UNIVERSIDAD NACIONAL DE GENERAL SARMIENTO

Trabajo Práctico 3

Integrante:

Ezequiel Tomas Ramallo

Legajo: 36827376/2017

E-mail: ezeramallo@gmail.com

Profesores:

Patricia Bagnes

Javier Marenco

Asignatura: Programación III – Primer Cuatrimestre 2020

Comisión: COM-1

Fecha de entrega: 12/05/2020

Introducción

El presente informe tiene como objetivo exponer, de forma breve, el desarrollo de una aplicación que permitirá carga un grafo de forma manual y, mediante el uso de un procedimiento de heuristica, obtener una clique "aceptable" con respecto a lo pedido en los requisitos del trabajo.

Esta aplicación contiene 2 pestañas "carga y calculo" y "Estableces conexiones" . La primera de ellas contiene los botones que nos permiten calcular y pintar la clique obtenida mediante el procedimiento heuristico, ademas de un panel de color gris que nos permite, mediante un click, ingresar un vértice y dibujarlo en pantalla. En cambio , la segunda pestaña, contiene los botones para establecer aristas entre vértices ya cargados en la pestaña "carga y calculo", como también un botón que nos permite borrar todo el grafo cargado a fin de poder comenzar de nuevo. Como esta app fue pensada para un grafo simple, la pestaña "Estableces conexiones" no permite establecer aristas múltiples ni loops.

Implementación

Para llevar a cabo implementación de esta aplicación decidimos utilizar el patrón MVC (modelo-vista-controlador), el cual nos permite separar el código de negocio del código de la interfaz. Para llevar a cabo la implementación de este patrón, decidimos dividir el proyecto en 5 package: codNegocio, AuxDibujoGrafo (auxiliar del package vista), Controlador, Vista y MVCPrincipal.

- ## Paquete AuxDibujoGrafo (paquete con clases que utilizaremos en el package vista)
-
  - **Interfaz "figura"**: Contiene los métodos a implementar de las clases "Circulo" y "Linea" contenidas es este package
  - **Clase "Círculo"**: clase que nos permite representan un vértice de forma grafica mediante el dibujo de un círculo en una interfaz grafica. En esta, tendremos como atributos: una instancia Point que contendrá 2 enteros, un int que corresponderá al tamaño del radio y una instancia de Color, que contendrá el color con el que se rellenara el circulo. La funcionalidad mas importante que puede mencionar de esta clase es:
    - paintComponent: Este método nos permitirá dibujar un círculo en la interfaz grafica

- **Clase “LInea” :** Esta clase nos permite representa una aristas de forma grafica, es decir, nos permitirá dibujar una linea en pantalla. En ella tendremos como atributos 2 instancias Point que representaran los extremos de esta linea y una instancia Color que contendrá el color de la linea a dibujar. El método mas importante de esta clase es el mismo descripto en la clase circulo.

• Paquete CodNegocio

- **Clase “Vértice”**: Esta clase contendrá los siguientes atributos: un entero que representa la etiquete del vértice, una variable de tipo genérica para almacenar un dato (En nuestro caso sera BigDecimal) y una List con todos los vértices vecinos.
- **Clase “Vertice2D”:** extension de la clase vértice, contiene todos los atributos y métodos de la clase ya mencionada, agregando dos atributos enteros que representaran las posición de un vértice en un plano cartesiano.
- **Clase “Grafo”:** Esta clase contiene los métodos y atributos necesarios para resolver el problema en la clase negocio. En ella nos encontraremos métodos como agregar o eliminar una arista o vértice, o bien, preguntar si alguno de los dos esta contenido en el grafo.
- **Clase “Negocio”:** clase encargada de la inteligencia de la aplicación, en ella utilizaremos 4 atributos;: el primero de estos sera un Comparator (usado para implementar el patrón strategy); el segundo, es una instancia a la clase grafo; el tercero es una instancia de la clase vértice, el cual nos permite almacenar un vertice de origen para implementar el procedimiento heuristico(se actualizara si el valor de ingreso de un nuevo vértice es mayor al del vértice que esta almacenado en este atributo); el cuarto atributo es un entero, el cual asigna automáticamente una etiqueta a cada vértice ingresado. EL método mas importantes de esta clase son:
  - CllíquePorHeuristica: método que nos devuelve una lista con los vértice que componen la clique obtenido por un procedimiento heuristaco.
- **Clase “SolucionNegocio”:** Esta clase se encarga del procesamiento de la clique obtenida en la clase negocio, a fin obtener los datos necesarios para ser mostrados en la interfaz grafica

• Paquete Controlador

- **clase “Controlador”**: esta clase implementa la interfaces de ActionListener y MouseListener, y se encarga de funcionar como una especie de intermediario entre el código de interfaz y el código negocio; es decir, mediante la reacción a un evento producido en el interfaz grafica, esta clase de encargar de actualizar y/o procesar el código de negocio para la posterior actualización de la parte grafica. En ella tendremos cuatro variables, dos de ella harán referencia tanto a la clase encargada del código de inteligencia como a la clase del código

encargado de la parte grafica; mientras que las dos restante, se encarga de almacenar los vértices seleccionados en la pestaña “Estableces conexiones” a fin de establecer su unión. Las funcionalidades que cumplen serán:

- crear_vertice: Este método se encargar de agrega un vértice al código de negocio y a una lista de la interfaz grafica, ademas de dibujarlo en el panel de la pestaña “carga y calculo” mediante la detección de un click en este panel ya mencionado.
- agregar_arista: estable una conexión entre dos vértice en el código de negocio, ademas de agregar una referencia de esta arista a una lista de la interfaz grafica y dibujar en el panel una linea entre vértices que la represente.
- pintar_clique: colorea los círculos que representan los vértices de la clique de color negro
- limpiar_cargar: borra todos los datos ingresados del código de negocio y de la interfaz grafica
- calcular_clique: nos brinda los vértices que componen la clique obtenido por el procedimiento heuristico junto a la suma de los peso de estos vértices.
- actionPerformed: Este método ejecutara un funcionalidad diferente dependiente del tipo de parámetro que reciba. Los método que puede ejecutar, son los descriptos anteriormente (calcular_clique, limpiar_cargar, pintar_clique, agregar_arista), ademas de las funcionalidades de selección de vértices implementadas dentro de esta función.
- mouseClicked: Metodo utilizado para llamar, mediante la detección de un click, a la función crear_vertice descripta anteriormente

• Paquete Vista

- **clase “PanelGrafo”**: clase que extiende de un JPanel. En ella tendremos 2 atributos que serán 2 list con figuras, una para representar los círculos (vértices) y otra para representar las lineas (aristas). En esta clase, se encontraran con métodos que permite dibujar y colorear círculos y lineas en el panel, como por ejemplo:
  - agregarDibujoVertice2D: agrega un círculo(vértice) al atributo list en basa a datos obtenido por parámetros, para posteriormente dibujarlo.
  - AgregarDibujoArista: Este método realiza la misma accion que el método descripto anteriormente, solo que es vez de círculo lo hace con lineas
  - pintarClique: pinta vértices y lineas ya ingresadas de otro color, en base a los datos obtenidos por parámetro.

- **clase “GuiApp”**: En esta clase se encuentra implementada la ventana principal de la aplicación , la cual contendrá componentes como JtabbedPanel, Jpanel, Jbutton, JList Jlabel, entre otros. Estos componentes, nos permitirán ingresar y dibujar un grafo y, calcular y pintar la clique de este grafo gracias a su interacción con el código de negocio. Los métodos mas importantes de esta clase son:
  - MostrarVentanaEmergente: Nos permite mostrar un mensaje pasado por parámetro a través de un JoptionPane. Este método se usa, básicamente, para ensañar mensajes que especificar las excepciones generadas por el ingreso de algún valor incorrecto
  - VentanaIngresoPeso: Nos permite tomar el peso de un vértice a ingresar a través del uso de un JoptionPane.
  - GUI: este método se encarga de distribuir y acoplar los paneles de componentes en el frame, así como también de activar la visualización de la aplicación.
  - CreaGUI: método encargado de inicial la aplicación

- ## Paquete MVCPrincipal
  - **Clase “Principal”:** es la clase que contendrá el man utilizado para aunar las partes del modelo, vista y controlador, a fin de lanzar la aplicación.

Problemas durante el desarrollo y decisiones tomadas.

- Al igual que en el TP1 y TP2, tuve inconvenientes al implementar el patrón MVC con la extensión windowBuilder (Imposibilita el setteo del código de negocio y de vista en controlador debido al main del “aplication window”) por lo que se decidió quitar el main predeterminado de la “aplication window” de la GuiApp e implementarlo en una clase aparte.