicializado, los parámetros que se envían al método deben ser string valido para el nombre y doubles válidos para las coordenadas, asimismo se debe pasar una cedula valida de un productor previamente registrado y un entero valido para la capacidad.
//Postcondiciones: Se registra en el grafo una plantación en dichas coordenadas y se da un mensaje de retorno de OK, en caso de que ya exista un punto en esas coordenadas se da error

public Retorno registrarPlantacion(String nombre, Double coordX, Double coordY, String cedula_productor, int capacidad);

//Precondiciones: El sistema debe estar inicializado, los parámetros que se envían al método deben ser string valido para el nombre y doubles válidos para las coordenadas, asimismo un entero valido para la capacidad.
//Postcondiciones: Se registra en el grafo un silo en dichas coordenadas y se da un mensaje de retorno de OK, en caso de que ya exista un punto en esas coordenadas se da error

public Retorno registrarSilo(String nombre, Double coordX, Double coordY, int capacidad);

//Precondiciones: El sistema debe estar inicializado, los parámetros que se envían al método deben ser doubles válidos para las coordenadas, asimismo un entero valido para el peso.
//Postcondiciones: Se registra en la correspondiente lista de adyacencias el tramo seleccionado y se da un mensaje de retorno de OK y en caso de que el tramo no sea válido por no existir los vértices se da el error correspondiente

public Retorno registrarTramo(Double coordXi, Double coordYi, Double coordXf, Double coordYf, int peso);

//Precondiciones: El sistema debe estar inicializado, los parámetros que se envían al método deben ser doubles válidos para las coordenadas.
//Postcondiciones: Se elimina en las correspondientes listas de adyacencias el tramo seleccionado y se da un mensaje de retorno de OK y en caso de que el tramo no sea válido por no existir se da el error correspondiente

public Retorno eliminarTramo(Double coordXi, Double coordYi, Double coordXf, Double coordYf);

//Precondiciones: El sistema debe estar inicializado, los parámetros que se envían al método deben ser doubles válidos para las coordenadas.
//Postcondiciones: Se elimina el punto del grafo y sus correspondientes aristas, se retorna un mensaje de OK o en caso de no existir el ramo el error correspondiente

public Retorno eliminarPunto(Double coordX, Double coordY);

//Precondiciones: El sistema debe estar inicializado
//Postcondiciones: Se abre un navegador que muestra el mapa de Uruguay centrado donde están marcadas con colores las ciudades, silos y plantaciones.

public Retorno mapaEstado();

//Precondiciones: El sistema debe estar inicializado, se deben pasar doubles válidos para las coordenadas, las coordenadas deben ser de una plantación correctamente registrada
//Postcondiciones: Se imprime la ruta al silo más cercano con la capacidad requerida por la plantación desde donde se la llama y se devuelve un retorno OK, en caso de no existir un silo con la capacidad requerida se da el error correspondiente. Si la plantación no existe también se da el error correspondiente.

public Retorno rutaASiloMasCercano(Double coordX, Double coordY);

//Precondiciones: El sistema debe estar inicializado, se deben pasar doubles válidos para las coordenadas, las coordenadas deben ser de una ciudad correctamente registrada
//Postcondiciones: Se imprimen las coordenadas de todas las plantaciones que estén a 20KM de la ciudad seleccionada y se da un registro de OK, si no existe ninguna plantación en ese rango entonces se da un error correspondiente.

public Retorno listadoDePlantacionesEnCiudad(Double coordX, Double coordY);

//Precondiciones: El sistema debe estar inicializado
//Postcondiciones: Se imprimen las coordenadas de todos los silos.

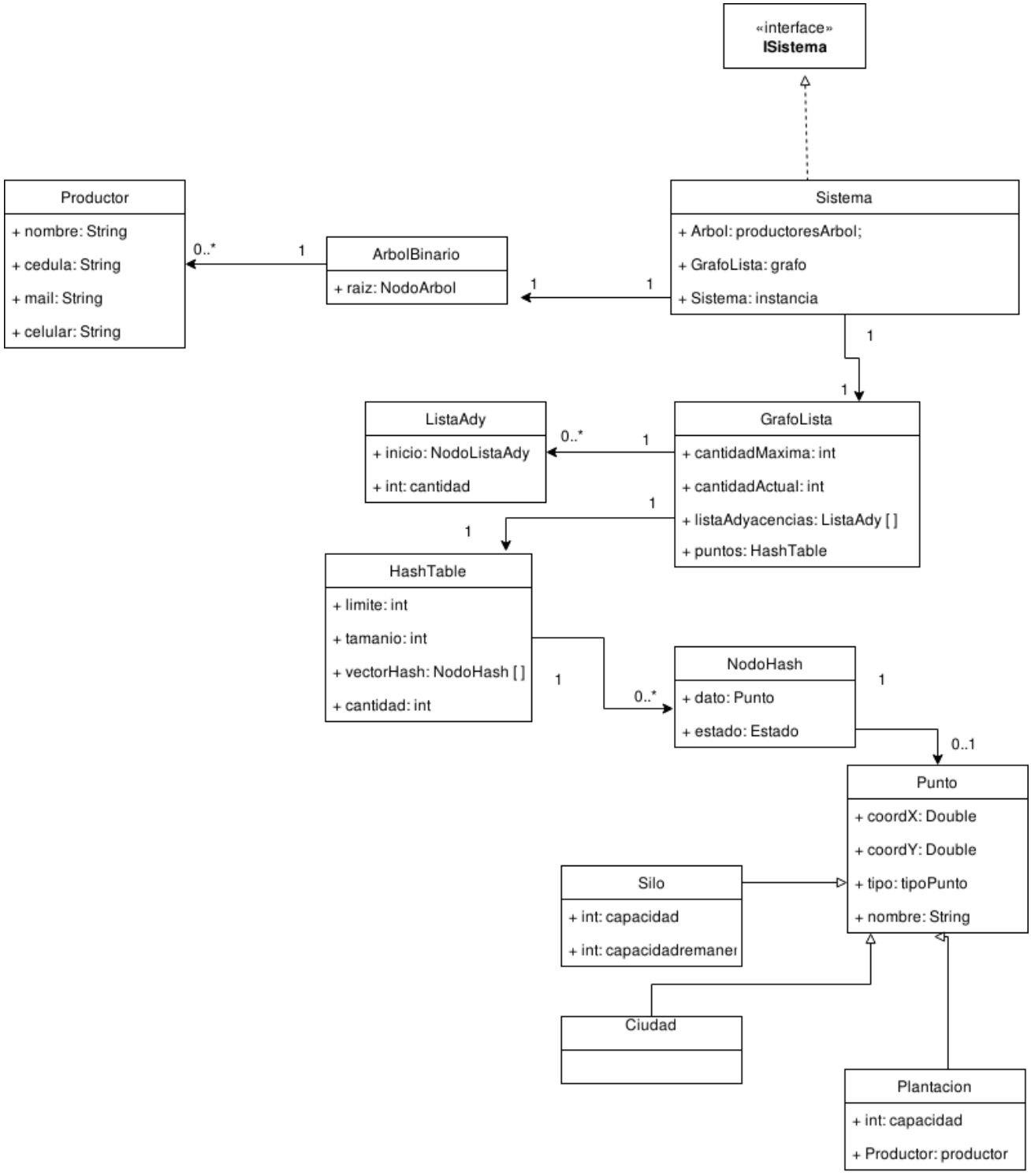
public Retorno listadoDeSilos();

//Precondiciones: El sistema debe estar inicializado
//Postcondiciones: Se devuelve la lista de productores ordenados.

public Retorno listadoProductores();

Solución Escogida

Diagrama de la estructura de datos



Justificación

Justificación de las estructuras elegidas para modelar las entidades del problema.

Árbol Binario

Se seleccionó la estructura de Árbol Binario para balancear la distribución de productores por cédula, y así, poder buscarlos con mayor eficiencia que en un vector o una lista, además de poder listarlos ordenados con mayor facilidad también. El registro del productor se requería en orden (log n) promedio.

Grafo

Se seleccionó la estructura de Grafo para cumplir los requerimientos de relación y distancia entre puntos. Se utilizó una lista de adyacencias en vez de una matriz, por preferencia y no dejar espacios vacíos.

Hash

Dentro del grafo, se utilizó una estructura de Hash para beneficiarnos de la eficiencia de la dispersión, y obtener la ubicación de un punto en el Hash a través de sus coordenadas, sin tener que recorrerlo. El registro de los puntos se requería en orden (1) promedio.

Punto

Se utiliza una Clase 'Punto' de la que heredan **Plantación**, **Ciudad** y **Silo** ya que todas son vértices del grafo y representan un punto en el mapa. Es el método que consideramos óptimo para la representación de la realidad planteada por la letra.

Productor

Se utiliza una Clase Productor para representar a los productores

Paquete Utilidades

Se generó un paquete de utilidades donde incluimos dos clases, una la clase Retorno, que usamos para devolver las respuestas de los métodos y la otra una clase de Útiles donde colocamos las funciones que validan datos como por ejemplo la cedula del Productor o el email usando expresiones regulares.

Testing

Se hicieron paquetes JUnit para generar las pruebas unitarias que corroboren que todas las funcionalidades requeridas están presentes y funcionando correctamente.
Resumen de los más significativos:

Dominio	Resumen de pruebas y resultados
Ciudades	Se hicieron pruebas de registrar una ciudad en coordenadas disponibles y también en coordenadas ocupadas, dando OK en el primer caso y el error esperado en el segundo. En las pruebas TestMapaGoogle también se prueba el 3.3 donde se pide armar el mapa de plantaciones alrededor de 20km de una ciudad
Plantaciones	Se hicieron pruebas de agregar, agregar en coordenadas ya ocupadas y agregar con capacidad igual a cero dando en todos los casos el resultado esperado. Plantaciones comparte algunas pruebas como la del mapa 3.3
Productores	Se hicieron las pruebas de agregar, eliminar, buscar por CI, listar ascendente, agregar un productor con CI duplicada y comparar dando los resultados esperados en cada caso. Para la clase Productor se hicieron algunos métodos de apoyo para poder cumplir con los requerimientos a saber: a) En la clase Utilidades se hicieron los métodos para validar teléfono, email y cédula usando expresiones regulares. b) También en la clase Utilidades se crearon los métodos para formatear las cédulas para pasarlas a un número ya que cuando comparamos a los productores y los listamos lo hacemos por orden de CI c) En la clase Productor implementamos el método CompareTo para comparar dos productores
Silo	Se hicieron las pruebas de registra un silo correctamente, luego de registrar un silo en coordenadas ocupadas y de registrar un silo con capacidad de cero, en todos los casos el resultado es el esperado
Sistema	Se hacen pruebas satisfactorias de crear y destruir el sistema
Grafo	Se hacen pruebas de registrar tramos válidos y también inválidos como con peso cero o entre puntos inexistentes, eliminar tramos existentes e inexistentes, ruta al silo más cercano con capacidad y plantaciones alrededor de los 20KM de una ciudad