MAHEEN ASAD – 241839
JAHANZEB RAJA MOHAMMAD - 242097

PROGRAMACIÓN ORIENTADA A OBJETOS

LAB 1 | INFORME RESULTADOS

1. INTRODUCCIÓN

En el primer laboratorio del curso Programación Orientada a Objetos, se ha tenido que implementar clases para calcular las distancias entre dos puntos. Además de ampliar estas funciones para crear matrices de distancias entre dos puntos, y así tener un concepto global sobre la aplicación de este programa en la vida práctica.
Durante la implementación, se puede observar que los puntos son ciudades y la matriz guarda las distintas ciudades y las distancias entre ellas. A nivel práctico, este tipo de clases son útiles para crear mapas sencillos.

Las clases implementadas son GeometricPoint i DistanceMatrix, adicionalmente, se han hecho TestGeometricPoint i TestDistanceMatrix. A parte, como opcional, se ha añadido una interfaz simple, utilizando las clases DisplayMatrix y Matrix. Para comprobar la ejecución de estas últimas, se ha implementado TestDisplayMatrix.

La primera clase creada ha sido GeometricPoint, la cual tiene como atributos dos puntos: double PointX y double PointY, a parte de un String name. El PointX y PointY son las coordenadas de un nombre y el String name, contiene el nombre de la ciudad establecida en estas coordenadas.

En esta clase, a parte de crear el constructor, se han creado los getters y setters necesarios para los atributos anteriores. El método que se considera importante a explicar es el public double Distance(GeometricPoint p). Este se utiliza para calcular la distancia entre dos puntos en un espacio. Adicionalmente, se ha añadido una función public void PrintPoint(), la cual imprime un punto por la pantalla.

Seguidamente, se ha creado la segunda clase DistanceMatrix. Esta contiene como atributos, una linkedlist de tipo GeometricPoint. A parte de una matriz de doubles. Para poder simplificar el trabajo, y centrarse en conceptos más importantes de la materia, se

ha preferido crear una matriz con una capacidad máxima de 20 columnas y 20 filas, en vez de crear un matriz dinámica. A parte de crear los constructores y los getters necesarios, se considera importante mencionar las siguientes funciones: public void addCity(double x, double y, String name) i public void createDistanceMatrix().
La primera función, addCity, añade un GeometricPoint en la linkedlist creada. Este GeometricPoint contiene el double x, double y e String name. La segunda función inicializa la matriz con las distancias que hay entre un punto y otro.

Finalmente, para la parte adicional, en el TestDisplayMatrix se ha añadido lo que marca el documento.

Se considera que entre clase DistanceMatrix y GeometricPoint existe una relación de agregación, y entre el GeometricPoint y DisplayMatrix existe una relación de implementación de la interfaz.

2. EJECUCIÓN Y COMPROBACIÓN DE ERRORES

Después de implementar todas las clases necesarias, se ha efectuado un test de ejecución de cada clase.

Primeramente, se ha realizado el test de GeometricPoint, ya que, sin la correcta implementación de este, era imposible continuar con las otras clases.
En el TestGeometricPoint se han utilizado los prints por pantalla, para observar hasta que punto los métodos de la clase GeometricPoint funcionan correctamente. Para ello, se han creado dos puntos, y se ha comprobado que carguen correctamente en sus respectivas variables, y una vez hecho esto, se ha calculado la distancia entre los dos puntos y se ha imprimido en la pantalla. Se ha comprobado que, con reales positivos y negativos da el resultado correcto. Sin embargo, se ha utilizado math.pow para calcular reales grandes. En este caso, se ha observado que para números mayores a 10^155, el programa no se ejecuta bien, y da como respuesta que la distancia entre dos puntos es infinita. No obstante, se ha comprobado que los resultados obtenidos son iguales a los calculados manualmente. Adicionalmente, se ha utilizado el debugger para analizar si las variables guardan correctamente los valores calculados.

Seguidamente, se ha testeado el DistanceMatrix a través de la clase TestDistanceMatrix. Para ello se ha creado una nueva matriz y como en el caso anterior se ha puesto prints en el inicio y al final de cada función de la clase de DistanceMatrix, para ver que funcionan correctamente. Cabe destacar, que no se pueden añadir más de 20 ciudades para crear la matriz, ya que se ha definido con una dimensión máxima de 20x20. Para esta prueba, se ha tenido en cuenta que el geometric point se ha implementado bien ya que hace un uso directo de esa parte del programa y una vez hecho se ha comprobado que la matriz se crea correctamente, usando nuevamente el debugger e imprimiendo por pantalla sus valores.

3. CONCLUSIÓN

En conclusión, esta práctica ha sido bastante útil para entender los conceptos explicados en la clase teórica y el como adaptar la parte de los seminarios. Personalmente, nos ha parecido muy bien la relación entre los seminarios y las prácticas, ya que en los seminarios se han hecho los diagramas, y se ha podido observar de forma global cómo se implementarían las clases, y luego en la práctica se ha hecho su implementación de forma manual. Sin embargo, nos hubiera gustado si se hubiera aclarado más si la matriz tenía que ser dinámica obligatoriamente o no, ya que en el momento de la implementación teníamos dudas.

Para efectuar el trabajo, como que consideramos que no era una práctica muy larga, se ha decidido que era mejor que cada persona implementara el código por su cuenta, ya que de esta forma cada uno aprende más sobre sus dificultades, errores... Finalmente, se han compartido los resultados y se han puesto en común. Compartiendo las posibles soluciones y seleccionando las que se consideraron mejores y óptimas, para el final hacer el test y redactar el informe final.