

CS562 Final Project

—final war

ImaginDragons

Yujie Du(10372723)
Chuanhui Zhang(10387654)

what we're about to present

-Project Overview

-Project Structure & User Interface

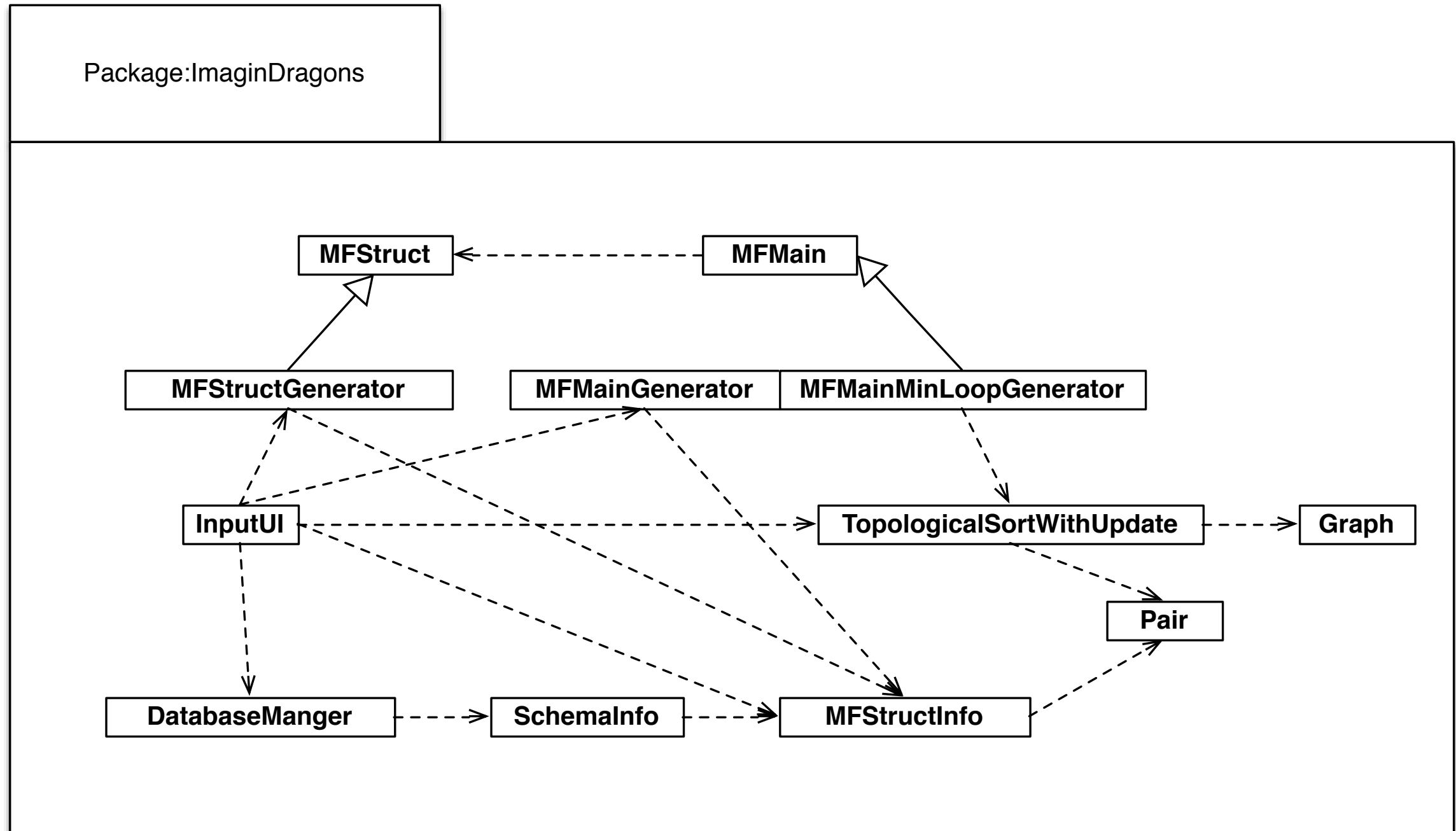
-HighLights

-Limitations and Future Work

Project Overview

1. **Problem:** Build a query processing engine for the MF queries because traditional optimizers lead to poor performance.
2. **Solution:** The project introduces the Φ operator, and scan each tuple for each grouping attributes instead of using joins
3. **Technology We Are Using**
 - Language: Java
 - DBMS: PostgreSQL
 - Library: PostgreSQL Java Library
4. **HighLights**
 - Check if select clause is valid? “”: selectAttr
 - Generate Minimum Loop using revised topological sort
 - Can check whether a particular condition is valid
(ex. if avg_1_quant is not created for some group, then this group should not ignored for 2.quant > avg_1_quant)

Project Structure



InputUI -- Yujie
DatabaseManager dm List of JFrame variables ...
initUI() test() //initialized test data initPanelNW() initPanelNE() initPanelSW() initPanelSE() addListeners() stageChanged() setTextAreaSE() main() //Entry of the whole project

Graph -- Yujie
HashSet<Integer>[] adj HashSet<Integer> end HashSet<Integer> allSet
addEdge() adj() start()

DatabaseManager -- Chuanhui
String usr String pwd String url
connect() retrieve() importData() getters() ...

Schemainfo -- Yujie
table_name TypeMap
getTableName() addAttribute() getValue() getSchema() schemainit() getSchemaInfo() updateType()

MFStructInfo -- Yujie&Chuanhui
SchemaInfo si //Instance of SchemaInfo ArrayList<String> selAttrList //Select attributes int numOfGv //Number of Grouping variables ArrayList<String> gaList //Grouping attributes ArrayList<String> afList //Aggregate functions ArrayList<String> condList //Conditions in "such that" Clause Map<String, String> gaNameToTypeMap Map<String, String> afNameToTypeMap //ex: Map<loop, HashMap<aggregate, Stack<String>>> Map<String, HashMap<String, Stack<String>>> afCoreMap //ex: Map<selAttr,false-left/true-right> Map<String, Boolean> selAlignMap //Map<selAttr, parsedSelAttrs> Map<String, ArrayList<String>> selAttrMap //Identifying whether a type is numeric type HashSet<String> numTypeSet //ex: Map<loop, ArrayList<afs>> //Identifying whether dependency condition is valid Map<Integer, ArrayList<String>> condMap //Paris of condition dependencies for //TopologicalSortWithUpdate ArrayList<Pair> condDepPairList
analyzeGa() analyzeAf() analyzeCond() analyzeSelAttr() getters...

TopologicalSortWithUpdate -- Yujie
Graph g int[] step int num boolean[] marked Map<Integer, HashSet<Integer>> sehdule
run() depthFirstOrder() dfs() update()

MFMainMinLoopGenerator	MFMainGenerator -- Chuanhui
String path String code String StaticStringsToOutput ...	String path String code String StaticStringsToOutput ...
MFMainMinLoopGenerator() ScanMinLoopInfo() PrintOutInfo() run()	MFMainGenerator() ScanLoopInfo() PrintOutInfo() run()

TopologicalSortWithUpdate -- Yujie
Graph g int[] step int num boolean[] marked Map<Integer, HashSet<Integer>> sehdule
run() depthFirstOrder() dfs() update()

User Interface

The screenshot shows a window titled "Simple ESQL Application" with a light gray background. It is divided into several sections:

- Top Left:** Input fields for "url:" (jdbc:postgresql://localhost:5), "username:" (cz), and "password:" (melody2211). Below these are "connect" and "clear" buttons.
- Top Right:** A panel for query configuration with labels "SELECT ATTRIBUTE(S):", "NUMBER OF GROUPING VARIA...", "GROUPING ATTRIBUTES(V):", "F-VECT([F]):", and "SELECT CONDITION-VECT([δ]):". It contains text boxes with values like "cust, prod, avg_0_quant, sum", "3", "cust, prod", "avg_0_quant, sum_1_quant, a", and "quant > avg_2_quant, quant >". There are checkboxes for "Min Loop" and "Output Here", and "submit" and "clear" buttons.
- Middle Left:** A "Waiting for input" label above "stage1" and "stage2" buttons.
- Middle Right:** A "Waiting for input" label above a large empty rectangular area.
- Bottom Left:** Input fields for "tableName:" (sales) and "schema:" (create table sales (cust varchar). Below are "execute" and "clear" buttons.
- Bottom Right:** "Import Data" and "Import ESQL" buttons.
- Status:** A label at the bottom left.

Stage 1:

- **NW panel:** input db info (uri, usr, pwd). Other panels will unlock if successfully connect to the db.

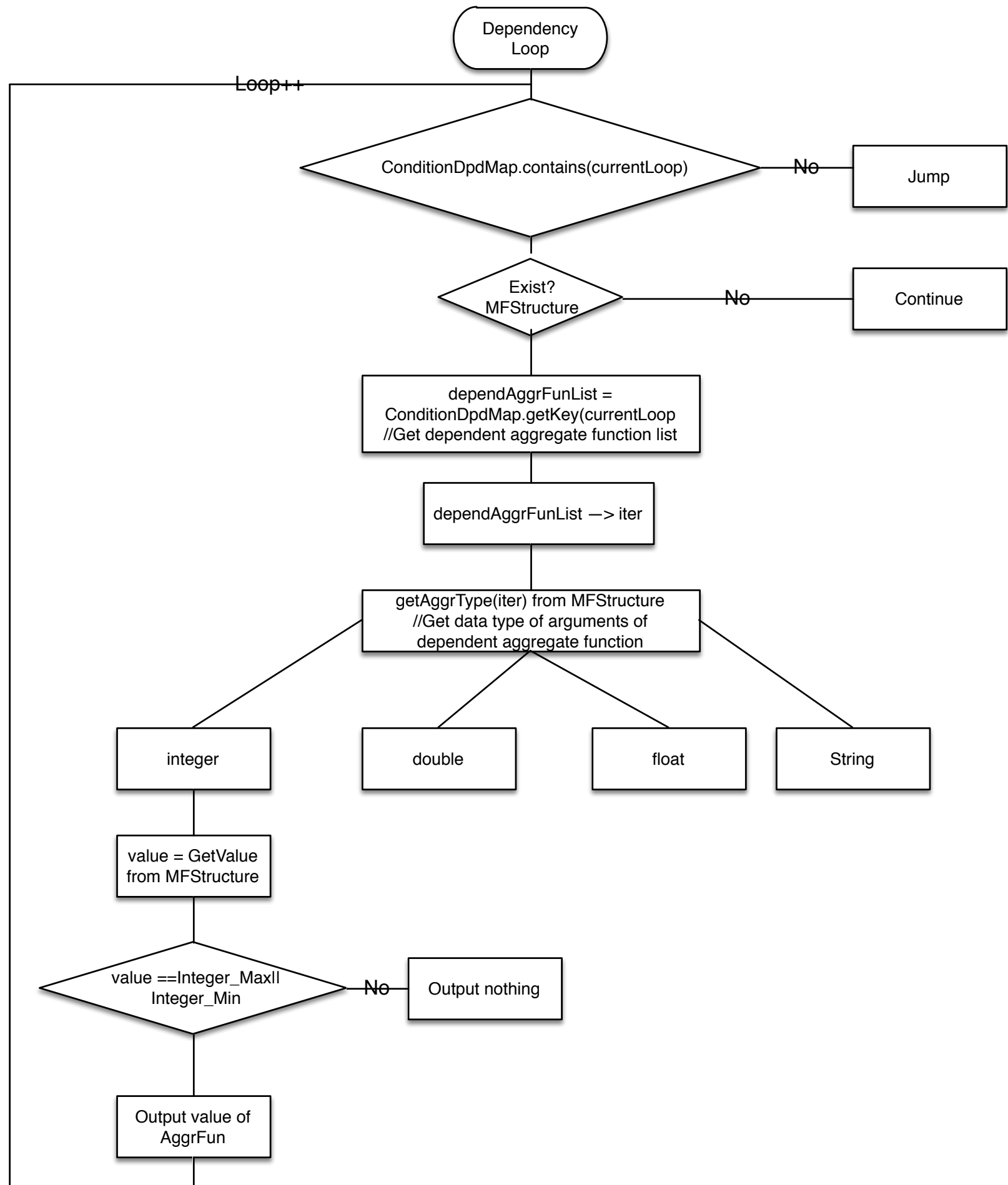
Stage 2:

- **SW panel:** db operations (create schema, import data/ESQL). You can unlock the NW panel by stage buttons.
- **NE panel:** input five Φ arguments to output MFMain.java and MFStruct.java. You can select MinLoop option.
- **SE panel:** output schema info after schema is created

HighLights

Check if certain condition is valid:

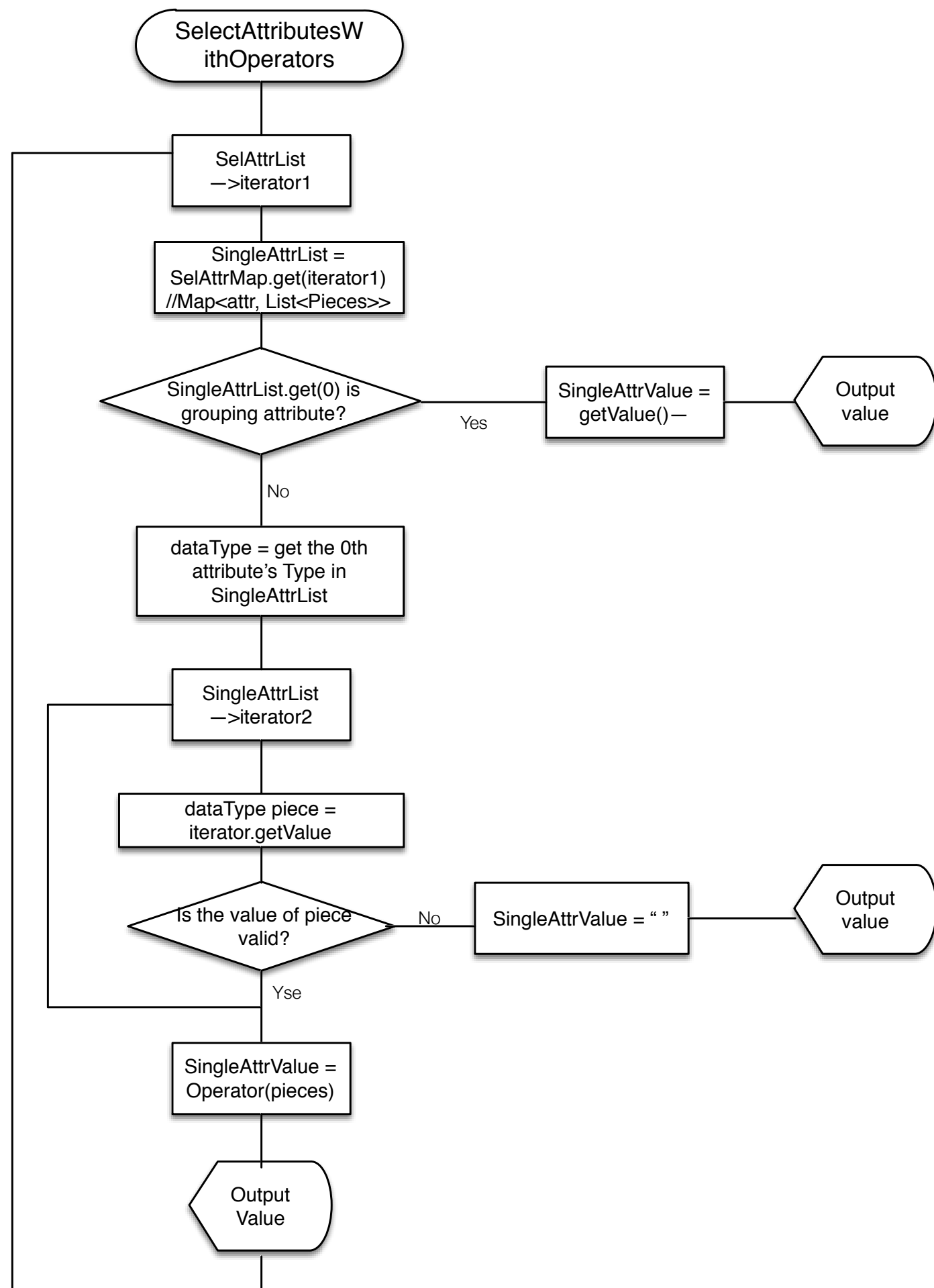
if avg_1_quant is not created for some group, then this group should not process 2.quant > avg_1_quant



HighLights(Cont'd)

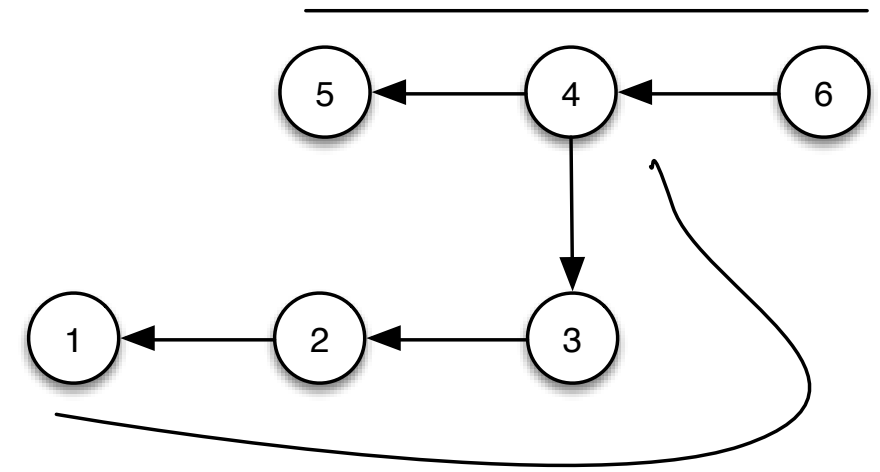
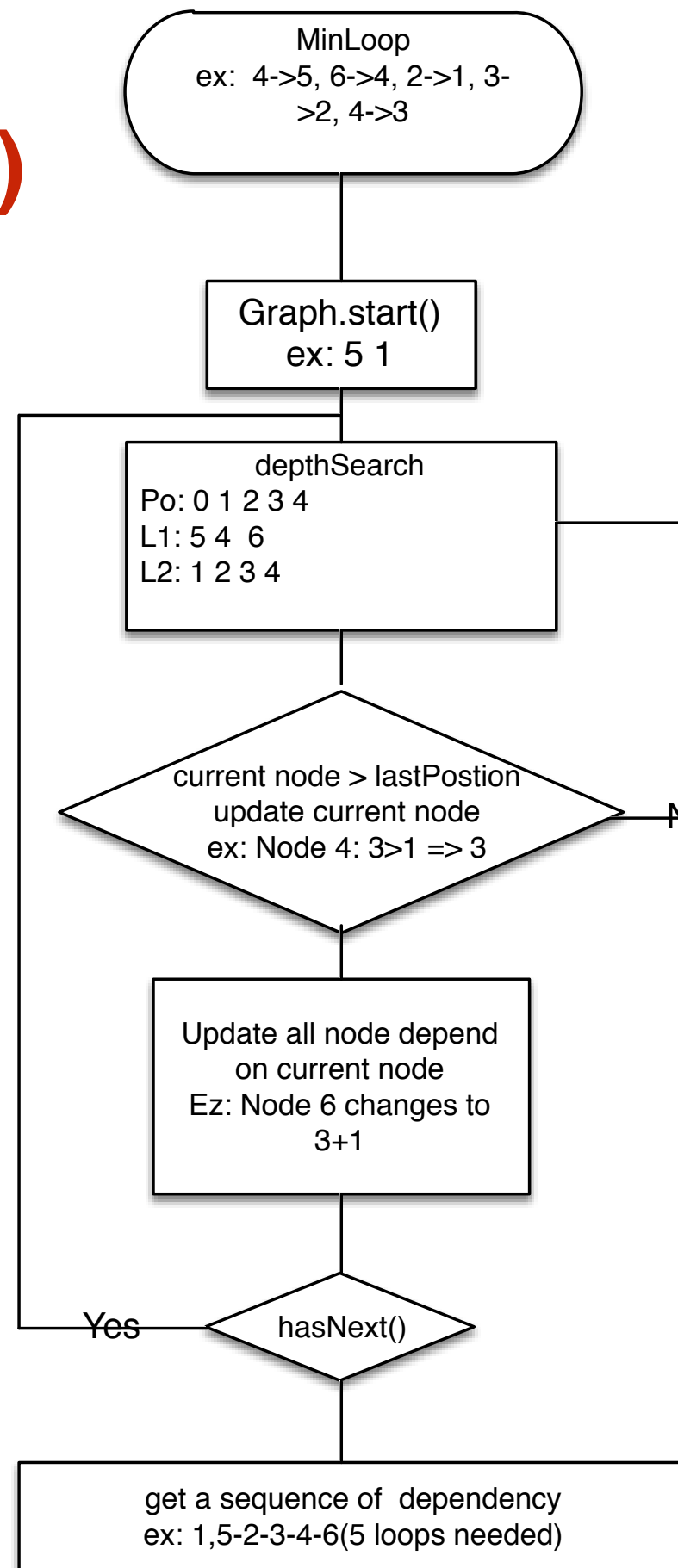
Check if Select Clause is valid especially when there are operators in it:

(Ex. if max_1_quant is not valid for a certain MFStruct Object, then max_1_quant + max_2_quant should output “”)



HighLights(Cont'd)

Generate Minimum Loop using revised topological sort



Limitation

1. Select Clause only include “+”, “-“, “*“, “/“
2. Due to tech difficulty, User Interface cannot print the output by MFMain.java
3. No error checking for ESQL syntax
3. As a two-member team, our project only focuses on MF Queries

Future Work

1. ESQL Parser (Ongoing...)
2. User Interface prints the output by MFMain.java
3. Support for EMF Queries