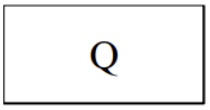
UNIVERSIDADE FEDERAL DE OURO PRETO
DEPARTAMENTO DE COMPUTAÇÃO

SPRINT 1

CAMILA APARECIDA SILVA OLIVEIRA - 22.2.4007

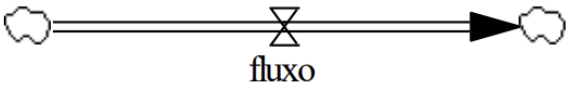
OURO PRETO
15/12/2024

1) Critério - Sistema sozinho.



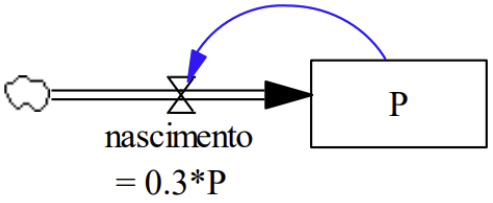

```C/C++
System *Q = new System(10);

Model model = new Model();

model.add(Q);

model.run(100);
```

2) Critério - Fluxo sozinho.



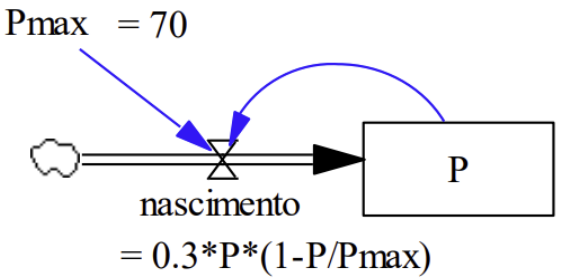

```C/C++
class FlowTest : Flow {
public:
	executeEquation(){
		return 0;
	}
}

Flow *fluxo = new Flow();

Model model = new Model();

model.add(fluxo);

model.run(100);
```

3) Critério - Fluxo sem origem conectado a um sistema (Modelo exponencial)




```C/C++
class ExponencialFlow : Flow {
public:
	double executeEquation(){
		return 0.3 * this->getTarget()->getValue();
	}
}

Model model = new Model();
System *P = new System(10);
ExponencialFlow *nascimento = new ExponencialFlow();

nascimento.setTarget(P);

model.add(P);
model.add(nascimento);

model.run(100);
```

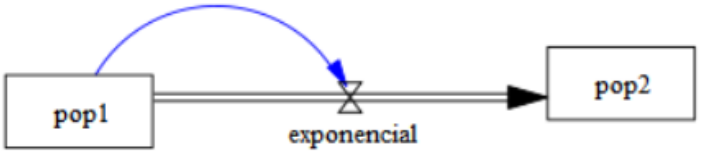
4) Critério - Fluxo sem origem conectado a um sistema (Modelo logístico)




```C/C++
class LogisticFlow : Flow {
public:
	double executeEquation(){
		double value = this->getTarget()->getValue();
```

```
                return 0.3 * value * (1 - value / Pmax);
        }
}

Model model = new Model();
System *P = new System(10);
ExponencialFlow *nascimento = new ExponencialFlow();

nascimento.setTarget(P);

model.add(P);
model.add(nascimento);

model.run(100);
```

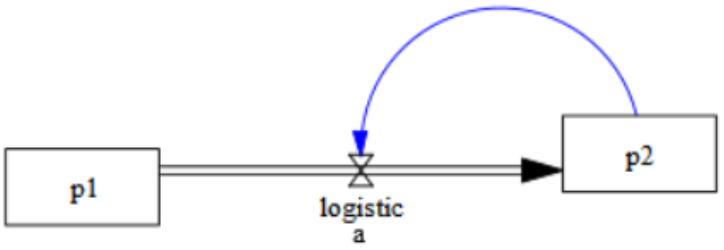
5) Critério - Fluxo conectando dois sistemas (Modelo exponencial).




```
C/C++
class ExponencialFlow : Flow {
public:
        double executeEquation(){
                return 0.01 * this->getSource()->getValue();
        }
}

Model model = new Model();
System* pop1 = new System(100);
System* pop2 = new System();
ExponencialFlow *exponencial = new ExponencialFlow();

exponencial->setSource(pop1);
exponencial->setTarget(pop2);

model.add(pop1);
model.add(pop2);
```

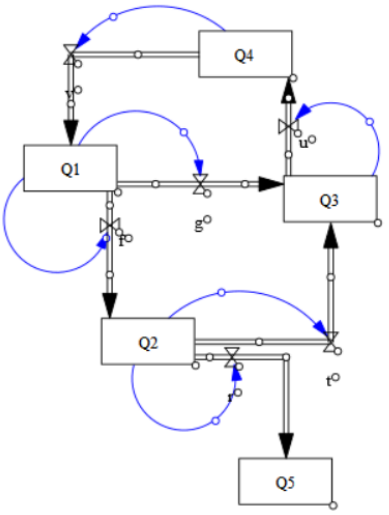
```
model.add(exponencial);

model.run(100);
```

6) Critério - Fluxo conectando dois sistemas (Modelo logístico).




```C/C++
class LogisticFlow : Flow {
public:
	double executeEquation(){
		double value = this->getTarget()->getValue();
		return 0.01 * value * (1 - value / Pmax);
	}
}

Model model = new Model();
System* p1 = new System(100);
System* p2 = new System(10);
LogisticFlow *logistic = new LogisticFlow();

exponencial->setSource(pop1);
exponencial->setTarget(pop2);

model.add(pop1);
model.add(pop2);
model.add(logistic);

model.run(100);
```

7) Critério - Múltiplos sistemas e fluxos interconectados.




```
C/C++
class ExponencialFlow : Flow {
public:
	ExponentialFlow(System *source, System *target) : Flow(source,
target){}
	double executeEquation(){
		return 0.01 * this->getSource()->getValue();
	}
}

Model model = new Model();

System* Q1 = new System();
System* Q2 = new System();
System* Q3 = new System();
System* Q4 = new System();
System* Q5 = new System();

ExponencialFlow *v = new ExponencialFlow(Q4, Q1);
ExponencialFlow *u = new ExponencialFlow(Q3, Q4);
ExponencialFlow *f = new ExponencialFlow(Q1, Q2);
ExponencialFlow *g = new ExponencialFlow(Q1, Q3);
ExponencialFlow *t = new ExponencialFlow(Q2, Q3);
ExponencialFlow *r = new ExponencialFlow(Q2, Q5);

model.add(Q1);
model.add(Q2);
model.add(Q3);
model.add(Q4);
model.add(Q5);
```

```
model.add(v);
model.add(u);
model.add(f);
model.add(g);
model.add(t);
model.add(r);

model.run(100);
```

DIAGRAMA UML