UNIVERSIDADE FEDERAL DE OURO PRETO – UFOP
Ciência da Computação

Sprint I: Desenvolvimento de um Framework C++ para a Construção de Simulações Baseadas na Dinâmica de Sistemas.

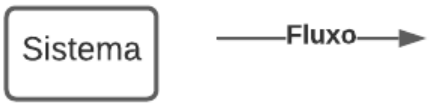
Professor: Tiago Garcia de Senna Carneiro
Aluno: João Francisco Bittencourt, 19.1.4016

Funcionalidades:

A API deve simular um sistema dinâmico da Teoria Geral de Sistemas apresentada em aula. Associando fórmulas matemáticas a fluxos que representam variações de atributos (energia, massa, etc) entre diferentes sistemas:
Sistema: representa um estoque de qualquer coisa, um acumulador.
Fluxo: representa uma variação de qualquer coisa, associada a uma fórmula algébrica.

Casos de uso:

Para o desenvolvimento das funcionalidades descritas, são analisados vários casos de uso, que servem para entender o comportamento da API para diferentes composições e casos.
Caso de sistema ou fluxos isolados:

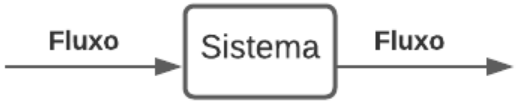



```
Flow f1;
System s1;
```

Caso de sistema com entrada ou saída único:

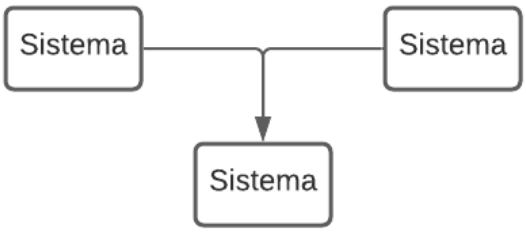

```
System s1;
Flow f1;
f1.link(NULL,s1); ou f1.link(s1,NULL);
```

Caso de sistema com entrada e saída:



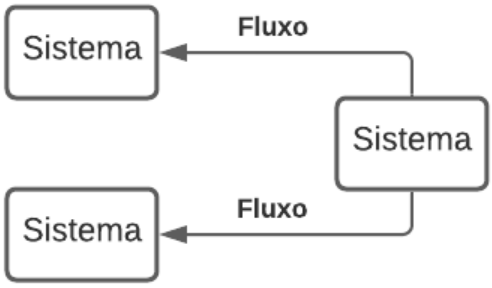

```
System s1;
Flow f1,f2;
f1.link(NULL,s1)~;
f2.link(s1,NULL);
```

Caso de um sistema recebendo fluxo de outros sistemas:



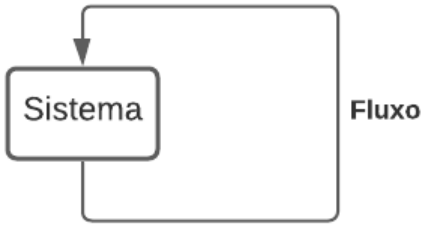

```
System s1,s2,s3;
Flow f1,f2;
f1.link(s1,s3);
f2.link(s2,s3);
```

Caso de um sistema enviando fluxos diferentes para outros sistemas:




```
System s1,s2,s3;
Flow f1,f2;
f1.link(s1,s3);
f1.link(s1,s2);
```

Caso de um sistema cíclico:




```
System s1;
Flow f1;
f1.link(s1,s1);
```

Casos de teste:

Criando o modelo e componentes:

```
Model m;
System s1, s2;
Flow f1,f2;
```

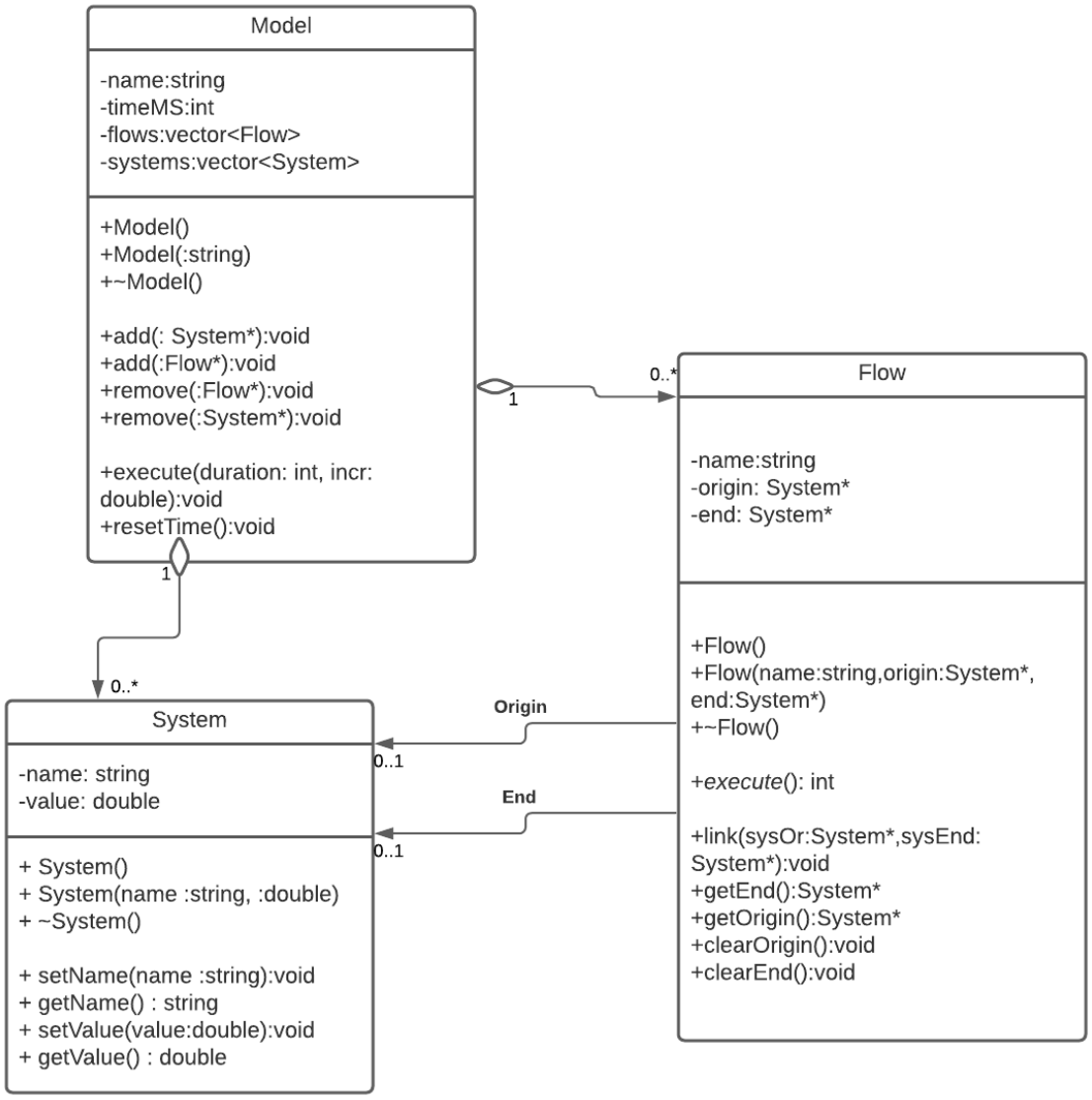
Adicionando componentes ao modelo:

```
m.add(s1);
m.add(s2);
m.add(f1);
m.add(f2);
f1.link(s1,s2);
f2.link(s2,s1);
```

Executando o modelo:

```
m.execute(5000, 1000);
```

Modelo UML:


Model
-name:string
-timeMS:int
-flows:vector<Flow>
-systems:vector<System>
+Model()
+Model(:string)
+~Model()
+add(: System*):void
+add(:Flow*):void
+remove(:Flow*):void
+remove(:System*):void
+execute(duration: int, incr: double):void
+resetTime():void
1
0..*
1
0..*
Flow
-name:string
-origin: System*
-end: System*
+Flow()
+Flow(name:string,origin:System*, end:System*)
+~Flow()
+execute(): int
+link(sysOr:System*,sysEnd: System*):void
+getEnd():System*
+getOrigin():System*
+clearOrigin():void
+clearEnd():void
Origin
0..1
End
0..1
System
-name: string
-value: double
+ System()
+ System(name :string, :double)
+ ~System()
+ setName(name :string):void
+ getName() : string
+ setValue(value:double):void
+ getValue() : double