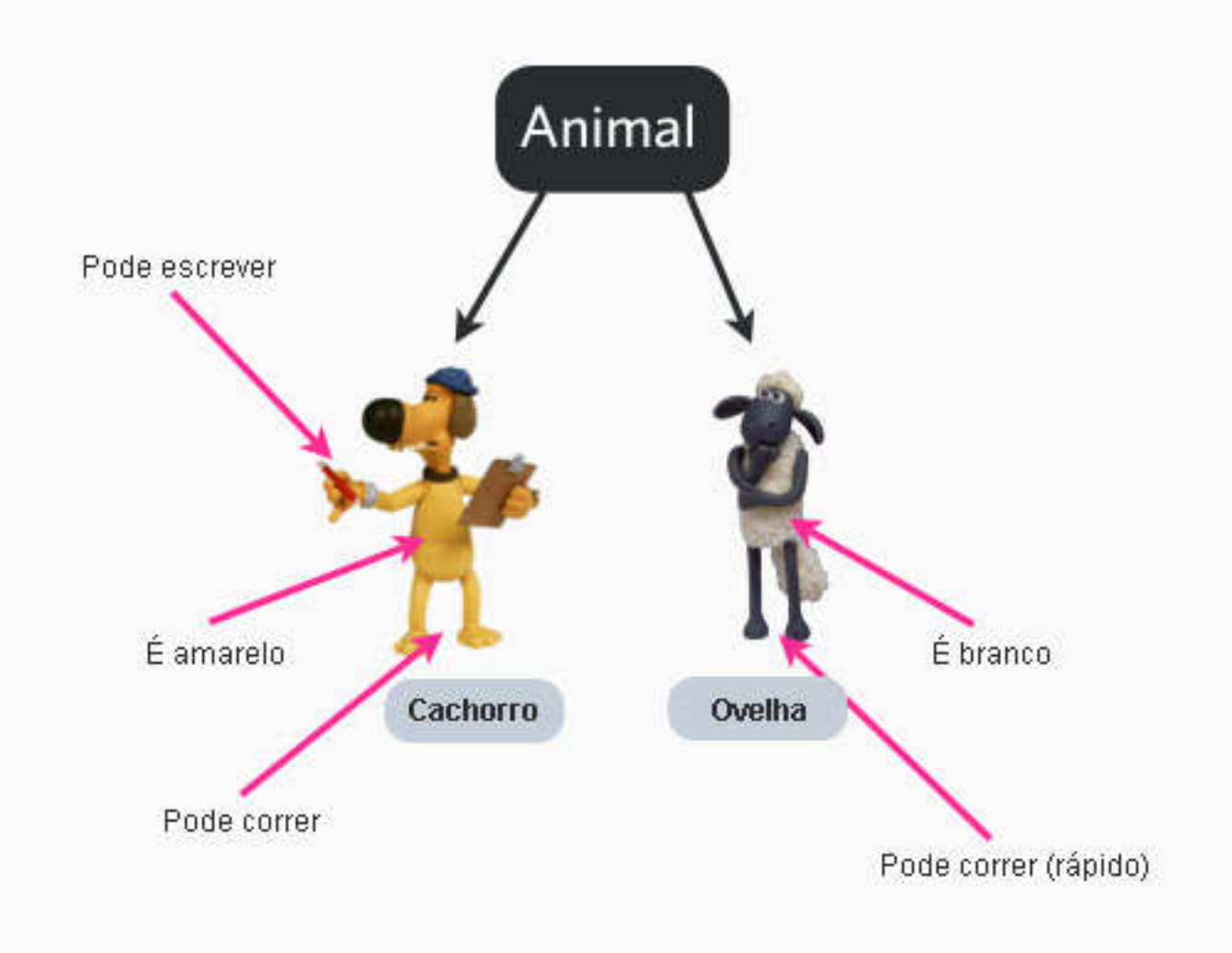
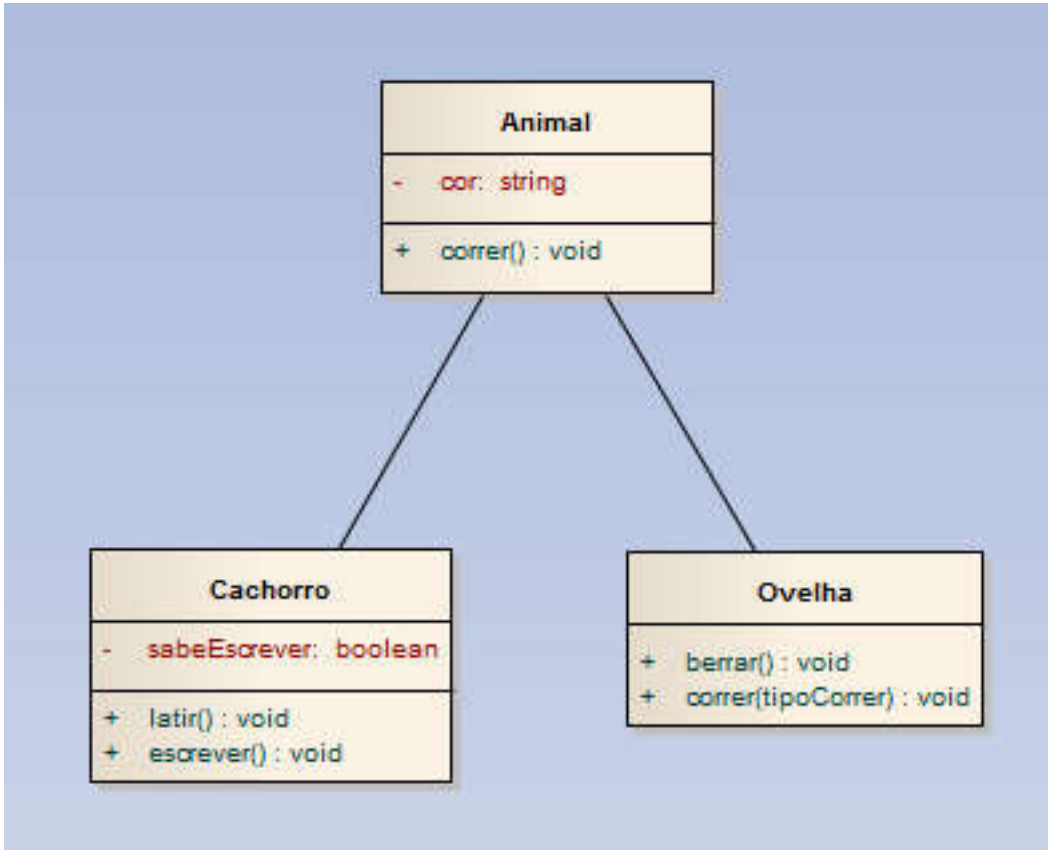
Introdução à Programação Orientada a Objetos

Objetos, propriedades, ações



Relações entre objetos, diagrama classes



Classes – propriedades e construtores

```
class Ponto {
    // Propriedades
     double x;
     double y;

     // Construtores
     Ponto(double x, double y) {
          this.x=x;
          this.y=y;
     }
     Ponto() {
          this.x=0;
          this.y=0;
     }}
```

Os construtores permitem inicializar os campos de um objeto

- Têm o mesmo nome da classe e não têm tipo nem return
- Podem existir vários construtores (***overloading /sobrecarga***), são distinguidos pelo tipo de argumentos
- O identificador ***this*** refere o próprio objeto

Classes – exemplo

```
public class Pontos {
   public static void main (String args[]) {
        Ponto p1 = new Ponto(1.0,1.0);
        Ponto p2 = new Ponto();
        System.out.printf("p1(%.1f,%.1f)\np2(%.1f,%.1f)\n",
            p1.x,p1.y,p2.x,p2.y);
   }
}
```

p1(1.0,1.0)
p2(0.0,0.0)

Classes – definição métodos (ações)

```
class Ponto {
    double x;
    double y;
    // Métodos
    public void printP() {
        System.out.printf("ponto(%.1f,%.1f)\n",this.x,this.y);
    }
    public String toString(){
        return String.format("ponto(%.1f,%.1f)\n",this.x,this.y);
    }
    public double dist(double x1,double y1) {
        return Math.sqrt((x-x1)*(x-x1)+(y-y1)*(y-y1));
    }
    public double dist(Ponto p) {
        return Math.sqrt((x-p.x)*(x-p.x)+(y-p.y)*(y-p.y));
    }
    // Construtores
    Ponto(double x, double y) {
        this.x=x;
        this.y=y;
    }
    Ponto() {
        this.x=0;
        this.y=0;
    }
}
```

Pode haver métodos, tal como os construtores, com o mesmo nome (***Overloading* /sobrecarga**)
- A distinção é feita pelos diferentes argumentos

Classes – Criar e usar objetos

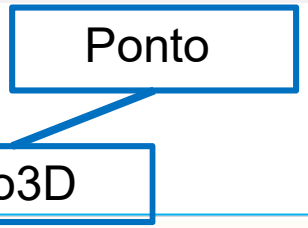
```
import java.util.*;
public class Classes_pontos_metodos {
    static Scanner sc = new Scanner(System.in);
    public static void main (String args[]) {
        Ponto p1, p2;
        // Uso dos construtores
        p1 = new Ponto(1.0,1.0);
        p2 = new Ponto();
        p2 = lerPonto();
        // Teste dos métodos
        System.out.printf("Distancia entre p1 e p2 = %.2f; %.2f\n",distancia(p1,
        p2), p2.dist(p1) );
        p1.printP();
        System.out.println(p2.toString());
        System.out.printf("Distancia de p1 e Origem = %f\n",p1.dist(0.0,0.0));
    }
    public static Ponto lerPonto() {
        Ponto p = new Ponto();
        System.out.print("x: ");
        p.x = sc.nextDouble();
        System.out.print("y: ");
        p.y = sc.nextDouble();
        return p;
    }
    public static double distancia(Ponto a, Ponto b) {
        return Math.sqrt(Math.pow(b.x-a.x, 2) + Math.pow(b.y-a.y, 2));
    }

}
```

```
C:\WINDOWS\SYSTEM32\cmd.exe
x: 3
y: 3
Distancia entre p1 e p2 = 2.83; 2.83
ponto(1.0,1.0)
ponto(3.0,3.0)

Distancia de p1 e Origem = 1.414214
```

Classes – herança




```
// classe Ponto3D herda da classe Ponto2 (classe mãe) - extends
class Ponto3D extends Ponto{
    // Propriedades
    // privadas, são só do conhecimento do objeto
    private double z;

   // as propriedades são lidas e atualizadas através de métodos
   public double getZ(){
    return this.z;
   }
   public void setZ(double v){
      this.z = v;
   }

}
```

```
import java.util.*;
public class Pontos3 {
    static Scanner sc = new Scanner(System.in);
    public static void main (String args[]) {
        Ponto2 p1, p2;
        // Uso dos construtores
        p1 = new Ponto2(1.0,1.0);
        p2 = new Ponto2();
        p2 = lerPonto();
        // Teste dos métodos
        System.out.printf("Distancia entre p1 e p2 = %.2f; %.2f\n",
                                    distancia(p1, p2), p2.dist(p1) );
        p1.printP();
        System.out.println(p2.toString());
        System.out.printf("Distancia de p2 para origem = %f\n",p2.dist(0.0,0.0));

        // teste da herança (Ponto3D extends Ponto2)
        Ponto3D p3 = new Ponto3D();
        p3.setZ(2.33);
        System.out.println(p3.getZ());
        System.out.println(p3.dist(p2));
    }
    public static Ponto2 lerPonto() {
    public static double distancia(Ponto2 a, Ponto2 b) {

}
```