```
grammar bankWithExpr ;

// @header block is for imports and optional package statement
@header {
  import java.util.*;
}

// @parser block is for member functions and data
@parser::members {

  // member data  ---
  Map<String, Integer> customers = new HashMap<String, Integer>() ;

  // member functions ---
  void makeDeposit(String user,  int amt) {
      if (customers.containsKey(user) ) {
         System.out.println("Welcome back " + user);
         customers.put(user, (customers.get(user) + amt));
      }
      else {
         System.out.println("Welcome new customer " + user);
         customers.put(user,amt);
      }

      System.out.println("Your deposit of " + amt + " has been processed");
  }

  void makeWithdrawal(String user,  int amt) {
      if (customers.containsKey(user) ) {
         System.out.println("Welcome back " + user);
      }
      else {
         System.out.println("Welcome new customer " + user);
         customers.put(user,0);
      }

      if(customers.get(user) < amt){
            System.out.println("Your withdrawal of " + amt +
            " cannot be processed due to a balance of " +
customers.get(user));
      }
      else{
            customers.put(user, (customers.get(user)-amt));
            System.out.println("Your withdrawal of " + amt + " has been
processed");
      }

  }

  int doMath(int v1, int v2, int op) {
     int retval = -1;
     switch(op) {
       case MOD : retval = v1 % v2;
                  break;
       case MUL : retval = v1 * v2;
              break;
       case DIV : retval = v1 / v2;
```

```
                    break;
       case ADD : retval = v1 + v2;
                    break;
       case SUB : retval = v1 - v2;
              break;
     }
     return retval;

  }
}  // end of @parser block



// Parser Rules -------------------------

transaction       : (deposit | withdraw )+  ;
deposit           : ID  DEPOSIT  expr  { makeDeposit   ($ID.text, $expr.v); }
;
withdraw    : ID  WITHDRAW expr  { makeWithdrawal($ID.text, $expr.v); } ;

expr returns [int v]
    :   a=expr op=MOD b=expr { $v = doMath($a.v, $b.v, $op.getType() ); }

    | a=expr op=(MUL|DIV) b=expr { $v = doMath($a.v, $b.v, $op.getType() ); }

    |   a=expr op=(ADD|SUB) b=expr { $v = doMath($a.v, $b.v, $op.getType() );
}

    |   NUM                         {$v = Integer.valueOf($NUM.getText());}
    |   ID                          {
                                    String id = $ID.getText();
                                    if ( customers.containsKey(id) ) {
                                        $v = customers.get(id);
                                    }
                                    else {
                                     $v = 0;
                                    }
                                    }
    |   '(' e=expr ')'              {$v = $e.v;}
    ;

//LEXER RULES  --------------------------
MOD :   '%' ;
MUL :   '*' ; // assigns token name to '*' used above in grammar
DIV :   '/' ;
ADD :   '+' ;
SUB :   '-' ;


DEPOSIT     : 'dep' ;
WITHDRAW    : 'withdraw' ;
NUM         : DIGIT+ ;
DIGIT       : [0-9]  ;
ID          : [a-z]+ | [A-Z] DIGIT DIGIT DIGIT;
WS          : [ \n\r\t]+ -> skip ;
```

```
ReplRunner : type 'bye' or 'quit' to terminate
>>>joe dep 10
Welcome new customer joe
Your deposit of 10 has been processed
>>>joe withdraw 10
Welcome back joe
Your withdrawal of 10 has been processed
>>>joe withdraw 1
Welcome back joe
Your withdrawal of 1 cannot be processed due to a balance of 0
>>>joe dep 5*7+3*2-8
Welcome back joe
Your deposit of 33 has been processed
>>>joe dep 15%4
Welcome back joe
Your deposit of 3 has been processed
Welcome new customer G875
Your deposit of 46 has been processed
>>>G875 withdraw 45
Welcome back G875
Your withdrawal of 45 has been processed
>>>G875 withdraw 2
Welcome back G875
Your withdrawal of 2 cannot be processed due to a balance of 1
```