class HeatEquationSolver(α, timeInterval, h, Ω, Γ_N, Γ_D, u_0, g^D, g^N)

timeInterval = I

- α: constant almost everywhere
- timeInterval: Object with variables for $I=[t_0, t_{max}]$ and h
- Ω, Γ_N, Γ_D: Object Ω with
  - Mesh of Ω { List of Points; List of Simplices referring to Points
  - functions that return
    - NeumannFaces
    - DerichlefSimplices
    - DerichletPoints

    they all refer to the points in the Mesh of Ω
- u_0, g^D, g^N: function handles
  - $u_0(x)$
  - $u^D(t,x)$
  - $u^N(t,x)$

that solves the PDE

$$\dot{u} = \alpha \Delta u \quad \text{on } I \times \Omega$$
$$u = g^D \quad \text{on } I \times \Gamma_D$$
$$\frac{\partial}{\partial \eta} u = g^N \quad \text{on } I \times \Gamma_N$$
$$u|_{t=0} = u_0 \quad \text{on } \Omega$$

includes public function "solve()", which solves the PDE

```
solve() {
    (A,M,N,D) = initializeSystem(α, Ω, u0, g^D, g^N)
    T = TimeSolver(u0, I, A, M, N+D)
    u = T.solve
    u.addBoundary
    return u
}
```

private functions:

- (A,M,N,D) = initializeSystem(α, Ω, u0, g^D, g^N)
  - A = getStiffnessMatrix()
  - M = getODEMatrix()
  - N = getNeumannVector()
  - D = getDirichletVector()
- getStiffnessMatrix()
- getODEMatrix()
- getNeumannVector()
- getDeirdletVector()
- addBoundary()

---

class TimeSolver (u_0, O_I, A, M, N+D)

that solves the system of ODEs

$$M\dot{u} = Au + V$$