SmarTest

Dan Walker (Project Manager)
Harpreet Singh (Language Guru)
Aiman Najjar (System Architect)
Parth Solanki (System Integrator)
Kshitij Raj Dogra (System Tester)

Introduction

Dan Walker (Project Manager)

Computerized Adaptive Tests

- Questions given to test-taker vary based on performance
- Precise evaluation of performance
  - Weiss, D. J., & Kingsbury, G. G. (1984).
- Can we leverage these strategies to improve the way we evaluate?

What is the SmarTest language?

- Declarative
- Easy to use
- Powerful
- Reusable

SmarTest Code

Harpreet Singh (Language Guru)

Example 1

Day before final exam

Professor just thought of an easy yet tricky question that he wants to append to list of questions for the final exam (with a wicked smile)

```
void main()
        {
        %%%Loads Math-Tricky questions from repository “smartest” into set
         math_tricky%%%
        set math_tricky = load("jdbc:mysql://localhost:8889/smartest", "root", "root", "Math-Tricky");

        %% Defines new question q
        question q = $"Math-Tricky":"What is (-4) X(-4)/(-2)?" ["-8":5, "-4":-4, "-1":-5]$;

        int len1 = len(math_tricky);
        print("Length of the set before adding a question: ");
        printInteger(len1);

        %% adding  tricky question to the set math_tricky
        math_tricky << q;

        len1=len(math_tricky);
        print("Length of the set after adding the question to it: ");
        printInteger(len1);

        %%saving the set  math_tricky into the database
         save("jdbc:mysql://localhost:8889/smartest", "root", "root", math_tricky);
        }
```

Example 2

Final Exam Day

Professor wants to ask easy questions till the score is 50 and tricky questions after that

```
void main()
{
%% Loads Math-Easy questions from repository “smartest” into set
%% math_easy

set math_easy = load("jdbc:mysql://localhost:8889/smartest", "root", "root", "Math-Easy");

%% similar to the previous statement

set math_tricky = load("jdbc:mysql://localhost:8889/smartest", "root", "root", "Math-Tricky");
int i = 0;
int total_score = 0;
int len1 = len(math_easy);
int len2 = len(math_tricky);
int count;
if(len1<len2)
{
        count=len1;
}
```

```
        else
        {
                count=len2;
        }
        loop while (i < count)
        {
                if (total_score < 50)
                {
                        total_score = total_score + askQuestion(math_easy);
                }
                else
                {
                        total_score = total_score + askQuestion(math_tricky);
                                }
                i = i + 1;
        }
}
```

Architecture

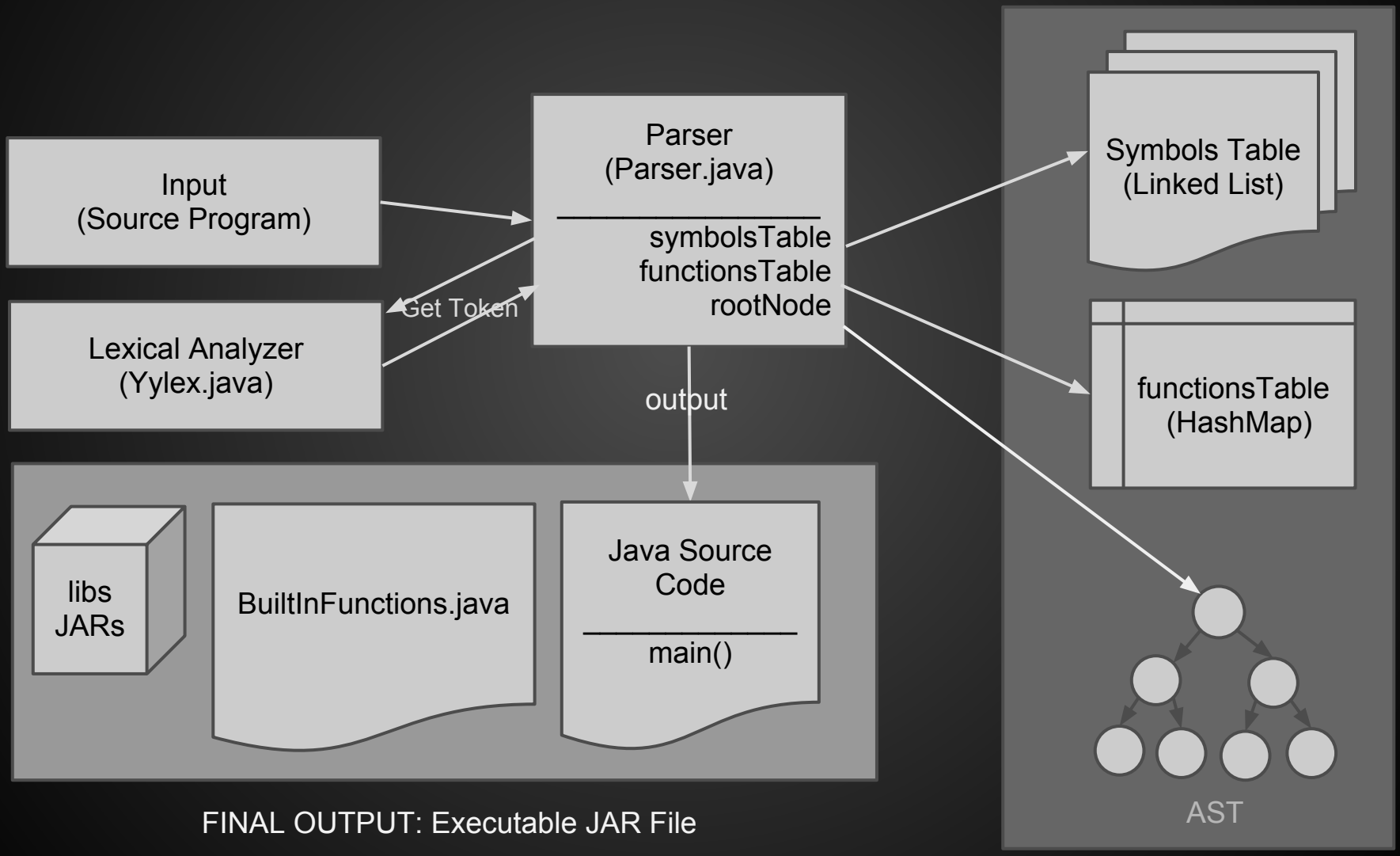
Aiman Najjar (System Architect)

Design Principles

- Self-Contained (One JAR File)
- Portability (Leveraged the power of Java!)
- Divide and Conquer (Java Polymorphism)
- Error Reporting in Mind

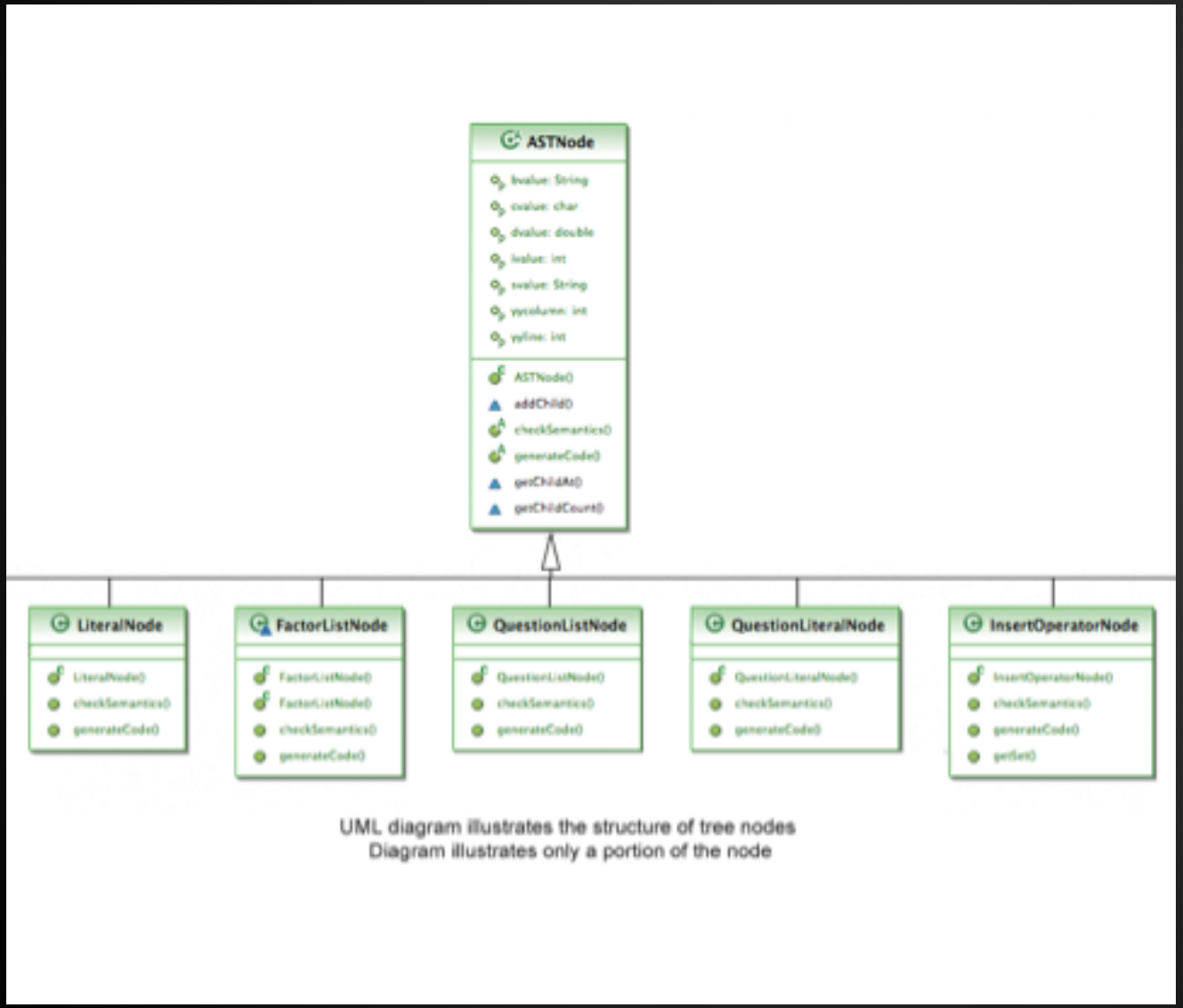
(Line numbers maintained throughout parsing phase and in symbols table)

Overview of Components



FINAL OUTPUT: Executable JAR File

AST Representation
Each Node Type => Java Object (Polymorphism)



UML diagram illustrates the structure of tree nodes
Diagram illustrates only a portion of the node

- *Each Node Type has a Java object*
- *All nodes inherit abstract class ASTNode.java*

Each Node must implement two methods:

checkSemantics
generateCode

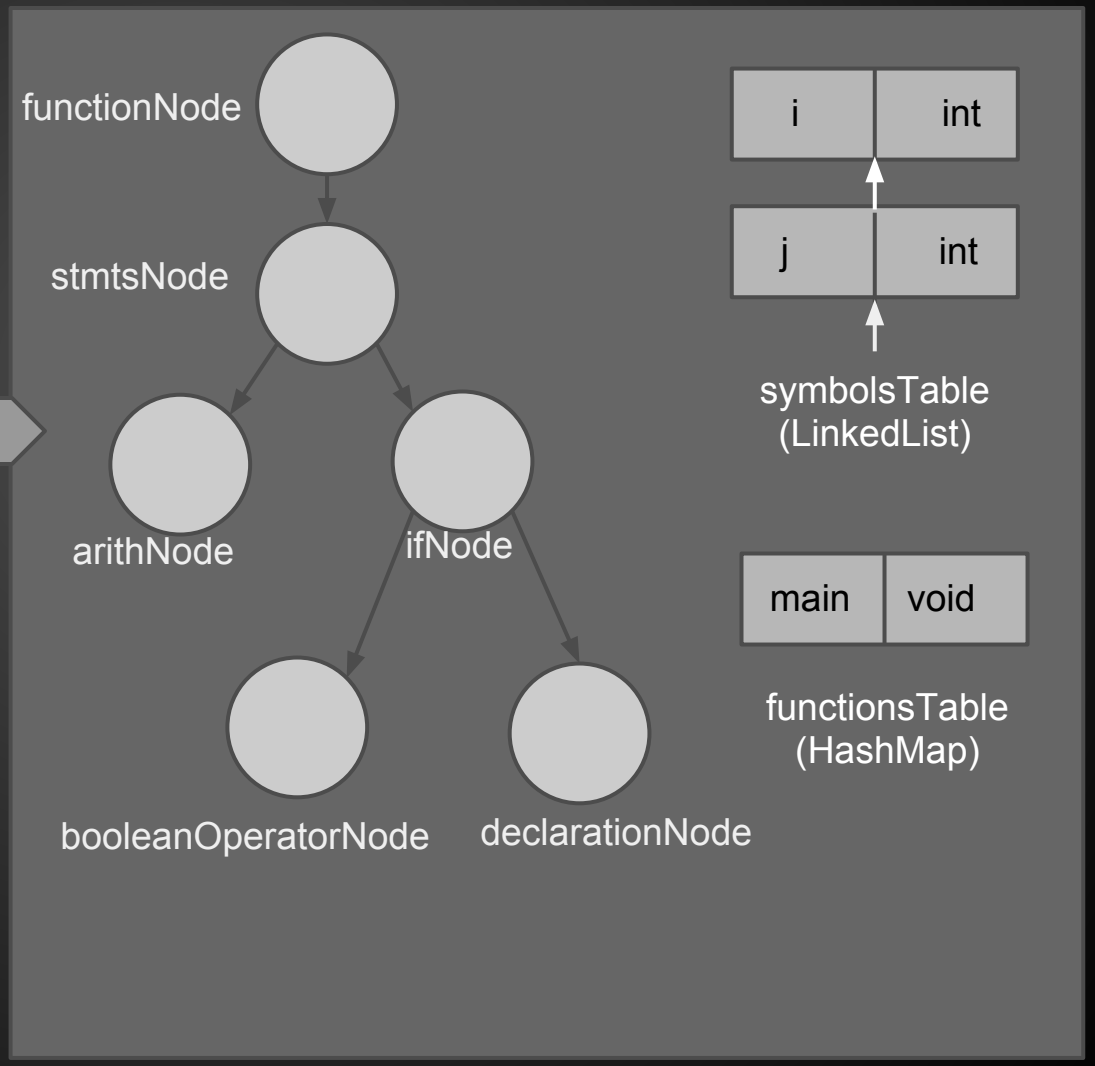
Each team member implements a set of nodes (divide and conquer)

Database

- MySQL
- Managed Completely by SmarTest (BuiltInFunctions.java)
- Abstracted using built-in save/load functions. (User cannot execute SQL queries)
- Java Library already included in output JAR (no pre-installation required)

Example: During Parser Runtime

```
void main()
{
    int i = 3;
    if (i > 1)
    {
        int j = i + 1;
    }
}
```



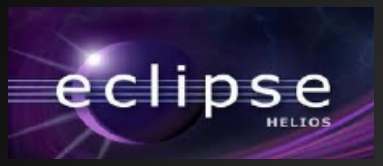
System Integration

Parth Solanki (System Integrator)

Run-Time Environment

- *Operating System*
  - *Ubuntu 11.04 or above*
- *Database*
  - *MySql Server 5.1.61*
  - *Table to store questions*
- *Java 1.6 or above*
- *Scripts to compile and run SmarTest program*
  - *stlc*
  - *stl*

Development tools and Utilities



Testing

Kshitij Raj (System Tester)

Test Plan

Black Box Testing

White Box Testing

Regression Testing

Integration Testing

Testing Methodology

1. Black Box Testing
    - Tested for use of the default values
    - Tested for a combination of variables types
    - For a given set of questions checked the resulting score

2. White Box Testing
    - Performed syntactic and semantic error checking
    - Tested loops and conditional statements
    - Exception and error handling in the system

Testing Methodology

3. Regression Testing
    - Built a script for running the suite of 100+ test cases at once
    - Tested all files when any modifications were made to ensure backward compatibility

4. Integration Testing
    - Tested with the MySQL server as a standalone component using stubs
    - Performed analysis after integrating it with the rest of the system
    - Tested with a number of categories in the database

Live Demo

Conclusions

-
- Things can look easy at the beginning and a team may commit a deliverable unrealistic to produce in the given time frame. In-depth and far sighted analysis is a must in order to avert this.
- Regular meetings with the Instructor and TAs helps focus the effort on the relevant deliverable.
- Referring to past projects is helpful in order to be able to design something that at that point in the course we did not know how to implement.