ELS - Checkpoint 2

João Félix - up202008867
João Malva - up202006605
Sofia Teixeira - up201806629

Problem Statement

- **Objetives:**
  - Consolidate results from XML, YAML, and JSON files into a single row of a table
  - Export the table as an HTML file for visualization
- **Challenges**
  - Different file formats and data structures
  - Parsing and extracting specific data elements based on format and necessity
  - Allow workflows to be used for repeated tasks with a similar structure

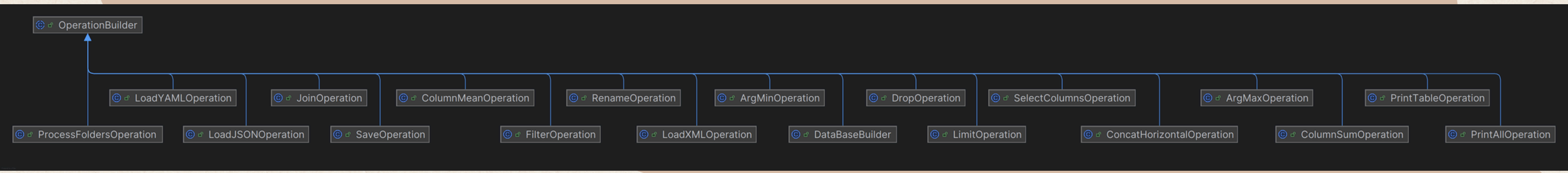
Design Decisions

Discarded Configuration File

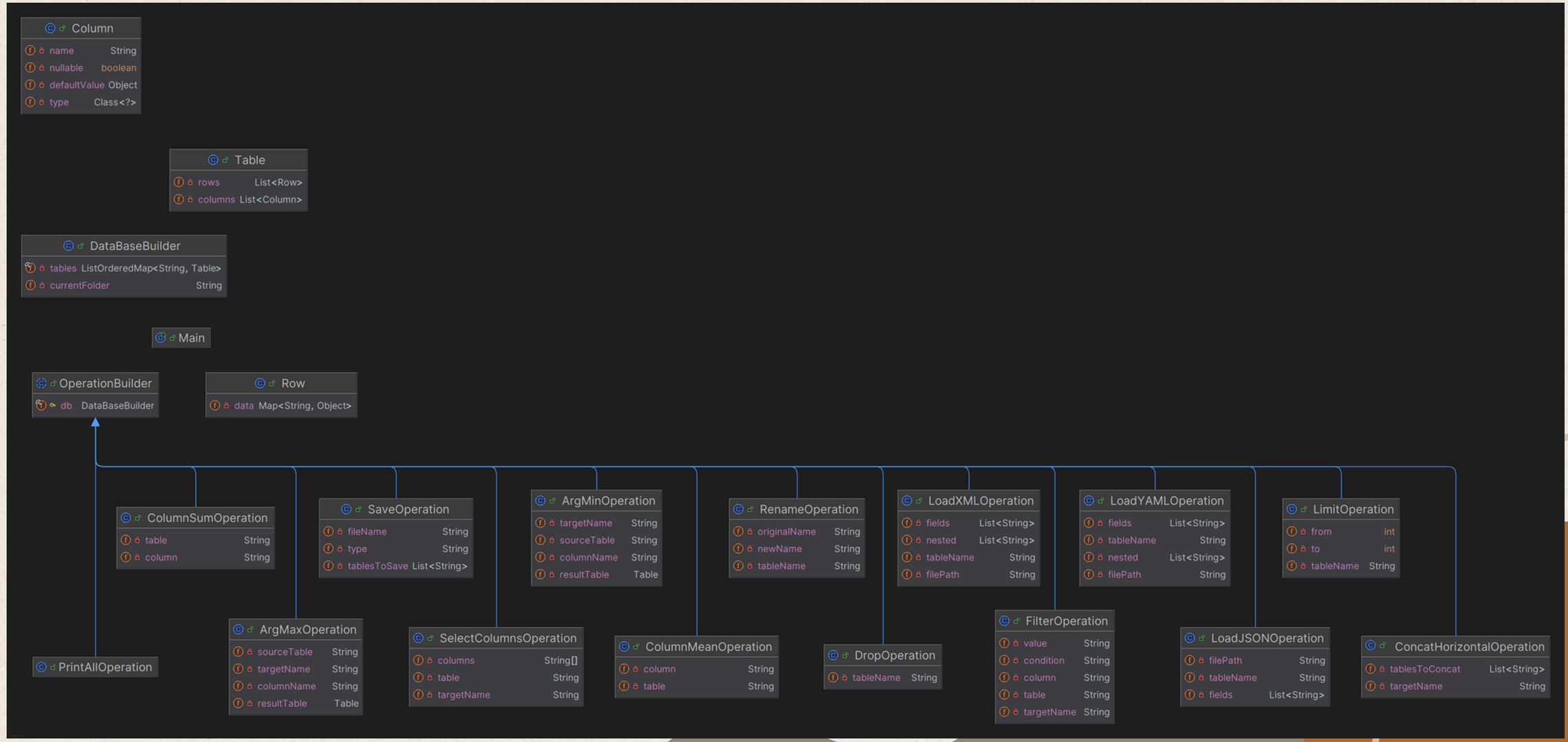
- We discarded the idea of using the configuration file and the built operations, since we restructured the way we represented the data

Builder Pattern

- Design Pattern that lets us produce different types and representations of an object using the same construction code



Semantic Model



Implemented Features

- **Parsing Files**
  - YAML
  - XML
  - JSON

```
.loadJSON() LoadJSONOperation
.from( filePath: "profiling.json")
.into( tableName: "time")
.withAttributes( ...fields: "time%", "seconds", "name")

.loadXML() LoadXMLOperation
.from( filePath: "vitis-report.xml")
.into( tableName: "vitis")
.nestedIn("AreaEstimates", "Resources")

.loadYAML() LoadYAMLOperation
.from( filePath: "decision_tree.yaml")
.into( tableName: "decision_tree")

.loadYAML()
.from( filePath: "decision_tree.yaml")
.into( tableName: "decision_tree2")
.nestedIn("params")
```

Implemented Features

- **Operations:**
  - **Max -** Gives the row with maximum value from a given column
  - **Min -** Gives the row with minimum value from a given column
  - **Drop -** Deletes a table
  - **Select -** Selects specific columns from a given table
  - **Filter -** Filters the values from a column with some given condition
  - **Concatenate -** Concatenates tables on rows with the same index in the table
  - **Rename -** Renames columns
  - **Limit -** Limits the table to specific indexes
  - **Print All -** Prints all the tables
  - **Print Table** - Prints one given table
  - **Save** - Saves in the inputed file the tables that were given

- All operations extend the OperationBuilder abstract class, which contains the methods common to all operations in the DSL and also the methods required to implement them.

Checkpoint 1 vs 2

config.yaml

```yaml
- load:
    files:
      - "assignment1Files/decision_tree_1.yaml"
      - "assignment1Files/decision_tree_2.yaml"
    as: "some_table"

- rename:
      table: "some_table"
      columns:
        - classes_: "Classes"
        - splitter: "Splitter"
        - ccp_alpha: "CPP Alpha"
        - min_samples_split: "Min Samples Split"

- save:
    table: "some_table"
    columns: ["Criterion", "Splitter", "CPP Alpha", "Min Samples Split"]
    out: "resources/configurationFile.csv"
```

```
db
        .processFolders()
        .folders("assignment2Files/run1", "assignment2Files/run2", "assignment2Files/run3") // Specify folders
        .operations(() -> { // Define reusable operation sequence
            db.loadJSON()
                    .from("profiling.json")
                    .into("time")
                    .withAttributes("time%", "seconds", "name")

                    .loadXML()
                    .from("vitis-report.xml")
                    .into("vitis")
                    .nestedIn("AreaEstimates", "Resources")

                    .loadYAML()
                    .from("decision_tree.yaml")
                    .into("decision_tree")

                    .loadYAML()
                    .from("decision_tree.yaml")
                    .into("decision_tree2")
                    .nestedIn("params")

                    .selectMax()
                    .onColumn("time%")
                    .onTable("time")

                    .concactHorizontal()
                    .toTable("final")
                    .onTables("decision_tree", "decision_tree2", "time", "vitis")

                    .end();
        })
        .printTable("final")
        .end();
```

Demo

Questions?