# Tarea Independiente 22/09/2025

David Núñez Franco

September 28, 2025

## Inventario de Conceptos Claves

- FP Pattern-matching alias desestructuración en ES67Java
- Noción de patrón. De Parámetro formal a Descripción de estructura del dato
- Eliminación de "boiler-plate" code
- Desestructuración en ES6:
- Uso en funciones
- Uso válido en let (similar en lambdas, const, var).
  - let [first, ...rest] = [1,2,3]; // first es 1 rest =[2, 3]
- Problemas con "fallas silenciosas" (undefined)
- Caso java: record como clase simplificada. Solución a DTO, POJO
- interfaces/clases selladas: restringiendo la 'O' de SOLID.
- switch como expresión patrones y cases
- throw como expresión en switch

## Ejercicio 3

```
3) Represente un tuple (x, y) en Java (coordenadas x y y cada una de cualquier tipo)
Debe funcionar lo siguiente:
record Tuple<X, Y>(X x, Y y){}
var manzana_peso_100 = new Tuple<String, Integer>("manzana", 100)
System.out.println("El peso de la" + manzana_peso_100.x() + "es:" + manzana_peso_100.y
var longitud_latitud_Heredia = new Tuple<Double, Double>(9.99872, -84.11587)
```

### Solución

```java
package demo;

// Tuple generico de dos elementos
record Tuple<X, Y>(X x, Y y) { }

public class TupleDemo {
    public static void main(String[] args) {
        // EJEMPLO 1: String e Integer
        var manzana_peso_100 = new Tuple<String, Integer>("manzana",100);
        System.out.println("El peso de la " + manzana_peso_100.x() + " es: " + manzana_peso_100.y());

        // EJEMPLO 2: Double y Double
        var longitud_latitud_Heredia = new Tuple<Double, Double>(9.99872, -84.11587);
        System.out.println("Coordenadas Heredia: (" + longitud_latitud_Heredia.x() + ", " + longitud_latitud_Heredia.y() + ")");
    }
}
```

Listing 1: class Tuple

# Ejercicio 4

Considere el modelo demo Ast.java. Añada las demás operaciones aritméticas. Evite repe

### Solución

```java
package demo;

import java.util.Map;
import java.util.function.IntBinaryOperator;

// Interfaz sellada, permite solo que se implemente Num, Op
sealed interface Node permits Num, Op, BinaryOp {}
// El compilador automaticamente agrega getters, toString, hasheable
record Num(int value) implements Node { }
record Op(String name) implements Node { }
```

```java
    record BinaryOp(Op oper, Node left, Node right) implements Node {}

    public class Ast {
        // Tabla de operaciones
        private static final Map<String, IntBinaryOperator> OPS = Map.of(
        "+", (a, b) -> a + b,
        "-", (a, b) -> a - b,
        "*", (a, b) -> a * b,
        "/", (a, b) -> a / b,
        "%", (a, b) -> a % b
        );

        // Aqui aplicamos pattern-matching (metodos deben ir dentro de la clase AST, no fuera)
        static int evaluate(Node n) {
            return switch(n) {
                case Num(var value) -> value;
                case Op(var name) -> throw new IllegalStateException("Op not allowed");
                // aplicar operaciones aritmeticas basicas
                case BinaryOp(var oper, var left, var right) -> {
                    var f = OPS.get(oper.name());
                    if (f == null)
                    throw new IllegalStateException("Unknown op: " + oper.name());
                    yield f.applyAsInt(evaluate(left), evaluate(right));
                }
            };
        }

        static public void test_0() {
            System.out.println("***test_0***");

            var ten = new Num(10);
            System.out.println("ten=" + ten + " " + ten.value());

            var twenty = new Num(20);

            var oper = new Op("+");
            System.out.println("oper=" + oper.name());

            var operation = new BinaryOp(oper, ten, twenty);
```

```
50
51                  System.out.println("operation=" + operation);
52
53                  System.out.println("operation=" + evaluate(operation
                        ) + "== 30?");
54
55                  operation = new BinaryOp(new Op("+"), ten, twenty);
56                  System.out.println("10 + 20 = " + evaluate(operation
                        ));
57
58                  operation = new BinaryOp(new Op("-"), twenty, ten);
59                  System.out.println("20 - 10 = " + evaluate(operation
                        ));
60
61                  operation = new BinaryOp(new Op("*"), ten, twenty);
62                  System.out.println("10 * 20 = " + evaluate(operation
                        ));
63
64                  operation = new BinaryOp(new Op("/"), twenty, ten);
65                  System.out.println("20 / 10 = " + evaluate(operation
                        ));
66
67                  operation = new BinaryOp(new Op("%"), twenty, ten);
68                  System.out.println("20 % 10 = " + evaluate(operation
                        ));
69              }
70
71              static public void main(String... args) {
72                  test_0();
73              }
74          }
```

Listing 2: class Ast