GIT Department of Computer Engineering

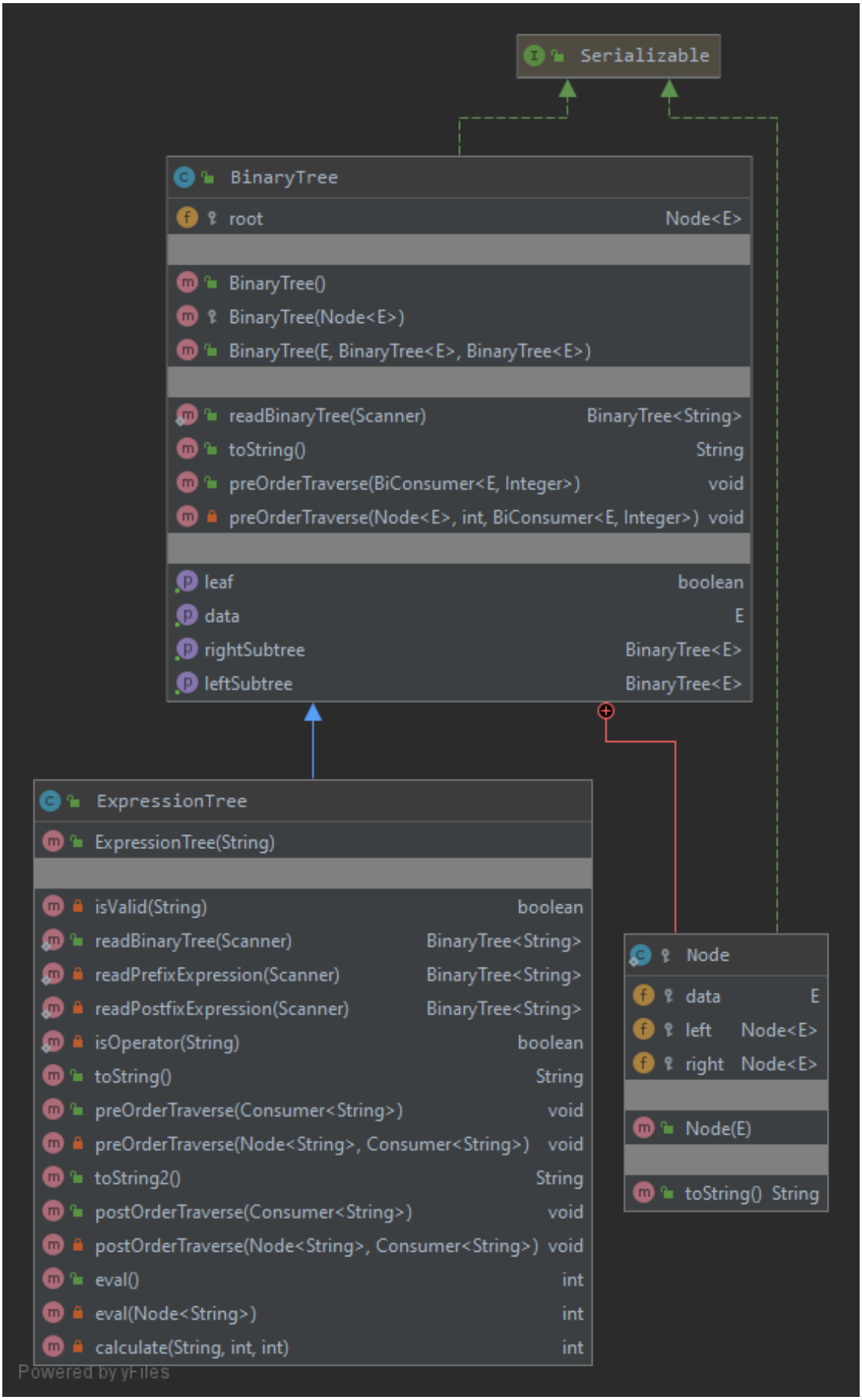
CSE 222/505 – Spring 2020

Homework #05 Part 2 Report

Abdullah ÇELİK

171044002

Class Diagram



Problem Solution Approach

Firstly, readBinaryTree method can read prefix and postfix expressions. For this reason, I have to decide whether the given expression is prefix or postfix. After this decision a method for prefix, a method for postfix will read these expressions and builds the expression trees. Secondly, it is important to override the consumer method of the preOrderTraverse and postOrderTraverse methods correctly. Therefore, paying attention to this part, the consumer of the preOrderTraverse method in the toString method should override the consumer method of the postOrderTraverse method in the toString2 method correctly. To implement the remaining methods with the correct algorithms.

Test Cases

Test ID	Scenerio	Test Data	Expected Results	Actual Results	Pass/Fail
TEST01	ExpressionTree constructor when expression is valid prefix expression	ExpressionTree constructor expression : "+ + 10 * 5 15 20"	Successfully created	As expected	Pass
TEST02	ExpressionTree constructor when expression is valid postfix expression	ExpressionTree constructor expression : "10 5 15 * + 20 +"	Successfully created	As expected	Pass
TEST03	ExpressionTree constructor when expression is null	ExpressionTree constructor expression : null	Expected throwed NullPointer Exception	As expected	Pass
TEST04	ExpressionTree constructor when expression is invalid expression	ExpressionTree constructor expression : "10 5 + ^ A ."	Expected throwed IllegalArgument Exception	As expected	Pass
TEST05	BinaryTree<String> readBinaryTree(Scanner scan) method called when scanner scans valid prefix expression	expression to scan : "- * 10 + 3 2 20"	Successfully created and returned BinayTree	As expected	Pass
TEST06	BinaryTree<String> readBinaryTree(Scanner scan) method called when scanner scans valid postfix expression	expression to scan: "10 3 2 + * 20 -"	Successfully created and returned BinayTree	As expected	Pass

TEST07	BinaryTree<String> readBinaryTree(Scanner scan) method called when scanner is null	expression to scan: null	Successfully returned null	As expected	Pass
TEST08	BinaryTree<String> readBinaryTree(Scanner scan) method called when scanner has no elements	expression to scan: ""	Successfully returned null	As expected	Pass
TEST09	String toString() method called	A valid expression tree	Successfully returned expression tree as prefix expression	As expected	Pass
TEST10	String toString2() method called	A valid expression tree	Successfully returned expression tree as postfix expression	As expected	Pass
TEST11	int eval() method called	A valid expression tree	Successfully returned correct result	As expected	Pass

Running and Results

```
TEST01 - Testing ExpressionTree constructor
When expression is valid prefix expression
Expression : + + 10 * 5 15 20
ExpressionTree was created successfully!
Prefix Expression(toString method) :
+ + 10 * 5 15 20
Postfix Expression(toString2 method) :
10 5 15 * + 20 +

TEST02 - Testing ExpressionTree constructor
When expression is valid postfix expression
Expression : 10 5 15 * + 20 +
ExpressionTree was created successfully!
Prefix Expression(toString method) :
+ + 10 * 5 15 20
Postfix Expression(toString2 method) :
10 5 15 * + 20 +

TEST03 - Testing ExpressionTree constructor
When expression is null
Expression : null
NullPointerException was caught!

TEST04 - Testing ExpressionTree constructor
When expression is invalid expression
Expression : 10 5 + ^ A .
IllegalArgumentException was caught!
Invalid token!
```

```
TEST05 - Testing BinaryTree<String> readBinaryTree(Scanner scan)
When scanner scans valid prefix expression
Expression : - * 10 + 3 2 20
Before method expression tree is :
null
After method expression tree is :
-
   *
      10
         null
         null
      +
         3
            null
            null
         2
            null
            null
   20
      null
      null

TEST06 - Testing BinaryTree<String> readBinaryTree(Scanner scan)
When scanner scans valid postfix expression
Expression : 10 3 2 + * 20 -
Before method expression tree is :
null
After method expression tree is :
-
   *
      10
         null
         null
      +
         3
            null
            null
         2
            null
            null
   20
      null
      null

TEST07 - Testing BinaryTree<String> readBinaryTree(Scanner scan)
When scanner is null
Before method expression tree is :
null
After method expression tree is :
null
```

```
TEST08 - Testing BinaryTree<String> readBinaryTree(Scanner scan)
When scanner has no next element(expression is empty)
Expression :
Before method expression tree is :
null
After method expression tree is :
null

TEST09 - Testing toString()
Prefix Expression(toString method) :
+ + 10 * 5 15 20

TEST10 - Testing toString2()
Postfix Expression(toString2 method) :
10 5 15 * + 20 +

TEST11 - Testing int eval()
Prefix Expression(toString method) :
+ + 10 * 5 15 20
Postfix Expression(toString2 method) :
10 5 15 * + 20 +
Result : 105
```