Project 6

Due 11:59 PM Wednesday November 29
This project **must** be submitted on time.
Project 6

• Implement the methods in the Hashtable and ExpressionTree classes shown on the following slides

• The operators in an infix expression can include +, -, *, /, %, ^, !, =. The operators +, -, *, /, % use Java precedence and associativity rules. The exponentiation operator, ^, has higher precedence than +, -, *, /, % and is right associative. ! means unary minus and is the highest precedence operator and is also right associative. The assignment operator, =, is the lowest precedence operator. Parentheses are used to change the order of operations.

• The operands will be variables (a letter followed by 0 or more letters or digits) or ints (one or more digits)

• You must not use Java’s Hashtable in this project.

• I will use my own test driver when I test your project. Look at the test driver I used in the postfix example. You can create something similar to test this project.

• There is no demo for this project.
import java.io.*;
import java.util.*;

public class Hashtable<K, V> {
    //implements a separate chaining hash table
    private class Node {
        K key;
        V value;
        Node next;
        private Node(K key, V value, Node n) {
            this.key = key;
            this.value = value;
            next = n;
        }
    }

    ArrayList<Node> table;
    int tableSize;

    private int hash(K key) {
        return (Math.abs(key.hashCode()))%tableSize;
    }
}
public Hashtable(int size) {
    table = new ArrayList<>((size);
    tableSize = size;
}

public V get(K key) {
    //if key is not in the table return null
    //otherwise return the value associated with key
}

public void put(K key, V value) {
    //if key is in the table update the value associated with key
    //otherwise insert key and its associated value into the table
}

public Set<K> keySet() {
    //return a set of the keys in the table
    //You can use Java's HashSet to create the set that is returned
}
import java.io.*;
import java.util.*;

public class ExpressionTree {
    private class Node {
        Node left;
        String data;
        Node right;
        private Node(Node L, String d, Node R) {
            left = L;
            data = d;
            right = R;
        }
    }

    public ExpressionTree(String exp) {
        //PRE: exp is an syntactically correct infix expression
        //     in the expression the first token is a variable and the
        //     second token is a = sign
        //Build an expression tree from exp
    }

    public void eval(Hashtable<String, Integer> varMemory) {
        //evaluate the expression tree
        //varMemory contains values for variables
    }
}
Programming Project 6 Submission

- Upload one zip file to Canvas. The zip file must contain only two files: `ExpressionTree.java` and `Hashtable.java`. Do not upload your whole Eclipse project!

- You have to write code to test your program but I will write my own test driver. Do not upload your test driver code.
Hard Deadlines For Projects

• Project 5: last day to demo is Saturday November 19
• Project 6: upload to Canvas by 11:59 PM Wednesday November 29
• Project 7: last day to demo is Saturday December 10