*CLASES ABSTRACTAS*

|  |
| --- |
| *NODO\_LISTA* |
| ATRIBUTOS  - Elemento: Dato\*  - Psig: Nodo\_lista\* |
| MÉTODOS  //Constructor con un parametro  + Nodo\_lista(Dato\* d);  //Constructor con dos parámetros  + Nodo\_lista(Dato\* d, Nodo\_lista\* psig)  + Asignar\_elemento(Dato\* d)  + Asignar\_siguiente(Nodo\_lista\* psig);  + Obtener\_elemento()  + Obtener\_siguiente()  //Destructor  + ~Nodo\_lista(); |

|  |
| --- |
| *LISTA* |
| ATRIBUTOS  - primero: Nodo\_lista<Dato>\*  - tamanio: integrer |
| MÉTODOS  //Constructor  + Lista();  - cambiar\_nodo(Nodo\* nodo,int pos)  - retirar(int pos)  + obtener\_tamanio()  + agregar(Dato\* d, int pos)  + eliminar(int pos)  + es\_vacia()  + consultar(int pos)  + obtener\_nodo(int pos)  + swap(int pos1, int pos2)  + esta(Dato\* dato\_a\_buscar)  + ordenar()  //Destructor  + ~Lista(); |

|  |
| --- |
| *NODO\_ABB* |
| ATRIBUTOS  - hijo\_izq: Nodo\_abb\*  - hijo\_der: Nodo\_abb\*  - telefono: string\*  - Cliente: Cliente\* |
| MÉTODOS  //Constructor sin parametros  + Nodo\_abb()  //Constructor con parámetros  + Nodo\_abb(Nodo\_abb\* nuevo\_hijo\_izq,Nodo\_abb\* nuevo\_hijo\_der,string nuevo\_telefono,Cliente\* nuevo\_cliente)  + obtener\_hijo\_der()  + obtener\_hijo\_izq()  + obtener\_telefono()  + obtener\_cliente()  + asignar\_hijo\_der(Nodo\_abb\* nuevo\_hijo\_der)  + asignar\_hijo\_izq(Nodo\_abb\* nuevo\_hijo\_izq)  + asignar\_telefono(string nuevo\_telefono)  + asignar\_cliente(Cliente\* nuevo\_cliente)  //Destructor  + ~Nodo\_abb() |

|  |
| --- |
| *ABB* |
| ATRIBUTOS  - raiz: Nodo\_abb\* |
| MÉTODOS  //Constructor sin parametros  + Abb()  //Constructor con parámetros  + Abb(Nodo\_abb\* nueva\_raiz)  + obtener\_raiz()  + asignar\_raiz(Nodo\_abb\* nueva\_raiz)  + eliminar(int pos)  + es\_vacio()  + agregar(Nodo\_abb\* nodo\_a\_agregar)  + borrar(Nodo\_abb\*)  //Destructor  + ~Abb() |

|  |
| --- |
| *INDIVIDUO* |
| ATRIBUTOS  - nombre: string\* |
| MÉTODOS  //Constructor con parámetros  + Individuo(string\* nuevo\_nombre,bool nueva\_alta,string nuevo\_legajo, int nuevo\_precio\_base, char nuevo\_tipo)  + obtener\_individuo()  + asignar\_individuo(string\* nuevo\_nombre)  + obtener\_tamanio\_cliente()  + mostrar\_cliente()  //Destructor  + ~Individuo() |

|  |
| --- |
| *CLIENTE* |
| ATRIBUTOS  - alta: boolean  - legajo: string\*  - precio base: integrer  - tipo: char |
| MÉTODOS  //Constructor por defecto  + Cliente()  //Constructor con parámetros  + Cliente(bool nueva\_alta, string nuevo\_legajo, int nuevo\_precio\_base, char nuevo\_tipo)  + obtener\_alta()  + obtener\_legajo()  + obtener\_precio()  + obtener\_tipo()  + asignar\_alta(bool nueva\_alta)  + asignar\_legajo(string nuevo\_legajo)  + obtener\_precio\_base()  + obtener\_tamanio\_cliente()  + mostrar\_cliente()  //Destructor  + virtual ~Cliente() |

|  |
| --- |
| *FAMILIA* |
| ATRIBUTOS  - familiares: Lista<string>\* |
| MÉTODOS  //Constructor con parametros  + Familia(Lista<string>\* nueva\_familiares,bool nueva\_alta,string nuevo\_legajo, int nuevo\_precio\_base, char nuevo\_tipo)  + mostrar\_familia()  + asignar\_familia(Lista<string>\* nueva\_familiares)  + obtener\_tamanio\_cliente()  + mostrar\_cliente()  //Destructor  + ~Familia() |

ACLARACIONES:

- LAS RELACIONES ENTRE CLASES ESTÁN DEFINIDAS EN EL ARCHIVO “Diagrama de relación de clases” DONDE SE VE CON MAYOR PROLIJIDAD Y MAS ACOTADAMENTE CADA CLASE.

- EL ARCHIVO “Funciones\_main.h” ESTÁ PROGRAMADO ESTRUCTURADO, ASIQUE NO ESTÁ INCLUIDO EN EL UML PERO SÍ EN EL DIAGRAMA PARA REFLEJAR COMO ES EL MANEJO DEL PROGRAMA POR PARTE DE CADA CLASE