**Requerimientos funcionales:**

|  |  |
| --- | --- |
| Nombre | **1A-** Obtener el ranking de las N franjas horarias que tengan más infracciones. |
| Descripción | Obtener el ranking de las N franjas horarias que tengan más infracciones. Para cada franja horaria hay que dar el intervalo que cubre la franja, el total de infracciones, el porcentaje de infracciones sin accidente, el porcentaje de infracciones con accidente y el valor total a pagar por las infracciones. |
| Entradas | El valor N como la cantidad de las franjas horarias. |
| Salidas | Una Cola de Prioridad orientada a mayor número de infracciones en cada franja horaria. |
| Estimación de complejidad temporal | O(1) para insertar en Cola de Prioridad |

|  |  |
| --- | --- |
| Nombre | **2A-** Realizar el ordenamiento de las infracciones por Localización Geográfica (Xcoord, Ycoord). |
| Descripción | Realizar el ordenamiento de las infracciones por Localización Geográfica (Xcoord, Ycoord). Si hay valores de Xcoord iguales, el criterio de ordenamiento lo define el valor Ycoord. Todas las infracciones con la llave (Xcoord, Ycoord) deben agruparse bajo un mismo objeto. |
| Entradas | Arreglo Dinamico con los objetos cargados, tupla con (Xcoord, Ycoord) |
| Salidas | Una Tabla de Hash ordenada por localización geográfica. |
| Estimación de complejidad temporal | O(n) para agregar en la tabla Hash |

|  |  |
| --- | --- |
| Nombre | **3A-** Buscar las infracciones por rango de fechas [Fecha Inicial (Año/Mes/Día), Fecha Final (Año/Mes/Día)]. |
| Descripción | Representar las infracciones por fecha (Año/Mes/Día) en un Árbol Balanceado por Fecha (TicketIssueDate). Para cada posible fecha en el rango definido con al menos una infracción, se requiere informar la fecha (Año/Mes/Día), el total de infracciones, su porcentaje de infracciones sin accidente, su porcentaje de infracciones con accidente y el valor total a pagar por las infracciones. Las fechas deben informarse en orden ascendente (cronológicamente). |
| Entradas | Arreglo Dinamico con los objetos cargados, rango de fechas. |
| Salidas | Árbol Balanceado por Fecha (TicketIssueDate). |
| Estimación de complejidad temporal | O(log (n)) + O(log(n)) (insertando y buscando) |

|  |  |
| --- | --- |
| Nombre | **1B-** Obtener el ranking de las N tipos de infracción (ViolationCode) que tengan más infracciones. |
| Descripción | En el ranking deben aparecer los tipos de infracción de mayor a menor número de infracciones. Para cada tipo de infracción debe mostrarse su código, total de infracciones, su porcentaje de infracciones sin accidente, su porcentaje de infracciones con accidente y el valor total a pagar por las infracciones. |
| Entradas | Tipo de infracción (ViolationCode) |
| Salidas | Una Cola de Prioridad orientada a cantidad de infracciones. |
| Estimación de complejidad temporal | O(1) para insertar en Cola de Prioridad. |

|  |  |
| --- | --- |
| Nombre | **2B-** Realizar el ordenamiento de las infracciones por Localización Geográfica (Xcoord, Ycoord). |
| Descripción | Realizar el ordenamiento de las infracciones por Localización Geográfica (Xcoord, Ycoord). Si hay valores de Xcoord iguales, el criterio de ordenamiento lo define el valor Ycoord. Todas las infracciones con la llave (Xcoord, Ycoord) deben agruparse bajo un mismo objeto. |
| Entradas | Arreglo Dinámico con los objetos cargados, tupla con (Xcoord, Ycoord). |
| Salidas | Un Árbol Balanceado ordenado por localización geográfica. |
| Estimación de complejidad temporal | O(log(n)) agregando al árbol. |

|  |  |
| --- | --- |
| Nombre | **3B-** Buscar las franjas de fecha-hora donde se tiene un valor acumulado de infracciones en un rango dado [US$ valor inicial, US$ valor final]. |
| Descripción | Buscar las franjas de fecha-hora donde se tiene un valor acumulado de infracciones en un rango dado [US$ valor inicial, US$ valor final]. Usar una representación de Árbol Balanceado por el valor acumulado de infracciones para cada fecha-hora. Es decir, para cada fecha en el semestre seleccionado, se debe calcular el valor acumulado (US$) de las infracciones para cada franja de hora completa (00:00:00 – 00:59:59, 01:00:00 – 01:59:59, ..., 23:00:00 – 23:59:59). |
| Entradas | Arreglo Dinamico con los objetos cargados, rango de US$ inicial y US$ final. |
| Salidas | Un Árbol Balanceado ordenado por el valor acumulado de infracciones en cada fecha-hora. |
| Estimación de complejidad temporal | O(log(n)) agregando |

|  |  |
| --- | --- |
| Nombre | **1C-** Obtener la información de una localización dada (AddressID). |
| Descripción | Para la localización de consulta (AddressID), se requiere informar el total de infracciones, el porcentaje de infracciones sin accidente, el porcentaje de infracciones con accidente, el valor total a pagar por las infracciones, y el Id de la calle (StreetSegId). |
| Entradas | AddressID. |
| Salidas | Tabla de Hash con todos los AddressID con sus respectivos datos. |
| Estimación de complejidad temporal | O(n) agregando + O(n) consultando. |

|  |  |
| --- | --- |
| Nombre | **2C-** Obtener las infracciones en un rango de horas [HH:MM:SS inicial, HH:MM:SS final]. |
| Descripción | Para el rango de horas de consulta, se requiere dar:  **Información general:** el total de infracciones, el porcentaje de infracciones sin accidente, el porcentaje de infracciones con accidente, y el valor total a pagar por las infracciones.  **Información por código (ViolationCode):** Para las infracciones resultantes, hay que agruparlas por código. Informar el detalle de código y el total de infracciones. |
| Entradas | Arreglo dinámico con los datos cargados, rango de horas. |
| Salidas | Tabla de Hash con la información requerida según el rango de horas. Usando el rango de horas como key. |
| Estimación de complejidad temporal | O(n) agregando + O(n) consultando. |

|  |  |
| --- | --- |
| Nombre | **3C –** Obtener el ranking de las N localizaciones geográficas (Xcoord, Ycoord) con la mayor cantidad de infracciones. |
| Descripción | Por cada localización (Xcoord, Ycoord) hay que reportar el total de infracciones, su porcentaje de infracciones sin accidente, su porcentaje de infracciones con accidente, el location, el Id de la dirección (AddressID) y el Id de la calle (StreetSegId). |
| Entradas | Tupla con Xcoord y Ycoord, número (N) de localizaciones geográficas a consultar. |
| Salidas | Cola de prioridad. |
| Estimación de complejidad temporal | O(1) agregando a Cola de Prioridad. |

|  |  |
| --- | --- |
| Nombre | **4C –** Mostrar una gráfica ASCII con la información de las infracciones por código (ViolationCode). |
| Descripción | Mostrar una gráfica ASCII con la información de las infracciones por código (ViolationCode). La gráfica debe mostrar los tipos de infracción ordenados de mayor a menor por su número de infracciones. Se espera mostrar una representación ASCII. |
| Entradas | ViolationCode a buscar. |
| Salidas | Cola de Prioridad orientada a mayor número de infracciones. |
| Estimación de complejidad temporal | O(1) agregando. |

**JUSTIFICACIÓN DE ESTRUCTURAS A USAR:**

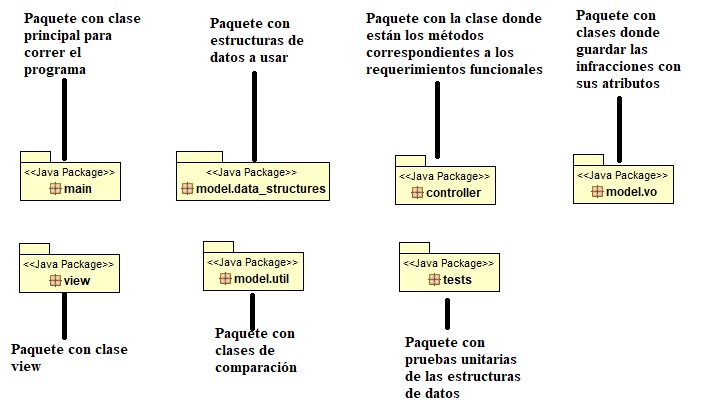
**1C)** Se quiere usar Cola de Prioridad orientada a AddressID mayor. Esto pues a pesar de que no hay necesidad de mostrar datos mayores que otros, es O(1) insertando y en espacio ocupa N. No se usa Hash porque no hay que guardar muchos arreglos en muchas llaves, simplemente los datos requeridos por AddressID.

**2C)** Se quiere usar Tabla de Hash porque se deben almacenar varias infracciones según un rango de horas. Luego se requieren ciertos datos de estas infracciones en el rango, por lo cual es más fácil acceder a los arreglos guardados en la llave de la tabla (generadas con una de las horas del rango) y luego extraer la información requerida.

**3C)** Se usará Cola de Prioridad orientada por cantidad de infracciones porque es rápida insertando O(1) (asumiendo que los datos están previamente ordenados) y ocupa O(n). Además, dado que el requerimiento habla de pedir N datos mayores, Cola de Prioridad lo facilita con la función delMax(). de complejidad O(1).

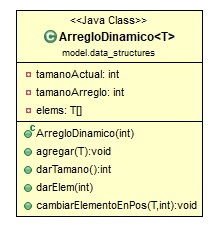
**4C)** Dado que el requerimiento habla de pedir los datos mayores para generar una tabla ASCII, una Cola de Prioridad lo facilita con la función delMax(). de complejidad O(1). Con esto se consiguen las infracciones por código con más infracciones.

**UML solución:**

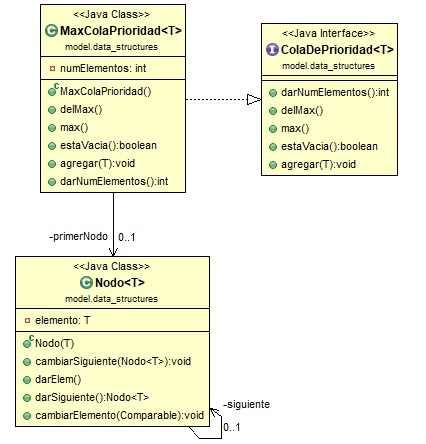
****

**UML estructuras:**

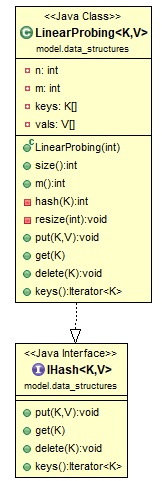
* **UML Arreglo Dinámico:**

****

* **UML Cola de Prioridad**

****

* **UML Tabla de Hash Linear Probing:**

****

* **Evidencia de carga:**

