Your Life After TT!

Conclusions
- Code Early
- Code Often
- Code Together

Time & Task
Zhengji: The PM
Peter: The Guru
Jason: The Architect
Michelle: The Tester
Akhila: The Integrator
Time & Task

Zheng: The PM
Peter: The Guru
Jason: The Architect
Michelle: The Tester
Akhila: The Integrator
Can We meet today?

Jason: No, I’m at work!

Michelle: Nope, I have classes.

Zheng: I am free in the afternoon.

Peter: I am free all day.

Akshila: No, I am meeting with my research group
What is TT?

- DSL for manipulating Dates, Tasks and Calendars
- Easy to learn for experienced and novice programmers
IT Properties

- Imperative
- Simple and Familiar
- Readable

```python
import "<std>";

//FAMILIAR
main() {
    //SIMPLE
    Date a = 2013.05.16;
    Date b = a + 1 day;
    //output 2013.05.17
    print(b);

    //READABLE
    Calendar myCalendar;
    every Task t in myCalendar {
        print(t);
    }
}
```
Motivation

- Java deprecated Date class
- Relevant context Calendar applications
- Gather useful statistics
Collect useful statistics

The windows dev team @MSFT spent:

40 hours on code design

55 hours on dev & testing

100 hours fixing user bugs (Something fishy)
To Do

1. fix grammar bug
2. operating systems assignment :( 
3. Meet my project team mates
4. pay rent
5. call mom
```cpp
import "<std>";

Date start = 2013.05.16;
Date end = 2013.05.17;

main() {
  Calendar c1;
  Calendar c2;

  c1.name = "Michelle's Day";
  c1.start = start;
  c1.end = end;

  Task t;
  t.name = "Team Meeting";
  t.start = 2013.05.16.10;
  t.end = 2013.05.16.12;
  addTask(c1, t);

  c2.name = "Jason's Day";

  Task t4;
  t4.name = "Tennis Lesson";
  t4.start = 2013.05.16.11;
  t4.end = 2013.05.16.12;
  addTask(c2, t4);

  Calendar c3;
  c3 = getAvailableOneHourMeetingSlots(c1, c2);

  // loop through c3, and print out every Task meeting in c3
  for (Task meeting in c3) {
    print(meeting);
  }
}
```
Code!

Compile Video

Syntactic Conom
Symtactic Constructs

- Functions
- Loops

- Data Types
  - Calendar
    - Time Frame: 1 hour, 2 days, 3 weeks
    - Time Entity: Monday, Weekend, May
    - Date: 2013.05.16, 2013.05.16.16.00
  - Task
    - Number: 1, 2, 3, 4, 5, ...
  - String: "str1", "str2"
  - Boolean: true, false

- Conditionals
  - if...else...
- Functions
- Loops
populateTasks(Calendar c, String name) {
    every Date d from 2013.05.16.12.00 to 2013.05.16.17.00 by 30 minutes {
        Task t;
        t.name = name;
        t.start = d;
        t.end = d + 30 minutes;
        addTask(c, t);
    }
}

Boolean conflict(Calendar c, Task t) {
    Boolean conflicts = false;
    every Task task in c{
        if (not (task.end < t.start || t.end < task.start)) {
            conflicts = true;
            break;
        }
    }
    return conflicts;
}
- Data-Types
  
  Calendar
  TimeFrame: 1 hour, 2 days, 3 weeks
  TimeEntity: Monday, Weekend, May
  Date: 2013.05.16, 2013.05.16.16.00
  
  Task
  Number: 1, 2, 3, 4, 5, ...
  String: "Str1", "Str2"
  Boolean: true, false

- Conditionals
  if...else...
Boolean isMay(Date d) {
    if (is(d, May))
        return true;
    return false;
}
Project Management

Software Development Process:  
spiral model

Version control & Process tracking:  
github + fogbugs

Process tracking:  
timeline + weekly goals

Team communication:  
weekly meeting + google drive
Project Management

Software Development Process: spiral model

Version control & Process tracking: github + fogbugs

Process tracking: timeline + weekly goals

Team communication: weekly meeting + google drive
Objectives determining

Evaluation and planning

Risk identification and resolving

- propose B-plan
- leaving emergency
- basic & advanced for

- divide and conquer
- keep consistent
- test before expand and move forward

Identify target users
get proper cases

Set project rules
one and weekly goals
-date before move forward
Project Management

Software Development Process: spiral model

Version control & Process tracking: github + fogbugs

Process tracking: timeline + weekly goals

Team communication: weekly meeting + google drive
Architectural Block Diagram

Source Program (Program.tt) → Lexer (Scanner) → Token Stream → Parser → AST → Semantic Checking → Symbol Table → Interpreter → Output → Input

AST → AST
t.start = 2013.5.16.16.30;

- breaks the source code into tokens
- detects syntax errors
Parser

- generates an AST
- detects syntax errors
Semantic Checking

- Type Checking
- analyze functions and variables and put them into Symbol Table
Interpreter

Input

Output
Symbol Table

Scope (ID = 2)

Scope (ID = 1)

Global Scope (ID = 0)

Symbol: Name, Type, Value, etc.
Symbol: Name, Type, Value, etc.
Symbol: Name, Type, Value, etc.
TT.g

```plaintext
declStmt
  : t=type (WS*)! IDENT (WS*)! ';' -> ^ (DECLARE $t IDENT) 

;
HelloWorld.tt

Task t;
t.name = "HelloWorld";
Calendar c;
addTask(c.t)
print(c);
athresh@sld01:~ TT helloworld.tt

TT.sh

#!/bin/bash

java TTMain helloworld.tt
TT.Sh

#!/bin/bash

java TTMain helloworld.tt
TTMain.java

FileInputStream fis
= new FileInputStream("helloworld.tt")

Interpreter interpreter = new Interpreter();
interpreter.interp(fis);
TITMain.java

FileInputStream fis = new FileInputStream("helloworld.tt")

Interpreter interpreter = new Interpreter();
interpreter.interp(fis);
public void interp(InputStream input) {

    // Invoke the Lexer and the Parser

    CharStream stream = new ANILRInputStream(input);
    lexer = new TILexer(stream);
    tokenStream = new CommonTokenStream(lexer);
    parser = new TIParser(tokenStream);

    ... }

Interpreter.java
public void interp(InputStream input) {

    // Invoke the Lexer and the Parser

    CharStream stream = new ANTLRInputStream(input);
    lexer = new TILexer(stream);
    tokenStream = new CommonTokenStream(lexer);
    parser = new TIParser(tokenStream);

    ... }

Output

Helloworld!
Syntax:

// Start with translationUnit
// Test imports and importedLibraries
import "testLibrary";
import "michelleLib";

// Test programBody and methodsAndFieldsDeclarations
// test GLOBAL declarationStatement
Calendar c;

// test GLOBAL definitionStatement
String S = "Hello World";

// Test main and block
main() {
// Test statement.type
// Test type and declarationStatement
String S;
Number n;
Date d;
Task t;
TimeFrame tf;
Calendar c;
TimeEntity time;

// Test basic definitionStatement for all type
String S = "hello there!";
Number n = 12;
Date d = 2013.02;
Task t = "Do Work";
TimeFrame tf = 1 day + 4 days;
Calendar c = "My Calendar";
TimeEntity mon = Monday;

// Test assignmentStmt
var = 3;

// Test ifThenStatement
if (3 + a) {
print ("3 is + a");
}
...
FogBugz:

https://plt-tt.fogbugz.com/

<table>
<thead>
<tr>
<th>Case</th>
<th>Title</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>String does not accept spaces commenting out the two rules</td>
<td>Closed (Fixed)</td>
</tr>
<tr>
<td>3</td>
<td>IDENT and STRING conflict use fragment if two Lexer rules</td>
<td>Closed (Fixed)</td>
</tr>
<tr>
<td>4</td>
<td>YEAR and NUMBER conflict use fragment keyword</td>
<td>Closed (Fixed)</td>
</tr>
<tr>
<td>5</td>
<td>Force bracket use in if and if else statements? Yes, force</td>
<td>Closed (Fixed)</td>
</tr>
<tr>
<td>6</td>
<td>timeFrame and timeFrameConstant break allowing spaces</td>
<td>Closed (Fixed)</td>
</tr>
<tr>
<td>7</td>
<td>logical statements should take IDENT ex ( e &lt; 3 ) some statements</td>
<td>Resolved (Fixed)</td>
</tr>
<tr>
<td>8</td>
<td>what are all the every loop options Yes, include the last 2</td>
<td>Resolved (Fixed)</td>
</tr>
<tr>
<td>9</td>
<td>punctuation should be allowed inside of strings try to use</td>
<td>Closed (Fixed)</td>
</tr>
<tr>
<td>10</td>
<td>should Read be upper case and print lower case</td>
<td>Closed (Fixed)</td>
</tr>
<tr>
<td>11</td>
<td>timeFrameDefnStmt should allow assignment with 'until'</td>
<td>Closed (Fixed)</td>
</tr>
<tr>
<td>12</td>
<td>definition of a variable from another variable? var1 = var2;</td>
<td>Closed (Fixed)</td>
</tr>
<tr>
<td>13</td>
<td>we shouldn't need WS* in between definitions in .g file, it is</td>
<td>Closed (Fixed)</td>
</tr>
<tr>
<td>14</td>
<td>dateConstant does not work I think we can just accept 01-</td>
<td>Resolved (Fixed)</td>
</tr>
<tr>
<td>15</td>
<td>Discontinuity Timeframe or TimeFrame</td>
<td>Resolved (Fixed)</td>
</tr>
<tr>
<td>16</td>
<td>Java.util.Calendar vs our Calendar.java We are probably</td>
<td>Closed (Fixed)</td>
</tr>
</tbody>
</table>
JUnit:

```java
public void testIfElse()
{
    FileInputStream fileStream = null;
    try {

        fileStream = new FileInputStream("Src/columbia/plt/tt/programs/ifelse.tt");
        interpreter.interp(fileStream);

        String[] programOut = outFileContent.toString().split("\n");
        assertEquals(programOut[0], "YES n is in range");
        assertEquals(programOut[1], "c is less than d");
    }
    catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (RecognitionException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (RecognitionException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Test Programs
- decalaredefine.tt
- ifelse.tt
- Loops.tt
- methods.tt
- operator.tt
- Program1.tt
- standard_library.tt
- strcmp.tt

Tutorial Programs
- tutorial_daySLeft.tt
- tutorial_HelloWorld.tt
- tutorial_MeetingTime.tt
- tutorial_RecurringTask.tt
- tutorial_ScheduleTaskS.tt
- tutorial_StudyTime tt
Time & Task

Zheng: The PM
Peter: The Guru
Jason: The Architect
Michelle: The Tester
Akhila: The Integrator
Conclusions

- Code Early
- Code Often
- Code Together
Conclusions

- Code Early
- Code Often
- Code Together
Questions?
TIParseR, TT Lexer, TT Main, Interpreter.java

Build.xml ➔ ant

.class files ➔ JVM