Programación Básica 1 – Mesa de examen final Marzo 05-03-2025 TN

Se nos solicitó desarrollar un producto software para realizar la gestión de traslados y viajes de una empresa de traslados de mascotas. Este software deberá contar con un inicio de sesión basado en un numero de gestión el cual deberá comenzar y verificar que el mismo comience con el carácter de '#'.

En esta primera entrega, realizaremos un desarrollo que nos permita agregar traslados y asignarles uno o muchos viajes.

Para lograr esto, deberemos completar las siguientes clases del proyecto suministrado:

Clase PruebaSistema:

Será la clase que contenga al método main y deberemos desarrollar lo siguiente para garantizar el correcto funcionamiento:

A- Para iniciar el sistema, el usuario debe ingresar un número de gestión. Este número debe comenzar con el carácter especial #. El sistema deberá verificar que el número ingresado cumpla con esta condición. En caso contrario, se solicitará nuevamente hasta que el usuario ingrese un número de gestión válido.

Casos a examinar:

○ ***AGREGAR_DESTINO***: * Se debe agregar un nuevo destino para el traslado. Si el destino se agrega correctamente, el sistema deberá mostrar el mensaje: "Destino agregado correctamente". En caso de error, se deberá mostrar el mensaje: "No se pudo agregar el destino".
○ ***AGREGAR_VIAJE*:** * Se debe agregar un viaje según el traslado determinado. Si la operación se realiza con éxito, el sistema deberá mostrar el mensaje: "El viaje se agregó correctamente". En caso de error, se deberá mostrar el mensaje: "No se pudo agregar el viaje".
○***VER_RESUMEN***: * Una vez asignados los viajes a sus respectivos destinos, se deberá generar y mostrar un resumen de la información que se podrá consultar en todo momento para visualizar la información actual.

Nombre y Apellido DNI FIrma

○ **SALIR:** Termina la ejecución del programa e imprime un mensaje de finalización del programa de manera correcta.

- **private static MenuPrincipal ingresarOpcionDelMenuPrincipalValidada():** El usuario debe ingresar una opción y el sistema deberá verificar que esté dentro del rango permitido. Si la opción ingresada no es válida, se solicitará nuevamente hasta que se ingrese una opción correcta.
  Además, debe utilizar estrategias de normalización de datos para procesar la información (ver clase String).
- **private static boolean agregarNuevoDestino(GestionDeTraslado actual) :** Se debe solicitar el código, la ciudad y la distancia para definir el destino correspondiente y agregarlo al sistema.
- **private static boolean agregarViaje(GestionDeTraslado actual):** * Se debera ingresar el codigo de destino (-1 salir), verificar si el destino ya ha sido ingresado con anterioridad e informar por pantalla el mensaje de "Error. El destino [codigoDestino] ya se ha ingresado o no ha sido definido por el usuario. Si el destino no se encuentra registrado, solicitar el nombre del cliente y añadirle un numero aleatorio definido en el rango de 7 a 99. A continuación Ingrese el porcentaje de descuento (0~100) y agregue el nuevo viaje solicitado.

  private static void verResumen(GestionDeTraslado actual): * Se debe imprimir un resumen de los viajes realizados, indicando: El viaje con el mayor precio. El importe total acumulado.

Clase GestionDeTraslado

Esta clase contiene la lógica principal de las operaciones que soportará el software. Completar el constructor, atributos, constantes, getters, setters y métodos necesarios para asegurar el correcto funcionamiento, además de los siguientes métodos:

A- La cantidad de destinos totales esta predeterminado en 20 y para los Viajes sera de 50.

- **public boolean agregarDestino(Destino destino):** Agrega una nuevo destino si no se encuentra previamente registrado con anterioridad.
- **public Destino buscarDestino(int codigo) :** debe verificar si el destino ya se encuentra previamente o no ingresado en el sistema.
- **public boolean agregarViaje(Viaje viaje):** Agrega una nuevo viaje si no se encuentra previamente registrado con anterioridad.

Nombre y Apellido DNI FIrma

- **public double calcularCostoDelViaje(Viaje viaje):** realiza el calculo del costo del viaje.
- **public double calcularImporteTotal():** Calcula el importe total de los viajes.
- **public double calcularImportePromedio():** Calcula el importe promedio de los viajes.
- **public Viaje[] obtenerViajesOrdenadosPorCostoAscendente():** Obtiene los viajes ordenados por costo ascendente.
- **public String viajeDeMayorPrecio():** Debe determinar el viaje de mayor precio y devolver todos los datos del destino y el valor del viaje hallado.

Clase Destino

Representa la abstracción del destino del viaje. Está formado por un código, la ciudad y la distancia a la que se encuentra. Puede tener mas información que deberá desarrollar en caso de ser necesario.

Clase Viaje:

Representa la abstracción del destino del viaje. Está formado por un destino, nombre y un porcentaje De Descuento. Puede tener mas información que deberá desarrollar en caso de ser necesario.

Lineamientos

- Deberá ajustar el nombre del proyecto suministrado indicando sus apellidos y nombres en donde dice “ApellidosNombres”: PB12025FM2619-TN-**ApellidosNombres.**
- No es posible modificar la firma de los métodos existentes, pero es posible agregar todos los métodos que se consideren necesarios.
- Se pueden aplicar las mejoras que considere necesarias, siempre cumpliendo el punto anterior.
- Se deberán completar los métodos y código faltante para que el sistema funcione correctamente.
- La versión de java del proyecto deberá ser ajustada a la del laboratorio.

Condiciones de aprobación

Se considerará desaprobado todo proyecto que:

a. **no compile**, o,
b. no se identifiquen los atributos necesarios para resolver lo solicitado, o,
c. los tipos de datos elegidos para los atributos no sean los adecuados, o,
d. no cumpla con las condiciones mínimas descritas a continuación.

Se considerará aprobado aquel examen que cumpla con el 70% de puntaje.

Clase Prueba Sistema	void main(String[] args)	10%
	private static MenuPrincipal ingresarOpcionDelMenuPrincipalValidada()	5%
	private static boolean agregarNuevoDestino(GestionDeTraslado actual)	5%
	private static boolean agregarViaje(GestionDeTraslado actual):	10%
	private static void verResumen(GestionDeTraslado actual):	5%
Clase GestionDeTraslado	public boolean agregarDestino(Destino destino):	5%
	public Destino buscarDestino(int codigo)	5%

Nombre y Apellido DNI FIrma

	public boolean agregarViaje(Viaje viaje):	5%
	public double calcularCostoDelViaje(Viaje viaje)	10%
	public double calcularImporteTotal():	5%
	public double calcularImportePromedio():	10%
	public Viaje[] obtenerViajesOrdenadosPorCostoAscendente():	5%
	public String viajeDeMayorPrecio():	10%
	public String toString()	5%
Todas las Clases	Completar el constructor, atributos, getters y setters, así como también, cualquier otro método necesario. Importante: No está permitido modificar el constructor.	5%
Total		100%

Nombre y Apellido DNI FIrma