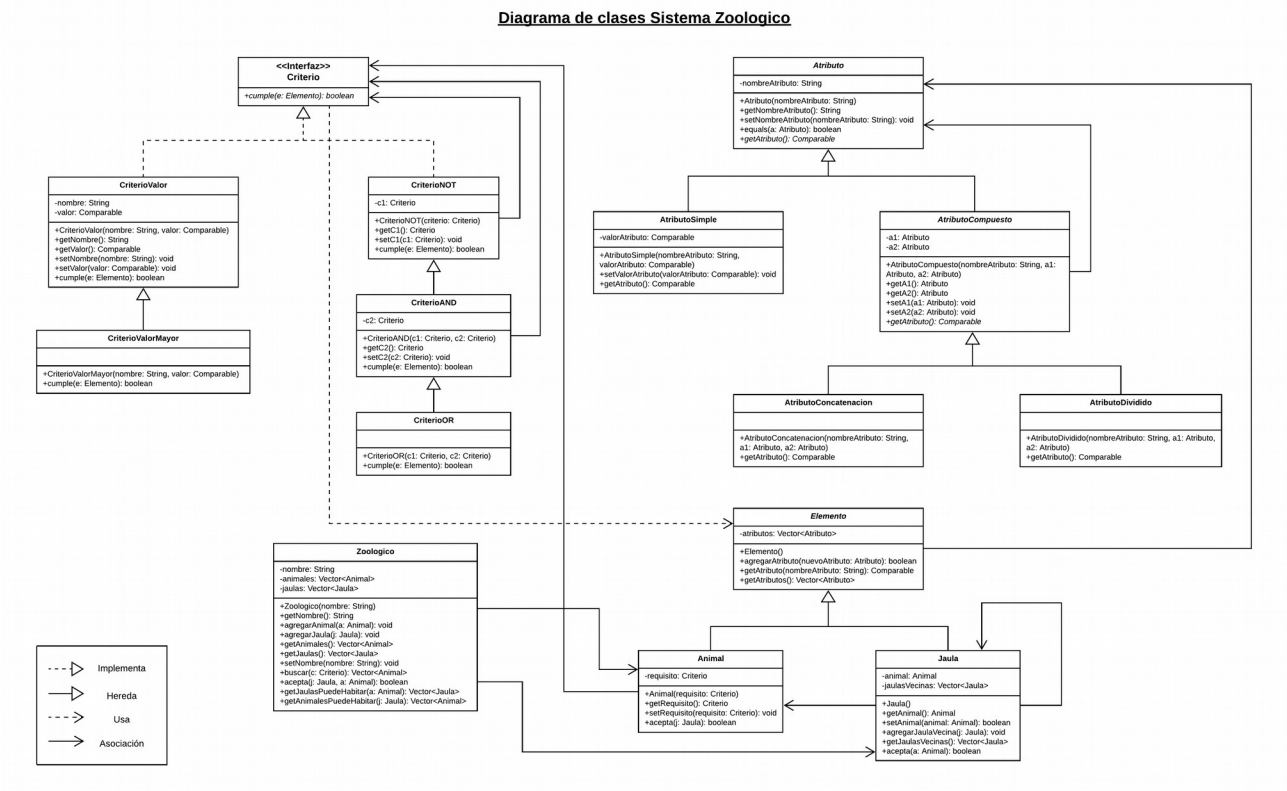
Resolución Sistema del Zoológico

Hechos 16.31: Cree en el Señor Jesucristo, y serás salvo, tú y tu casa.



Palabras, tipos y métodos reservados de Java.
Clases creadas para la resolución.

```
public abstract class Atributo {
    private String nombreAtributo;
    public Atributo(String nombreAtributo){
        this.nombreAtributo = nombreAtributo;
    }
    public String getNombreAtributo() {
        return this.nombreAtributo;
    }
    public void setNombreAtributo(String nombreAtributo) {
        this.nombreAtributo = nombreAtributo;
    }
    public boolean equals(Atributo a) {
        return this.getNombreAtributo().equals(a.getNombreAtributo());
    }
    public abstract Comparable getAtributo();
}

public class AtributoSimple extends Atributo {
    private Comparable valorAtributo;
    public AtributoSimple(String nombreAtributo, Comparable valorAtributo){
        super(nombreAtributo);
        this.valorAtributo = valorAtributo;
```

```
    }
    public void setValorAtributo(Comparable valorAtributo) {
        this.valorAtributo = valorAtributo;
    }
    public Comparable getAtributo() {
        return this.valorAtributo;
    }
}

public abstract class AtributoCompuesto extends Atributo {
    private Atributo a1;
    private Atributo a2;
    public AtributoCompuesto(String nombreAtributo, Atributo a1, Atributo a2){
        super(nombreAtributo);
        this.a1 = a1;
        this.a2 = a2;
    }
    public Atributo getA1() {
        return this.a1;
    }
    public Atributo getA2() {
        return this.a2;
    }
    public void setA1(Atributo a1) {
        this.a1 = a1;
    }
    public void setA2(Atributo a2) {
        this.a2 = a2;
    }
}

public class AtributoConcatenacion extends AtributoCompuesto {
    public AtributoConcatenacion(String nombreAtributo, Atributo a1, Atributo
    a2){
        super(nombreAtributo, a1, a2);
    }
    public Comparable getAtributo() {
        return this.getA1().getAtributo() + " " + this.getA2().getAtributo();
    }
}

public class AtributoDividido extends AtributoConcatenacion {
    public AtributoDividido(String nombreAtributo, Atributo a1, Atributo a2){
        super(nombreAtributo, a1, a2);
    }
    public Comparable getAtributo() {
        return (Double)this.getA1().getAtributo() / (Double)
        this.getA2().getAtributo();
    }
}
```

```
public abstract class Elemento {
    private Vector<Atributo> atributos;
    public Elemento(){
        this.atributos = new Vector<Atributo>();
    }
    public boolean agregarAtributo(Atributo nuevoAtributo){
        if(!this.atributos.contains(nuevoAtributo)){
            this.atributos.add(nuevoAtributo);
            return true;
        }
        return false;
    }
    public Comparable getAtributo(String nombreAtributo){
        for(int i = 0; i < this.atributos.size(); i++){
       if(this.atributos.elementAt(i).getNombreAtributo().equals(nombreAtributo))
        {
                return this.atributos.elementAt(i).getAtributo();
            }
        }
        return null;
    }
    public Vector<Atributo> getAtributos(){
        Vector<Atributo> retorno = new Vector<Atributo>();
        for(int i = 0; i < this.atributos.size(); i++){
            retorno.add(this.atributos.elementAt(i));
        }
        return retorno;
    }
}

public class Animal extends Elemento {
    private Criterio requisito;
    public Animal(Criterio requisito){
        super();
        this.requisito = requisito;
    }
    public Criterio getRequisito() {
        return this.requisito;
    }
    public void setRequisito(Criterio requisito) {
        this.requisito = requisito;
    }
    public boolean acepta(Jaula j){
        return this.requisito.cumple(j);
    }
}

public class Jaula extends Elemento {
    private Animal animal;
    private Vector<Jaula> jaulasVecinas;
    public Jaula(){
        super();
```

```
        this.animal = null;
        this.jaulasVecinas = new Vector<Jaula>();
    }
    public Animal getAnimal() {
        return this.animal;
    }
    public boolean setAnimal(Animal animal) {
        if(this.acepta(animal)){
            this.animal = animal;
            return true;
        }
        return false;
    }
    public void agregarJaulaVecina(Jaula j){
        this.jaulasVecinas.add(j);
    }
    public Vector<Jaula> getJaulasVecinas() {
        Vector<Jaula> retorno = new Vector<Jaula>();
        for(int i = 0; i < this.jaulasVecinas.size(); i++){
            retorno.add(this.jaulasVecinas.elementAt(i));
        }
        return retorno;
    }
    public boolean acepta(Animal a){
        return a.acepta(this);
    }
}

public class Zoologico {
    private String nombre;
    private Vector<Animal> animales;
    private Vector<Jaula> jaulas;
    public Zoologico(String nombre){
        this.nombre = nombre;
        this.animales = new Vector<Animal>();
        this.jaulas = new Vector<Jaula>();
    }
    public String getNombre() {
        return this.nombre;
    }
    public void agregarAnimal(Animal a){
        this.animales.add(a);
    }
    public void agregarJaula(Jaula j){
        this.jaulas.add(j);
    }
    public Vector<Animal> getAnimales(){
        Vector<Animal> retorno = new Vector<Animal>();
        for(int i = 0; i < this.animales.size(); i++){
            retorno.add(this.animales.elementAt(i));
        }
        return retorno;
```

```
    }
    public Vector<Jaula> getJaulas() {
        Vector<Jaula> retorno = new Vector<Jaula>();
        for(int i = 0; i < this.jaulas.size(); i++){
            retorno.add(this.jaulas.elementAt(i));
        }
        return retorno;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public Vector<Animal> buscar(Criterio c){
        Vector<Animal> retorno = new Vector<Animal>();
        for(int i = 0; i < this.animales.size(); i++){
            if(c.cumple(this.animales.elementAt(i))){
                retorno.add(this.animales.elementAt(i));
            }
        }
        return retorno;
    }
    public boolean acepta(Jaula j, Animal a){
        return a.acepta(j);
    }
    public Vector<Jaula> getJaulasPuedeHabitar(Animal a){
        Vector<Jaula> retorno = new Vector<Jaula>();
        for(int i = 0; i < this.jaulas.size(); i++){
            if(this.jaulas.elementAt(i).acepta(a)){
                retorno.add(this.jaulas.elementAt(i));
            }
        }
        return retorno;
    }
    public Vector<Animal> getAnimalesPuedeHabitar(Jaula j){
        Vector<Animal> retorno = new Vector<Animal>();
        for(int i = 0; i < this.animales.size(); i++){
            if(this.animales.elementAt(i).acepta(j)){
                retorno.add(this.animales.elementAt(i));
            }
        }
        return retorno;
    }
}

public interface Criterio {
    boolean cumple(Elemento e);
}

public class CriterioValor implements Criterio {
    private String nombre;
    private Comparable valor;
    public CriterioValor(String nombre, Comparable valor) {
        this.nombre = nombre;
```

```
        this.valor = valor;
    }
    public String getNombre() {
        return this.nombre;
    }
    public Comparable getValor() {
        return this.valor;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public void setValor(Comparable valor) {
        this.valor = valor;
    }
    public boolean cumple(Elemento e) {
        if(e.getAtributo(this.nombre) != null){
            return e.getAtributo(this.nombre).equals(this.valor);
        }
        return false;
    }
}

public class CriterioValorMayor extends CriterioValor {
    public CriterioValorMayor(String nombre, Comparable valor) {
        super(nombre,valor);
    }
    public boolean cumple(Elemento e) {
        if(e.getAtributo(this.getNombre()) != null){
            return (e.getAtributo(this.getNombre()).compareTo(this.getValor()) >
            0);
        }
        return false;
    }
}

public class CriterioNOT implements Criterio {
    private Criterio c1;
    public CriterioNOT(Criterio criterio){
        this.c1 = criterio;
    }
    public Criterio getC1() {
        return this.c1;
    }
    public void setC1(Criterio c1) {
        this.c1 = c1;
    }
    public boolean cumple(Elemento e) {
        return !this.c1.cumple(e);
    }
}
```

```
public class CriterioAND extends CriterioNOT {
    private Criterio c2;
    public CriterioAND(Criterio c1, Criterio c2){
        super(c1);
        this.c2 = c2;
    }
    public Criterio getC2() {
        return this.c2;
    }
    public void setC2(Criterio c2) {
        this.c2 = c2;
    }
    public boolean cumple(Elemento e) {
        return this.getC1().cumple(e) && this.getC2().cumple(e);
    }
}

public class CriterioOR extends CriterioAND {
    public CriterioOR(Criterio c1, Criterio c2){
        super(c1, c2);
    }
    public boolean cumple(Elemento e) {
        return this.getC1().cumple(e) || this.getC2().cumple(e);
    }
}
```