Instituto de Tecnología ORT
Carrera de Analista de Sistemas
Materia: Taller De Programación 1

FINAL

Leé con cuidado el enunciado y por lo menos dos veces para detectar las clases y estructuras necesarias para resolver lo pedido. Pensá bien la estrategia de resolución antes de comenzar el desarrollo. El objetivo de este examen es **evaluar la correcta aplicación de los conceptos y técnicas** vistos hasta el momento:

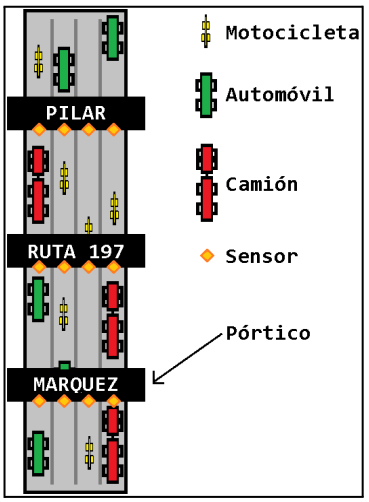
- Correcta definición de clases y asignación adecuada de sus responsabilidades.
- Correcta aplicación de Herencia y Polimorfismo, incluyendo interfaces.
- Uso adecuado de excepciones
- Uso adecuado de TADs
- Modularización reutilizable y mantenible con uso de métodos con correcta parametrización y correcto encapsulamiento, utilizando *setters* y *getters* sólo cuando corresponda y con la visibilidad pertinente.
- Correcta validación de los datos que ingresan al sistema.
- Importación y Exportación de proyectos Java desde Eclipse.

Antes de empezar, recordá:

- Descargar el archivo TP1-Final-C2-2023.zip. del aula virtual e importarlo en el Eclipse renombrandolo cómo TP1-Sede-Curso-APELLIDOyNOMBRE.zip
- Al finalizar el examen, exportarlo en un archivo ZIP y subirlo en el aula virtual en el link correspondiente; el archivo a subir debe llamarse igual que el proyecto.

ENUNCIADO

Una empresa dedicada a realizar sistemas de tráfico inteligente nos encarga el desarrollo de un prototipo de sistema de peajes del tipo “Free-Flow” (los vehículos nunca frenan para abonar el peaje en las casillas de peaje).
La idea es simular el tráfico de diferentes tipos de vehículos que pasan por debajo de un pórtico y así obtener estadísticas de acuerdo a lo sucedido.
Para el desarrollo, va a existir una empresa **concesionaria** de la autopista que tiene un nombre (no puede ser nulo ni vacío) por ejemplo *“Panamericana”*.



La concesionaria (que es mostrable) debe estar preparada para contener N pórticos, aunque para este ejemplo vamos a generar solo tres (3) pórticos.
Cada **pórtico** (que es mostrable) tiene un nombre (por ejemplo *“Marquez”, “Ruta 197”, “Pilar”, etc.*) y 4 carriles en los que en cada uno de ellos se encuentra un **sensor**.
Estos sensores tienen la capacidad de detectar y procesar únicamente vehículos **automóviles**, **motocicletas** y **camiones**. Cualquier otro tipo de vehículo que haya sido detectado por un sensor pero que no sea de ese grupo de vehículos, deberá ser almacenado internamente en la concesionaria, como un error en una estructura en la que el último elemento en ingresar queda inmediatamente disponible para su utilización. Luego estos errores serán visualizados.
Los sensores son responsables de almacenar todos los vehículos que van procesando en orden en el que van ingresando.

Instituto de Tecnología ORT
Carrera de Analista de Sistemas
Materia: Taller De Programación 1

INFORMACIÓN SOBRE LOS VEHÍCULOS:
Todos los vehículos cuentan con una **patente** alfanumérica (cualquier caracter) y un **peso** en kilos que no puede ser menor que cero (0). Observá la siguiente tabla:

- **Automóviles**: La patente de un automóvil tiene que tener exactamente 6 caracteres (letras/números/etc.).
  Por ejemplo: *"AAA111", "123456", "ABCDEF" o "1+^H2%"*.
- **Camiones**: La patente de un camión tiene que tener más de 6 caracteres (letras/números/etc.) y debe contener al menos un espacio " ".
  Por ejemplo: *"AAA 112 11", "14 23456" o "ABCDE F5"*.
  Los camiones también tienen información acerca de su **tipo de acoplado**, que pueden ser (*CAMIÓN, TANQUE, TOLVA, PLATAFORMA*; ver imagen adjunta).
- **Motocicletas**: La patente de una motocicleta tiene que tener exactamente 4 (letras/números/etc.) y comenzar con una 'A'.
  Por ejemplo: *"A123", "A A5", "A%%%" o "AA66"*.
  Las motocicletas también tienen información acerca de su **cilindrada** (número entero) que no puede ser menor que 0 ni mayor a 5000.

ATENCIÓN: Para todos los vehículos, al asignarle una patente inválida se debe arrojar una excepción indicando lo sucedido.

CÁLCULO DEL IMPORTE SEGÚN EL VEHÍCULO:
Cuando un sensor detecta un vehículo, el mismo debe procesar el importe que debe abonar ese vehículo, según cierta lógica.
Para calcular el importe del peaje de un vehículo, siempre se utiliza la misma fórmula:
IMPORTE_BASE + **IMPORTE_ADICIONAL**.

- **Automóviles**:
  - IMPORTE_BASE: **250**
  - IMPORTE_ADICIONAL: **0** (cero).
- **Camiones**:
  - IMPORTE_BASE: **500**
  - IMPORTE_ADICIONAL: TipoDeAcoplado * MultiplicadorTipoDeAcoplado.
- **Motocicletas**:
  - IMPORTE_BASE: **100**
  - IMPORTE_ADICIONAL: Si la cilindrada es menor a 1500 se le suma la mitad del IMPORTE_BASE. Sino, se le suma la totalidad del IMPORTE_BASE.

NOTA:
Para el caso de los **Tractores**, se considera que el importe que pagan en los peajes es despreciable (o regalado) y que sus patentes siempre son válidas.

CÓDIGO A DESARROLLAR:

1. Crear las clases necesarias, con sus constructores, y validaciones necesarias.
2. Implementar, donde sea necesario, la interfaz Mostrable.

Clase **`Concesionaria`**

3. Desarrollar el método `void agregarPortico(String nombre)` que agrega el pórtico a la concesionaria.
4. Desarrollar el método `void procesarVehiculo(String nombrePortico, Vehiculo vehiculo, int carril)` que procesa el vehículo del carril en el pórtico enviado por parámetro. En caso de que el proceso falle o el pórtico no exista, se debe agregar a los errores que luego serán visualizados. (existen dos constantes, para esto).
5. Desarrollar el método `void mostrarEstadisticasPorticos()` que muestra por consola las estadísticas de cada uno de los pórticos ordenados por nombre del pórtico.
6. Desarrollar el método `void mostrarErrores()` que muestra por consola los errores recolectados según su orden cronológico de aparición.

Clase **`Portico`**

7. Desarrollar el método `float procesarVehiculo(Vehiculo v, int carril)` procesar el vehículo en el carril/sensor indicado y retornar el importe.
8. Desarrollar el método `float getImporteAcumulado()` retorna el importe total acumulado de todos los sensores del portico.

Clase **`Sensor`**

9. Desarrollar el método `float procesarVehiculoRetornarImporte(Vehiculo v)` que procesa el vehículo y retorna el importe. Este método arroja una excepción en el caso de que la clase del vehículo no sea soportada (ver el punto anterior).
10. Desarrollar el método `float getImporteProcesados()` que retorna el importe total acumulado de los vehículos que fueron procesados por el Sensor.
11. Desarrollar el método `int getCantidadProcesados()` que retorna la cantidad de vehículos que fueron procesados por el Sensor.

Luego de desarrollar los métodos, la salida debe arrojar un resultado como el que se muestra a continuación:

```
--------------- AGREGANDO PORTICOS ---------------
Se ha agregado el pórtico [Pilar]
Se ha agregado el pórtico [Ruta 197]
El nombre del pórtico no puede ser nulo ni estar vacío.
Se ha agregado el pórtico [Marquez]

--------------- PROCESAR VEHICULOS ---------------
Se procesó [Vehiculo (Automovil) [patente=CAR111, pesoEnKilos=2100]] Carril/Sensor [1]
Se procesó [Vehiculo (Automovil) [patente=CAR111, pesoEnKilos=2100]] Carril/Sensor [2]
Se procesó [Vehiculo (Automovil) [patente=CAR111, pesoEnKilos=2100]] Carril/Sensor [1]
Se procesó [Vehiculo (Automovil) [patente=CAR222, pesoEnKilos=2500]] Carril/Sensor [1]
Se procesó [Vehiculo (Motocicleta) [patente=A000, pesoEnKilos=1100]] Carril/Sensor [3]
Se procesó [Vehiculo (Motocicleta) [patente=A000, pesoEnKilos=1100]] Carril/Sensor [3]
```

```
                                                                                Se
procesó [Vehiculo (Motocicleta) [patente=AAAA, pesoEnKilos=800]] Carril/Sensor [2]
Se procesó [Vehiculo (Motocicleta) [patente=AAAA, pesoEnKilos=800]] Carril/Sensor [1]
Se procesó [Vehiculo (Camion) [patente=GGG GGG, pesoEnKilos=4500]] Carril/Sensor [2]
Se procesó [Vehiculo (Camion) [patente=GGG GGG, pesoEnKilos=4500]] Carril/Sensor [1]
Se procesó [Vehiculo (Camion) [patente=YYY YYY, pesoEnKilos=4500]] Carril/Sensor [1]
Se procesó [Vehiculo (Camion) [patente=YYY YYY, pesoEnKilos=4500]] Carril/Sensor [1]

--------------- MOSTRAR ESTADISTICAS ---------------

--------------- MOSTRAR ESTADISTICAS PORTICOS ---------------
Estadísticas del pórtico [Marquez   ] es [ 3654.40]
Estadísticas del pórtico [Pilar     ] es [  400.00]
Estadísticas del pórtico [Ruta 197  ] es [ 1551.20]

--------------- MOSTRAR ERRORES ---------------
ERROR: El pórtico [YYYYYYYYYY] no existe.
ERROR: El pórtico [XXXXXXXXXX] no existe.
ERROR: Error procesando el vehículo [El tipo de vehículo [Tractor] no está soportado por la
plataforma.].
ERROR: Error procesando el vehículo [Index 15 out of bounds for length 4].
ERROR: Error procesando el vehículo [El tipo de vehículo [Tractor] no está soportado por la
plataforma.].
```