Ejercicio 6 objetos:
Imágenes satelitales representadas como topografías.

Tipos:

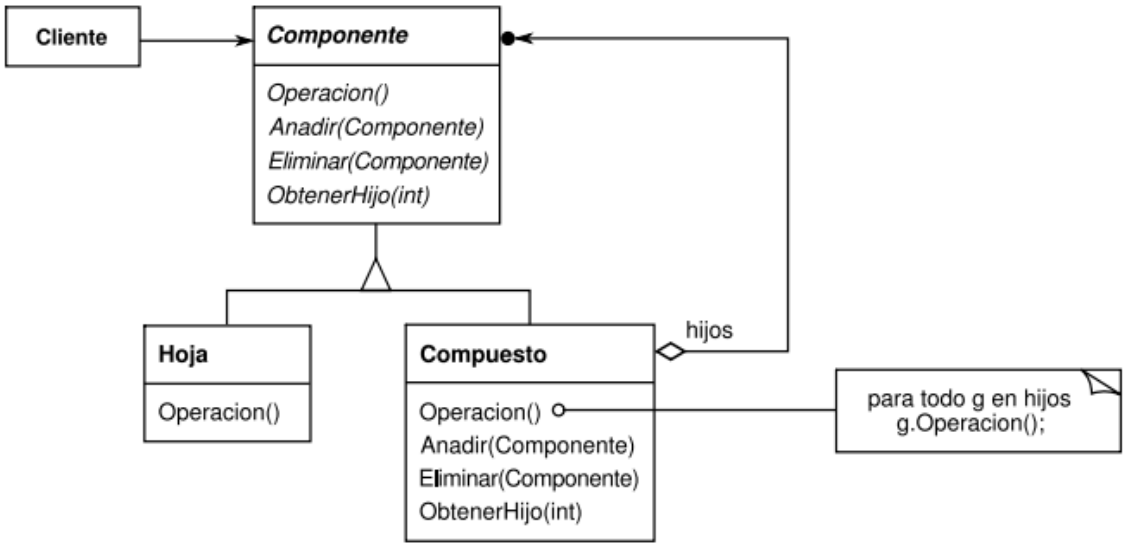
- Agua
- Tierra
- Mixta (varía proporción de agua y tierra).

La topología se puede anidar infinitamente, puede contener topografías parece.
Es decir una mixta contiene otras topografías (4) incluida la mixta, mientras que la de agua o tierra solo son de lo que son.
Hay elementos simples y complejos.

Cabe aclarar, la proporción de agua de las mixtas puede ser la misma, pero la distribución las haría distintas.

Composite, sirve porque no sabemos la complejidad de la cantidad de topografías mixtas anidadas que tenemos, y hay tratar las complejas y simples de forma uniforme y polimórfica. Ejemplo de composite:

ESTRUCTURA



Página 196 de patrones de diseño, se ve el composite.

```
public class Topografia {  }

public class Agua extends Topografia {
   public Agua(){
   // nada particular                    }

public class Tierra extends Topografia {
   public Tierra(){
   // nada particular                    }

public class Mixta extends Topografia {
   List<Topografia> partes
   public Topografia(List <Topografia> topografias){
      partes = topografias;          }
```

Cálculo de proporción.

```
public class Agua extends Topografia {
   public int proporcionDeAgua() {  return 1;  }
}

public class Tierra extends Topografia {
   public int proporcionDeAgua() {  return 0;  }
}

public class Mixta extends Topografia {
   public int proporcionDeAgua() {
      return partes.stream().mapToInt(parte -> parte.proporcionDeAgua()).sum() / 4;
   }
}
```

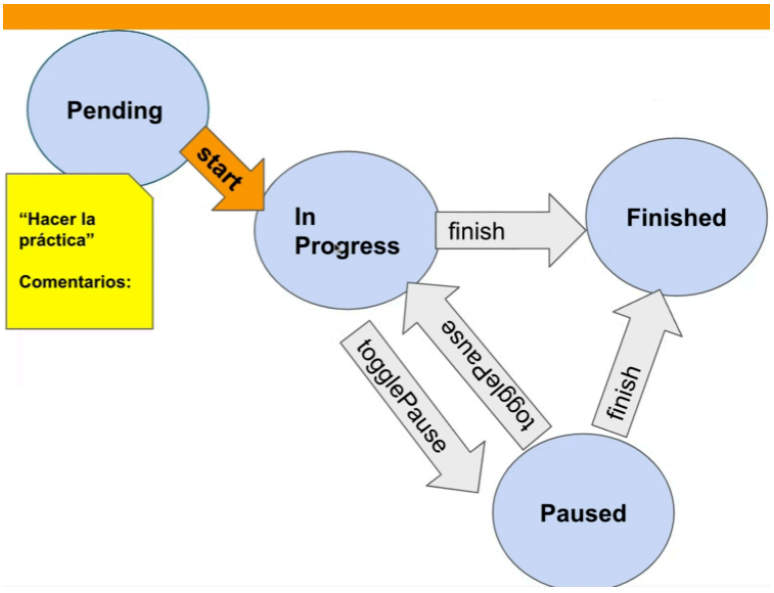
Igualdad:

```
public class Agua extends Topografia {
   public boolean igual(Topografia otraTopografia){
      return otraTopografia.igualAgua(); } }

   public boolean igualAgua(){
      return true; } }

public class Tierra extends Topografia {
   public boolean igualAgua(){
      return false;  }  }

public class Mixta extends Topografia {
   public boolean igualAgua(){
      return false; }  }
```

```
public class Mixta extends Topografia {
   public boolean igual(Topografia otraTopografia){
      return otraTopografia.igualMixta(this) } }

   public boolean igualMixta(TopografiaMixta otraTopografia){
            // ojo… lo siguiente es pseudocódigo!
      return  otraTopografia[1].igual(this.partes[1]) &&
            ...
} }

public class Agua extends Topografia {
   public boolean igualMixta(TopografiaMixta otraTopografia){
      return false  } }
```

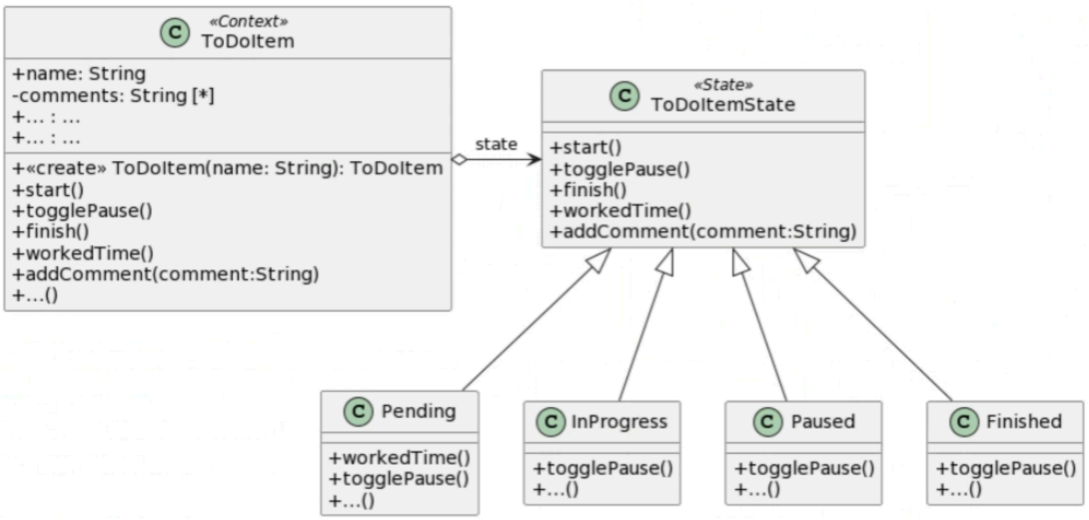
```
public class Mixta extends Topografia {
   public boolean igual(Topografia otraTopografia){
      return otraTopografia.igualMixta(this) } }

   public boolean igualMixta(TopografiaMixta otraTopografia){
            // ojo… lo siguiente es pseudocódigo!
      return  otraTopografia[1].igual(this.partes[1]) &&
            ...
} }

public class Agua extends Topografia {
   public boolean igualMixta(TopografiaMixta otraTopografia){
      return false  } }
```

El cliente no está definido, pueden ser los tests, como tal no es parte del patrón.
Ejercicio 7 objetos: nada.
Ejercicio 8 objetos:
ToDoItem manejador de tareas:



Los globos son estados, y las flechas de transición son métodos.
Además de estos métodos podemos preguntarle datos, como duración de la tarea (desde el comienzo hasta el final si hay o tiempo actual).
Añadir comentarios, si no termino la tarea. Si termino no hay efecto.

State, patrón página 345 patrones de diseño.

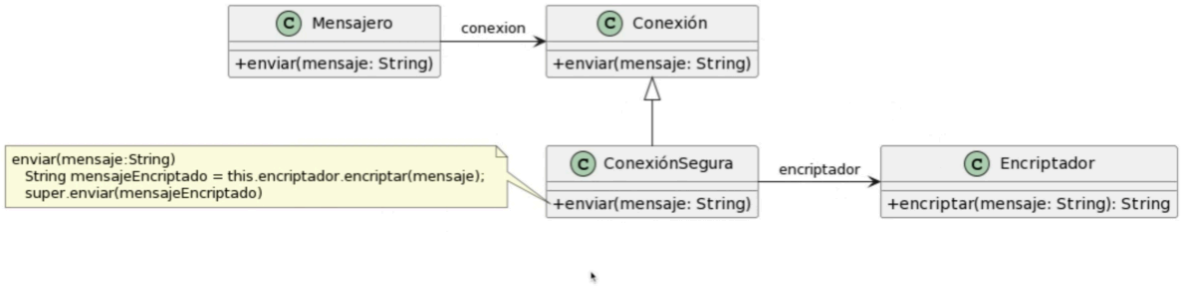


El estado sabe cuando cambiar el estado, pero quién es el que va cambiarlo? Contexto podría ser, el que modifica su estado, pero el estado sigue desciende cuando lo va a hacerlo. (Están necesariamente acopladas).

```
public class ToDoItem {
  private String name;
  private ToDoItemState state;
  . . .

  public void togglePause(){
    state.togglePause(this);
  }

  . . .
}
```

```
public class InProgress extends ToDoItemState {
  . . .

  public void togglePause(ToDoItem context){
    context.setStatus(new Paused());
  }


 . . .


}
```

Ejercicio 10: Encriptador



Tareas:

1. Modifique el diseño para que el objeto Encriptador pueda encriptar mensajes usando los algoritmos Blowfish y RC4, además del ya soportado RSA.
2. Documente mediante un diagrama de clases UML indicando los roles de cada clase.

Podemos usar strategy. Podemos generalizar el encriptador.