Instituto Tecnológico de Costa Rica
Campus Tecnológico Local San José

Escuela de ingeniería en computación
[IC-2001] Estructura de datos

Profesor:
Mauricio Avilés

Fecha:
II semestre 2022

Elaborado por:
Danny Jiménez Sevilla, c.2020018418
Emily Sánchez Orozco, c.2021067314
Daniel Cruz Oviedo, c. 2020087592

Proyecto #0
Sistema de administración de colas

Introducción

En la mayoría de las empresas es común ver diferentes métodos para llevar un orden a la hora de atender los clientes, el más común es por medio de colas físicas las cuales dependiendo de la cantidad de gente que llegue a requerir un servicio dentro de la empresa puede llegar a ser muy tedioso tener que estar atentos a moverse un espacio en la fila, así como es tedioso para la empresa el mantener un orden, esto sumado al tener que clasificar a los clientes según el servicio requerido, su prioridad y el área en que estos deben ser atendidos hacen que el manejo de una cola física sea un verdadero reto para las empresas y los clientes. Además de todo lo mencionado cabe destacar que el espacio físico en muchas empresas los espacios son reducidos, lo cual hace casi imposible el tener una cola física llegando incluso convertirse en un desorden dentro del área de atención, siendo también una gran incomodidad tanto para los clientes como para los propietarios de la empresa y/o trabajadores.

Partiendo de este punto la principal razón de diseñar una solución que nos ayuda a mantener este orden de una manera más automática, es la minimización de este trabajo el cual se puede lograr mediante recursos tecnológicos como lo son los sistemas informáticos, en este caso en específico el más practico sería un sistema de administración de colas, para esto se requiere diferentes partes y un análisis profundo de las necesidades.

En primer lugar, la necesidad más obvio que se debe satisfacer es el llevar un orden según la entrada de personas al lugar la cual es la función principal de la estructura de datos conocida como colas, pero también hay que tomar en cuenta que en estas empresas también existen diferentes tipos de clientes y que cada uno de ellos tiene un propósito diferente que es atendido según conveniencia de la empresa en una área específica es por ello que para este proyecto se ha decidido implementar un sistema de administración de colas un poco más específico, en el cual tenemos que enfocarnos en diferentes partes, si esto lo modelamos o pensamos de una manera simple las partes serian:

- Las áreas, las cuales serán manejadas dentro de programa como objetos que a su vez contienen más elementos, como lo son las ventanillas, estas áreas tienen una cola específica para personas de nivel prioritaria y para personas de propósito general
- Las ventanillas, en este caso hablamos de una forma específica del espacio en el que el cliente puede realizar sus gestiones, las ventanillas están contempladas dentro de un área en específico y son las encargadas de atender cliente por cliente.

- Las colas por supuesto son lo más importante del sistema, en este caso se deberían de manejar una cola distinta por área y a su vez una variación de las colas son las colas de prioridad, las cuales nos ayudaran a mantener un orden dependiendo del tipo de cliente.
- Dentro del sistema también se deben contemplar ciertas estadísticas que podrán determinar la eficiencia del sistema, para ello se deberían de diseñar una forma de llevar estadísticas como el tiempo de espera, la cantidad de tiquetes que se dan por área, entre otras.
- Otra parte para considerar es la versatilidad que debe tener el sistema, esto porque estamos diseñando un sistema que pueda funcionar para cualquier tipo de empresa, por lo cual es importante que el usuario pueda tener la capacidad de moldear aspectos del sistema a su gusto.
- Dentro de las partes más importantes esta la posibilidad de atender a un cliente, la cual sino existe tendríamos una cola sin terminar de clientes, lo cual haría el sistema obsoleto.

Tomando en cuenta estas partes se puede satisfacer las necesidades del problema que sucede en muchas empresas, además de ser una solución eficiente para olvidarse de llevar un orden físico de las personas que se atienden dentro de una empresa.

Presentación y análisis del problema

Descripción del problema

En las instituciones bancarias se maneja diariamente un gran flujo de personas que buscan diferentes servicios en distintas áreas ubicadas en el mismo edificio, dicha actividad es bastante compleja a la hora de organizar que cada persona que llega se dirija al servicio que desea adquirir por lo cual en este proceso se pierde veracidad y se aumentan los tiempos de espera de cada cliente entonces esto conlleva a que cada diligencia no se haga de la mejor manera como se deseara.

La falta de un sistema de información eficiente hace que los procesos se vuelvan más lentos, dispendiosos y se tenga que recurrir a métodos manuales, como por ejemplo que cuando los clientes ingresan tengan que primero consultar a que área se deben dirigir y cuál es la cola en la que se deben colocar según el servicio deseado por lo cual se pueden concluir que estos procesos no son los más adecuados para el manejo de clientes dentro del edificio.

Por otro lado en los bancos se maneja una cola para cada servicio que se ofrece y además en cada uno de esos servicios hay una cola para clientes preferenciales (personas con alguna discapacidad visual y/o auditiva, adultos mayores y mujeres embarazadas) y luego está la cola en donde se colocan todas las demás personas, por lo que esto causa que se den confusiones entre los clientes

y no sepan cual es la cola que les corresponde por lo que para realizar distintos trámites provoca pérdida de tiempo de los clientes.

Para lograr que el proceso de administración en las colas no sea deficiente y los clientes no tengan que esperar una gran cantidad de tiempo para poder ser atendidos en el orden de la cola se identifica la siguiente formulación del problema: ¿Cómo desarrollar un software que permita un proceso eficiente y efectivo para la administración de colas?

Este Proyecto de Software viene a ofrecer una alternativa de solución a la permanente necesidad por parte de las instituciones bancarias u otra empresa, de poder crear una solución virtual o procesos más satisfactorios para que cuando los clientes ingresan puedan tener toda la información necesaria que requieren para dirigirse al área y cola que les corresponde según el servicio que desean adquirir y el tipo de cliente dependiendo si es preferencial o normal haciendo que el proceso sea de manera más ágil y oportuno.

Metodología

Para la realización del proyecto se empezó realizando un diagrama de clases según UML, de esta forma fue mucho más fácil entender que clases debíamos crear, cuales métodos llevaba cada clase y cuales estructuras aprendidas durante el curso usar para las funcionalidades necesarias, después de crear el diagrama de clases se empezó realizando las clases Area, Service y ServiceWindow. En el caso de la clase Area es la que más peso tiene, ya que se encarga de realizar una gran parte las operaciones que se pide para la realización del proyecto. Para dicha clase, nos encargamos de crear varias operaciones, entre las que se podría mencionar son las siguientes:

1. El constructor y constructor de copia.
2. Método para añadir y eliminar los servicios con respecto a un área en específico.
3. Método para añadir un cliente a una ventanilla de un respectivo servicio elegido previamente.
4. Métodos accesores para los atributos de la clase Area.
5. Método para imprimir la situación de las colas sea normal o preferencial.

La clase Service se encarga de inicializar los atributos que debe llevar como lo son el nombre y el código, después de que la implementamos decidimos añadir un tercer atributo que se llama ticketsGiven para llevar un control de los clientes que han elegido ese servicio de este modo se

harían más fáciles algunas implementaciones. Para esta clase solo se añadió como métodos el constructor, constructor de copia y los métodos accesores.

La clase ServiceWindow al igual que la anterior lleva la inicialización de los atributos código, cantidad de tiquetes atendidos en esa ventana y un puntero de tipo ticket. Como métodos tiene un constructor, método para atender un cliente y un método el cual accesor para retornar la cantidad de clientes que fueron atendidos en esa ventanilla.

Posteriormente se decidió crear otra clase para el Tiquete, esta clase tiene como atributos el código del tiquete, si es para un cliente preferencial, la hora de llegada y la hora de salida. Para sus métodos se le añadió el constructor y un método que calcula la duración del cliente en la cola calculando su hora de llegada y su hora de salida.

Analizando las clases se tomó la decisión de implementar un Controlador esta clase seria de bastante utilidad para acceder más rápido a las funcionalidades de todas las clases creadas anteriormente, en esta clase se crearán las listas de todas las áreas disponibles y los servicios que se ofrecen y como métodos se añadieron los siguientes:

1. Generar un tiquete.
2. Añadir servicio.
3. Eliminar servicio.
4. Añadir área.
5. Eliminar área.
6. Atender.
7. Ver el estado de las colas.
8. Promedio de espera de los clientes en un área.
9. Cantidad de tiquetes atendidos en una ventanilla.
10. Cantidad de tiquetes que se han repartido.
11. Cantidad de tiquetes preferenciales repartidos.

Para las estructuras se decidió utilizar ArrayList, para las listas de servicios y áreas, LinkedQueue y LinkedPriorityQueue para las colas que se asignaran en cada área o ventanilla, a continuación, se adjunta dentro de los documentos el diagrama de clases creado con las respectivas clases creadas y sus relaciones entre las mismas para que sea más entendible como lo realizamos.

Análisis crítico

Durante la implementación del proyecto, se pudo resolver e implementar exitosamente las clases que fueron las más importantes durante la realización del proyecto, las cuales fueron Area, Service, ServiceWindow y Ticket. En el caso de la clase Area le implementamos varios métodos que, entre los más importantes se detallan a continuación: En la clase Controlador, se pudo implementar satisfactoriamente, ya que su principal función era controlar los métodos implementados anteriormente, por lo que facilito bastante el trabajo y pudimos hacerlo más útil a la hora de hacer el menú.

El diagrama de clases nos facilitó bastante entender el proyecto ya que verlo en texto y luego diagramado es bastante más entendible por lo que se logró realizar con éxito. Después de algunas pruebas se tuvo que cambiar algunos métodos en las clases de las estructuras ya disponibles como la LinkedQueue o LinkedPriorityQueue para que funcionara correctamente con las otras clases creadas.

Conclusiones

1) Las listas simplemente enlazadas, así como también otro tipo de estructuras similares, son útiles a la hora de trabajar problemas como pilas y colas, ya que se maneja la misma lógica de agregar, borrar o buscar elementos. Como, por ejemplo, puede ser la fila de un banco en donde se necesita de estas tres funciones principales de listas simplemente enlazadas, para registrar quien es el primero de la lista, el último o el tiempo que lleva en espera, etc.
2) Los diagramas de clases son uno de los tipos de diagramas más útiles en UML a la hora de comenzar a realizar un proyecto, ya que trazan claramente la estructura de un sistema al modelar sus clases, atributos, operaciones y relaciones entre objetos, haciendo que a la hora de pasarlo al código sea mucho más sencillo, además que para muchas personas es más fácil entender de que se trata y cuáles son las funcionalidades del proyecto con los diagramas.
3) La aplicación de pruebas durante las etapas del proyecto puede acelerar el proceso de depuración al diagnosticar el comportamiento de componentes codificados más simples y nuevos en lugar de ejecutar todo el proyecto al final.
4) La librería ctime/time de C++ es bastante útil y eficiente para un método donde se requiera obtener una hora en específico, en este caso se utilizó para obtener la hora de llegada y la hora de salida de un cliente, haciendo que fuera mucho más fácil obtener el tiempo de espera de cada cliente.

5) Utilizar una clase Controlador según el MVC es bastante eficiente ya que la aplicación se puede desarrollar rápidamente, sea modificable y mantenible, además que separar las funciones de la aplicación en modelo, vista y controlador hace que la aplicación sea más ligera.
6) Cuando se realiza un menú para la vista del proyecto se suele utilizar un menú principal y algunos submenús, si se utiliza un switch en el principal, para que el sistema sea más eficiente en los submenús utilizar funciones “if” para los mismos, así no se harán más ciclos que deterioren o hagan más lento el programa.
7) La abstracción es bastante útil para realizar un proyecto, porque al utilizarla se seleccionan los datos más importantes y mostrar solo los detalles relevantes de los objetos, además ayuda a reducir la complejidad del proyecto.
8) Cuando se trabaja en el leguaje de C++ se debe de recordar el constructor, la inicialización y el constructor de copia por lo que si se realiza un constructor se debe de poner también el constructor de copia porque, los constructores de copia generados por el compilador que son automáticos como los constructores de copia definidos por el usuario, tienen un único argumento de tipo "referencia a class-name".
9) Para la creación del menú es preferible y más eficiente que sean opciones relacionadas con un número, es decir que cada acción disponible se pueda acceder por medio de un número ya que para el usuario y para las pruebas es mucho más fácil poner solo un número que escribir toda la frase de la acción, además que se pueden prevenir varios errores en la codificación.
10) Cuando se desea almacenar datos y acceder a ellos fácilmente es bastante útil usar una lista de Arreglos ya que esta estructura es bastante eficiente para buscar elementos según un índice indicado o recorrido por un ciclo, haciendo que para buscar elementos su lógica sea bastante útil.

Recomendaciones

1) Cuando se va a desarrollar un proyecto de software en donde se tengan funcionalidades que implemente filas o colas se recomienda utilizar lista enlazadas ya que esta estructura es bastante útil para esto y además su lógica de desarrollo es bastante eficiente.
2) Utilizar las clases ArrayList, LinkePriorityQueue, LinkedQueue y ArrayQueue es de mucha ayuda para diferentes tareas dentro del proyecto, en el caso del ArrayList, es muy útil a la hora de agregar áreas y servicios para poder manejarlos como arreglos, en el caso de las estructuras de cola son muy eficientes para cuando se debe atender a una persona que tenga una condición de preferencial (agregar prioridades)

3) Se recomienda implementar una clase controlador ya que esta clase se encarga de crear independencia en el funcionamiento del proyecto, facilitar el mantenimiento en caso de errores y ofrecer maneras más sencillas para probar el correcto funcionamiento del sistema.
4) Antes de iniciar a programar el proyecto, es bueno primero realizar un diagrama de clases que pueda ilustrar lo que se quiera implementar, cuáles son las clases, atributos y métodos que se requiere ayuda mucho a ordenar las ideas y entender mejor las funcionalidades necesarias, ya que irse directamente a programar puede dificultar la elaboración del proyecto.
5) Tener las estructuras de datos vistas en clase bien implementadas ahorra mucho tiempo a la hora de realizar el proyecto, ya que los métodos facilitados por dichas estructuras son de gran ayuda para las operaciones que se necesitan implementar.
6) Se recomienda no usar tantos ciclos en el main, si ya se ha implementado un switch en el menú principal para los otros submenús utilizar if ya que como son submenús no van a ser tan repetitivos como se requiere que sea el menú principal.
7) Se recomienda utilizar la abstracción en aquellas clases que sea posible ya que esto ayudara a que se obtengan solo los datos que sean completamente necesarios y deshacerse de los que no se requieren para el proyecto.
8) Se recomienda que cuando se crea una clase nueva, apenas se crea realizar el constructor la inicialización y el constructor de copia, ya que si se llega a olvidar a la hora de hacer pruebas se puedan dar errores que luego sea difícil de identificar y así se evita que el constructor de copias modifique accidentalmente el objeto copiado.
9) Es recomendable ir haciendo las pruebas necesarias en el proceso de la creación del código así se va depurando el programa, ya que muchas veces se crea código y no se hacen todas las pruebas ya que cuando hay muchas clases creadas y además están relacionadas entre sí y se dan errores es más difícil arreglarlos o identificar cual es la clase/método que da problemas.
10) Para el menú se puede utilizar la opción que ofrece C++ que posee una estructura condicional múltiple que se llama: switch se recomienda para el menú principal porque ejecuta un bloque de sentencias si una variable o expresión coincide con alguno de los valores proporcionados, además se le puede crear un ciclo para que se repita las veces que sea necesario.

Referencias

Cruz, M. (2012). *Cola con prioridad en C++*. Martin Cruz Blog Sobre Programación. https://blog.martincruz.me/2012/11/cola-con-prrioridad-en-c.html

Oliet, N. M., Mallén, Y. O., y López, J. A. V. (2004). *Estructuras de datos y métodos algorítmicos: ejercicios resueltos*. Pearson educación. https://books.google.co.cr/books?hl=es&lr=&id=pwVv8SGVZtcC&oi=fnd&pg=PA3&dq=estructuras+de+datos&ots=pg9S_hEoQ1&sig=dLTGyDG992lJ8oqj5cZIeg88g&redir_esc=y#v=onepage&q=estructuras%20de%20datos&f=false