|  |
| --- |
| **Vertice** |
| - siguiente: Vertice\*  - arista: Arista\*  - nombre: string |
| + asignarProxVertice(Vertice\*): void  + agregarArista(Arista\*):void  + aliminarArista(Arista\*):void  + obtenerProxVertice(): Vertice\*  + obtenerAristas(): Arista\*  + obtenerNombreVertice(): string |

|  |
| --- |
| **Arista** |
| - sig: Arista\*  - destino: Vertice\*  - precio: int  - horasVuelo: double |
| + asignarSiguiente (Arista\*): void  + asignarPrecio(int): void  + asignarHoras(double): void  + consultarSiguiente(): Arista\*  + ConsultarDestino(): Vertice\*  + cunsultarPrecio(): int  + consultarHoras(): double |

|  |
| --- |
| **Grafo** |
| - primero: Vertice\*  - ultimo: Vertice\*  - tamanio: int |
| + agregarVertice (string): void  + existeVertice (string): bool  + cantVertices (): int  + obtenerVertice (string): Vertice\*  + agregarArista (Vertice\*, Vertice\*, int, double): void  + obtenerPeso1 (Vertice\*, Vertice\*): int  + obtenerPeso2 (Vertice\* inicio, Vertice\* destino): double  + obtenerTamanio (): unsigned  + caminoMinimo (Vertice\*, Vertice\*, int ): void  + mostrarVer3 (list<Etiqueta>, Vertice\*, Vertice\*, stack<TuplaCompleta>, bool, int ): void  + mostrarPila (stack<TuplaCompleta>, int): void  + mostrarVertices (): void  + marcadoComoVisitado (list<Vertice\*>, Vertice\*): bool  + verificarPesoVerticeMarcado (Vertice\*, Vertice\*, list<Etiqueta>&, int, int, list<Vertice\*>&, ColaPrioridad &): void  + mostrarVerticesMarcados (list<Vertice\*>): void  + existeCamino (list<Etiqueta>, Vertice\*): bool  + mostarEtiquetas (list<Etiqueta>): void    + obtenerPrimero(): Vertice\* |

|  |
| --- |
| **Etiqueta** |
| - vertice: Vertice\*  - anterior: list<Vertice\*>  - pesoAcumulado: int  - pesoDouble: double  - iteracion: int |
| Etiqueta (Vertice\*)  ~Etiqueta ()  + setAnterior (Vertice\*): void //no va mas  + sumoAnterior (Vertice\*): void  + setPesoAcumulado (int): void  + setPesoDouble (double ): void  + setIteracion (int iteracion): void  + getVertice(): Vertice\*  + getAnterior(): list<Vertice\*>  + getPesoAcumulado(): int  + getPesoDouble(): double  + getIteracion(): int |

|  |
| --- |
| **Menu** |
| - opcion: char  - opcionVuelos: char  - partida: string  - destino: string  - aeropuertos: BST<Aeropuerto\*>\*  - vuelos: Grafo\* |
| Menu ()  ~Menu()  + getOpcion(): char  + getOpcionVuelos(): char  + setAeropuertos (BST<Aeropuerto\*>\*): void  + setVuelos (Grafo\*): void  + hayVuelosCargados (): bool  + limpiarPantalla (): void  + mostrarMenuPrincipal (): void  + hacerEleccion (): void  + consultarAeropuerto (): void  + agregarAeropuerto (): void  + buscarMenor (BSTNode<Aeropuerto\*>\*):BSTNode<Aeropuerto\*>\*  + reemplazar (BSTNode<Aeropuerto\*>\*, BSTNode<Aeropuerto\*>\*): void  + destruir (BSTNode<Aeropuerto\*>\*): void  + eliminarAeropuerto(string): void  + darDeBajaAeropuerto(): void  + solicitarPartidaYDestino(): void  + mostrarMenuVuelos(): void  + hacerEleccionMenuVuelos(): void  + pausa(): void |

|  |
| --- |
| **VerificadorDatos** |
| VerificadorDatos();  ~VerificadorDatos ();  + esCadena (string): bool  + esIATA (string): bool  + esDouble (string): bool  + esEntero (string): bool  + esAeropuerto (string): bool  + esVuelo (string ): bool |

|  |
| --- |
| **ManejoDeArchivos** |
| - chequear: VerificadorDatos |
| ManejoDeArchivos()  ~manejoDeArchivos()  + existe (string): bool  + cargarAeropuertos (BST<Aeropuerto\*>\*, string ): void  + agregarAero (BST<Aeropuerto\*>\* , string): void  + cargarVuelos (Grafo\*, string): void  + agregarVuelo (Grafo\*, string): void  + quitarGuion(string texto): string |

|  |
| --- |
| **NodoColaPrioridad** |
| - Vertice: Vertice\*  - prioridad: int  - pesoEntero: int  - pesoDouble: double  - siguiente: NodoColaPrioridad\* |
| NodoColaPrioridad (Vertice\*, int, int, double)  + obtenerVertice (): Vertice\*  + getPrioridad (): int  + getEntero (): int  + getDouble (): double  + setSiguiente (NodoColaPrioridad\*): void  + getSiguiente (): NodoColaPrioridad\*  ~NodoColaPrioridad() |

|  |
| --- |
| **ColaPrioridad** |
| - primero: NodoColaPrioridad\*  - primeroSegunPrioridad: NodoColaPrioridad\*  - ultimo: NodoColaPrioridad\*  - criterioOrden: int |
| ColaPrioridad (int)  + vacia (): bool  + topAndPop(): Vertice\*  + topVertice(): Vertice\*  + topPrioridad(): int  + topEntero(): int  + topDouble(): double  + pop(): void  + push(Vertice\*, int, double, int): void    ~ColaPrioridad() |

|  |
| --- |
| **Aeropuerto** |
| - IATA: string  - nombre: string  - ciudad: string  - pais: string  - superficie: double  - terminales: unsigned  - destinosNacionales: unsigned  - destinosInternacionales: unsigned |
| Aeropuerto()  ~Aeropuerto()  + asignarCodigo(string): void  + asignarNombre(string): void  + asignarCiudad(string): void  + asignarPais(string): void  + asignarSup(double): void  + asignarTerminales(unsigned): void  + asignarDestNac(unsigned): void  + asignarDestInternac(unsigned): void  + obtenerCodigo(): string  + obtenerNombre(): string  + obtenerCiudad(): string  + obtenerPais(): string  + obtenerSup(): double  + obtenerTerminales(): unsigned  + obtenerDestNac(): unsigned  + obtenerDestInternac(): unsigned  + mostrarInformacion(): void |

|  |
| --- |
| **BSTNode** |
| - codIATA: string  - data: T // ¿Esta bien “T” o tendria que poner el valor ingresado por el template?  - left: BSTNode<T>\*  - right: BSTNode<T>\*  - parent: BSTNode<T>\* |
| BSTNode (T, string)  ~BSTNode()  + get\_data(): T  + get\_IATA(): string  + set\_IATA(string IATA): void  + set\_data(T data): void  + set\_right (BSTNode<T>\* , BSTNode<T>\*): void  + set\_left (BSTNode<T>\* , BSTNode<T>\*): void  + set\_left(BSTNode<T>\* left): void  + set\_right (BSTNode<T>\* right): void  + set\_parent(BSTNode<T>\* parent): void  + get\_right(): BSTNode<T>\*  + get\_left(): BSTNode<T>\*  + get\_parent(): BSTNode<T>\*  + isLeaf (): bool  + rightChildOnly (): bool  + leftChildOnly (): bool |

|  |
| --- |
| **BST** |
| - root: BSTNode<T>\* |
| - insert (BSTNode<T>\*, T, string): BSTNode<T>\*  - liberarNodo (BSTNode<T>\*): void  - obtenerAltura (BSTNode<T>\*): int  - mostrarNivel (BSTNode<T> \*, int): void  - print\_in\_order (BSTNode<T> \*): void  - imprime\_acostado (BSTNode<T> \* , int): void  - imprime\_en\_ancho (BSTNode<T> \*): void  - search (BSTNode<T>\* , T): BSTNode<T>\*  - buscarIATA (BSTNode<T>\*, string): BSTNode<T>\*  - buscarNombre (BSTNode<T>\* , string: BSTNode<T>\*  - find\_min (BSTNode<T>\*): T  - find\_max (BSTNode<T>\*): T  - successor (BSTNode<T>\* ): T  - predecessor (BSTNode<T>\*): T  - remove (BSTNode<T>\* , T ): BSTNode<T>\*  - delete\_all (BSTNode<T>\*): void  - quitar (BSTNode<T>\* , string ): BSTNode<T>\*  - aeropuertoReemplazante (BSTNode<T>\*, T): T  - buscarMenorPorIATA (BSTNode<T>\*, T): T  + BST()  + buscar (string&): BSTNode<T>\*  + quitar (string): BSTNode<T>\*  + aeropuertoReemplazante (string, T): T  + buscarMenorPorIATA(): T  + obtenerAltura (): int  + imprime\_acostado (): void  + imprime\_en\_ancho (): void  + insert (T, string): void  + print\_in\_order(): void  + search (T ): bool  + find\_min (): T  + find\_max (): T  + successor(T ): T  + predecessor(T ): T  + remove (T): void  + eliminar (string ): void  + get\_root(): BSTNode<T>\*  + empty(): bool  + delete\_all(): void  ~BST<T>()  }; |