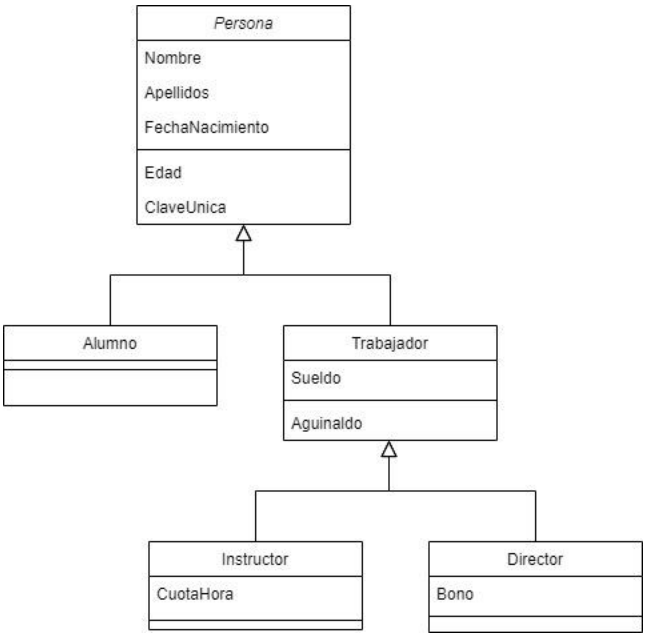


```
public abstract class Persona
{
    private string _nombre;
    private DateTime _fechaNacimiento;
    /// <summary> Propiedad de lectura y escritura
    10 referencias
    public string nombre...
    /// <summary> Propiedad de lectura y escritura
    6 referencias
    public DateTime fechaNacimiento...
    /// <summary> Propiedad de lectura y escritura
    11 referencias
    public string apellidos { get; set; }

    public Persona()
    {
    }

    /// <summary> Sobrecarga del Constructor, con valores de las propiedades
    2 referencias
    public Persona(string nombre, string apellidos, DateTime fechaNacimiento)
    {
        _nombre = nombre;
        this.apellidos = apellidos;
        _fechaNacimiento = fechaNacimiento;

    }
    /// <summary> Método Virtual la clase derivada puede sobreescribirlo (overv
    3 referencias
    public virtual int edad()
    {

        DateTime hoy = DateTime.Now;
        int edad = hoy.Year - _fechaNacimiento.Year;
        return edad;
    }
    /// <summary> Método Abstracto la clase derivada debe implementarlo
    7 referencias
    public abstract string ClaveUnica();
```

```
public class Alumno : Persona
{
    0 referencias
    public Alumno()
    {

    }
    1 referencia
    public Alumno(string nombre, string apellidos, DateTime fechaNacimiento) : base(nombre, apellidos, fechaNacimiento)
    {

    }
    7 referencias
    public override string ClaveUnica()
    {
        return nombre.Substring(0, 2) + apellidos.Substring(0, 2) + fechaNacimiento.ToString().Substring(0, 4);
    }
}
```

```
public class Trabajador : Persona
{
    10 referencias
    public virtual decimal sueldo { get; set; }
    1 referencia
    public Trabajador()
    {

    }
    2 referencias
    public Trabajador(string nombre, string apellidos, DateTime fechaNacimiento) : base(nombre, apellidos, fechaNacimiento)
    {

    }
    7 referencias
    public override string ClaveUnica()
    {
        return nombre.Substring(0, 2) + edad().ToString() + apellidos.Substring(0, 2);
    }
    5 referencias
    public virtual  decimal Aguinaldo(int diaslaborados)
    {
        decimal aguinaldo;
        aguinaldo = sueldo / 24 * (diaslaborados / 365);
        return aguinaldo;
    }
    5 referencias
    public override string ToString()
    {
        return "Trabajador: " + nombre + " " + apellidos + "sueldo: " + sueldo.ToString();
    }

}
```

```
public class Instructor: Trabajador
{
    2 referencias
    public decimal cuotaHora { get; set; }

    0 referencias
    public Instructor()
    {

    }
    1 referencia
    public Instructor(string nombre, string apellidos, DateTime fechaNacimiento) : base(nombre, apellidos, fechaNacimiento)
    {

    }
    7 referencias
    public override string ClaveUnica()
    {
        return "I" + nombre.Substring(0, 2) + edad().ToString() + apellidos.Substring(0, 2);
    }
    1 referencia
    public decimal CalcularSueldo(int horaslaborados)
    {
        return cuotaHora * horaslaborados;
    }
}
```

```
public class Director : Trabajador
{
    3 referencias
    public decimal bono { get; set; }

    0 referencias
    public Director()
    {

    }
    1 referencia
    public Director(string nombre, string apellidos, DateTime fechaNacimiento) : base(nombre, apellidos, fechaNacimiento)
    {


    }
    5 referencias
    public override decimal Aguinaldo(int diaslaborados)
    {
        decimal aguinaldo;
        aguinaldo = sueldo / 24 * (diaslaborados / 365) + 4 * bono;
        return aguinaldo;
    }

    5 referencias
    public override string ToString()
    {
        return "Director: " + nombre + " " + apellidos + "bono: " + sueldo.ToString();
    }
}
```