Clases de Inicialización

- Aplicación: clase principal que extiende Application de JavaFX y configura la ventana principal
- Launcher: punto de entrada que inicia la aplicación

Clase: Aplicación

Paquete: com.omarcisho.calculadorabasicaip

Descripción: clase principal que extiende Application de JavaFX y es responsable de iniciar la interfaz gráfica.

Métodos:

- **start(Stage stage):** método sobrescrito de la clase Application. Carga el archivo FXML de la interfaz, configura la escena y muestra la ventana principal.

Clase: Launcher

Paquete: com.omarcisho.calculadorabasicaip.App

Descripción: clase que actúa como punto de entrada de la aplicación.

Métodos:

- **main(String[] args):** método principal que inicia la aplicación llamando al método launch de Application con la clase Aplicacion.

Clase: EvaluadorJEP

Paquete: com.omarcisho.calculadorabasicaip.Model

Descripción: clase que se encarga de evaluar expresiones matemáticas utilizando la biblioteca JEP.

Métodos:

- **evaluaExpresion(String expression):** método estático que toma una expresión matemática como string, la evalúa y devuelve el resultado como string. Maneja errores sintácticos y matemáticos.
- **aplicaPorcentajes(String operacion):** método privado estático que procesa porcentajes en la expresión, convirtiéndolos a su forma decimal para su evaluación.

Clase: Operación

Paquete: com.omarcisho.calculadorabasicaip.Model

Descripción: clase de modelo que representa una operación matemática junto con su resultado para ser almacenada en el historial.

Atributos:

- **numeroOperacion:** número entero que identifica la operación en el historial.
- **operacion:** String que contiene la expresión matemática.
- **resultado:** String que contiene el resultado de la operación.

Métodos:

- **Constructor Operacion(int, String, String**): inicializa una nueva instancia con un número de operación, la expresión y su resultado.

- **getNumeroOperacion():** devuelve el número de operación.

- **getOperacion():** devuelve la expresión matemática.

- **getResultado():** devuelve el resultado de la operación.

Clase: Controlador

Paquete: com.omarcisho.calculadorabasicaip.Controller

Descripción: clase controlador principal que extiende EvaluadorJEP y maneja la interacción entre la interfaz gráfica y el modelo. Gestiona eventos de usuario y actualiza la vista.

Atributos:

- **Elementos FXML:** referencias a componentes de la interfaz como tablas, columnas y áreas de texto.
- **historialDeOperaciones:** lista observable para almacenar el historial de operaciones.
- **Flags de control:** variables booleanas para controlar estados como punto decimal, resultado y operación perpetua.
- **Variables de control:** almacenan información sobre la última operación e índices.
- **StringBuilders:** para controlar y construir la operación paso a paso.

Métodos principales:

colocaOperacionEnPantalla(): actualiza el área de texto con la operación actual.

colocaCaracterDesdeBoton(Button): añade un carácter desde un botón presionado.

botonNumericoPresionado(ActionEvent): maneja eventos de botones numéricos.

botonOperadorAritmeticoPresionado(ActionEvent): maneja eventos de botones de operadores.

limpiaEntrada(): limpia el text area de pantalla

borraCaracter(): elimina el último carácter de la entrada.

colocaPuntoDecimal(): añade un punto decimal a la operación.

evalua(): evalúa la expresión matemática actual y muestra el resultado.

initialize(): inicializa la tabla de historial.

annadeOperacionAlHistorial(String, String): añade una operación y su resultado al historial.

inicializarTablaDeHistorial(): configura la tabla de historial.

buscaUltimaOperacion(String): encuentra el índice del último operador en una expresión.

eliminaPrimerNumero(String): elimina el primer número de una expresión.

borraUltimaOperacion(): elimina la última operación ingresada.

extraeParteDecimalYEntera(): separa y maneja las partes decimal y entera de un número.

Flujo de Trabajo

1. Launcher inicia la aplicación.
2. Aplicacion carga la interfaz gráfica desde el archivo FXML.
3. El Controlador inicializa la interfaz y establece los manejadores de eventos.
4. El usuario interactúa con la interfaz presionando botones.
5. El Controlador recibe eventos, actualiza la vista y utiliza EvaluadorJEP para procesar operaciones.
6. Los resultados se muestran en la interfaz y se almacenan en el historial utilizando la clase Operacion.
7. Este diseño MVC permite un mantenimiento más sencillo, separando claramente las responsabilidades de cada componente del sistema.