**Manual de usuario**

**Harold Guerrero Caicedo**

**Juan Esteban Salas**

**Proyecto #1**

**Programación Orientada a Objetos**

**Índice**

***Clase Aeronave***

Esta es la clase base que representa una aeronave genérica.

Atributos:

* registro\_de\_marca (número de registro de la aeronave), capacidad\_de\_pasajeros (número de pasajeros que puede transportar), velocidad\_max (velocidad máxima de la aeronave), autonomía (autonomía de vuelo en millas), ano\_fabricación (año de fabricación de la aeronave), estado (estado actual de la aeronave, por ejemplo, “en el vuelo” o “en tierra”), coordenadas (un par de coordenadas de latitud y longitud) y coor\_aeronaves (un vector que almacena las coordenadas de todas las aeronaves), también se crea un identificador “id” para la torre de control.

***Funciones:***

* “guardar\_coordenadas” genera coordenadas aleatorias y las asigna a la aeronave.
* “obtenerCoordenadas” devuelve las coordenadas de la aeronave.
* “consultarCoordenadasAero” permite actualizar las coordenadas de una aeronave específica en el vector “coor\_aeronaves”.
* La función “reiniciar” actualiza las coordenadas de todas las aeronaves en función de un vector de coordenadas proporcionado.

***Clase Avión***

Representa un tipo de aeronave que es un avión.

* Tiene atributos adicionales como altitud\_maxima(la latitud máxima a la que puede volar),cantidad\_de\_motores(número de motores del avión) y categoría (categoría de avión).

***Clase Helicóptero (hereda de la clase Aeronave)***

Representa un tipo de aeronave que es un helicóptero.

* Tiene atributos adicionales como cantidad\_de\_rotores(número de rotores del helicóptero), capacidad\_de\_elevación (capacidad de carga que puede elevar) y uso específico (propósito o uso específico del helicóptero).
* El constructor de esta clase llama al constructor de la clase base ”Aeronave” y establece atributos específicos del helicóptero)

***Clase Jetprivado***

Representa un tipo de aeronave que es un jet privado.

* Tiene atributos adicionales como propietario (nombre del propietario del jet privado), lista\_de\_servicios\_a\_bordo (lista de servicios disponibles a bordo) y lista\_de\_destinos\_frecuentes (lista de destinos frecuentes para esta aeronave).
* El constructor de esta clase llama al constructor de la clase base “aeronave” y establece los atributos específicos del jet privado.

***Clase PuertaDeEmbarque***

Este código representa una puerta de embarque en un aeropuerto. La clase tiene los siguientes atributos y funciones:

* Vuelo\* vuelo\_asignado: Un puntero a un objeto de la clase Vuelo, que representa el vuelo asignado a esta puerta de embarque.
* bool disponible: Un booleano que indica si la puerta de embarque está disponible o no.
* Int identificacion: Un identificador único para la puerta de embarque.
* String ubicacion: Una cadena que describe la ubicación de la puerta de embarque.
* String hora\_embarque: Una cadena que representa la hora de embarque para el vuelo asignado.
* vector<Vuelo>historia: Un vector que almacena el historial de vuelos que han utilizado esta puerta de embarque en el pasado.
* PuertaDeEmbarque(int id, string ubi): constructor que recibe una identificación (id) y una ubicación (ubi) como parámetros y establece la disponibilidad inicial en verdadero.

Esta clase de PuertaDeEmbarque se utiliza para gestionar y registrar información relacionada con las puertas de embarque de un aeropuerto, incluyendo detalles sobre el vuelo asignado, su disponibilidad y el historial de vuelo que han utilizado esa puerta en el pasado.

***Clase TorreDeControl***

Esta clase representa una torre de control de un aeropuerto.

* vector<pair<double, double>> corrdenadas\_aeronaves: Un vector que almacena las coordenadas de todas las aeronaves en el espacio aéreo controlado por la torre de control,
* vector<PuertaDeEmbarque> puertas\_embarques: Un vector que almacena objetos de la clase PuertaDeEmbarque, que representan las puertas de embarque del aeropuerto.
* vector<Vuelo\*> vuelos: Un vector que almacena punteros a objetos de la clase Vuelo, que representan los vuelos gestionados por la torre de control.
* vector<Aeronave\*> aeronaves: Un vector que almacena punteros a objetos de la clase Aeronave, que representan las aeronaves controladas por la torre de control.
* TorreDeControl(int numeroDePuertasEmbarque): Constructor que inicializa las puertas de embarque en función del número proporcionado. Divide las puertas de “Vuelos Nacionales” y “Vuelos Internacionales” en una proporción de 1/3 a 2/3, respectivamente.
* void actualizar\_ubi\_aero(int id): Actualiza las coordenadas de una aeronave epecífica en el vector coordenadas\_aeronaves en función de sus coordenadas actuales. También actiualiza las coordenadas de todas las aeronaves en relación con la aeronave modificada.
* pair<double, double> obtener\_ubi(int id): obtiene las coordenadas de una aeronave especifica en función de sus identificador.
* void agregarAeronave(Aeronave\* aero): Agrega una aeronave al vector de aeronaves. Asigna un identificador único a la aeronave, guarda coordenadas aleatorias y actualiza todas las coordenadas de las aeronaves.
* void actualizarTodasLasAeronaves(): Actualiza las coordenadas de todas las aeronaves en función de sus ubicaciones actuales.
* void eliminarAeronave(int id): Elimina una aeronave del vector de aeronaves y actualiza todas las coordenadas de las aeronaves.
* Int asignar\_puertas(Vuelo\* flight): Asigna una puerta de embarque a un vuelo si hay disponibilidad. Genera una hora de embarque aleatoria y registra el vuelo en el historial de la puerta de embarque correspondiente.
* void desocupar\_puerta(int cualPuerta): Marca una puerta de embarque como disponible después de que un vuelo ha despegado.

La idea de esta clase es gestionar las operaciones de una torre de control en un aeropuerto, incluyendo las puertas de embarque, el seguimiento de las aeronaves y los vuelos, registro de coordenadas y el historial de las puertas de embarque que se utilizan.