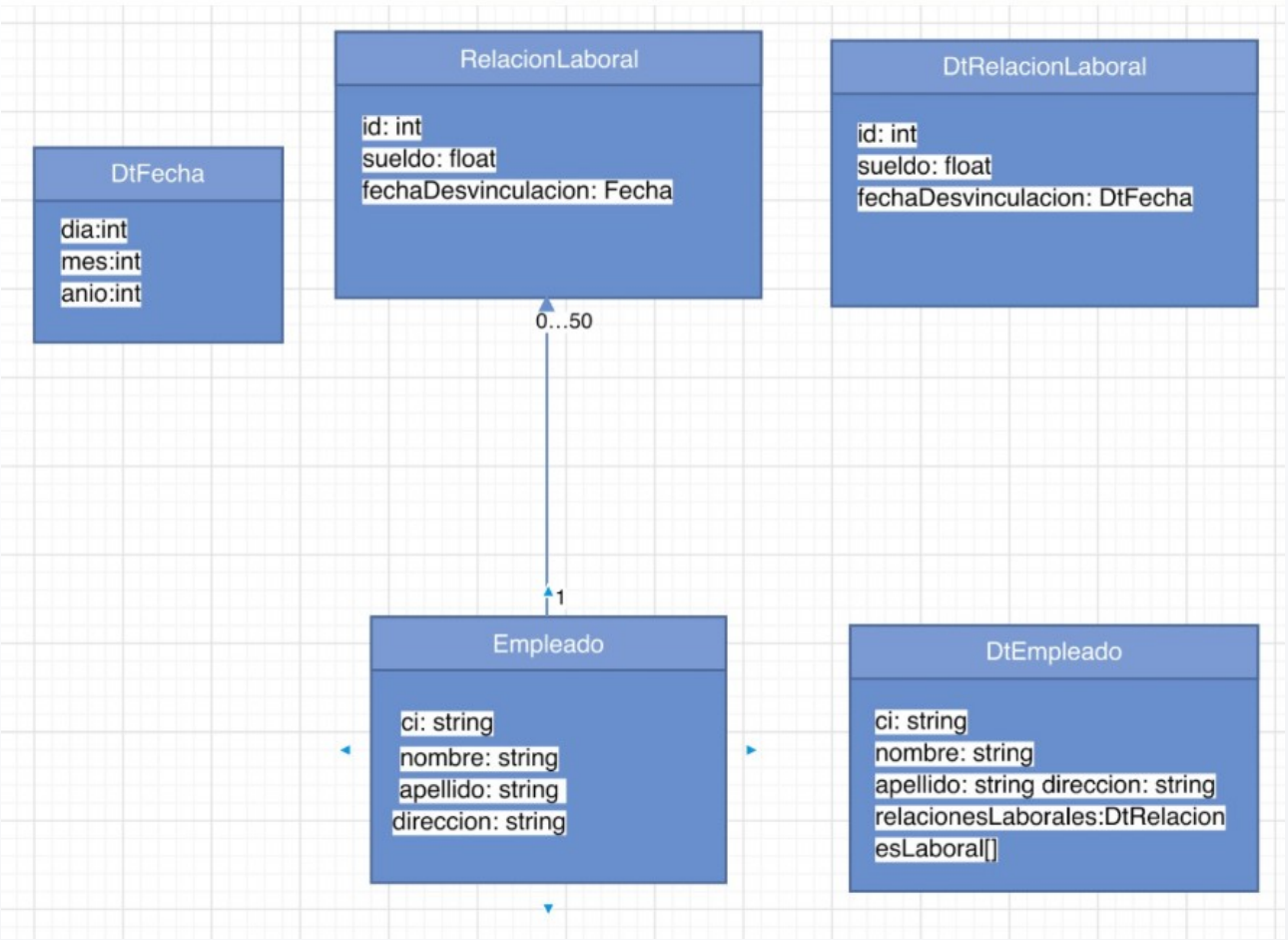
Ejercicio – Parte 2

Luego de realizar la primera parte, ahora tenemos la necesidad de mejorar el manejo de las excepciones y ademas implementar las demás clases como datatype para poder mostrar los datos al usuario.



Funcionalidades a realizar

1 – Crear datatypes, atributos de estos, getters y constructores.

2 – Modificar las siguientes funciones a como se muestra a continuación:

- **Agregar empleado**:

  void agregarEmpleado(DtEmpleado empleado);

- **Agregar relaciones laborales a un empleado X**

  void agregarRelacionLaboral(DtRelacionLaboral);

- **Listar todos los empleados**

<>Prof. Lucas Cíceri
Año 2021<\>

DtEmpleado[] listarEmpleados();

- **Listar todas las relaciones laborales de un empleado X.**

DtRelacionLaboral[] listarRelacionesLaborales(ci);

- **Desvincular una relación laboral de un empleado X.**

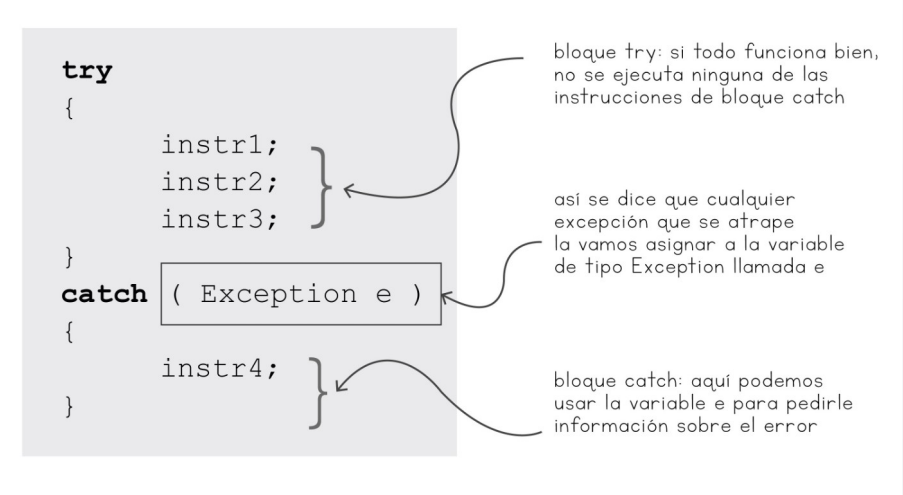
void desvincularRelacionesLaborales(ci, id);

3 – Pasar todos los controles realizados en el sistema a excepciones utilizando try/catch.

¿Que son las excepciones ?

Una expresión es la manera en qué expresamos en un lenguaje de programación algo sobre el estado de un objeto. Es el medio que tenemos para decir en un programa algo sobre el problema.

Bloques try/catch




```
void MyFunc(int c)
{
    if (c > numeric_limits< char> ::max())
        throw invalid_argument("MyFunc argument too large.");
    //...
}

int main()
{
    try
    {
        MyFunc(256); //cause an exception to throw
    }

    catch (invalid_argument& e)
    {
        cerr << e.what() << endl;
        return -1;
    }
    //...
    return 0;
}
```

Lucas Cíceri
Año 2021<\>