Java

オリジナル電卓作成

(GUI 電卓)

内容

仕様：

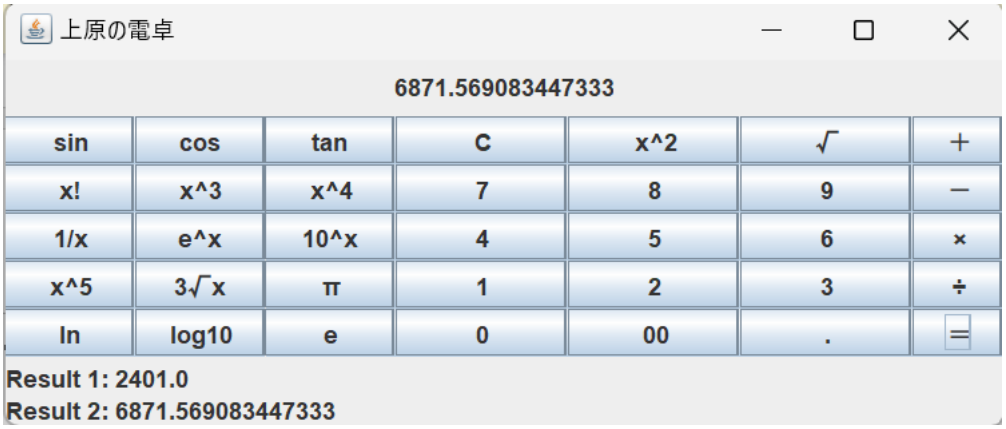
- フレームの上に計算結果を表示するラベルを設置、デフォルト値“0”
- 数字ボタン：0~9 と小数点
- 演算子ボタン：＋ － × ÷ ＝ C

演算仕様：

- 除数 0 の場合、表示結果を 0 とする

１、計算結果スクリーンショット

入力　5023117÷731



２、プログラム説明

2.1、クラスとメソッド構造

クラスの詳細

1. Calculator クラス

Frame を継承。計算機の全体的なレイアウトを管理し、主要な機能を提供する役割を担う。

2. Dentaku クラス

JFrame を継承し、電卓の GUI と機能を実装。numBTN および funBTN クラスを内部に持ち、それぞれのボタンの動作を定義している。

3. numBTN クラス

JButton を継承し、数字ボタンを表すクラス。ActionListener インターフェースを実装しており、ボタンが押されたときの動作を定義。

4. funBTN クラス

JButton を継承し、演算子や特殊機能ボタンを表すクラス。ActionListener インターフェースを実装しており、ボタンが押されたときの動作を定義。

カプセル化と属性変数

Dentaku クラスでは、電卓の動作に必要な変数をカプセル化して使用。これらの変数はクラス内部でのみ操作され、外部から直接アクセスできない。以下は主要な属性変数のリストと

その説明。

・label：現在の入力または計算結果を表示するための JLabel オブジェクト。
・resultLabel1：直前の計算結果を表示するための JLabel オブジェクト。
・resultLabel2：さらに前の計算結果を表示するための JLabel オブジェクト。
・operand1：演算の最初のオペランドを保持するための変数。
・operand2：演算の二番目のオペランドを保持するための変数。
・operator：現在の演算子を保持するための文字列変数。
・startNewNumber：新しい数値入力を開始するかどうかを示すブール値変数。
・justCalculated：直前に計算が行われたかどうかを示すブール値変数。
・isFirstResult：結果ラベル 1 または結果ラベル 2 に表示するかを示すブール値変数。

メソッド
Dentaku クラスには、以下の主要なメソッドがあります。

1. createDentaku
全体の電卓 GUI を構築するメソッド。表示パネル、数字パネル、演算子パネル、追加機能パネル、結果パネルのレイアウトを設定。

2. setDisplay
電卓の表示パネルにラベルを追加するメソッド。

3. setNumPanel
数字ボタンを配置するパネルを設定するメソッド。

4. setFunPanel
基本的な演算子ボタンを配置するパネルを設定するメソッド。

5. setAdvFunPanel
拡張機能の演算子ボタンを配置するパネルを設定するメソッド。

6. setResultPanel
結果を表示するラベルを配置するパネルを設定するメソッド。

7. updateResults
計算結果を結果ラベル 1 または結果ラベル 2 に更新するメソッド。

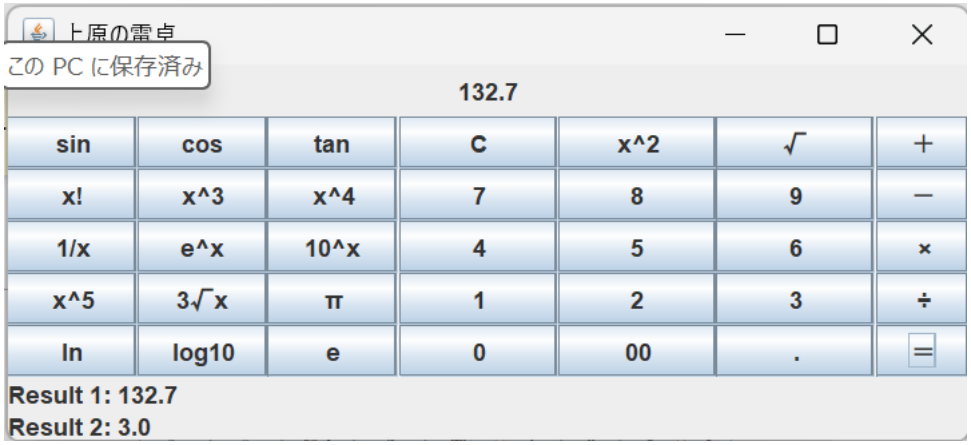
8. factorial
階乗を計算するためのメソッド。

以下は具体的なボタンの動作を定義する内部クラスのメソッド。

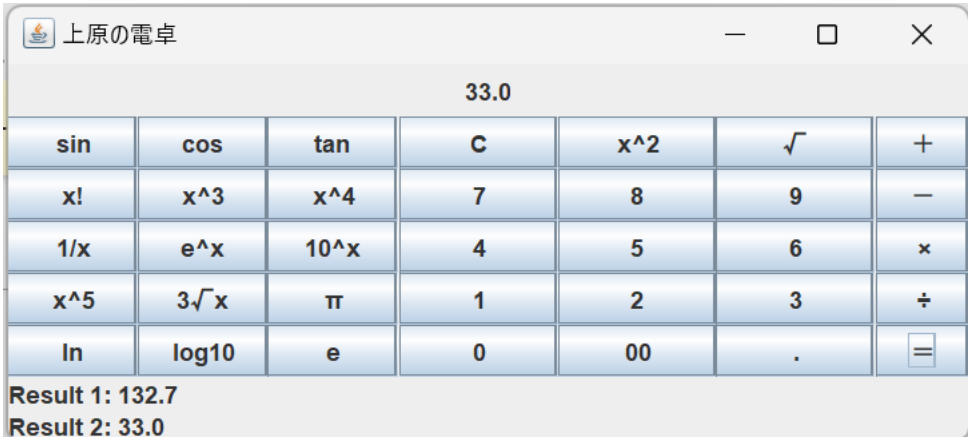
・numBTN クラスの actionPerformed メソッド
数字ボタンが押されたときの動作を定義している。現在の入力状態に応じてラベルのテキストを更新する。

・funBTN クラスの actionPerformed メソッド
演算子ボタンや特殊機能ボタンが押されたときの動作を定義する。各演算子や機能に応じた処理を行い、結果をラベルに表示する。

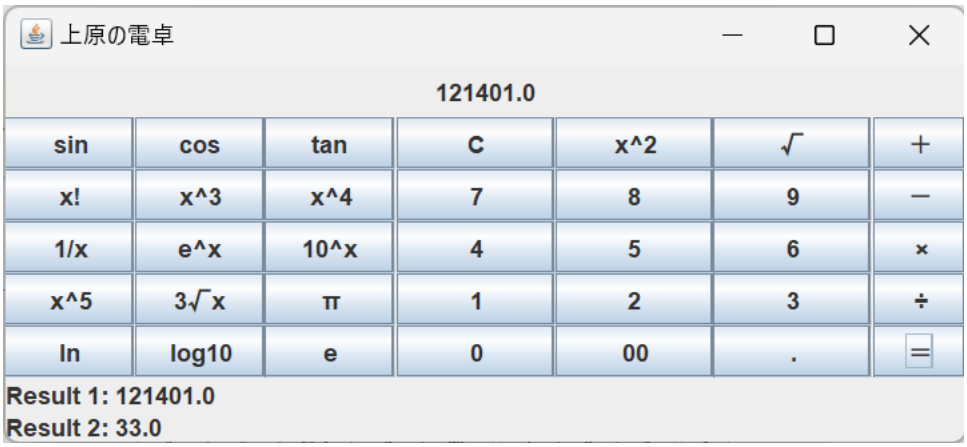
「各種演算子」
・“＋” 入力→45.7+87



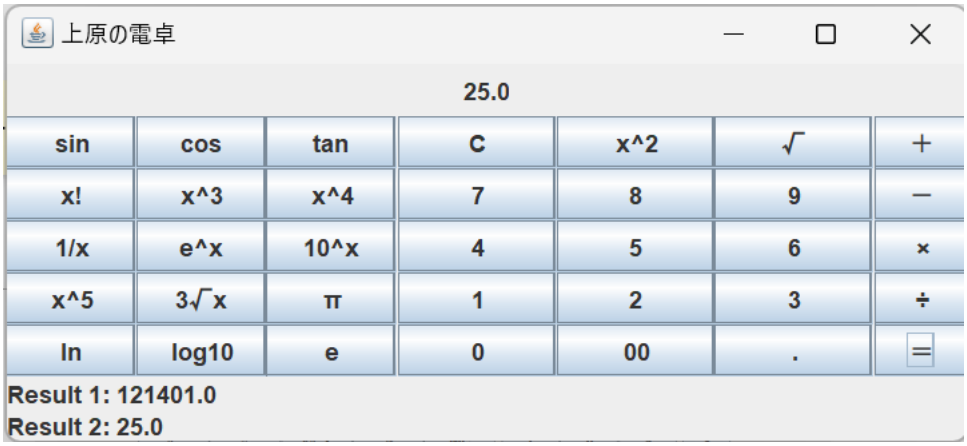
・“－” 入力→67—34



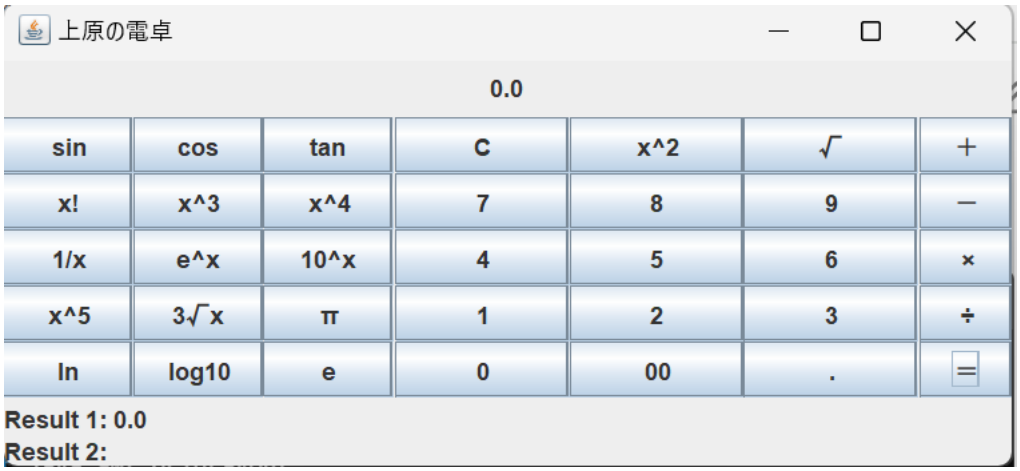
・“×”　入力→987×123



・“÷”　入力→625÷25



・“÷”　入力→33÷0



2.2、工夫点

・例えば、3＋4＝7 という計算をしたとする。その後に 6 を記入して＝を入力すると答えが 9 となった。これが意味する事は 6 と＝を押したら初期で設定し、残存していたオペランド 1 とオペレーターが勝手に機能してしまっていることである。新たな計算を開始したいのに最初の 3＋が残っているので、破棄したはずのものが残っている事になる。本当は、3＋4＝7 を計算し終わった後、6 を押したら自動的に初期の状態（ホーム状態）に戻らないといけない。つまり、計算し終わった後、数字を入れて＝を押したら入れた数字が表示されないといけない。
計算し終わった後に新たな数字を入れたらオペランド 1、オペレーター、オペランド 2 がリセットされるように修正した。具体的には justCalculated という新たなフラグを作った。これによって計算が終了した後、新たな数字が入力されたらこのフラグでチェックする。funBTN の＝の部分で startNewNumber と justCalculated が true の時、オペランド 1、オペレーター、オペランド 2 がリセットするようにした。
オペレーターである演算子を押したら自動的に現在表示されているラベルがオペランド 1 に格納されるようになっている。私が作った電卓の仕様は計算が終わってもオペレーターのどれかを押したら、その計算し終わった数字を再利用できるという物である。

・内部クラスを利用したボタン操作の統一
numBTN と funBTN の 2 つの内部クラスを作成し、それぞれ数字ボタンと機能ボタンの動作を統一的に管理した。これにより、ボタンごとの動作を明確に分け、コードの可読性とメンテナンス性を圧倒的に向上した。また、ボタンごとの動作を直接記述することで、if 文による複雑な分岐を避け、コード行数の節約にも役立った。

・拡張機能の追加を容易にする設計
拡張機能の部分のコードは、全て同じ形式で記述されているため、新たな機能の追加や既存

機能の編集がしやすい。例えば、新しい数学関数を追加する際にも、同じ形式で記述するだけで簡単に追加できる。将来的に機能を拡張する際の手間が省け、コードの一貫性が保たれ、可読性も高まる。

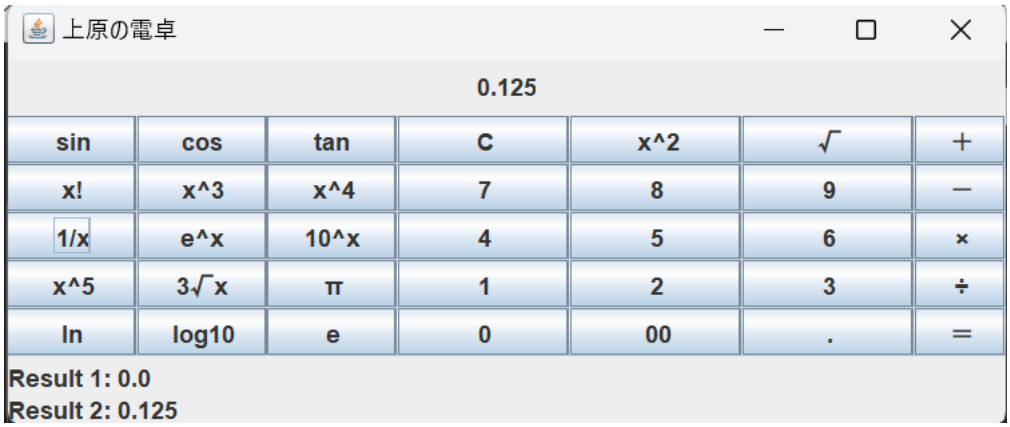
・各ボタンの動作をメソッド内に明確に分離
各ボタンの動作を個別のメソッド内で定義し、ボタンが押された際の処理を明確に分離した。これで各ボタンの機能を確認しやすい。

・エラーハンドリング
Try&catch ブロックで、操作ミスが発生した際にプログラムがクラッシュしないようにした。例えば小数点が 2 回押された場合は無視する処理を追加している。

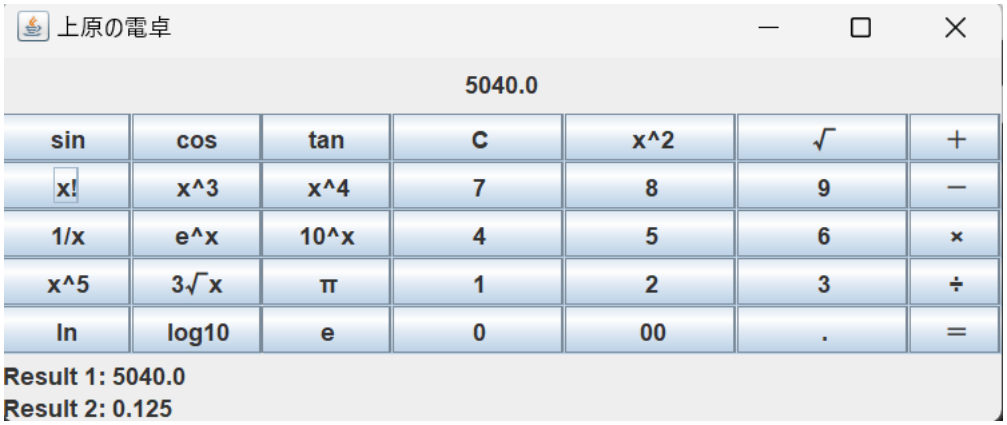
３、拡張課題の説明（あれば）

基本的な四則演算に加えて以下のような演算機能を追加した。全部で 18 個。
三角関数、累乗計算、階乗、逆数、指数関数、対数関数、立方根、特殊定数。
拡張機能の部分のコードは全て同じ形式で記述しているため、新たな機能の追加は比較的しやすかった。難しかった点は各数学関数の正確な計算を行うための実装である。どのパッケージを使えば良いのか、e やπなどの特殊定数はボタンを押せば値が表示されるようにするにはどうしたらよいのか。これらは、Java の標準ライブラリで既に提供数学関数を利用した。よって複雑な数学計算を比較的簡潔に実装できたと思う。
以下拡張機能のうち、代表的なものをスクリーンショットして貼る。

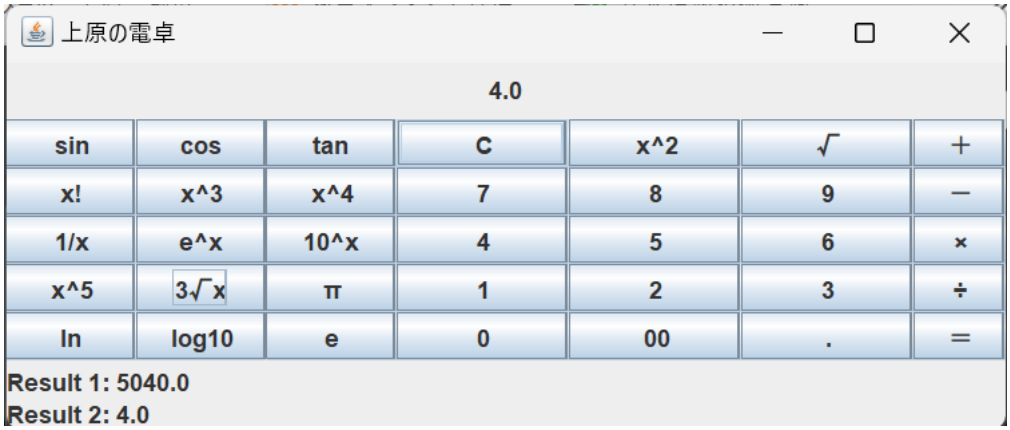
・“1/x”　入力→8



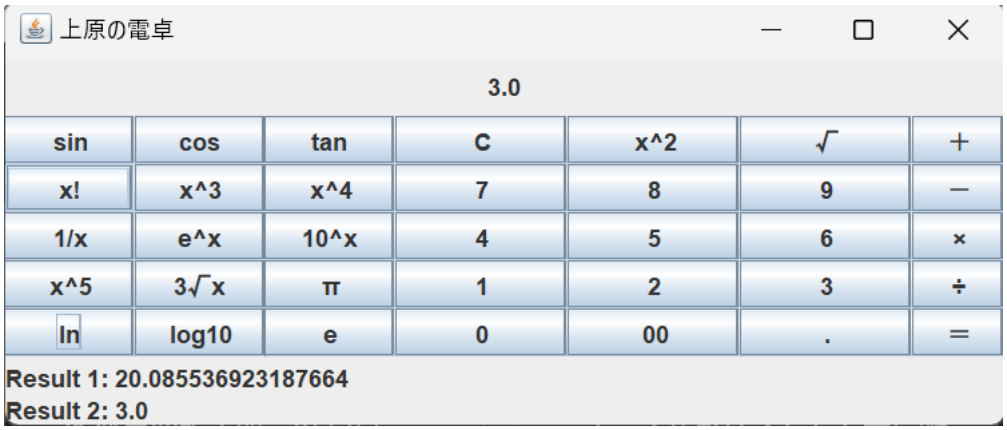
・“x!”　入力→7



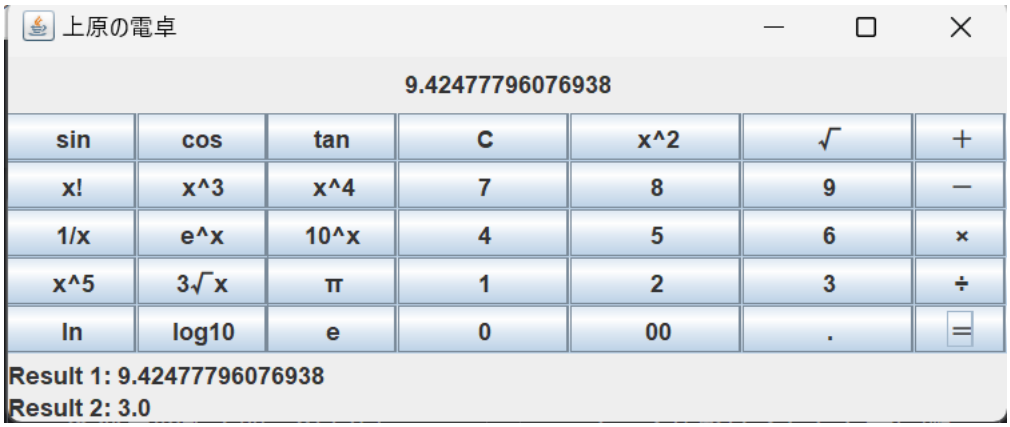
・“3√x”　入力→64



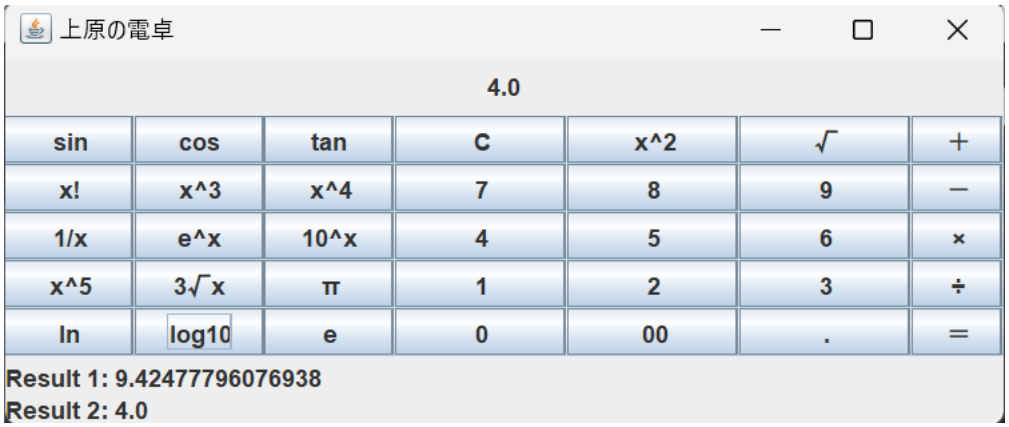
・“e”と“x^3”と“ln”　入力→e→x^3→ln



・“π”　半径 3 の円の面積



・“log10”　入力→10000



・“x^4”　入力→7