INF-253 Lenguajes de Programación
Tarea 3: Java

Profesor: Roberto Nicolas Diaz Urra
Ayudante Cátedras: Sebastián Godínez
Ayudante Tareas: Monserrat Figueroa - Sebastián Campos

Lunes 7 de Octubre de 2019

1. Historia

Estás desorientado, tienes muchas lagunas mentales y no sabes exactamente donde te encuentras. Lo último que recuerdas es la destrucción de tu hogar,y que inmediatamente después de ese hecho emprendiste un viaje a un lugar extraño y oscuro para ti. Tu cabeza da vueltas y los agujeros en tu memoria no ayudan a tranquilizarte, alguien jugó con tus recuerdos. Revisas tus bolsillos y consigues tranquilidad en dos piedras que encuentras en él, sea lo que sea que haya sucedido tienes la certeza de algo, saliste victorioso. Miras a tu alrededor y te das cuenta que estas en un lugar desolado, solo observas tierra a tu alrededor y un pequeño letrero a un par de metros de tu posición. No tienes que acercarte mucho para poder leer lo que dice: 'En el paquete principal, se ha llamado al método de la clase publica principal llamado "comunicar". Éste método llama al paquete sistema, el cuál llama al sub-paquete salida, quien a su vez tiene un método llamado mostrar-linea. Utilizando éste método se te entregan las palabras "Hola". No alcanzas ni a asimilar la mitad de lo que se te acaba de comunicar, cuando de repente se abre el suelo bajo tus pies y caes por un tobogán que no parece tener fin. Pasado unos segundos la oscuridad se esclarece, y a tu alrededor comienzas a ver una ciudad enorme a través de las paredes del tobogán , excesivamente complicada, y bastante aparatosa sin razón. Sin embargo, parece que de alguna extraña forma todo funciona y lo peor, funciona eficientemente. En la lejanía puedes observar una torre alzada en el centro que tiene una piedra brillante en su punta, no entiendes ni sabes como vas a llegar a ella, pero has llegado muy lejos como para retirarte.

2. Objetivos

El alumno utilizará conceptos de programación orientada a objetos en la implementación de la solución a un problema de optimización.

3. Contexto del Problema

Un país contiene múltiples ciudades, en cada una de estas ciudades residen miles de personas en edificios o casas. Cada edificio o casa necesita un suministro de gas que puede ser otorgado por solo una compañía en el país.

Esta compañía está buscando optimizar sus recursos para aumentar sus utilidades, ya que el principal costo a disminuir es el del viaje que tienen que hacer los camiones para satisfacer toda la demanda de gas del país, se quiere determinar el lugar estratégico (una ciudad) para colocar su planta distribuidora.

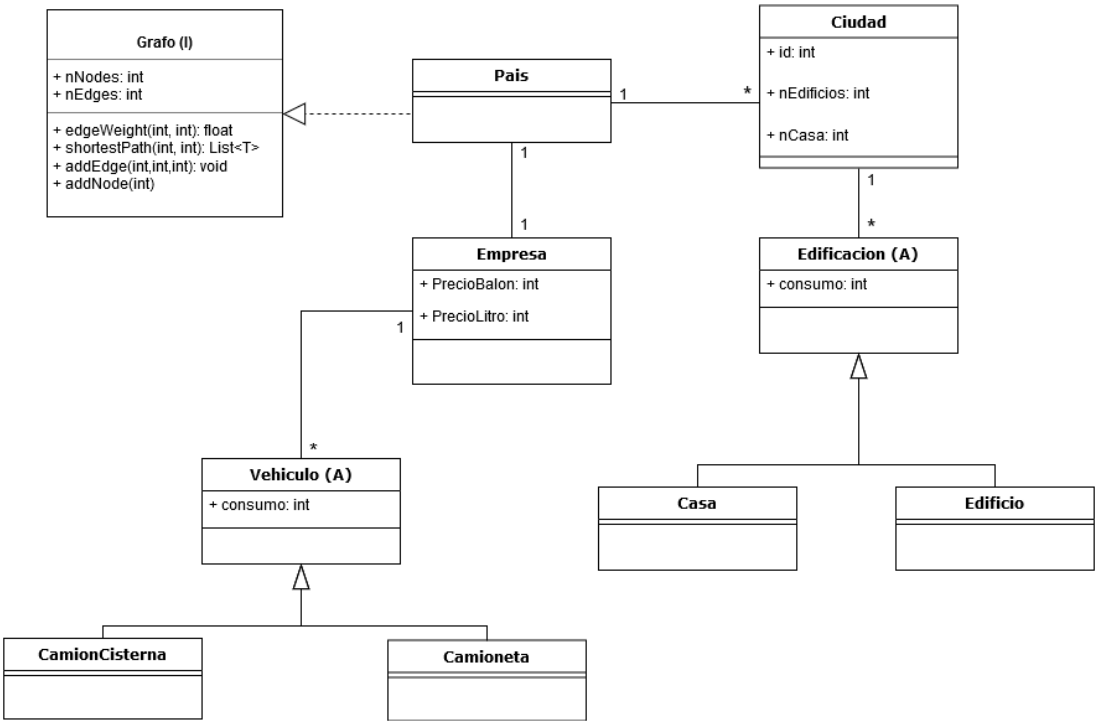
Entre cada ciudad existe una distancia muy significativa, mucho mayor a la distancia que existe entre residencias, por lo que esta última no se considerará como una distancia relevante para la compañía. Los vehículos que se utilizan para transportar gas presentan cierto consumo por kilometro recorrido y cada edificación presenta un consumo determinado, por lo que la utilidad final obtenida en una ciudad es la venta de gas menos el gasto que significó el transporte.

3.1. Detalles del problema

1. Un país puede ser representado mediante un grafo, donde cada nodo es una ciudad y el peso de los arcos es la distancia entre ellas.
2. En cada ciudad existen dos tipos de residencias, **casas** y **edificios**. El gasto se mide en balones de gas y litros de gas respectivamente
3. Existen dos tipos de vehículos, camiones cisternas y camionetas, que se encargan de los edificios y las casas respectivamente.
4. Un camión cisterna puede satisfacer la demanda de un solo edificio, sin embargo, se sabe que una camioneta basta para satisfacer a todas las casas de una ciudad.
5. El precio por litro o balón consumido es determinado por la empresa.
6. La planta de distribución es capaz de almacenar infinitos vehículos.
7. Todos los vehículos tienen el mismo consumo.

4. Estructura de clases

El siguiente modelo de clases representa las relaciones existentes entre las clases a implementar, los atributos y métodos básicos que deben presentar.



Las relaciones 1-a-1 mostradas en el modelo indican que ambas clases solo pueden relacionarse con sólo una instancia de la otra clase. En el ejemplo del modelo, se puede ver que un país solo puede estar relacionado a una sola empresa, en cambio, las relaciones 1-a-muchos (*) muestran que una clase puede relacionarse con múltiples instancias de la otra clase pero no inversamente, en este modelo se puede ejemplificar con la clase empresa, la cual puede tener muchos vehículos pero un vehículo solo puede estar relacionado a una empresa.

1. Grafo: Interfaz.
   - nEdges: número de arcos del grafo.
   - nNodes: número de nodos del grafo.
   - edgeWeight: recibe como parámetro el identificador de dos nodos vecinos y retorna el peso de este arco.
   - shortestPath: recibe el identificador de dos nodos y retorna el camino mas corto.
   - addEdge: recibe el identificador de dos nodos y el peso de este arco respectivamente.
   - addNode: recibe el identificador de un nuevo nodo.
2. País: clase que implementa a la interfaz Grafo.
   - Empresa: empresa de gas que pertenece al país.
3. Ciudad: Clase
   - id: identificador de la ciudad.
   - nEdificios: cantidad de edificios que hay en la ciudad.
   - nCasa: cantidad de Casas que hay en la ciudad

4. Edificación: clase abstracta
    - Consumo: representa el consumo de gas de la edificación
5. Casa y Edificio: clases que extienden a la clase Edificación
6. Vehículo: clase abstracta
    - Consumo: representa el consumo del vehículo
7. CamionCisterna y Camioneta: clases que extienden a la clase Vehiculo

5. A implementar

El alumno debe implementar un programa que logre obtener una ubicación óptima para la planta de distribución. Toda la información para la resolución del problema será entregada a través de archivos de texto.

- **mapa.txt**: Este archivo corresponde las ciudades y los caminos existentes, donde la primera linea corresponde a la cantidad de ciudades y la segunda linea corresponde a la cantidad de caminos existentes. Las siguientes lineas detallan la unión entre dos ciudades y su respectiva distancia, donde los dos primeros números separados por un espacio corresponden a los identificadores de las ciudades involucradas y el tercer numero separado por un espacio corresponde a la distancia entre ellas.

  Ejemplo:

  ```
  10
  25
  0 1 3
  1 4 7
  1 5 3
  1 2 4
  2 3 6
  ...
  ```

- **edificaciones.txt**: Corresponde a las edificaciones de cada ciudad y su consumo correspondiente. El archivo presentará una linea con tres enteros que corresponde al identificador de la ciudad, cantidad de casas y cantidad de edificios respectivamente, seguido de una linea con el consumo respectivo de cada casa y otra linea con el consumo de cada edificio, este patrón continua sucesivamente hasta completar todas la ciudades.

  Ejemplo:

  ```
  0 5 6
  3 4 5 6 7 7
  20 100 50 1000 200 1000
  1 2 4
  ...
  ```

- **empresa.txt**: Corresponde a los precios del gas y el costo de transporte. el archivo presenta tres lineas correspondientes al precio por balon de gas, el precio por litro de gas y el costo por kilometro de transporte respectivamente.

  Ejemplo:

  ```
  1000
  100
  10
  ```

Con la información del problema dada, el programa debe construir el modelo de clases señalado anteriormente y mostrar por pantalla las ciudades donde se podría colocar la planta, además de mostrar las utilidades obtenidas y los vehículos utilizados en cada ciudad.

Ejemplo:

```
La ciudad 0 es la ubicación óptima
ciudad 0:
- Utilidad: 100
- Se utilizaron 10 camiones cisterna y 1 camionetas
ciudad 1:
- Utilidad: -10
- Se utilizaron 2 camiones cisterna y 0 camionetas
...
```

5.1. Consideraciones

1. Se pueden agregar más atributos y métodos a las clases señaladas.
2. Cada clase debe presentar los métodos `get` y `set` respectivos.
3. La manera de implementar la estructura del grafo es a libre elección

6. Archivo a entregar

- **La entrega debe realizarse en un tar.gz y debe llevar el nombre: Tarea3LP_RolIntegrante-1_RolIntegrante-2.tar.gz**
- El tar.gz debe incluir siguientes archivos:
  - *.java: Un archivo por cada clase solicitada
  - makefile : Archivo de compilación
- **El archivo README.txt debe contener nombre y rol de los integrantes del grupo e instrucciones detalladas para la correcta compilación y ejecución de su programa.**

7. Sobre la Entrega

- Se debe trabajar en grupos de dos personas.
- La tarea se debe poder compilar y ejecutar en los computadores del LDS.
- Si por alguna razón está sólo, debe realizar un anuncio en Moodle en busca de un compañero. En caso de que exista alguna situación que obligue al estudiante a trabajar sólo, se debe regularizar previamente con los ayudantes mediante correo electrónico.
- **El no cumplir con las reglas de entrega conlleva un descuento máximo de 30 puntos en su tarea.**
- La entrega será vía Moodle y el plazo máximo de entrega es hasta el **Domingo 27 de Octubre a las 23:55 hora Moodle**.
- Por cada día de atraso se descontarán 20 puntos (10 puntos dentro de la primera hora, después se descuenta por el día).
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- Los atributos agregados deben comentarse al inicio de la clase de forma:
  `//(tipo) (nombre) : Razón de existir.`
- En caso de crear un método se debe comentar (por encima de él) de forma:

```
/** (Nombre)
(tipo) (parámetro 1)
(tipo) (parámetro 1)
--------------------
Descripción breve
--------------------
*/
```

8. Calificación

- Código
  - Implementación de la clase Grafo. (20 puntos)
  - Implementación de algoritmos de camino mas corto. (25 puntos)
  - Inicialización de clases. (20 puntos)
  - Implementación de métodos. (25 puntos)
  - Encontrar óptimo. (10 puntos)
- No entrega Makefile. (-100 puntos)
- No compila. (-100 puntos)
- No utiliza las clases o métodos de forma adecuada. (-20 puntos)
- No comenta métodos. (-10 puntos c/u)
- No comenta atributos. (-5 puntos c/u)