

# Aclaraciones

Las siguientes aclaraciones facilitarán a los profesores la corrección y harán más rápido y sencillo el recorrido por la aplicación.

Cualquier otra cosa que necesiten podemos hacer una video-llamada por Hangouts:

[gerleven@gmail.com](mailto:gerleven@gmail.com)

O por whatsapp: 3426303635 (Germán)

## Principales estructuras de datos usadas por el sistema:

```
package logica;

import java.util.ArrayList;

public class Logica {

    //DATOS Y ESTRUCTURAS GLOBALES DEL SISTEMA:

    public static List<Insumo> listaInsumosIndustria = new ArrayList<Insumo>();
    public static List<Planta> listaPlantasIndustria = new ArrayList<Planta>();
    public static List<Stock> listaStocksIndustria = new ArrayList<Stock>();

    public static List<Camion> listaCamiones = new ArrayList<Camion>();

    public static final GrafoPlanta grafoDePlantas = new GrafoPlanta();

    public static Boolean cargaAutomatica = false; //usada para saber si se realizó una carga automática
    public static Integer tipoDeCarga = 0;

    public static Integer ultimoId=0;
    public static Integer getNewId() { //general para cualquier cosa, sea Insumo, Stock o lo que sea
        if(ultimoId==null) ultimoId=0;
        //System.out.println("Id asignado: "+ultimoId);
        return (++(ultimoId));
    }
}
```

## Botones de cargas de datos automáticas para probar la aplicación:

La aplicación cuenta con 3 cargas automáticas de datos de pruebas que permiten probar de manera abarcativa la totalidad de funcionalidades requeridas por el enunciado. Aun así el usuario puede agregar carga de datos adicional de manera manual como sucedería en un caso real de utilización de la aplicación.

Datos de prueba:

Ejemplo Grafo

Grafo Minimo

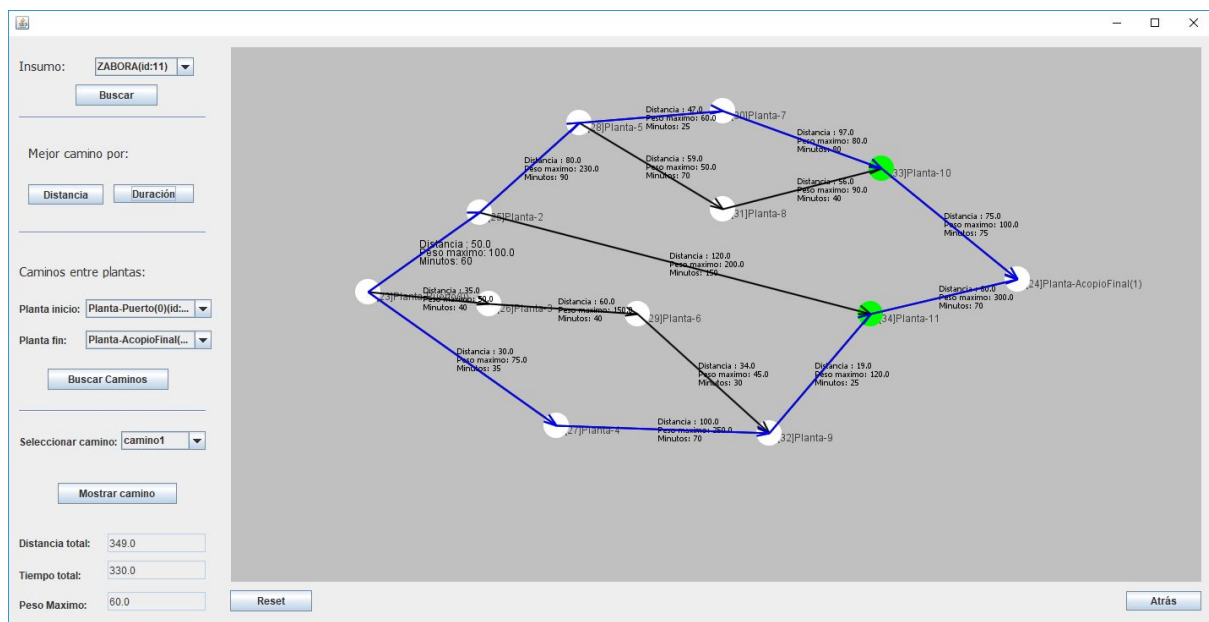
Mochila

Nota: El botón “Reset Datos de Prueba” limpia todas las listas del sistema dejando todo en cero como si recién abrieramos la aplicación.

Cada botón de carga de datos automática tiene incorporado internamente un “reset” que hace lo mismo que el botón Reset Datos de Prueba.

### Ejemplo Grafo:

Carga un Grafo similar al del ejemplo del enunciado:



Nota: Este grafo en particular tiene su propia forma de generarse, ya que la distribución de nodos tal como se la ve en la imagen es automática, (el usuario no tiene que acomodarlos manualmente, aunque puede moverlos luego si quiere).

### Grafo Mínimo:

Es un grafo mínimo de solo 2 plantas, el cual contiene 3 insumos con stock faltante: Nitrógeno Madera y Hielo, lo utilizamos más que nada para probar funcionalidades del grafo para no tener que lidiar con el grafo anterior que es más extenso y más difícil de testear.

Insumo:

NITROGENO LL.

NITROGENO LIQUIDO

MADERA(id:2)

HIELO(id:3)

Mejor camino por:

Distancia

Duración

Caminos entre plantas:

Planta inicio:

Planta-Puerto(0)(id:7)

Planta fin:

Planta-Puerto(0)(id:7)

Buscar Caminos

Seleccionar camino:

Mostrar camino

Distancia total:

0.0

Tiempo total:

0.0

Peso Maximo:

0.0

Reset

Atrás

## Mochila:

Creamos este caso para simular el caso de la mochila (programación dinámica) y testear que funcione bien el calculo de “Mejor Envio”. El caso consta de los mismos 4 items del ejemplo dado en clases (de cada insumo se tiene: el peso faltante en industria (osea el faltante de todas las plantas) y valor de ese faltante en plantas iguales al ejemplo original) y un camión de capacidad 4 simulando la mochila.

Resultados de búsqueda de insumos

Nombre-Id Insumo	Peso faltante	Valor
UNO	3	6
DOS	2	9
TRES	1	5
CUATRO	2	6

Seleccione camión:

Id-12 dominio-2.

Informacion del camión

Id:

12

Marca:

MOCHILA

Modelo:

BAG

Dominio:

2

Año:

2019

Costo por Km:

50.0

Apto para líquidos:

Si

Capacidad:

4.0

Generar solución

Atrás

Resultados de búsqueda de insumos		
Nombre-Id Insumo	Peso faltante	Valor
DOS	2	9
CUATRO	2	6

## Clase Lógica:

Esta clase se encarga de albergar métodos de clase usados por el sistema para la gestión de la logica, como ser realizar las cargas de datos automáticas, resetear listas, mostrar listas, validar carga de prueba, ect.

## Agregar

### Carga de datos manual por parte del usuario:

Orden de carga: Insumos, Planta, Stock

Bajo nuestro diseño, es necesario que el usuario ingrese primero el insumo, luego la planta y luego el stock, porque para setear el stock necesita previamente saber en qué planta estará y cual es en insumo

### Carga de camiones:

ID: Integer

Marca: String

Modelo: String

Dominio: Integer

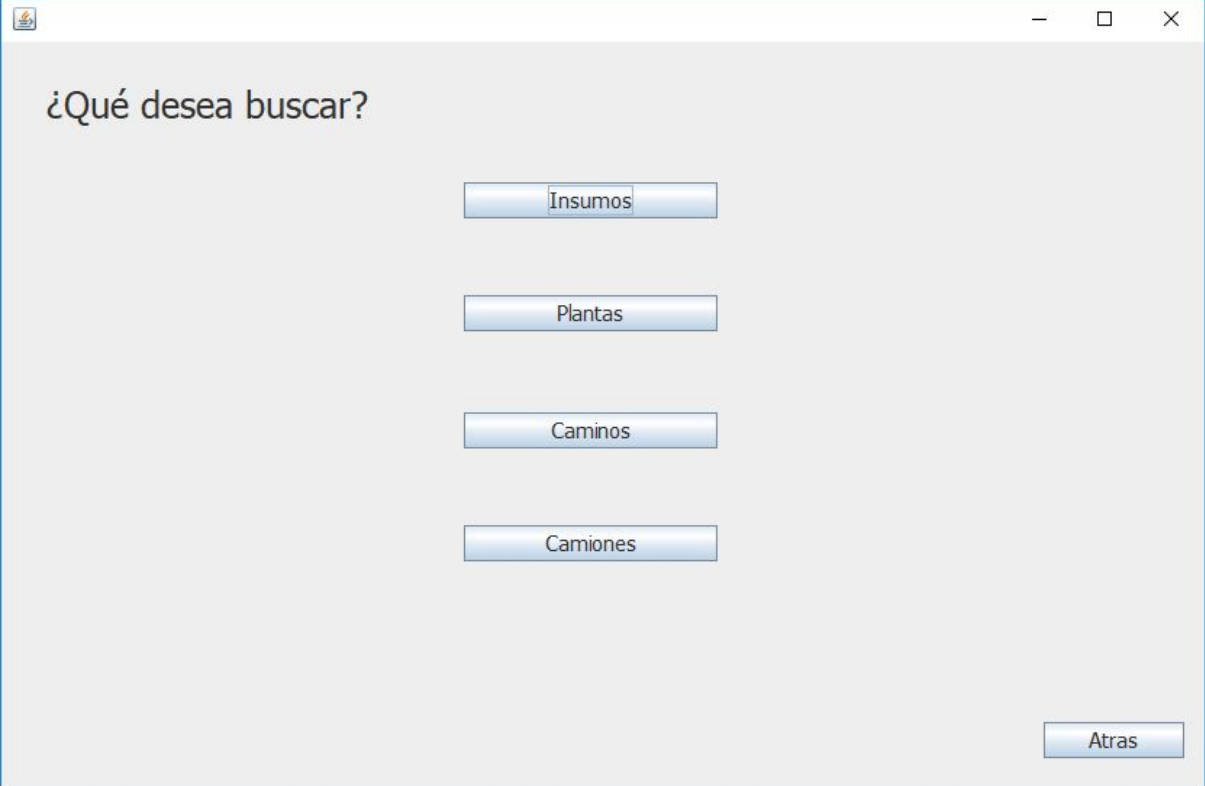
Anio: Integer

CostoKM: Double

AptoLiquido: Boolean

Capacidad: Double

# Búsqueda



¿Qué desea buscar?

Insumos

Plantas

Caminos

Camiones

Atras

## Búsqueda de Insumos:

Además de poder hacer la búsqueda Ascendente y Descendente, el sistema permite tanto hacer una búsqueda puntual de un Insumo (si se completa el campo de nombre con el nombre del insumo por ejemplo) o bien buscarlos a todos (Ascendente o Descendentemente) al dejar el campo vacío (aplica ya sea para búsqueda por nombre, costo o stock). Por lo tanto se puede buscar un solo insumo poniendo el dato de ese insumo o traerlos todos si se deja el campo vacío.

—□×

Buscar insumo por nombre

Nombre:

Busqueda ascendente

Busqueda descendente

Atras

Resultados de busqueda de insumos

—□×

Id Insumo	Nombre Insumo	Costo	Peso	Stock	Refrigerado
2	AGUA MINERAL	50.0	0.0	245	false
6	ARENA	250.0	200.0	265	false
3	GASEOSA NARA...	70.0	0.0	175	true
8	HIELO	50.5	1.0	456	true
10	LEVADURA	0.5	0.1	149	false
5	MADERA	150.0	100.0	355	false
4	MERCURIO	770.0	0.0	473	false
1	NITROGENO LIQ...	550.0	0.0	435	true
7	PISTON	750.5	3.0	246	false
9	TELA	78.5	0.2	279	false
11	ZABORA	10.5	0.5	357	false

Editar

Eliminar

Atras

Resultados de búsqueda de insumos					
Id Insumo	Nombre Insumo	Costo	Peso	Stock	Refrigerado
8	HIELO	50.5	1.0	456	true
<div> <div>Editar</div> <div>Eliminar</div> <div>Atras</div> </div>					

## Búsqueda de Plantas:

Al igual que con los insumos: si se deja el campo “id” vacío trae todas las plantas, si completamos un valor trae solo la planta encontrada.

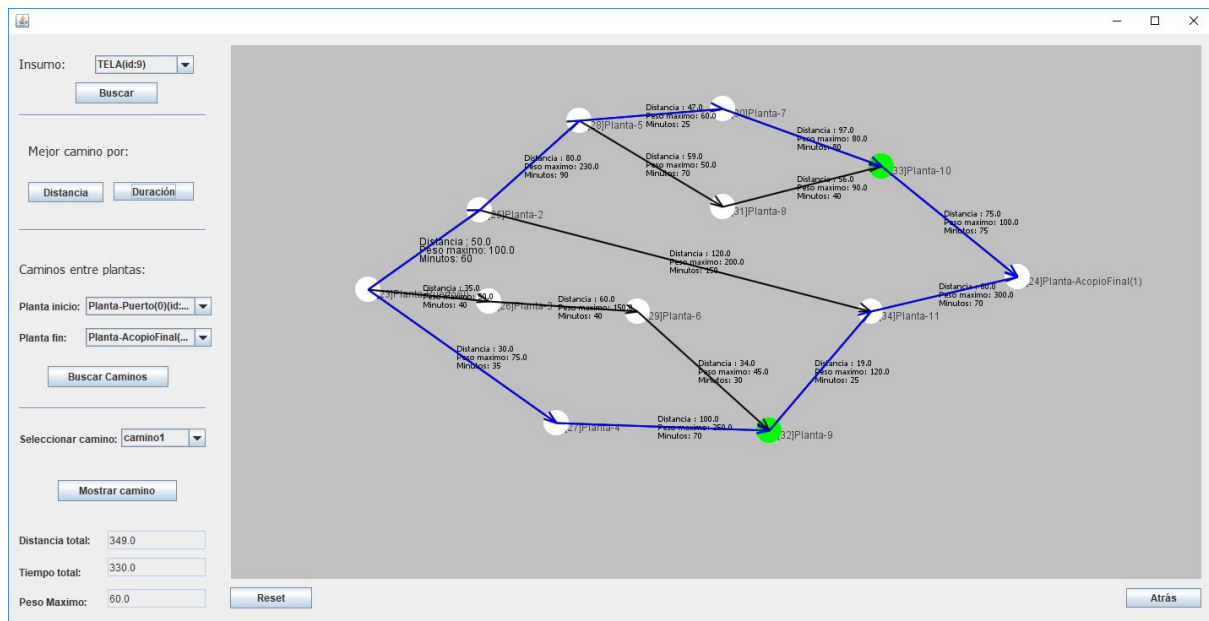
## Grafo

### Buscar:

El grafo permite seleccionar un insumo del sistema, al presionar “Buscar” pintará de verde las plantas que necesiten ese insumo.

por duración:

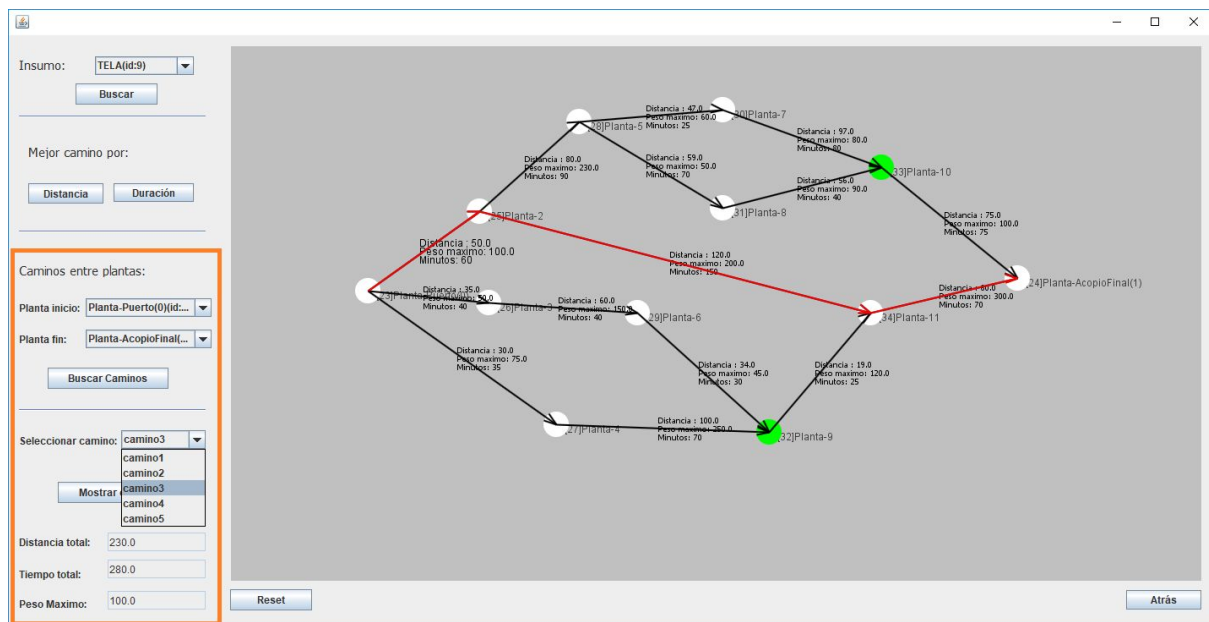




## Camino entre plantas:

Se seleccionan 2 plantas cualesquiera, se uno de todos los posibles caminos del desplegable y al presionar “Mostrar Camino” se pintara dicho camino en el grafo mostrando además los valores de dicho camino en la parte inferior izquierda del panel.

El botón “Reset” borra todo lo pintado en color.

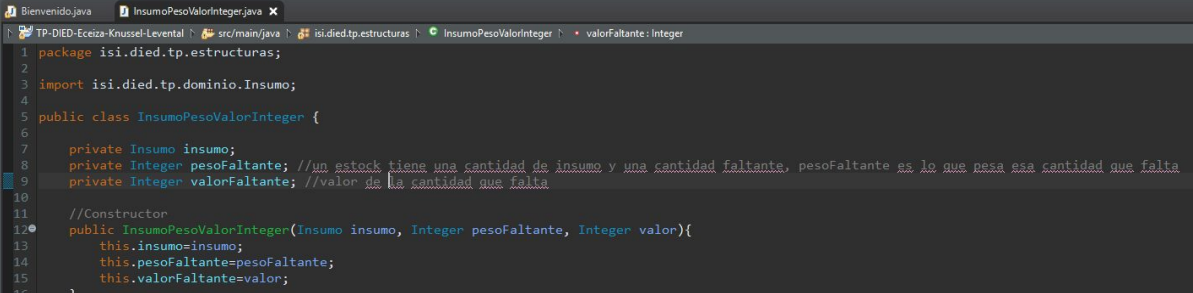


# Envíos

Para poder utilizar el algoritmo de la mochila tuvimos que adaptar los datos de nuestro sistema para que pueda entregar un input compatible al que necesitaba el algoritmo de la mochila, para lo cual creamos un tipo de dato nuevo llamado **InsumoPesoValorInteger** que se detalla a continuación.

## Clase InsumoPesoValorInteger:

Se usa para el cálculo de Mejor Viaje.



```
1 package isi.died.tp.estructuras;
2
3 import isi.died.tp.dominio.Insumo;
4
5 public class InsumoPesoValorInteger {
6
7     private Insumo insumo;
8     private Integer pesoFaltante; //un stock tiene una cantidad de insumo y una cantidad faltante, pesoFaltante es lo que pesa esa cantidad que falta
9     private Integer valorFaltante; //valor de la cantidad que falta
10
11     //Constructor
12     public InsumoPesoValorInteger(Insumo insumo, Integer pesoFaltante, Integer valor){
13         this.insumo=insumo;
14         this.pesoFaltante=pesoFaltante;
15         this.valorFaltante=valor;
16     }
17 }
```

Si por ejemplo en la planta A faltan 5 unidades de madera, en la planta B faltan 10 y en la C faltan 30. entonces el InsumoPesoValorInteger de Madera será:

Insumo = Madera

PesoFaltanteEnIndustria = 45

ValorFaltanteEnIndustria = PesoFaltanteEnIndustria \* Insumo.getCosto()

En la sección de Descargos se hace una importante aclaración respecto la incompatibilidad entre los Insumos Líquidos y el cálculo de Mejor Envío de esta sección.