**Parser - PKB API**

*public class Follow*

The class stores programs in tables efficient to respond to Follow queries, providing query evaluator methods to obtain answers. It creates data structures vector<int> followsTable, vector<int> followedByTable, vector<vector<int>> followsStarTable and vector<vector<int>> followedByStarTable on initialisation.

*void setFollows(int s1, int s2)*

Return type: void

Creates a vector<int> of size s1+1 called followsTable and stores s2 at position s1. If the size of followsTable is more than or equals to s1, an invalid argument exception is thrown.

*void setFollowedBy(int s1, int s2)*

Return type: void

Creates a vector<int> of size s1+1 called followsTable and stores s2 at position s1. If the size of followsTable is more than or equals to s1, an invalid argument exception is thrown.

*void setFollowsStar(int s1, int s2)*

Return type: void

Appends s2 to vector<int> at position s1 in followsStarTable

*void setFollowedByStar(int s1, int s2)*

Return type: void

Appends s2 to vector<int> at position s1 in followsStarTable

*vector<int>getFollows(int statementNum)*

Return type: vector<int>

Takes in argument passed in and returns vector<int> type at that position in followsTable

*vector<int>getFollowedBy(int statementNum)*

Return type: vector<int>

Takes in argument passed in and returns vector<int> type at that position in followedByTable

*vector<int>getFollowsStar(int statementNum)*

Return type: vector<int>

Takes in argument passed in and returns vector<int> type at that position in followsStarTable

*vector<int> getFollowedByStar(int statementNum)*

Return type: vector<int>

Takes in argument passed in and returns vector<int> type at that position in followedByTable

*public class Parent*

The class stores programs in tables efficient to respond to Parent queries, providing query evaluator methods to obtain answers. It creates data structures vector<int> parentTable, vector<int> childTable, vector<vector<int>> parentStarTable and vector<vector<int>> childStarTable on initialisation.

*void setParent(int s1, int s2)*

Return type: void

Creates a vector<int> of size s1+1 called parentTable and stores s2 at position s1. If the size of parentTable is more than or equals to s1, an invalid argument exception is thrown.

*void setChild(int s1, int s2)*

Return type: void

Creates a vector<int> of size s1+1 called childTable and stores s2 at position s1. If the size of childTable is more than or equals to s1, an invalid argument exception is thrown.

*void setParentStar(int s1, int s2)*

Return type: void

Appends s2 to vector<int> at position s1 in parentStarTable

*void setchildStar(int s1, int s2)*

Return type: void

Appends s2 to vector<int> at position s1 in childStarTable

*vector<int>getParent(int statementNum)*

Return type: vector<int>

Takes in argument passed in and returns vector<int> type at that position in parentTable

*vector<int>getChild(int statementNum)*

Return type: vector<int>

Takes in argument passed in and returns vector<int> type at that position in childTable

*vector<int>getParentStar(int statementNum)*

Return type: vector<int>

Takes in argument passed in and returns vector<int> type at that position in parentStarTable

*vector<int> getChildStar(int statementNum)*

Return type: vector<int>

Takes in argument passed in and returns vector<int> type at that position in childStarTable

*public class Modify*

The class stores programs in tables efficient to respond to Parent queries, providing query evaluator methods to obtain answers. It creates data structures vector<string> varIndexTable and vector<string> procIndexTable, for indexed and efficient access to string variables, modifiesTable, modifiedByTable, procModifiesTable and procModifiedByTable on initialisation.

*int varIndexer(string varName)*

Return type: int

Function searches for passed in argument in varIndexTable, and if found returns its index position in the table, otherwise, adds it to the table and returns its position.

*int procIndexer(string procName)*

Return type: int

Function searches for passed in argument in procIndexTable, and if found returns its index position in the table, otherwise, adds it to the table and returns its position.

*void setModifies(int s, string varName) {*

Return type: void  
Creates a vector<string> of size s+1 called modifiesTable and stores varName at position s. If the size of modifiesTable is more than or equals to s, an invalid argument exception is thrown.

*void setModifiedBy(string varName, int s)*

Return type: void

Calls function to find index of the argument varName from varIndexTable. Function goes to the index position in modifiedByTable; an invalid argument is thrown if s is found at location, otherwise, s is appended to vector<int> at location.

*void setProcModifies(string procName, string varName)*

Return type: void

Calls function to find index of the argument procName from procIndexTable. Function goes to the index position in procModifiesTable; varName is appended to vector<string> at location if not already existent.

*void setProcModifiedBy(string procName, string varName)*

Return type: void

Calls function to find index of the argument procName from procIndexTable. Function goes to the index position in procModifiedByTable; varName is appended to vector<string> at location if not already existent.

*vector<string>getModifies(int s)*

Return type: vector<string>

Takes in argument passed in and returns vector<string> type at that position in modifiesTable

*vector<int> Modify::getModifiedBy(string varName) {*

Return type: vector<int>

Takes in argument passed in, converts it to its index in the varIndexTable, and returns the vector<int> in the modifiedByTable at this position.

*vector<string>getProcModifies(string procName)*

Return type: vector<string>

Takes in argument passed in, converts it to its index in the procIndexTable, and returns the vector<string> in the procModifiesTable at this position .

*vector<string>getProcModifiedBy(string procName)*

Return type: vector<string>

Takes in argument passed in, converts it to its index in the procIndexTable, and returns the vector<string> in the procModifiesTable at this position.

**QueryEval- PQL API**

*public class QueryEval*

QueryEval(QE) takes in the PKB and PQL as its input to generate the result for each individual query statement. It evaluates each clause (select, such that and pattern) individually. Once the evaluation of each clause is finished, it will take the intersection of all three clauses. Should any clause be unsatisfied, the class will return an empty vector<string>. Otherwise, the result will be contained inside the vector<string>. Statement numbers are treated as a string in the final answer.

vector<string> runQueryEval();

Return type:vector<string>

After QE has been initialised, if it is the first time running the QE then QE.runQueryEval() will produce the result of the query. For subsequent queries, the function setQueryStatement should be called first to update the QE with the new query.

void setQueryStatement(QueryStatement qs)

Return type:void

updates the QE with the most recent query to be evaluated.