Primer Parcial Práctico - CAI 2021-1

Martes 27 de Abril de 2021

Sistema de Presentismo

Casos de Uso

Se requiere un sistema de presentismo que sea capaz de cumplir con los siguientes requerimientos:

1) Contar con Preceptores, Asistencias y Alumnos (pueden ser AlumnoRegular o AlumnoOyente)
2) Permitir que un preceptor pueda cargar la asistencia de un día en particular (día a ser ingresado por el usuario) a todos los alumnos **regulares** registrados. (Alta Asistencia)
3) Listar las asistencias de un día en particular (día a ser ingresado por el usuario) con un {FORMATO} específico. (Listar Asistencia)

Consideraciones punto 1

a) ToString solo se debe sobrescribir en Persona y en Asistencia (implementar Display de acuerdo al modelo de clases).

Consideraciones punto 2

a) Si el registro de alumno ingresado no es regular, mostrar por pantalla “El alumno {FORMATO} es oyente” y no pedir asistencia ni agregar a la lista de asistencia.
b) 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.
c) Solo se puede agregar a la lista (_asistencias) una vez por cada fecha (no puede haber el mismo alumno con la misma fecha más de UNA vez).
   Tip: Se puede usar el método AddRange para agregar colecciones a una lista.
d) En caso que la fecha ya exista en la lista (_asistencias) se deberá arrojar una AsistenciaExistenteEseDiaException.
e) La lista de alumnos incluye tanto alumnos Regulares como Oyentes.
f) En la lista que se **envía** al método AgregarAsistencia NO puede haber fechas distintas.

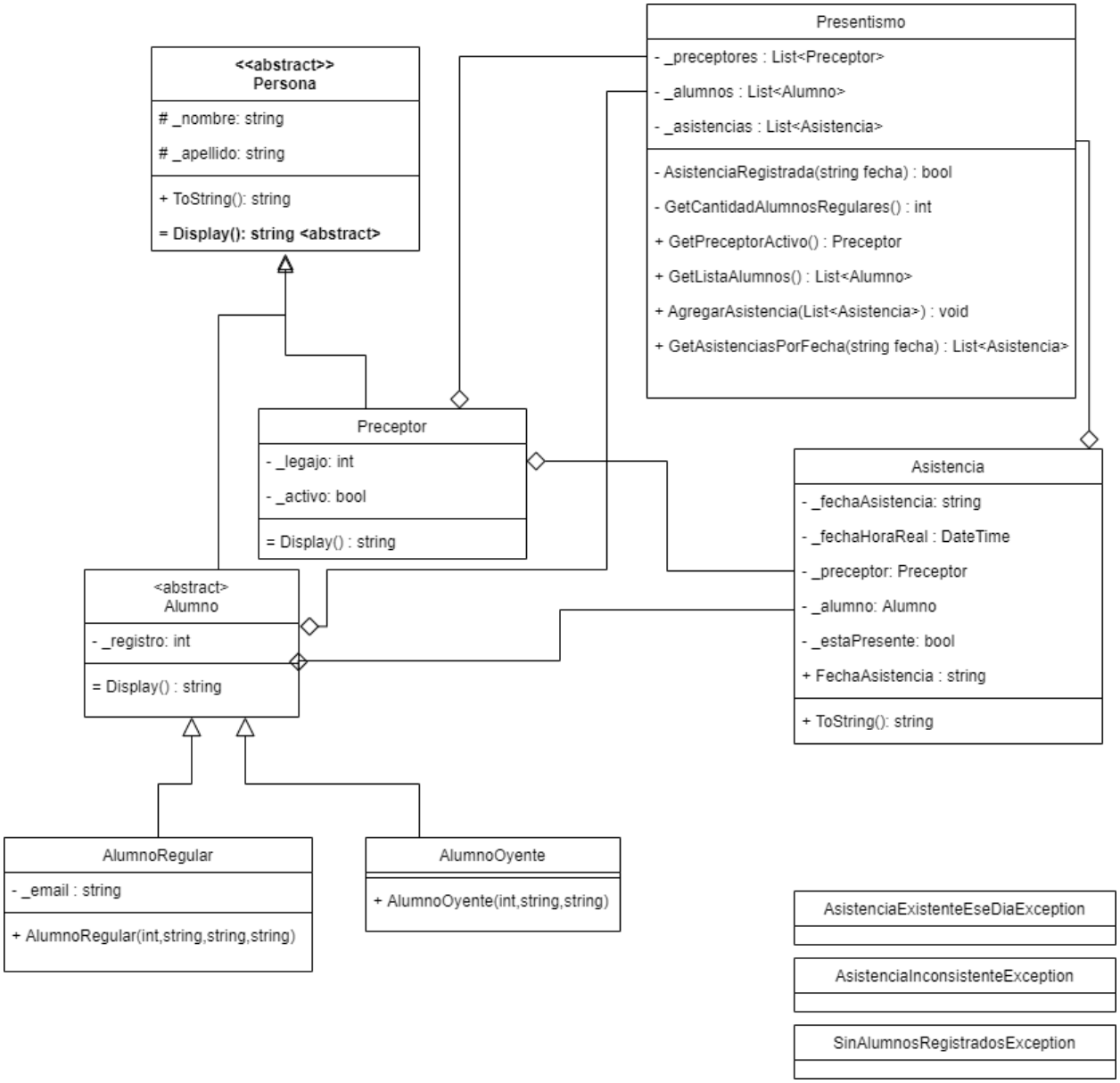
Consideraciones punto 3

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

Formatos para ToString

a) Preceptor “APELLIDO - LEGAJO”
b) Alumno “NOMBRE (REGISTRO)”
c) Asistencia "FECHAREFERENCIA {ALUMNO FORMATEADO} está presente {SI o NO} por {PRECEPTOR FORMATEADO} registrado el {FECHAHORAREAL}"
d) Fechas de tipo string “AAAA-MM-DD” (.ToString(“yyyy-MM-dd”)

Modelo



Consideraciones generales

- El proyecto principal debe ser una aplicación de consola
- Debe estar implementado en al menos dos capas
- Es necesario contar con validación de las entradas de usuario
- Tiene que tener manejo de excepciones
- La consola debe estar activa hasta que el usuario desee finalizar el programa (ingresando X)
- Se debe resolver en 2 horas
- Debe compilar sin errores

- Debe funcionar de acuerdo a las interfaces y definiciones dadas en los fragmentos de código provistos
- No debe terminar por excepción no controlada
- El prefijo del nombre de los proyectos debe ser **Parcial1.[APELLIDO].[NombreProyecto]**
- Se debe entregar la solución entera (comprimida en un archivo) en la dirección de drive previamente asignada (borrar la carpeta BIN de los 2 proyectos del archivo ZIP)
- Es válido agregar los constructores que necesiten
- Respetar las visibilidades del diagrama

+ Público	- Privado	# Protegido	= Internal

Template de la clase Program

El código provisto a continuación tiene que ser utilizado como estructura inicial para el desarrollo de la clase Program. Completar de acuerdo a lo que sea necesario.

```
public class Program
{
    private static Presentismo _presentismo;

    static Program()
    {
        _presentismo = new Presentismo();
    }
    // modificar lo que corresponda para que solo finalice el
    // programa si el usuario presiona X en el menú
    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 menú 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>();

    _preceptores.Add(new Preceptor(5, "Jorgelina", "Ramos", true));
    _preceptores.Add(new Preceptor(6, "Juan", "Gutierrez", false));

    _alumnos.Add(new AlumnoRegular(123, "Carlos", "Juárez", "cj@gmail.com"));
    _alumnos.Add(new AlumnoRegular(124, "Carla", "Jaime", "carla@gmail.com"));
    _alumnos.Add(new AlumnoOyente(320, "Ramona", "Vals"));
    _alumnos.Add(new AlumnoOyente(321, "Alejandro", "Medina"));
}
```