CSE 213 – Homework 6 Rubric

Student Name: Marisa Loraas

Grader Name: Rachel Powers

Problem	Score	Total
Style Guidelines (7 points)		**5**
Submission is named `cse213_<firstname>_<lastname>_hw5.tar.gz`	1	1
Packages are named: `oop.<firstinitial><lastname>.hw5.<number>` where <number> is the problem number	1	1
Code follows Google's style guide reasonably well, and uses four-space indentation	3	3
Traversal.java		**1**
DifferentiatesPREORDER, INORDER, and POSTORDER	1	1
BinarySearchTree.java **Note:** The implementations for add(), find(), and remove() will probably be "distributed" between the two classes BinarySearchTree and Node		**44**
Class is defined with a type parameter, <E>	1	1
Node inner class contains an element of type E, a left Node, and a right Node	1	1
Constructor takes a Comparator<E> as an argument and sets it as an attribute	2	2
add() inserts a new leaf node into the tree	8	8
remove() finds and removes a Node from the tree, and moves	9	9
find() returns true if the given element is in the tree, false otherwise	6	6
findMin() returns the minimum (left-most) element of the tree, or null if the tree is empty	4	4
findMax() returns the maximum (right-most) element of the tree, or null if the tree is empty	4	4
isEmpty() returns true if the root is null, false otherwise	1	1
clear() sets the root to null	1	1
addAll() takes an array-list of elements, and adds each one to tthe tree	4	4
print() prints each element of the tree, using either a PREORDER, INORDER, or POSTORDER traversal	3	3
Operator.java		**7**
Differentiates 8 operators: LPAREN, RPAREN, EXPONENT, MULTIPLY, DIVIDE, MODULO, ADD, and SUBTRACT	1	1

getPrecidence() returns: • 3 for LPAREN & RPAREN • 2 for EXPONENT • 1 for MULTIPLY, DIVIDE, & MODULO • 0 for ADD & SUBTRACT	2	2
eval() returns the result of applying an operator to a given pair of values, depending on the value of this	2	2
tostring() returns "(", ")", "^", "*", "/", "%", "+", or "-"; depending on the value of this	2	2
Token.java		**8**
One consructor sets a number, and sets the operator to null	2	2
One constructor sets an operator, and sets the number to null	2	2
isOperator() returns true if the operator is not null	1	1
parseToken() is static; returns a new Token if its input is one of the operator symbols *or* can be parsed as a double; returns null otherwise	3	3
RPN.java		**14**
eval() is static; implements the RPN algorithm on pages 4-5 of the assignment	6	6
main() loop prompts for RPN expressions, splits each input into an array-list of Tokens, and prints the result of RPN.eval()	6	6
The main loop catches any exceptions, and prints an error message instead of crashing	2	2
Calculator.java **Note:** The examples in the PDF show some expressions where parentheses are not surrounded by spaces. To clarify, this is not a requirement, and no points should be taken off if the program requires each input token to be space separated.		**16**
toRPN() implements Dijkstra's shunting yard algorithm on page 6 of the assignment	8	8
main() loop prompts for math expressions, splits each input into an array-list of Tokens, converts to an RPN expression, and prints the result of RPN.eval()	6	6
The main loop catches any exceptions, and prints an error message instead of crashing.	2	2
JUnit Testing		5
Student has a testing class for RPN.java	0	1
Student has static method with the @BeforeClass annotation that initializes a global ArrayList.	0	1
Student has static method with the @AfterClass annotation that prints a message to the console (.5 pts) and clears the ArrayList (.5 pts).	0	1
Student has methods with the @Test annotation that tests addition, subtraction, multiplication and division of the RPN calculator (0.25 pts each).	0	1

Student has a test that invokes an ArithmeticException based on their RPN calculator’s stack not returning one value.	0	1
Total Score	95	**100**
Comments:		