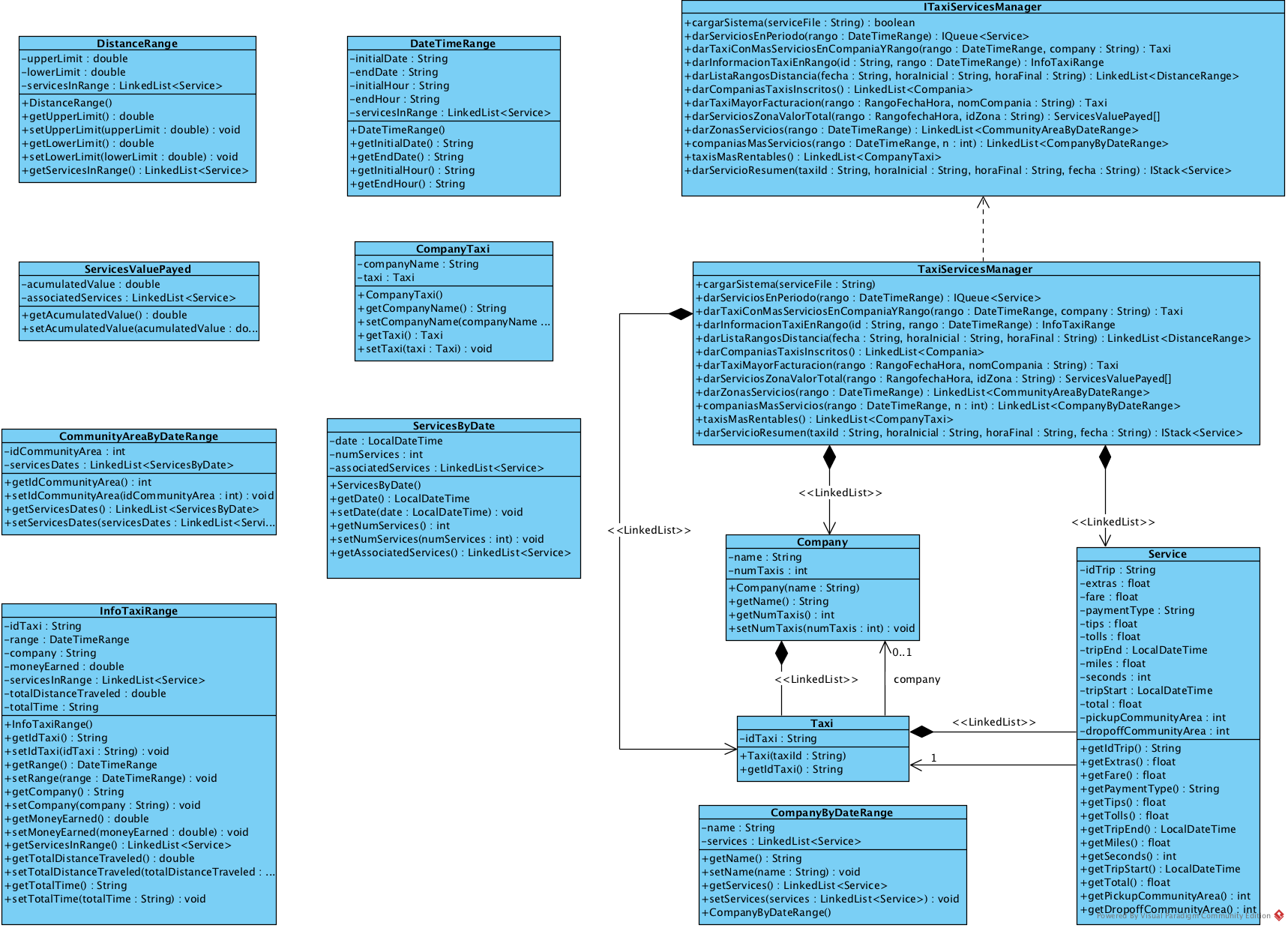
Juan Esteban Méndez Roys 201531707

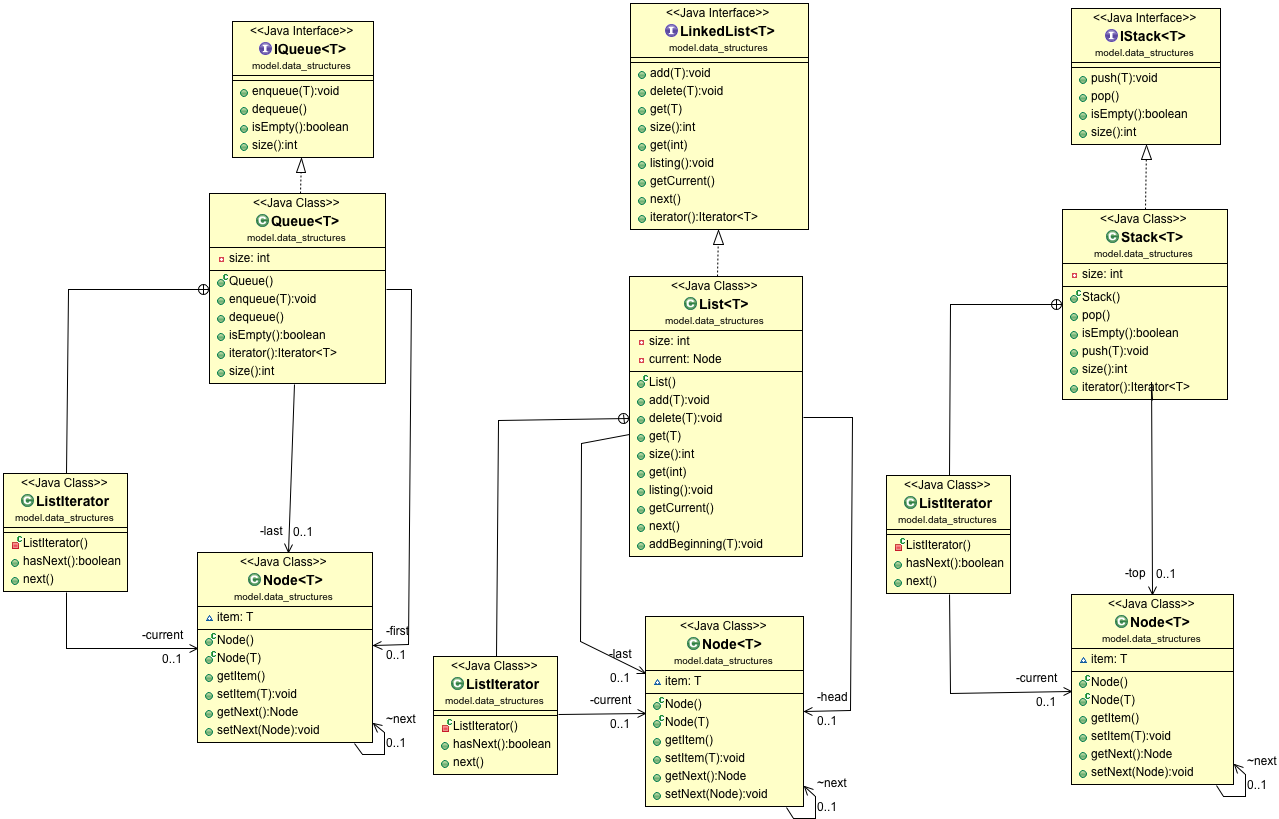
Santiago Rojas 2014

**Documento de Diseño Proyecto 1**

1. Modelo del Mundo en UML:



1. Modelo de estructuras de datos en UML:



1. Estimación de complejidad de requerimientos:

**1A**- Generar una Cola con todos los servicios de taxi que se prestaron en un periodo de tiempo dado por una fecha/hora inicial y una fecha/hora final de consulta. El inicio y terminación del servicio debe estar incluido dentro del periodo de consulta. Los servicios deben mostrarse en orden cronológico de su fecha/hora inicial.

Complejidad del algoritmo:

O(2N+2) ~ O(N)

Solo se hará un recorrido, habrá una asignación de variable n+1 comparaciones y n incrementos. (Un for)

**2A-** Buscar el taxi de una compañía dada que más servicios inició en un periodo de tiempo dado por una fecha/hora inicial y una fecha/hora final de consulta.

Complejidad del algoritmo:

O(). Se usarán 3 fors, el primero para ubicar la compañía dada, el segundo para hacer un recorrido de los taxis de cada compañía, y el tercero para seleccionar los servicios de cada taxi que cumplan los parametros de hora estipulados. La operación que mas se repetira es la comparación entre horas dentro de cada servicio.

**3A-** Buscar la información completa de un taxi, a partir de su identificador, en un periodo de tiempo dado por una fecha/hora inicial y una fecha/hora final de consulta. Incluye el nombre de su compañía y los valores totales de plata ganada, de servicios prestados, de distancia recorrida y de tiempo total de servicios.

Complejidad del algortimo:

O(). Se tendra que hacer un recorrido de la lista de taxis que contiene la clase “TaxiServicesManager”. Al encontrar el taxi se hara un recorrido de su lista de servicios con el uso de un segundo ‘for’, asignandole a un acumulador el valor de cada servicio para así encontrar el valor total de plata ganada por el taxi. Habrá una segunda asignación que será la operación que mas se repetirá junto a la anterior, para asi encontrar el acumulado de distancia recorrida por un taxi. La tercera y última asignacion que también será la mas repetida junto a las anteriores servirá para encontrar el tiempo total de servicios del taxi. El numero de servicios prestados sera simplemente el tamaño de la lista de el respectivo taxi.

Solo se hará un recorrido, habrá una asignación de variable n+1 comparaciones y n incrementos. (Un for)

**4A-** Retornar una lista de rangos de distancia recorrida, en la que se encuentran todos los servicios de taxis servidos por las compañías, en una fecha dada y en un rango de horas especificada. La información debe estar ordenada por la distancia recorrida, así la primera posición de la lista tiene a su vez una lista con todos los servicios cuya distancia recorrida esta entre [0 y 1) milla. En la segunda posición, los recorridos entre [1 y 2) millas, y así sucesivamente.

Complejidad del algoritmo:

O(). Se planea recorrer con un ‘for’ la lista de servicios de la clase ‘TaxiServicesManager’. Se irán instanciando objetos de la clase ‘DistanceRange’ dependiendo de la información encontrada en los servicios que se encuentran entre el rango de horas que entra como parametro. Se agregaran dichos objetos a una lista que estara ordenada de menor a mayor por el atributo ‘lowerLimit’. Para agregar a la lista se tendrá que recorrer parcialmente para asi poder agregar los objetos en orden , por eso el ‘n^2’, pues serán dos ciclos anidados.

**1B**- Mostrar la información de las compañías de taxi consistente en: El total de compañías que tienen al menos un taxi inscrito y el total de taxis que prestan servicio para al menos una compañía. Adicionalmente, generar la lista alfabética de compañías a las cuales aparecen inscritos los servicios de taxi de la fuente de datos de consulta. Por cada compañía debe informarse su nombre y el número de taxis que tiene registrados.

Complejidad del algoritmo:

O(N) Se usará un for para dar el total de compañias con al menos un taxi.

O(N) Se usará un for para dar el total de taxis que prestar servicio para al menos una compañía.

O() Se usará un lagoritmo de inserción para ordenar la lista de compañias.

O(+N+N) ~

**2B-** Buscar el taxi de una compañía dada que mayor facturación ha generado en un periodo de tiempo dado por una fecha/hora inicial y una fecha/hora final de consulta.

Compleljidad algoritmo:

O()

En este caso, se usaran 3 ciclos anidados. El primero para recorrer la lista de compañias hasta encontrar la que entra por parametro, el siguiente para encontrar el taxi y el ultimo para recorrer los servicios de ese taxi. La operación que más se repetira sera la comparación entre el acumulador de la plata ganada por ese taxi contra la cantidad contenida por la variable ‘mayor’.

**3B-** Buscar la información completa de una zona de la ciudad en un periodo de tiempo dado por una fecha/hora inicial y una fecha/hora final de consulta. El número total de servicios que se recogieron en la zona de consulta y terminaron en otra zona y el valor total pagado por los usuarios; el número total de servicios que se recogieron en otra zona y terminaron en la zona de consulta y el valor total pagado por los usuarios, y el total de servicios que iniciaron y terminaron en la misma zona de consulta y el valor total pagado por los usuarios.

Complejidad del algoritmo:

O(N) Ya que se usará solo un ciclo para recorrer la lista de servicios en “TaxiServicesManager”.

**4B-** Retornar una lista con todas las zonas de la ciudad (ordenadas por su identificador). Cada zona debe tener el total de servicios iniciados en dicha zona en un rango de fechas. Por ejemplo, la primera posición de la lista tiene todos los servicios de la primer zona, en dicha posición, se tiene una lista de fechas (ordenadas cronológicamente) con el total de servicios asociados a dicha fecha.

Complejidad del algoritmo:

O(N + ) ~

El primer termino antes de simplificar es N piuesto que es el ciclo que recorrerrá la lista de servicios de el sistema. El segundo termino corresponde a la del algoritmo de ordenamiento que utilizaremos el cuál será el de inserción, que tiene una complejidad de .

**2C-** Identificar el top *X* de compañías que más servicios iniciaron en un periodo de tiempo dado por una fecha/hora inicial y una fecha/hora final. El valor X es un dato de consulta. El resultado debe mostrar el Top *X* de compañías ordenadas por el número de servicios de mayor a menor. Por cada compañía debe informarse su nombre y su número de servicios de respuesta.

Complejidad del algoritmo:

O(N+)

Primer termino corresponde a los N servicios que seran recorridos. El segundo termino al del algoritmo de inserción que se implementara para ordenar los datos. Los datos son instancias de la clase “CompanyByDateRange” y se ordenaran alfabeticamente en orden ascendente.

**3C-** Buscar el taxi más rentable de cada compañía. El taxi más rentable de una compañía es aquel cuya relación de plata ganada y distancia recorrida en los servicios prestados es mayor.

Complejidad del algoritmo:

O()

Se usaran dos ciclos, uno que recorra cada compañía, y otro que recorra cada taxi de las compañias, ademas de las asignaciones de variables, como la que retornará el taxi más rentable. Las operaciones que mas se repetiran son las que estarán dentro del segundo ciclo de la anidación el cual será el mas interno.

**4C-** Dada la gran cantidad de datos que requiere el proyecto, se desea poder compactar información asociada a un taxi particular. Para ello usted debe guardar en una pila todos los servicios generados por el taxi en orden cronológico, entre una hora inicial y una hora final, en una fecha determinada. Para comprimir la información, usted debe ir guardando los servicios reportados por el taxi, siempre y cuando la distancia acumulada de dichos servicios sea inferior a un valor dado (por ejemplo 10 millas). Cuando la sumatoria de las distancias de los servicios acumulados en la pila sobrepasen el valor límite, usted deberá retirar los servicios de la pila y generar uno nuevo, que resuma los servicios analizados. Este servicio resumen, tiene como hora inicial, la menor hora de los servicios, como hora final la mayor hora de los servicios, como distancia la sumatoria de todas las distancias de los servicios; como duración, la sumatoria de las duraciones de los servicios; y como valor total de la carrera, la sumatoria de todos los valores de los servicios. Los demás campos deben quedar en el valor por defecto dependiendo del tipo de dato**.**

Complejidad algoritmo:

O().

Se utilizaran dos ciclos anidados. Uno que se encargara de recorrer la lista de taxis en el sistema y encontrar el que coincida con el ‘”ID” que entra como parametro al metodo, y el segundo que se encargará de recorrer la lista de servicios del taxi. Las operaciones dentro de este último ciclo serán las que más se repetirán, entre las cuales estará la instanciación de una pila y ocasionalmente la de nuevos objetos de la clase “Service” que seran añadidos a esta pila.