Simulación de cola de pedidos de un restaurante

Proyecto 1 del curso de Programación orientada a Objetos

1st Danny Jimenez Sevilla
Estudiante de ingeniería en computación
Carné: 2020018418
Cartago, Costa Rica
dannyjimenez118@gmail.com

2nd Larry Rivera Trejos
Estudiante de ingeniería en computación
Carné: 2019203344
Cartago, Costa Rica
larry.rivera.trejos@hotmail.com

I. Introducción

En este proyecto se tratará de implementar un programa que sea capaz de simular un restaurante haciendo posible el cargar un menú desde un archivo JSON (insertar referencia), poder crear una cola de clientes que automáticamente creen pedidos aleatorios según el menú proporcionado anteriormente, que se cree una lista de pedidos pendientes así como también una lista de productos en producción con respectivos contadores de tiempo, y también el manejar una lista de estadísticas generales al finalizar la ejecución de la simulación.

Por otra parte también se desea que se vea una eficacia en la manipulación de archivos JSON, el uso de herencia y polimorfismo, así como la aplicación de los conceptos de ingeniería de software.

Antes de leer este documento se deben de conocer los conceptos básicos de ciertas palabras que son partes del lenguaje JAVA, las cuales se describen a continuación:

JSON: acrónimo de JavaScript Object Notation (notación de objeto de JavaScript), es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera (año 2019) un formato independiente del lenguaje.[5]

Parseo (Parsing): Proceso de analizar una secuencia de símbolos a fin de determinar su estructura gramatical definida. También llamado análisis de sintaxis. Un parseador (parser) es un programa de computación que lleva a cabo esta tarea. El parseo transforma una entrada de texto en una estructura de datos (usualmente un árbol) que es apropiada para ser procesada.[1]

Thread: En sistemas operativos, un hilo o hebra (del inglés thread), proceso ligero o subproceso es una secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo.[7] Esto también puede ser aplicado a programas, como es el caso.

Herencia: En programación orientada a objetos, la herencia es, después de la agregación o composición, el mecanismo más utilizado para alcanzar algunos de los objetivos más preciados en el desarrollo de software como lo son la reutilización y la extensibilidad. A través de ella, los diseñadores pueden crear nuevas clases partiendo de una clase o de una jerarquía de clases preexistente (ya comprobadas y verificadas) evitando con ello el rediseño, la modificación y verificación de la parte ya implementada.[2]

Polimorfismo:En programación orientada a objetos, el polimorfismo se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos.[6]

GUI: La interfaz gráfica de usuario, conocida también como GUI (del inglés graphical user interface), es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.[3]

JFrame: Frame es una clase utilizada en Swing (biblioteca gráfica) para generar ventanas sobre las cuales añadir distintos objetos con los que podrá interactuar o no el usuario. A diferencia de JPanel, JFrame posee algunas nociones típicas de una ventana como minimizar, cerrar, maximizar y poder moverla.[4].

II. Diseño

A. Descripción de la solución

Para este problema necesitamos ayudarnos de las clases JFrame, JSON Simple (librería), de la clase Thread las cuales son indispensables para el funcionamiento del programa, este

programa se ejecuta usando varios Threads, y una GUI en la cual el usuario mediante botones podrá cargar el menú, parseando el JSOn mediante la librería JSON simple podremos obtener los datos, usando los condicionales podremos saber cuando es un objeto y cuando no. El usuario tendrá que llevar paso a paso la simulación desde la primera orden del primer cliente hasta que se generen las estadísticas, se hizo uso de las herramientas de diseño de netbeans para poder diseñar la GUI para que se viera de una manera más sobria y elegante [**Adobe**], se trato también de hacer una interfaz amigable con el usuario, en cuanto a las colas se hizo uso de los métodos que JAVA tiene para el uso de colas, y también de los métodos de las librerías MATH, FileReader y otrás más.

Para poder dar siguiente en las simulación se tiene que haber creado la cola de clientes, sino se tirará un error, el cual le dirá al usuario que debe de hacer, el botón de salir del programa esta hecho mediante código para que se vea más amigable con el usuario y así poder también desactivar la propiedad ”undecorated”.

En este proyecto se usarán herramientas como: Overaleaf para la edición del documento, GitHub para guardar el código, Git para hacer copias de seguridad en la memoria interna de cada una de las computadores, NetBeans como editor y compilador para el proyecto en JAVA, y LucidChart para hacer diagramas de clases.

B. Implementación de clases y Diagrama de clases

En esta subseccion explicaremos con una breve descripción como se implemento cada una de las clases que se hicieron para el funcionamiento de este programa.

PrincipalClass: Esta clase aloja el método main, donde se ejecuta el thread principal del programa, contiene una instancia llamada principalWindow, la cual pertenece a la clase Inicio que es la ventana principal y única que el usuario vera al ejecutar el programa

Producto: Esta clase se define para ser padre de varias clases como lo son la clase Acompannamiento, la clase Bebida, y la clase plato fuerte, debido a que estos tienen varios atributos en común y métodos que se estarían escribiendo innecesariamente si fuera por separado. Implementa los métodos get para cada uno de los atributos y tiene atributos como nombre, tiempo de producción y precio.

Acompannamiento: Esta clase hereda de la clase padre Producto, tiene los mismos atributos de la clase padre y además agrega el atributo de tipo, el cual sirve para saber si es un acompañamiento dulce o salado, en esta clase tenemos los métodos get para cada uno de los atributos extras, y hereda los mismos métodos de la clase padre.

Bebida: Esta clase hereda de la clase padre Producto, tiene los mismos atributos de la clase padre y además agrega el atributo de tipo, el cual sirve para saber si es una bebida es fría o caliente, en esta clase tenemos los métodos get para cada uno de los atributos extras, y hereda los mismos métodos de la clase padre.

PlatoFuerte: Esta clase hereda de la clase padre Producto, está misma es un objeto Bebida que tiene los mismos atributos de la clase padre y además agrega el atributo de tamaño, el cual sirve para saber si es un plato fuerte de tamaño medio o entero, en esta clase tenemos los métodos get para cada uno de los atributos extras, y hereda los mismos métodos de la clase padre.

Cliente: Esta clase se hace para el manejo de los clientes como objetos, esta tiene atributos como: numero de cliente, contador y paciencia, y los métodos get para cada uno, además de esto tiene un método llamado ”ordenar” que lo que hace es crear de forma aleatoria la lista de productos que el cliente desea comprar y implementa la interfaz Runnable la cual nos funciona para crear hilos con los cuales llevamos por separado el contador de espera

Cliente especial: Esta clase hereda de la clase Cliente, y es una variación donde se hace uso de paciencia ya que por motivos técnicos la solución más fácil para poder usar la paciencia fue sobreescríbanlo en esta nueva clase, para hacer así uso de esta con el método ”DecrementarPaciencia” el cual se encarga de decrementar la paciencia, valga la redundancia por su nombre, e igual hereda los métodos de la clase Cliente y sobre escribe el método ”getPaciencia”.

Ordenes: Esta clase es para el manejo de las ordenes que hace cada cliente, se encarga de calcular los precios totales de las ordenes, calcular si hay algún descuento por haberse creado un combo, también se encarga de administrar los pedidos de cada cliente haciéndolos como un objeto aparte para poder ir revisando si el pedido está listo o aún no. Esta clase hereda de la clase Thread la cual nos funciona para crear hilos de funcionamiento en el cual llevaremos contadores de las ordenes

Producción: Esta clase no hereda de ninguna otra, se encarga de hacer los pedidos (ordenes) y crear las instancias de las ordenes, manejar la cola de producción, y de otras cosas referentes a la producción, tiene atributos como: maximoProductos, el cual se refiere a la cantidad máxima de espacio que hay para los productos; el espacioDisponible que a como su nombre lo dice se encarga de dar el espacio que dispone la cola para agregar más productos, un array de productos el cual funciona para almacenar la cantidad y que productos lleva cada pedido. En esta clase también se hace uso del parseo para el JSON que contiene el menú. Esta clase hereda de la clase Thread la cual nos

ayuda a tener un metodo run en el cual podremos llevar un hilo para llevar los contadores de cada producto en producción

Inicio: está es una clase que hereda de la clase ”JFrame” por lo cual la llamamos nuestra GUI, en ella tenemos el código que nos ayuda a colocar todos los iconos en la ventana que verá el usuario, usando campos de texto, botones, labels, y demás objetos que nos facilita está clase, tiene un método que ayuda a poder cambiar la posición de la ventana teniendo el atributo ”undecoreted” desactivado, otro método al presionar el botón siguiente en el cual se va avanzando en la simulación, otro método para el botón crear cola, otro método para el botón cargar menú en el cual se parsea el JSON que utilizamos para albergar el menú.

Estadísticas: Esta clase se encarga de las estadísticas sobre cuantas personas se fueron sin pagar, cuantos clientes están satisfechos, ganancias del ”día” y demás datos importantes

Para poder ver más a fondo los métodos y atributos de cada una de las clases acceda al siguiente link a ver el diagrama de clases el cual no se pudo incluir por el tamaño de la imagen: https://estudianteccr-my.sharepoint.com/:f:/g/personal/danny_jimenez_estudiantec_cr/EvbpLvlXjaxLoaIadoLlR-IBmz1WpzvRkoM-Ra5sPLIIig?e=a0NbHs

C. División de actividades

En esta sección se explicará el como se dividieron las tareas entre cada uno de los integrantes del grupo.

Actividad	Integrantes
Documentación	DJ/LR
Menú Botón Cargar	DJ
Clases y Actividades	DJ/LR
Clases de Productos	LR
Interfaz Gráfica	DJ
Botón Iniciar / Seguir	DJ
Clase Manejo de Productos	LR
Clase Manejo de Productos	LR

Nota: se usaran las siglas DJ (Danny Jiménez) y LR(Larry Rivera)

D. Cronograma

En esta sección se describirá las fechas en las que se trabajaron las actividades por separado, siendo estas en orden cronológico:

Actividad	Fecha
Lectura y entendimiento del enunciado	21/03
Reunión para inicializar las herramientas a usar	23/03
Reunión para definir las actividades a realizar	26/03
División de las actividades a realizar	05/04l
Investigación sobre JSON	09/04
Creación de las Clases Productos	13/04
Creación de la Interfaz Gráfica	13/04
Creación de la Clase Producción	18/04
Creación de la Clase de Estadísticas	18/04
Investigación sobre Threads	23/04
Creación de clases con Threads	23/04
Modificaciones a la Interfaz Gráfica	25/04
Creación de clases de Ordenes	25/04
Conclusión del Diagrama de clases	27/04
Conclusión de la Documentación	27/04

III. Conclusión

Este proyecto resulto satisfactorio ya que se pudo implementar de una buena manera aunque no la totalmente eficiente ya qué se nos complico el uso de JSON simple por ser algo nuevo, pudimos implementar bien el uso de los Thread y funciones en los JFrames, creemos que para mejorar esta interfaz se pudo hacer uso de matrices para manejar los pedidos, se pudo hacer uso de otro tipo de diseño de interfaz y quizá usar más threads y poder correr la simulación sin hacerla paso a paso.

Para descargar el proyecto el cual está en un repositorio de GitHub acceda al siguiente link: https://github.com/dan-jimenez/Herencia-Polimorfismo.git

References

[1] Wikipedia. *¿Qué es parseo (parsing)?* URL: https://www.alarconnelson.com/2017/11/que-es-parseo-parsing.html#:~:text=Proceso%20de%20analizar%20una%20secuencia,lleva%20a%20cabo%20esta%20tarea..

[2] Wikipedia. *Herencia*. URL: https://es.m.wikipedia.org/wiki/Herencia_(inform%C3%A1tica).

[3] Wikipedia. *Interfaz gráfica de usuario*. URL: https://es.wikipedia.org/wiki/Interfaz_gr%C3%A1fica_de_usuario#:~:text=La%20interfaz%20gr%C3%A1fica%20de%20usuario,acciones%20disponibles%20en%20la%20interfaz..

[4] Wikipedia. *JFrame*. URL: https://es.wikipedia.org/wiki/Jframe.

[5] Wikipedia. *JSON*. URL: https://es.wikipedia.org/wiki/JSON.

[6] Wikipedia. *Polimorfismo*. URL: https://es.wikipedia.org/wiki/Polimorfismo_(inform%C3%A1tica).

[7] Wikipedia. *Thread*. URL: https://es.wikipedia.org/wiki/Hilo_(inform%C3%A1tica)#:~:text=En%20sistemas%20operativos%2C%20un%20hilo,ejecutada%20por%20un%20sistema%20operativo..