Se requiere un sistema de presentismo que sea capaz de:

1) Tener Preceptores, Asistencias y Alumnos (pueden ser AlumnoRegular o AlumnoOyente)

2) Se deberá poder tomar asistencia de un día en particular (ingresado por usuario) a todos

los alumnos **regulares** registrados. (Alta Asistencia)

3) Se deberá poder listar las asistencias de una fecha ingresada por el usuario con su {FORMATO}. (Listar Asistencia)

Consideraciones punto 1

1. ToString solo se debe sobrescribir en Persona y en Asistencia.

Consideraciones punto 2

1. Si el alumno no es regular, mostrar por pantalla “El alumno {FORMATO} es oyente” y no pedir asistencia ni agregar a la lista de asistencia.
2. En caso que la lista de asistencia ingresada no tenga una cantidad igual a la lista de alumnos regulares registrados, se debe arrojar una AsistenciaInconsistenteException.
3. Solo se puede agregar a la lista (\_asistencias) una vez por fecha y cada vez que se ingresen asistencias a la lista (\_asistencias) se debe ingresar un string a la lista de fechas (\_fechas).

Tip: firma de AgregarAsistencia(List<Asistencia> asistencias, string fecha)

1. En caso que la fecha ya exista en la lista (\_fechas) se deberá arrojar una AsistenciaExistenteEseDiaException.

Consideraciones punto 3

1. En caso que no haya asistencias mostrar “No hay registros para la fecha {FECHAINGRESADA}”.

Formatos para ToString

1. Preceptor “APELLIDO - LEGAJO”
2. Alumno “NOMBRE (REGISTRO)”
3. Asistencia "FECHAREFERENCIA {ALUMNO FORMATEADO} está presente {SI o NO} por {PRECEPTOR FORMATEADO} registrado el {FECHAHORAREAL}"
4. Fechas de tipo string “AAAA-MM-DD”

Se deben respetar las visibilidades del diagrama

* Publico - Privado # Protegido = Internal

\* El proyecto principal deber ser consola

\* Al menos en dos capas

\* Con validación de las entradas de usuario

\* Con manejo de excepciones

\* La consola se debe mantener activa hasta que el usuario desee finalizar el programa ingresando X

\* Se debe resolver en 2 horas

\* Debe compilar

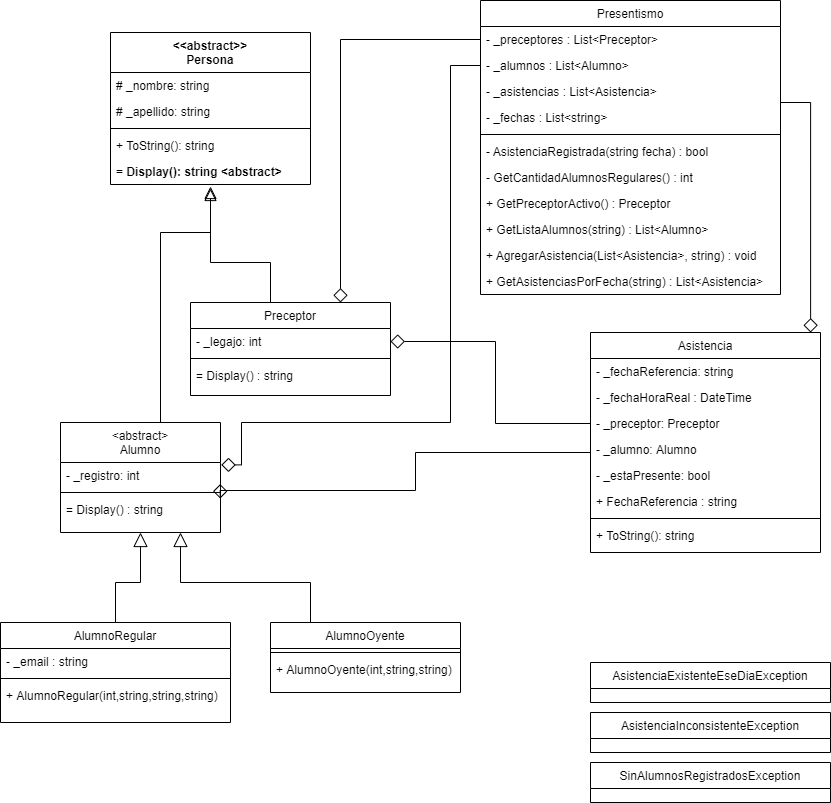
\* No debe terminar por excepción no controlada

\* El prefijo del nombre de los proyectos debe ser Parcial1.[APELLIDO].[NombreProyecto]

\* Se entrega la solución entera (comprimida) en la dirección de drive previamente asignada

\* Es válido agregar los constructores que necesiten

# Modelo



# Template de la clase Program (para completar)

public class Program

{

private static Presentismo \_presentismo;

static Program()

{

\_presentismo = new Presentismo();

}

static void Main(string[] args)

{

Preceptor preceptorActivo = \_presentismo.GetPreceptorActivo();

DesplegarOpcionesMenu();

string opcionMenu = ""; // pedir el valor

switch (opcionMenu)

{

case "1":

TomarAsistencia(preceptorActivo);

break;

case "2":

MostrarAsistencia();

break;

case "X":

// SALIR

break;

default:

break;

}

}

static void DesplegarOpcionesMenu()

{

Console.WriteLine("1) Tomar Asistencia");

Console.WriteLine("2) Mostrar Asistencia");

Console.WriteLine("X: Terminar");

}

static void TomarAsistencia(Preceptor p)

{

// Ingreso fecha

// Listar los alumnos

// para cada alumno solo preguntar si está presente

// agrego la lista de asistencia

// Error: mostrar el error y que luego muestre el menu nuevamente

}

static void MostrarAsistencia()

{

// ingreso fecha

// muestro el toString de cada asistencia

}

}

## Datos de Inicialización

// iniciar Presentismo con los siguientes datos

public Presentismo()

{

\_alumnos = new List<Alumno>();

\_asistencias = new List<Asistencia>();

\_preceptores = new List<Preceptor>();

\_fechas = new List<string>();

\_alumnos.Add(new AlumnoRegular(123, "Carlos", "Juarez", "cjua@gmail.com"));

\_alumnos.Add(new AlumnoRegular(124, "Carla", "Jaime", "cjai@gmail.com"));

\_alumnos.Add(new AlumnoOyente(320, "Ramona", "Vals"));

\_alumnos.Add(new AlumnoOyente(321, "Alejandro", "Medina"));

\_preceptores.Add(new Preceptor(5, "Jorgelina", "Ramos"));

}