Phase II

- Parser generator
  - takes as input an XML type description
  - generates a Java parser for documents of that type
    - group_name.parser.type_name
    - group_name.tree
- The generated parser
  - takes as input an XML document
  - rejects it if the format is incorrect
  - otherwise produces a parse tree for the program
- Validation of the system
  - correctness of the parser generator
  - correctness of the generated parser
  - need to write test code that uses generated parsers
  - at least two tests

Email Example

- XML document type description

```
<XML_TYPE NAME = "Email">
  <TAG NAME = "Header">
     <ATTR> From Date Subject </ATTR>
     <TAG NAME = "To">
       <ATTR> Receiver </ATTR>
     </TAG>
  </TAG>
  <TAG NAME = "Body">
  </TAG>
</XML_TYPE>
```


How about this?

```
<XML_TYPE NAME = "Email">
  <TAG NAME = "Header">
     <ATTR NAME = "From"> </ATTR>
     <TAG NAME = "To" REPEATABLE = "Yes" >
       <ATTR> Receiver </ATTR>
     </TAG>
  </TAG>
  <TAG NAME = "Body">
  </TAG>
</XML_TYPE>
```


Tree Example

```
<XML_TYPE NAME = "Tree Type">
  <TAG NAME = "Node">
     <ATTR>NAME</ATTR>
     <RECUR TAG = "Node"></RECUR>
  </TAG>
</XML_TYPE>
```


Parser Generator

- Takes an input type description
- Generates a parser
- For example
  - java ParserGen tree_type
  - The generated parser is in package "ding.parser.tree"
  - The tree class is in package "ding.tree"

Expression Example

"(2+6)/2"

```
<OP TYPE="/">
  <OP TYPE="+" >
    <NUM VAL="2"></NUM>
    <NUM VAL="6"></NUM>
  </OP>
  <NUM VAL="2"></NUM>
</OP>
```

•Chen Ding

```
import ding.parser.expr;
import ding.tree;

class Test {
 static public void main(...){
   ExprParser p = new ExprParser();
   Tree t = ExprParser.parse("a.expr");
   System.out.println(Calculate(t));
 }

 static calculate(Tree t) {
   if (t instanceof Num) return t.val;
   else {
     return (child1 op child2);
 }
}
```