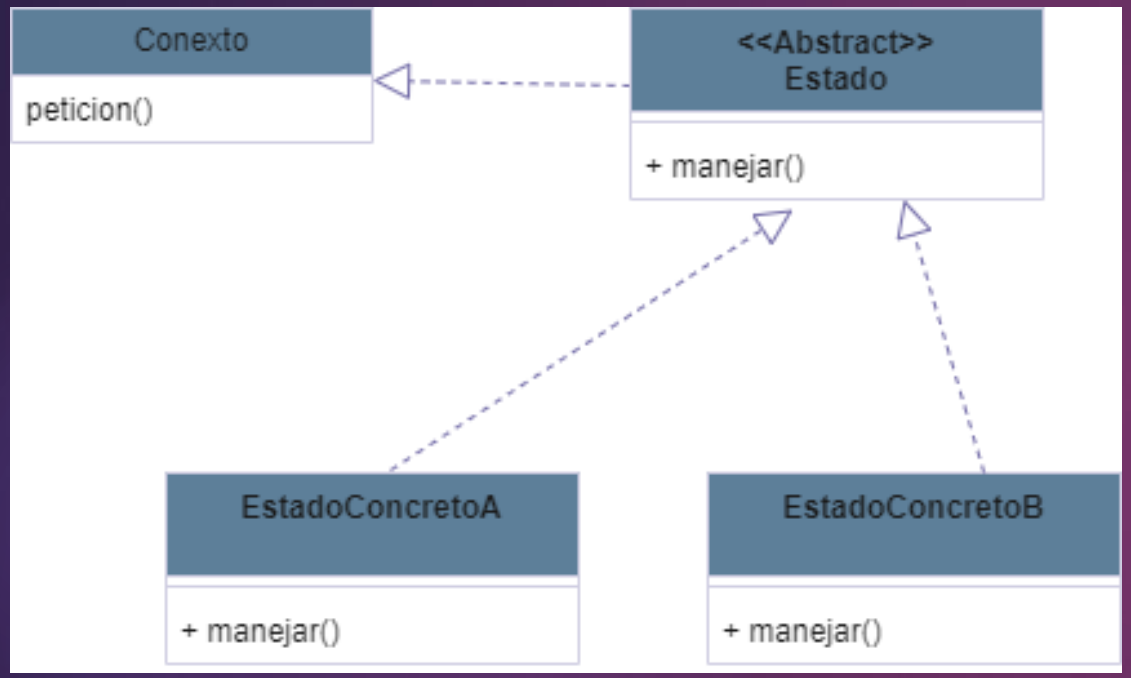
# Patrón de diseño

STATE – GRUPO 1

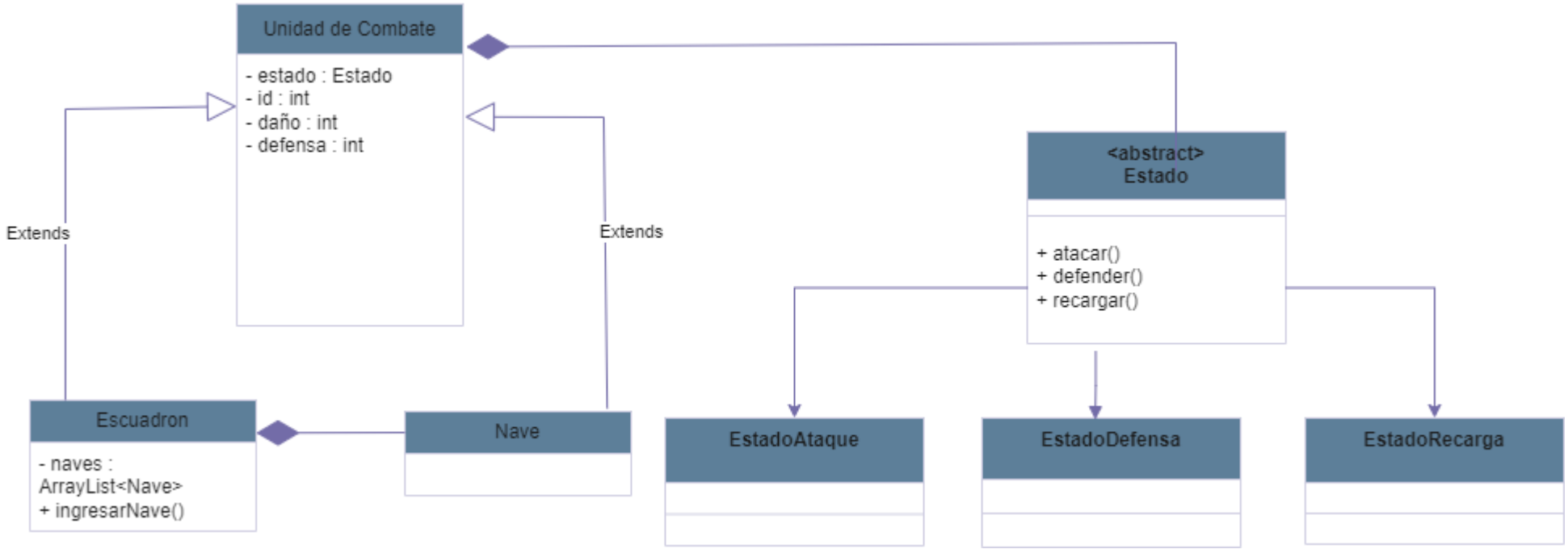
# Integrantes

- Clivio Lucas Ariel
- Guardati Francisco
- Fernandez Rocio Belen
- Peñafiel Huayta Abigail
- Rosano Matias
- Polito Thiago
- Barbaro Leonardo
- Satamaria Luciano Martin Adres
- Didolich Martin
- Zitzmann Yair
- Gonzalez Agustin Elias
- Aragon Rodrigo Ezequiel
- Goncete Franco Javier
- Modola Lautaro




EL PATRÓN STATE SUGIERE QUE CREES NUEVAS CLASES PARA TODOS LOS ESTADOS POSIBLES DE UN OBJETO Y EXTRAIGAS TODOS LOS COMPORTAMIENTOS ESPECÍFICOS DEL ESTADO PARA COLOCARLOS DENTRO DE ESAS CLASES.

EN LUGAR DE IMPLEMENTAR TODOS LOS COMPORTAMIENTOS POR SU CUENTA, EL OBJETO ORIGINAL, LLAMADO *CONTEXTO*, ALMACENA UNA REFERENCIA A UNO DE LOS OBJETOS DE ESTADO QUE REPRESENTA SU ESTADO ACTUAL Y DELEGA TODO EL TRABAJO RELACIONADO CON EL ESTADO A ESE OBJETO.



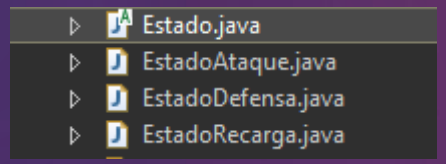

# UML en el contexto

Naves de combate

```java
package clase;

public class UnidadDeCombate {
    protected int id;
    protected int daño = 50;
    protected int defensa = 50;
    protected int energia = 100;
    Estado estado;

    public UnidadDeCombate(int id){
        this.id = id;
        this.estado = new EstadoAtaque(this);
    }

    public void modoDefensa(){
        this.estado = new EstadoDefensa(this);
    }

    public void modoAtaque(){
        this.estado = new EstadoAtaque(this);
    }

    public void modoRecarga(){
        this.estado = new EstadoRecarga(this);
    }

    public String atacar(){
        return this.estado.atacar();
    }
```

- La unidad de Combate es la clase que tiene los métodos a llamar y que cambia los Estados.
- Tiene definido un Atributo Estado el cual determinara como se comportan los diferentes métodos.

```java
package clase;

public abstract class Estado {

    protected UnidadDeCombate unidad;

    Estado(UnidadDeCombate unidad){
        this.unidad = unidad;
    }

    public abstract String atacar();
    public abstract String defender();
    public abstract String recargar();


}
```




- Se declaran tres estados diferentes como clases para cada uno de los Estados : Recarga, Ataque y Defensa.
- Como ventaja esto presenta la posibilidad de definir un comportamiento determinado para un Estado particular que solo esta activo en un momento especifico.
- La desventaja es el mantenimiento que requiere modificar un comportamiento, ya que habría que modificarlo en todas las clases particulares.

# Referencias


- https://refactoring.guru/es/design-patterns/state
- https://khalilstemmler.com/articles/uml-cheatsheet/
- https://refactorizando.com/patron-state-patron-java/
- https://refactoring.guru/es/design-patterns/state/java/example