

TRABAJO PRIMER CORTE ARQUITECTURA DE SOFTWARE



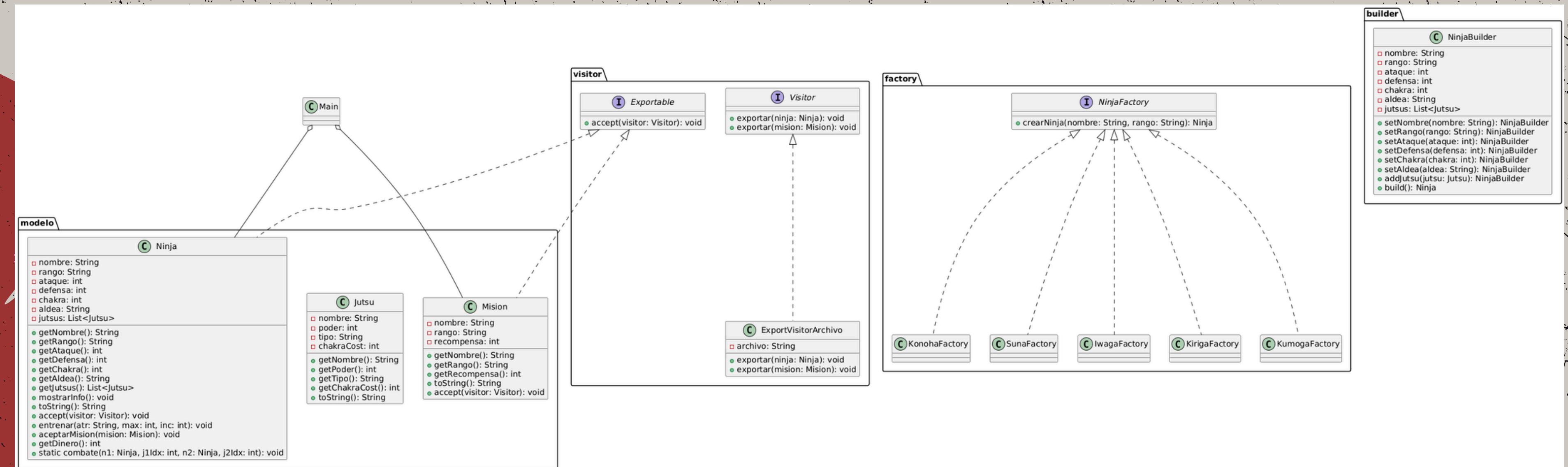
**Andres Julian Alaix Perez
Juan Sebastián Sánchez Rodríguez**



CODIGO FUENTE

https://github.com/juansanchez23/Parcial_ArqSoftware/tree/main/Desktop/Parcial_ArqSoftware/

DIAGRAMA UML





PATRONES DE DISEÑO

FACTORY:

Creacion de Ninjas en
aldeas

```
public interface NinjaFactory {  
    Ninja crearNinja(String nombre, String rango);  
}
```

```
public class KumogaFactory implements NinjaFactory {  
    @Override  
    public Ninja crearNinja(String nombre, String rango) {  
        return new Ninja(nombre, rango, ataque:30, defensa:30, chakra:30, aldea:"Kumoga", Arma:  
            new Jutsu(nombre:"Bunshinjutsu", poder:40, tipo:"Neutral", chakraCost:10), ne
```

PATRONES DE DISEÑO

BUILDER:

Construcción y personalizar atributos y Jutsus del Ninja

```
public class NinjaBuilder {  
    private String nombre;  
    private String rango;  
    private int ataque;  
    private int defensa;  
    private int chakra;  
    private String aldea;  
    private List<Jutsu> jutsus = new ArrayList<>();  
  
    public NinjaBuilder setNombre(String nombre) { this.nombre = nombre; return this; }  
    public NinjaBuilder setRango(String rango) { this.rango = rango; return this; }  
    public NinjaBuilder setAtaque(int ataque) { this.ataque = ataque; return this; }  
    public NinjaBuilder setDefensa(int defensa) { this.defensa = defensa; return this; }  
    public NinjaBuilder setChakra(int chakra) { this.chakra = chakra; return this; }  
    public NinjaBuilder setAldea(String aldea) { this.aldea = aldea; return this; }  
    public NinjaBuilder addJutsu(Jutsu jutsu) { this.jutsus.add(jutsu); return this; }  
  
    public Ninja build() {  
        return new Ninja(nombre, rango, ataque, defensa, chakra, aldea, jutsus);  
    }  
}
```

PATRONES DE DISEÑO

VISITOR:

Separa el algoritmo, de los objetos Ninja y Misión, permitiendo la operación sobre los objetos

```
package visitor;

import modelo.Ninja;
import modelo.Mision;

public interface Visitor {
    void exportar(Ninja ninja);
    void exportar(Mision mision);
}
```

```
import visitor.Visitor;

public class Mision implements Exportable{
    @Override
    public void accept(Visitor visitor) {
        visitor.exportar(this);
    }
    private String nombre;
    private String rango; // D, C, B, A, S
    private int recompensa;

    public Mision(String nombre, String rango, int recompensa) {
        this.nombre = nombre;
        this.rango = rango;
        this.recompensa = recompensa;
    }

    public String getNombre() { return nombre; }
    public String getRango() { return rango; }
    public int getRecompensa() { return recompensa; }

    @Override
    public String toString() {
        return "Mision{" +
            "nombre='" + nombre + '\'' +
            ", rango='" + rango + '\'' +
            ", recompensa=" + recompensa +
            '}';
    }
}
```

