calculator\Calculator.java

```java
/**
 * @author  Sebastian Diaz & Guillaume Dunant
 * Date   : 04.12.2023
 * Fichier: Calculator.java
 */

package calculator;

import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

 /**
  * Calculatrice en mode console
  */
public class Calculator {
    private static State state;
    private static Map<String,Operator> operationsMap;
    private static final String[] opeName =
        {"+", "-", "*", "/", "POW", "SQRT", "NEG",
        "INV", "MS", "MR", "C", "CE", "HELP", "EXIT"};

    private enum opeEnum {ADD, SUB, MULT, DIV, POW, SQRT,
         NEG, INV, MS, MR, C, CE, HELP, EXIT};

    /**
     * Converti un string en Double
     * @param input String à convertir
     * @return Double ou null si la conversion a échoué
     */
    private static Double convertInputToDouble(String input){
        try{
            return Double.parseDouble(input);
        }
        catch(NumberFormatException e){
            return null;
        }
    }

    /**
     * Programme principale pour la calculatrice en mode console
     * @param args Arguments de lancement
     */
    public static void main(String[] args) {
        //Affichage titre
        System.out.println("****************************\n" +
                           "*        Calculator        *\n" +
                           "****************************\n");

        System.out.printf("%s pour afficher les opérations ou %s pour quitter\n\n", opeName[12], opeName[13]);

        String input;
        Double val;
        Scanner scanner = new Scanner(System.in);
        boolean firstVal = true;

        while (true) {

            //Récupère l'entrée de l'utilisateur
            System.out.print("> ");
            input = scanner.nextLine();

            //Si l'input est un nombre
            if((val = convertInputToDouble(input)) != null){
                if(firstVal){
                    firstVal = false;
                }
                else{
                    state.pushCurrent();
                }
                state.setCurrent(val);
            }
            else{
                input = input.toUpperCase();
                Operator ope = operationsMap.get(input);

                //Si l'input est une opération
                if (ope != null) {
                    ope.execute();
                }
                //Si EXIT
                else if(input.equals(opeName[opeEnum.EXIT.ordinal()])){
                    break;
                }
                //Si HELP
                else if(input.equals(opeName[opeEnum.HELP.ordinal()])){
                    for(String s : opeName){
                        System.out.println(s);
                    }
                    continue;
                }
                //Input non reconnu
                else{
                    System.out.println("Opération inconnue");
                }
            }

            //Affiche les valeurs contenues dans la stack et la valeur courante
            String[] stackStrings = state.getStackInString();
            if(stackStrings != null){
                for(String s : stackStrings){
                    System.out.print(s + " ");
                }
            }
            System.out.println(state.getCurrentInString());
        }

        scanner.close();
    }

    static{
        state = new State();
        operationsMap = new HashMap<>();
        operationsMap.put(opeName[opeEnum.ADD.ordinal()], new Addition(state));
        operationsMap.put(opeName[opeEnum.SUB.ordinal()], new Subtraction(state));
        operationsMap.put(opeName[opeEnum.MULT.ordinal()], new Multiplication(state));
        operationsMap.put(opeName[opeEnum.DIV.ordinal()], new Division(state));
        operationsMap.put(opeName[opeEnum.POW.ordinal()], new Power(state));
        operationsMap.put(opeName[opeEnum.SQRT.ordinal()], new SquareRoot(state));
        operationsMap.put(opeName[opeEnum.NEG.ordinal()], new Negate(state));
        operationsMap.put(opeName[opeEnum.INV.ordinal()], new Inverse(state));
        operationsMap.put(opeName[opeEnum.MS.ordinal()], new MemoryStore(state));
        operationsMap.put(opeName[opeEnum.MR.ordinal()], new MemoryRecall(state));
        operationsMap.put(opeName[opeEnum.C.ordinal()], new Clear(state));
        operationsMap.put(opeName[opeEnum.CE.ordinal()], new ClearError(state));
    }
}

```