Simulación de un sistema de aviación

Proyecto 2 del curso de Programación orientada a Objetos

1st Danny Jimenez Sevilla
Estudiante de ingeniería en computación
Carné: 2020018418
Cartago, Costa Rica
danny.jimenez@estudiantec.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 programa de aviación que pueda manejar todas las funciones que necesita, cómo:

Por otra parte también se desea que se vea una eficacia en la manipulación del patron de diseño MVS, el uso de Sockets e hilo y tambien 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:

MVS: Modelo, vista y controlador. Estos tres componentes son los que definen no solo las entidades si no la forma en que se comunican unas con otras. MVC es un patrón de diseño basado en orientación a objetos. Es toda la funcionalidad de la misma, desarrollada en un determinado lenguaje de programación. No define cómo se muestra la información, solo lo que se hace con dicha información y dónde está almacenada[**MVS**]

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.[4] Esto también puede ser aplicado a programas, como es el caso.

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.[2]

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.[3].

II. Diseño

A. Descripción de la solución

Para este problema necesitamos ayudarnos de la clase JFrame, 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 utilizar la simulación. El usuario tendrá que llevar paso a paso la simulación controlando la llegada, aterrizaje y la salida de los aviones, tomando en cuenta su tamaño su capacidad, en otras palabras es el control aéreo de un aeropuerto.
El usuario a su vez podrá ver información entrante sobre los vuelos como el estado en qué están, si vienen con retraso o no, esto se logró mediante el uso de dos hilos principales que corren al mismo tiempo y generan dos GUI para que el usuario en una ventana pueda ver la información y en la otra pueda asignar las parámetros para que el vuelo aterrice de manera eficiente, también se manejo dos hilos más que son los de Flights que se encarga de generar los vuelos aleatoriamente apoyándose en los métodos Random de la clase Math y enviándolos cada cierto tiempo, el otro hilo es el del controlador que a su vez trabaja como servidor donde se envían todos los datos necesarios para que este inicie los contadores, asigne las pistas de aterrizaje y las puertas de

embarque mediante los mensajes recibidos por la clase de Controller Window la cual también funciona como cliente. Las demás clases principales como lo son: la clase Flight, la clase Information Window trabajan como clientes que se conectan al servidor Controller el cual les envía información y a la vez recibe de ellos. Cabe destacar que las cuatro clases principales contienen un metodo Main para poder correrlos como aplicaciones separadas y que la comunicación entre sockets sea más evidente

Donde se utilizan 4 aplicaciones principales:

Vuelos: Simula la llegada de los aviones al aeropuerto, esto lo hace mediante el uso de métodos Random de la clase Math como se menciono anteriormente, y creando objetos de una clase Airplane la cual se encarga de generar el avión para que sea tratado como un objeto. Esta clase trabaja con un hilo como cliente que se conecta a un servidor principal e intercambia mensajes con el.

Ventana Controlador: Esta es la ventana principal para el usuario. Conforme llegan los vuelos, el usuario debe asignarle a cada avión una pista de aterrizaje adecuada que esté libre.Consta de varios JPanel que están agrupados gracias al Tabbed Panel el cual se encarga de seccionarlos, en cada sección el usuario tendrá la información necesaria para asignar lo requerido, esto se debe asignar en la ultima pestaña. Esta clase trabaja con un hilo como cliente que se conecta a un servidor principal e intercambia mensajes con el.

Controlador: Es la aplicación que simula el movimiento de los aviones dentro del aeropuerto. Esta clase mediante ciclos lleva contadores de los tiempos que tarda un avión en aterrizar y descargar después de haber sido asignado, también es un Server que le manda información a las otras 3 clases principales de las cuales el también recibe información, es por esto que se ejecuta como un hilo aparte y además corre otro hilo de una clase extra que se encarga de llevar contadores de tiempo real e ir los restando para así generar el tiempo restante para que una pista o una puerta quede libre, se encarga también de crear las puertas de embarque y las pistas de aterrizaje y tratarlas como objetos separadas que tienen un estado mediante el cual se puede saber si están disponibles o no, o si tienen la capacidad necesaria

Ventana de información: Es la ventana que usualmente se ve en la sala de espera. Muestra los vuelos esperados y el estado actual, este estado esta dado por el Controller y cada uno de los objetos los posee como atributos, esta clase se comporta en el FrontEnd como un cliente que no envía información solo la recibe ya que la única función para la que se le necesita es poder mostrar la información de manera detallada y concisa.

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 sub-sección explicaremos con una breve descripción como se implemento cada una de las clases que se hicieron para el funcionamiento de este programa.

Gate: Esta clase es la encargada de crear las compuertas de embarque, otorga un nombre y también la capacidad de personas que pueden ingresar al avión, mantiene atributos de disponibilidad los cuales son necesarios para saber si se puede descargar ahí o no.

LandinStrip: Esta clase es la encargada de crear las compuertas, otorga un nombre y también la capacidad de personas que pueden ingresar al avión. Se comporta como un objeto en el cual se puede asignar un avión y tiene un atributo que muestra la disponibilidad de la misma.

ControllerPanel: Esta es una clase que hereda de la clase JPanel en la cual se edita de manera gráfica la interfaz de la ventana de controlador, esta a su vez se maneja como un objeto que será usado en la clase ControllerWindow para poder agregarlo al JFrame que se visualizará al final.

Airplane: Esta clase se encarga de crear un avión, pide y otorga un nombre, tamaño y tipo del avión que se va crear. Este objeto se utiliza en el generador de vuelos de la clase Flights, en la cual se genera este objeto y es tratado de manera independiente el cual se podrá asignar a una pista de aterrizaje por un tiempo estipulado o a una puerta de embarque, en la cuál simulará la descarga o el aterrizaje.

InformationPanel: Al igual que la clase ControllerPanel este hereda de la clase JPanel para ser manipulado de manera grafica y será usado en la clase InformationWindow para poder visualizar la información. En esta clase se implementan acciones dentro de eventos como por ejemplo al MousePressed o el MouseDragged para poder mover la ventana sin tener que usar el parametro ”undecored” en false, esto para que ayude a la interfaz visual a qué se vea más elegante.

Además de estas existen clases complementarias dentro del proyecto las cuales no son tan relevantes y tienen una documentación interna que las respalda en caso de surgir dudas de su utilidad

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://lucid.app/lucidchart/dd585677-5fe4-4526-a0f0-fc3c18401757/edit?page=0_0#

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
Creacion de herramientas de trabajo	DJ
Documentación	DJ/LR
Ventana Controlador	LR
Clases y Actividades	DJ/LR
Clases del Controlador	DJ
Ventana Informacion	LR
Clases de Vuelos	DJ
Clases de Creacion de Aviones	DJ
Interfaz Gráfica	DJ/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	15/05
Reunión para inicializar las herramientas a usar	15/05
Reunión para definir las actividades a realizar	22/05
División de las actividades a realizar	22/05
Creación de la clase Airplane	29/05
Creación de la Clase Gate	29/05
Creación de la Interfaz Gráfica	29/05
Investigación sobre Sockets	04/06
Investigación sobre Threads	04/06
Creación de clases con Threads	05/06
Modificaciones a la Interfaz Gráfica	05/06
Creación de Diagrama de clases	19/06
Conclusión del Diagrama de clases	21/06
Conclusión de la Documentación	21/06

III. Conclusión

Este proyecto resulto satisfactorio ya que se pudo implementar bien el uso de los Thread y funciones en los JFrames, se nos complico al principio el uso de los hilos ya que no estábamos familiarizados con el tema, pero se investigo y se hicieron consultas y se pudo implementar con éxito junto con las ventanas y todo unido, creemos que pudimos mejorar el proyecto mediante el uso de clases de JAVA como la interfaz Observer y la clase Observer u otro tipo de clases que hacen mucho más fácil la comunicación mediante sockets, también se pudo implementar las clases de una manera más eficiente si se usaba de mejor manera la distribución de métodos y clases heredadas, así cómo el uso de más hilos para el manejo de datos de una manera sincronizada y así aumentando la eficiencia y reduciendo el tiempo que tarda cada proceso.
Otra alternativa para mejorar seria el uso de fondos con imágenes dentro de los frame lo cual le daría una mejor apariencia, así como también una organización más a detalle de las ventanas, con implementación de fundamentos del diseño gráfico como bases y el uso de una paleta de colores más acorde a lo que se quería lograr
Para descargar el proyecto el cual está en un repositorio de GitHub acceda al siguiente link: https://github.com/dan-jimenez/designs-patterns.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. *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..

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

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