Susan Kare (1954)

- Deseñadora de iconas, elementos gráficos e tipografías dixitais en ordenadores, móviles, ... para as maiores empresas de informática (Microsoft, Apple, IBM, Facebook, ...)
- Pioneira do *pixel art*
- A máis incluínte deseñadora de iconas informáticos segundo o MoMA (Museo de Arte Moderna) de Nova Iorque

Clase en Octave

- Bloque *classdef*
- En arquivo co mesmo nome que a clase (*punto.m*)
- Seccións *properties* e *methods*
- Declaración de obxecto:

  p=punto(1,3);
- Acceder a variable: *p.x*
- Chamar a función: *p.mostra*()

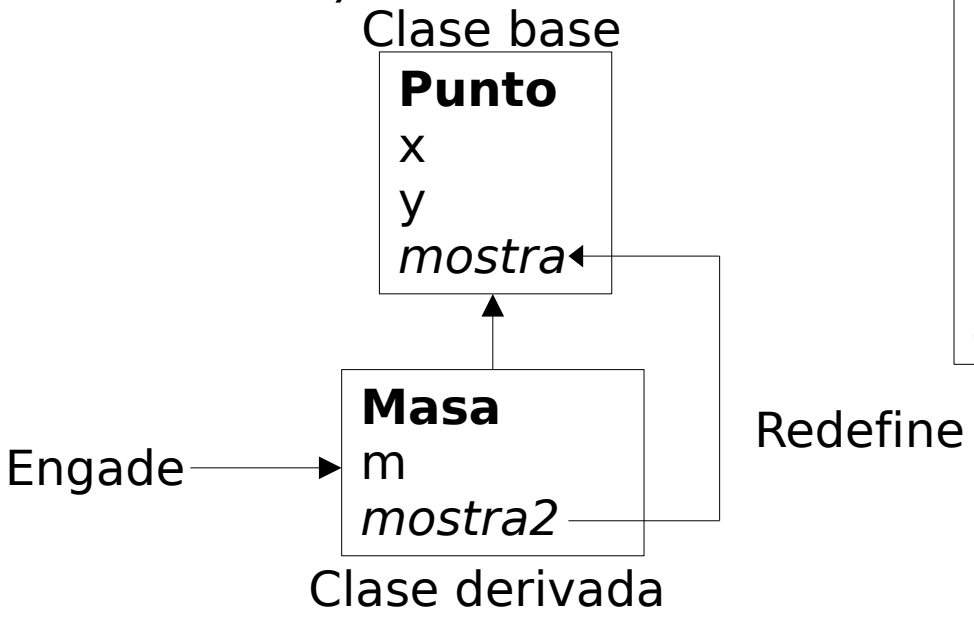
```
classdef punto
    properties
        x
        y
    end
    methods
        function p=punto(x,y)
            p.x=x;p.y=y;
        end
        function mostra(p)
            printf('punto: x=%g y=%g\n',p.x,p.y)
        end
    end
end
```

Herdanza en Octave

- Clase derivada: *masa*
- Hereda de *punto*
- Engade a variábel *m*
- Redefine a función *mostra*() de *punto* (sobrecarga da función)

Isto é o que fai que *masa* herede de *punto*

```
classdef masa < punto
    properties
        m
    end
    methods
        function ms=masa(x,y,m)
            ms=ms@punto(x,y);
            ms.m=m;
        end
        function mostra(p)
            printf('masa: x=%g y=%g m=%g\n',
              ms.x,ms.y,ms.m)
        end
    end
end
```



Polimorfismo en Octave

- Chamar a funcións distintas co mesmo nome sen ter que comprobar no programa qué función se executa
- *p.mostra*(): chama á función *mostra*() da clase base (*punto*)
- *m.mostra*(): chama á *mostra*() da clase derivada (*masa*)
- Se hai varias clases derivadas que redefinen *mostra*(), unha variable *x* pode chamar a distintas funcións *mostra*() dependendo de a que clase se refiran

Chamando manualmente a cada función

```
x=punto(1,2);
x.mostra()
x=masa(2,3,1);
x.mostra()
```

Chamando a cada función automáticamente nun bucle

```
p=punto(1,2);
m=masa(2,1,4);
y={p,m};
for i=1:2
    y{i}.mostra()
end
```

En cada iteración executa a función *mostra*() de cada obxecto

Sobrecarga de operadores

- Para redefinir (sobrecargar) un operador aritmético, relacional ou lóxico hai que definir unha función cun nome específico para cada operador
- Por exemplo, para a suma (operador +) a función débese chamar *plus*()
- As prioridades dos operadores redefinidos non cambian

a+b(plus)	a-b(minus)	a*b(mtiles)	a.*b(times)	a./b(rdivide)
a.\b(ldivide)	a/b(mrdivide)	a\b(mldivide)	a^b(power)	a.^b(mpower)
a<b(lt)	a>b(gt)	a<=b(le)	a>=b(ge)	a==b(eq)
a&b(and)	a|b(or)	~a(not)	a’(ctranspose)	a~=b(ne)

Sobrecarga de operador +

Ficheiro *sobrecarga.m*

```
clear all;clc
p=punto(1,3);
m=masa(4,5,1);
printf('sobrecarga de operador +:\n')
s=p+m;
s.mostra()
```

Ficheiro *punto.m*

```
classdef punto
    properties
        x
        y
    end
    methods
        function p=punto(x,y)
            p.x=x;p.y=y;
        end
        function mostra(p)
            printf('punto: x=%g y=%g\n',p.x,p.y)
        end
        function r=plus(a,b)
            x=a.x+b.x;y=a.y+b.y;
            r=punto(x,y);
        end
    end
end
```