

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

-----****-----



BÁO CÁO

ĐỒ ÁN CÔNG NGHỆ THÔNG TIN

GVHD: Nguyễn Đăng Quang

Sinh viên thực hiện:

**20110457
20110464**

**Trần Tiến Đạt
Lê Hải**

Tp. Hồ Chí Minh ngày 15 tháng 11 năm 2022

MỤC LỤC

I. ĐẶT TẢ ĐỀ TÀI	2
II. PHÂN CÔNG CÔNG VIỆC	3
III. THIẾT KẾ	4
1. Cách Hoạt Động Của Ứng Dụng	4
2. Thuật toán	5
3. Thiết kế lớp	6
A. Danh mục các lớp	6
B. Doanh mục phương thức của lớp	8
C. Doanh mục hàm JavaScript	10
4. Thiết kế giao diện	10
IV. CÀI ĐẶT VÀ KIỂM THỬ	11
V. KẾT LUẬN	13

I. ĐẶT TẢ ĐỀ TÀI

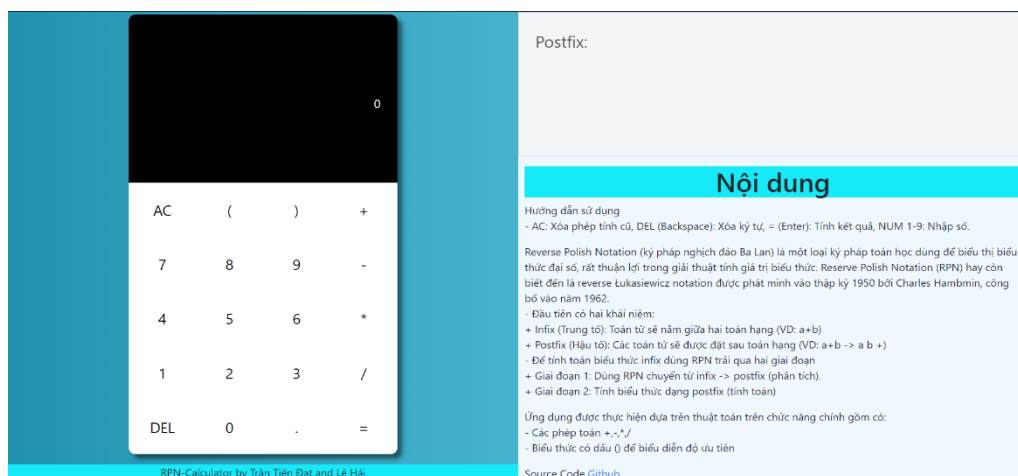
Hiện nay các tiện ích tính toán trên máy tính rất phổ biến nhưng người dùng thường không chú ý rằng máy tính không hiểu dạng biểu thức mà con người chúng ta hay sử dụng

ví dụ như các phép toán sử dụng dấu ngoặc để thể hiện độ ưu tiên của phép toán. Để giải quyết bài toán đó thì Reverse Polish Notation (RPN) là thuật toán “kinh điển” để giải quyết vấn đề này.

Nhóm em xây dựng một trang web mô phỏng lại máy tính cầm tay cho phép người dùng nhập vào biểu thức để tính toán, trả ra kết quả và cho người dùng xem biểu thức dưới dạng hậu tố (postfix) để người dùng biết được dạng biểu thức mà máy tính có thể hiểu và xử lý. Song song đó áp dụng các nguyên tắc thiết kế S.O.L.I.D và factory design pattern vào bên trong code của ứng dụng.

Người dùng có thể nhập vào các phép tính cơ bản như cộng, trừ, nhân, chia và các dấu ngoặc tròn để đánh dấu mức độ ưu tiên của các phép toán từ đó hệ thống sẽ xử lý trả về kết quả của biểu thức và kèm theo minh họa biểu thức dưới dạng máy tính có thể hiểu và tính toán.

Về giao diện sẽ có bố cục chia làm 2 một phần là ứng dụng chính (máy tính), phần còn lại sẽ in ra biểu thức dưới dạng postfix và các thông tin giới thiệu về thuật toán và ứng dụng.



Giao diện chính của ứng dụng sẽ phát triển

II. PHÂN CÔNG CÔNG VIỆC

TT	MSSV	Tên SV	Công việc	Mức độ
----	------	--------	-----------	--------

1	20110457	Trần Tiến Đạt	Thiết kế lớp, chức năng, backend	100%
2	20110XXX	Lê Hải	Nghiên cứu thuật toán, thiết kế giao diện, frontend, kiểm thử	100%

III. THIẾT KẾ

1. Cách Hoạt Động Của Ứng Dụng

Ứng dụng được triển khai dưới dạng Web Application cho nên sẽ theo mô hình 3 lớp gồm Model, View và Controller. Các lớp trên tương ứng với 3 package đảm nhiệm chức năng riêng biệt.

Phần View: gồm 1 file index.html để cho người dùng tương tác, nhập vào input sau đó gửi xuống server xử lý thông qua POST method.

Phần Controller: nhận các giá trị được người dùng gửi xuống server tiến hành gọi các class và xử lý tính toán trả ra kết quả

Phần Model: Gồm các lớp để xây dựng thuật toán RPN cũng như để tính toán kết quả.

Mô tả hoạt động của ứng dụng:

Người dùng nhập vào input → Tách chuỗi thành các object (toán tử / toán hạng) → Sử dụng thuật toán để tính toán giá trị → Trả về phía frontend hiển thị cho người dùng.

+ Đầu tiên người dùng sẽ nhập vào biểu thức cần tính toán gửi xuống server thông qua http request. Server nhận vào input của người dùng dưới dạng String, sau đó sử dụng đối tượng của lớp CalculatorString để tách các ký tự trong chuỗi thành các Object thuộc vào loại Operand hoặc Operator tùy vào giá trị của chuỗi.

+ Sau khi đã có một danh sách các đối tượng MathSymbol (là lớp dẫn xuất của lớp Operand và Operator) bao gồm các đối tượng thuộc 2 nhóm trên. Ta dùng đến đối tượng của lớp CalculatorInfix để chuyển từ dạng trung tố sang hậu tố theo thuật toán RPN. Kết thúc quá trình ta nhận được một danh sách các Operand và Operator được sắp xếp theo đúng thứ tự tính toán.

+ Kết thúc quá trình tính toán ta dùng đối tượng của lớp CalculatorPosfix để tính toán giá trị cuối cùng và CalculatorList nhằm để in ra các giá trị bên trong 1 danh sách cần xem, ở đây là Queue<MathSymbol>postfix. Sau khi đã có kết quả cùng với chuỗi biểu thức dạng postfix, ta nối 2 kết quả trên ngăn cách nhau bằng “;” để frontend dùng javascript để tách chuỗi hiển thị ra 2 phần kết quả.

2. Thuật toán

- Đầu tiên có hai khái niệm:

- + Infix (Trung tố): Toán tử sẽ nằm giữa hai toán hạng (VD: $a+b$)
- + Postfix (Hậu tố): Các toán tử sẽ được đặt sau toán hạng (VD: $a+b \rightarrow a\ b\ +$)

- Để tính toán biểu thức infix dùng RPN trải qua hai giai đoạn:

- + Giai đoạn 1: Dùng RPN chuyển từ infix \rightarrow postfix (phân tích).
- + Giai đoạn 2: Tính biểu thức dạng postfix (tính toán).

-Luồng dữ liệu

- + Đầu vào: chuỗi ký tự toán học VD: $(12*(24+4))$
- + Đầu ra: Hàng đợi gồm các toán hạng và toán tử đã theo dạng hậu tố (VD: $12\ 24\ 4\ +\ *$) và kết quả phép tính

- Các bước thuật toán

** Chuyển từ trung tố sang hậu tố*

- Duyệt đến hết chuỗi để tách ra toán hạng và toán tử bỏ vào 1 mảng
- Duyệt hết mảng vừa thu được ở bước trên

- Nếu là toán hạng $[0,9]$ ta push vào queue
- Nếu là toán tử $+, -, *, /$

→ Thực hiện vòng lặp kiểm tra nếu đỉnh Stack là toán tử, độ ưu tiên lớn hoặc bằng toán tử đang xét thì ta pop ra khỏi stack và push vào queue

→ Push toán tử hiện tại vào stack.

- Nếu gặp dấu (ta push vào stack.

- Nếu gặp dấu $)$, ta thực hiện lấy toán tử trong stack và push vào queue đến khi gặp dấu $($ và pop dấu $($ khỏi stack.
 - Kết thúc vòng lặp, nếu trong stack còn toán tử, ta push hết giá trị trong stack vào queue.
- * *Tính toán giá trị biểu thức sau khi chuyển về dạng postfix*
- Khởi tạo Stack
 - Cho vòng lặp duyệt các giá trị trong queue:
 - Nếu giá trị là số ta push vào stack.
 - Nếu gặp toán tử, ta pop 2 phần tử trong stack ra và thực hiện tính giá trị biểu thức với toán tử hiện tại và push giá trị vừa tính được vào stack.
 - Kết thúc vòng lặp, phần tử duy nhất ở trong stack là giá trị biểu thức cần tính.

3. Thiết kế lớp

A. Danh mục các lớp

Sinh viên phụ trách: Trần Tiên Đạt

TT	Tên lớp	Mục đích
1	MathSymbol (<i>Abstract class</i>)	Lớp trừu tượng cho các ký tự trong biểu thức toán học có các phương thức trừu tượng để các lớp con kế thừa.
2	Operand (<i>MathSymbol</i>)	Kế thừa từ MathSymbol mô tả các đối tượng thuộc lớp này là toán hạng có các phương thức để ép giá trị từ chuỗi sang số.
3	Operator (<i>MathSymbol</i>)	Kế thừa từ MathSymbol mô tả các đối tượng lớp này là toán tử có các phương thức để kiểm tra độ ưu tiên của từng loại toán tử.
4	MathSymbolFactory	Lớp này đóng vai trò như nhà máy sản xuất các đối tượng Operand/Operator thông qua tham số được truyền vào khởi tạo (constructor).

5	ICaculator<T> (Interface)	Giao diện chứ 1 hàm calculateResult sử dụng Generic để tham số hóa kiểu dữ liệu trả về. Để các lớp thuộc loại Calculuator sẽ implement phương thức này.
6	CalculatorString (ICalculator)	Dùng để tách các token bên trong chuỗi biểu thức đầu vào thành 1 mảng các đối tượng MathSymbol có thể thuộc vào Operand/Operator
7	CalculatorInfix (ICalculator)	Dùng RPN để chuyển một mảng các đối tượng Operand/Operator thành một hàng đợi các đối tượng như trên nhưng được sắp xếp theo đúng thứ tự postfix để tính toán.
8	CalculatorPostfix (ICalculator)	Nhận dữ liệu từ hàm tính của CalculatorInfix trả về, tính giá trị của biểu thức và trả về kết quả cuối cùng.
9	CalculatorList (ICalculator)	Lớp này dùng để show ra giá trị các phần tử bên trong 1 danh sách/mảng về kiểu chuỗi.
10	Home (HttpServlet)	Đóng vai trò là controller xử lý các request (GET) khi người dùng vào trang chủ của web.
11	Calculate (HttpServlet)	Controller xử lý các POST Request khi người dùng gửi xuống server biểu thức cần tính toán và Response cho Frontend giá trị kết quả để hiển thị cho người dùng.
12	Error (HttpServlet)	Controller xử lý lỗi khi người dùng tìm kiếm các tài nguyên không có sẵn trên web
13	RedirectPage	Chứa các biến static lưu các chuỗi để chương trình dễ dàng sử dụng. VD: public static final String HOME_SERVLET = "/home";

B. Doanh mục phương thức của lớp (bỏ qua getter/setter của thuộc tính không dùng đến)

Sinh viên phụ trách: Trần Tiến Đạt

TT	Phương thức	Mục đích	Tên file, số thứ tự chứa dòng báo cáo
1	getValue() Input: Không có Output: String	Lấy giá trị bên trong đối tượng Operand/Operator	MathSymbol.java (13) CalculatorPostfix.java (39) CalculatorList.java (16)
2	getFirstCharacter() Input: Không có Output: Char	Lấy ký tự đầu tiên của đối tượng	MathSymbol.java (21,27) Operator.java (11) CalculatorPostfix.java (54)
3	getTypeBracket() Input: Không có Output: Int	Lấy ra loại dấu ngoặc '(' hoặc ')'	MathSymbol.java (26) CalculatorInfix.java (32,34,36,40) CalculatorString.java (60)
4	calculateResult() Input: Không có Output: Generic type	Tính toán các giá trị là thuộc tính của từng loại máy tính (override từ Interface ICalculator)	ICalculator.java (4) CalculatorPostfix.java (24) CalculatorInfix.java (25) CalculatorList.java (13) CalculatorString.java (22) Calculate.java (45,48,51,52)
5	checkNegative(MathSymbol symbol) Input: MathSymbol Output: boolean	Kiểm tra số âm	CalculatorString.java (41,59)
6	calTowVar(Operand op1, Operand op2, Operator opt)	Tính toán biểu thức gồm 2 toán hạng và 1 toán tử	CalculatorPostfix.java (48)

	Input: 2 Operand, Operator Output: Operand	trả về kết quả là toán hạng	
7	createObject(String value) Input: String Output: MathSymbol	Nhận vào giá trị của 1 loại ký tự toán học sau đó kiểm tra và khởi tạo đối tượng thuộc vào Operand/Operator tương ứng với giá trị (Factory design pattern)	MathSymbolFactory.java (5) CalculatorString.java(33,45,49)
8	getPriority() Input: Không Output: int	Trả về mức độ ưu tiên của toán hạng	Operator.java(10) Calculator.java(47,48)
9	getDecimalValue() Input: Không Output: float	Trả về giá trị thập phân kiểu float	CalculatorPostfix.java(50,51) Operand.java(9)
10	doGet(req,resp) (Home.java) Input: HttpServletRequest /Response Output: Không có	Xử lý HTTP Method GET vào trang /home	Home.java (15)
11	doPost(req,resp) (Calculate.java) Input: HttpServletRequest /Response	Xử lý HTTP Method POST khi người dùng gửi yêu cầu tính	Calculate.java(21)

	Output: Không có	toán cho server và response kết quả cho phía client	
12	Caluculate(String expr) Input: String Output: String	Xử lý tính toán giá trị từ chuỗi của người dùng.	Calculate.java (42)

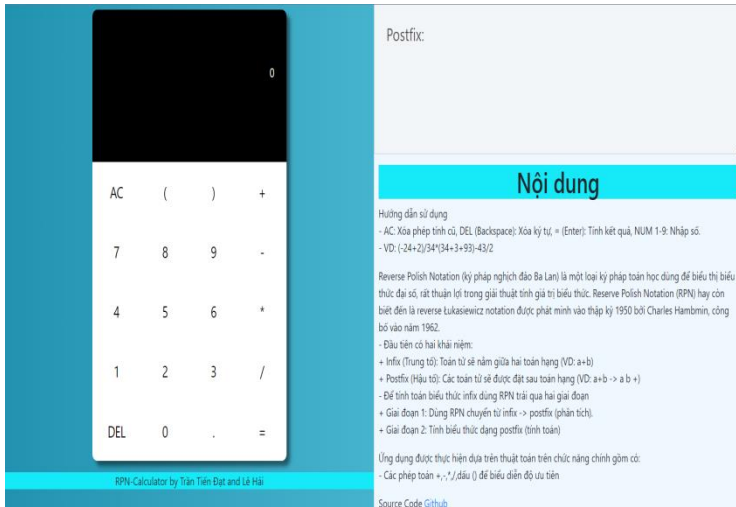
C. Doanh mục hàm JavaScript

Sinh viên phụ trách: Lê Hải

TT	Tên function	Mục đích	Tên file, số thứ tự chứa dòng báo cáo
1	cal()	-Hàm để thực thi các phím = , DEL,AC cũng như tạo hiệu ứng khi hiện kết quả	Calculator.js(11)
2	getDataToServer()	- Dùng để gửi dữ liệu lên với phương thức post. Hiện thị đáp số sau cùng và đáp số thuật toán RPN.	Calculator.js(49)
3	keydown()	-Giúp người dùng thực hiện đánh phím thay vì click	Calculator.js(78)
4	clearDisplay()	-Xóa hết tất cả những gì đã nhập trước đó	Calculator.js(64)
5	backSpaceDisplay()	-Xóa 1 phần tử bên góc trái ngoài	Calculator.js(68)

4. Thiết kế giao diện

TT	Thông tin	Mục đích	Nguyên nhân
----	-----------	----------	-------------

1	<p>Tên màn hình: Trang HOME</p> <p>Giao diện:</p> 	<p>Tạo ra 1 máy tính cho người dùng sử dụng và hiển thị biểu thức postfix như mong muốn</p>	<p>Lê Hải (Thiết kế)</p> <p>Lúc đầu nhóm chúng em đã tạo ra 1 cái giao diện calculator với các chức năng đơn giản để tính toán. Sau đó, vì để hiển thị kết quả của thuật toán + nội dung của bài, nên nhóm chia đôi màn hình.</p>
---	--	---	---

IV. CÀI ĐẶT VÀ KIỂM THỬ

Link Code: [Github](#)

Link Web (Khả dụng đến 28/11 do chính sách của Heroku):

[Máy tính online \(cal-rpn-itproject.herokuapp.com\)](http://cal-rpn-itproject.herokuapp.com)

TT	Tình Huống	Mục đích	Giải thích
1	<p>Dữ liệu vào: 10+2</p> <p>Kết quả dự kiến: 12.00</p>	<p>Kiểm tra tính năng cơ bản của máy tính</p>	<p>Phép tính cơ bản mà máy tính bắt buộc phải tính được</p>

2	Dữ liệu vào: -9-4 Kết quả dự kiến: -13.00	Xử lý số âm	Để kiểm tra số xem việc tách số và toán tử có chính xác không
3	Dữ liệu vào -10 -- 3 Kết quả dự kiến: -7	Xử lý số âm	Do hai dấu – nằm kề nhau nên phải kiểm tra việc tách số
4	Dữ liệu vào: 10+*5 Kết quả dự kiến: Lỗi	Kiểm tra tính hợp lệ của biểu thức	Biểu thức chính xác phải ở dạng trung tố gồm 2 toán hạng và 1 toán tử ở giữa
5	Dữ liệu vào: (14.5-43)/43 Kết quả dự kiến: -0.66	Xử lý phép tính thập phân	Máy tính xử lý toán hạng ở dạng thập phân và kết quả làm tròn đến số thứ 2
6	Dữ liệu đầu vào: $(-42+24-(325-23)/2+4)/12$ Kết quả dự kiến: -13.75	Xử lý số âm và phép toán có độ ưu tiên khác nhau	Yêu cầu máy tính phải xử lý được dấu – sau dấu (chính xác, phân tích đúng biểu thức dưới dạng postfix và làm tròn số thập phân thứ 2
7	Dữ liệu đầu vào: 340,282,346,638,528,860,000,000 * 340,282,346,638,528,860,000,000 Kết quả: Lỗi	Kiểm tra giới hạn tính	Do máy tính lưu kết quả ở kiểu Float Java (4 bytes) Giá trị lớn nhất biến float có thể lưu: 3.40282346638528860e+38
8	Dữ liệu vào: -340,282,346,638,528,860,000 * 340,282,346,638,528,860,000 Kết quả: lỗi	Kiểm tra giới hạn số âm	Kiểu float trong Java có thể lưu kiểu âm ở giới hạn là $1.4012985e-45$ (2^{-149})

--	--	--	--

V. KẾT LUẬN

Thông qua quá trình phân tích và thiết kế ứng dụng nhóm em đã hiểu rõ hơn về cách hoạt động của và tính toán các biểu thức phức tạp bên trong máy tính. Vận dụng được các kiến thức về cấu trúc dữ liệu và giải thuật, sử dụng các loại dữ liệu như mảng, ngăn xếp hàng đợi.

Ôn lại 4 tính chất của OOP (đóng gói, đa hình , kế thừa, trừu tượng). Tìm hiểu thêm về nguyên tắc thiết kế S.O.L.I.D trong lập trình để dễ dàng phát triển, chỉnh sửa code và áp dụng mẫu thiết kế nhà máy (factory design pattern). Sử dụng mô hình 3 lớp (Model-Controller-View) trong ứng dụng lập trình web. Các tính năng của ứng dụng cơ bản đã đáp ứng được của một ứng dụng máy tính cho người dùng. Mục tiêu đề ra ban đầu của nhóm coi như đã đáp ứng.

Bên cạnh đó việc tiếp cận nguyên tắc S.O.L.I.D trong việc phát triển ứng dụng với sinh viên vẫn còn khó khăn, quy mô ứng dụng còn hạn chế cho nên việc áp dụng vào vẫn chưa phát huy hết hiệu quả, kiến thức về design pattern yêu cầu người lập trình phải làm nhiều các ứng dụng trong thực tế để hiểu rõ khi nào nên áp dụng cho phù hợp. Nhóm em đã sửa phần thiết kế lớp trong suốt 4 tuần để hoàn thiện các tính năng ứng dụng. Để khắc phục các vấn đề trên nhóm đã tham khảo rất nhiều source code trên internet để tối ưu code xử lý.

Tổng kết lại ứng dụng của nhóm đã đáp ứng một số điểm sau: giao diện trực quan, hiện đại dễ tương tác, kết quả của các phép tính được làm tròn đến số thập phân thứ 2 nên độ chính xác khá cao. Tuy nhiên vẫn có một số điểm hạn chế như hiện nay các phép tính về lũy thừa (^) thực hiện vẫn chưa chính xác. Nhóm em đang tìm hiểu để khắc phục vấn đề này.

Nhóm xin được cảm ơn thầy đã theo dõi, hướng dẫn chỉ bảo cho phần đề tài của nhóm. Nhờ vào giúp đỡ của thầy mà nhóm đã hoàn thành được đồ án một cách trọn vẹn.

