Laboratorio N°2 - Operaciones con Matrices y Algoritmos

Programación Avanzada

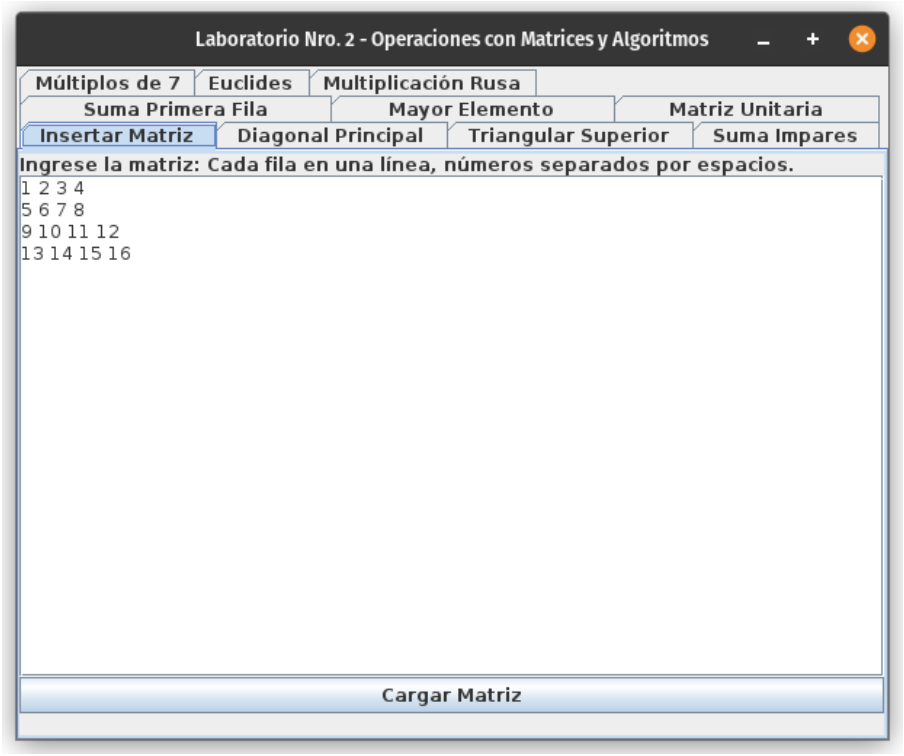
Nombres	Apellidos	Carnet	Iniciales
Andres Humberto	Chirinos Lizondo	13280260	C

1 Marco Práctico

1.1 Introducción

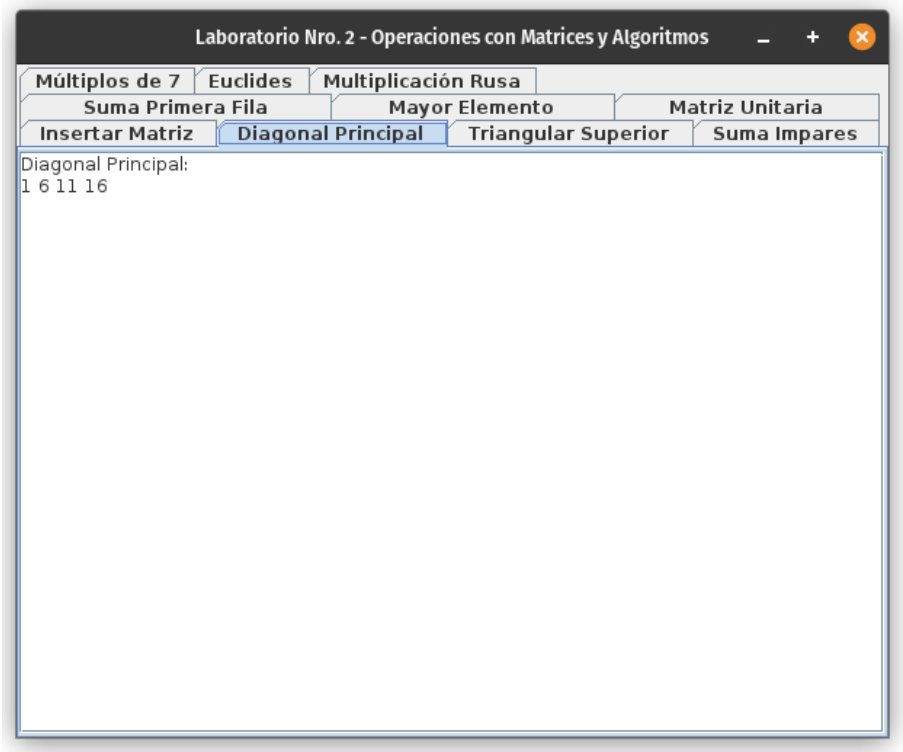
Tomemos la siguiente matriz de referencia para los siguiente algoritmos

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$



1.2 Mostrar los elementos de la diagonal principal de una Matriz cuadrada

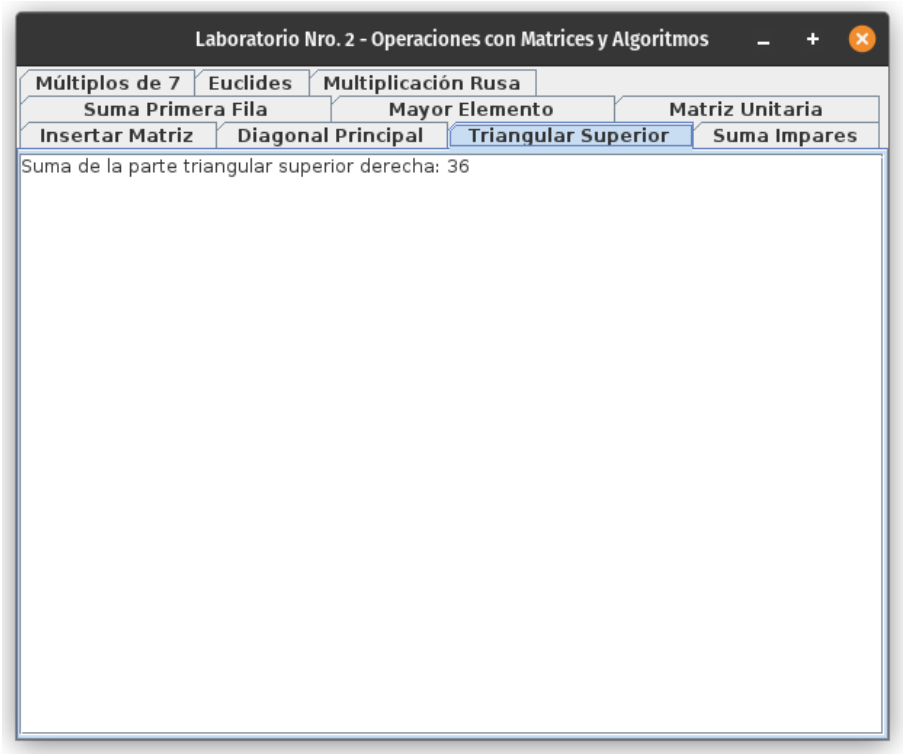
```
private void actualizarDiagonal() {
    StringBuilder salida = new StringBuilder("Diagonal Principal:\n");
    for (int i = 0; i < matriz.length; i++) {
        salida.append(matriz[i][i]).append(" ");
    }
    diagTextArea.setText(salida.toString());
}
```



1.3 Sumar los elementos de la triangular superior derecha

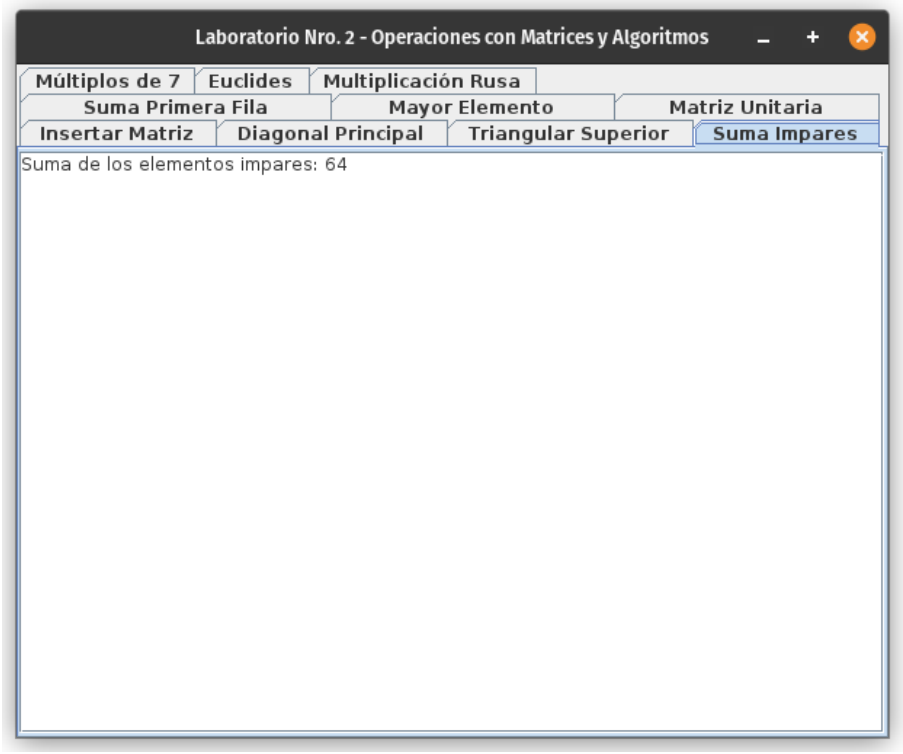
```
private void actualizarTriangularSuperior() {
    int suma = 0;
    for (int i = 0; i < matriz.length; i++) {
        for (int j = i + 1; j < matriz[i].length; j++) {
            suma += matriz[i][j];
        }
```

```
    }
    triSupTextArea.setText("Suma de la parte triangular superior derecha: " + suma);
}
```



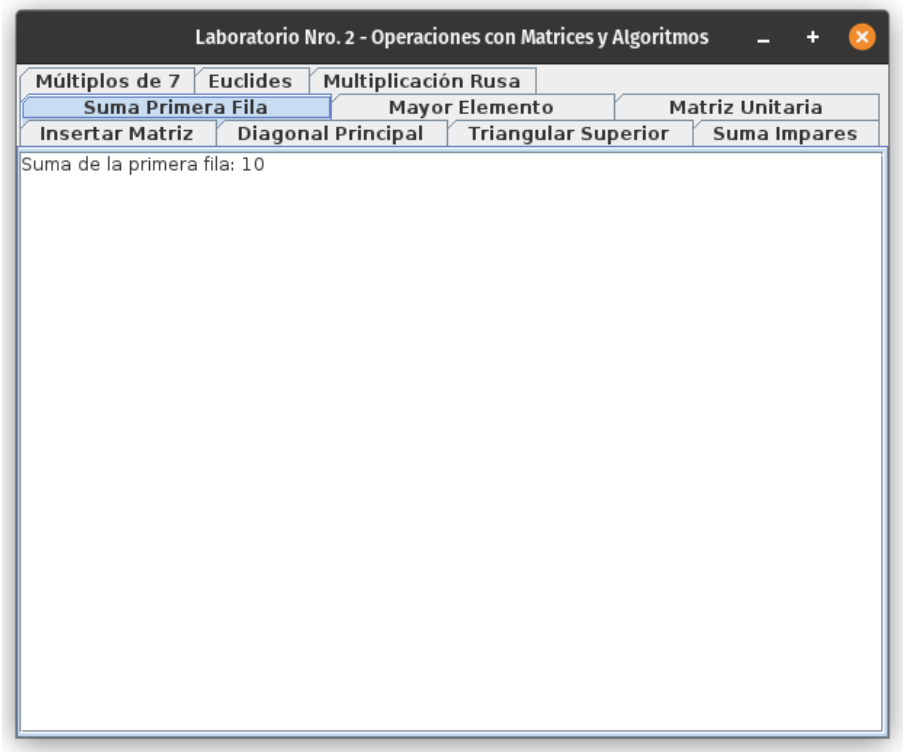
1.4 Sumar los elementos impares de la matriz

```
private void actualizarSumaImpares() {
    int suma = 0;
    for (int[] fila : matriz) {
        for (int valor : fila) {
            if (valor % 2 != 0) {
                suma += valor;
            }
        }
    }
    sumaImparesTextArea.setText("Suma de los elementos impares: " + suma);
}
```



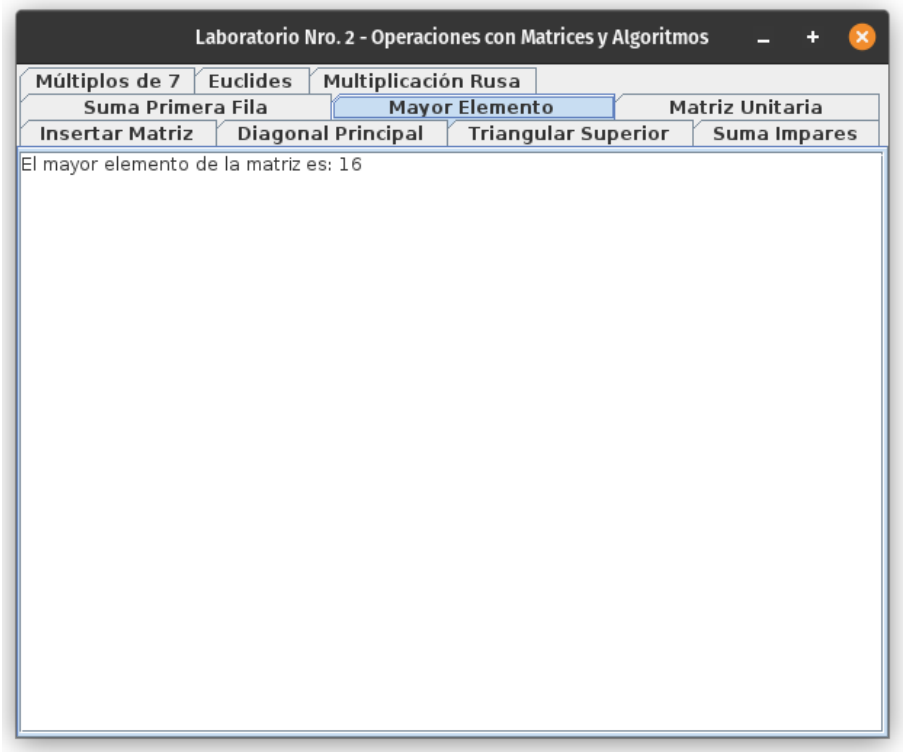
1.5 Sumar los elementos de la primera fila

```
private void actualizarSumaPrimeraFila() {
    int suma = 0;
    for (int valor : matriz[0]) {
        suma += valor;
    }
    sumaPrimeraFilaTextArea.setText("Suma de la primera fila: " + suma);
}
```



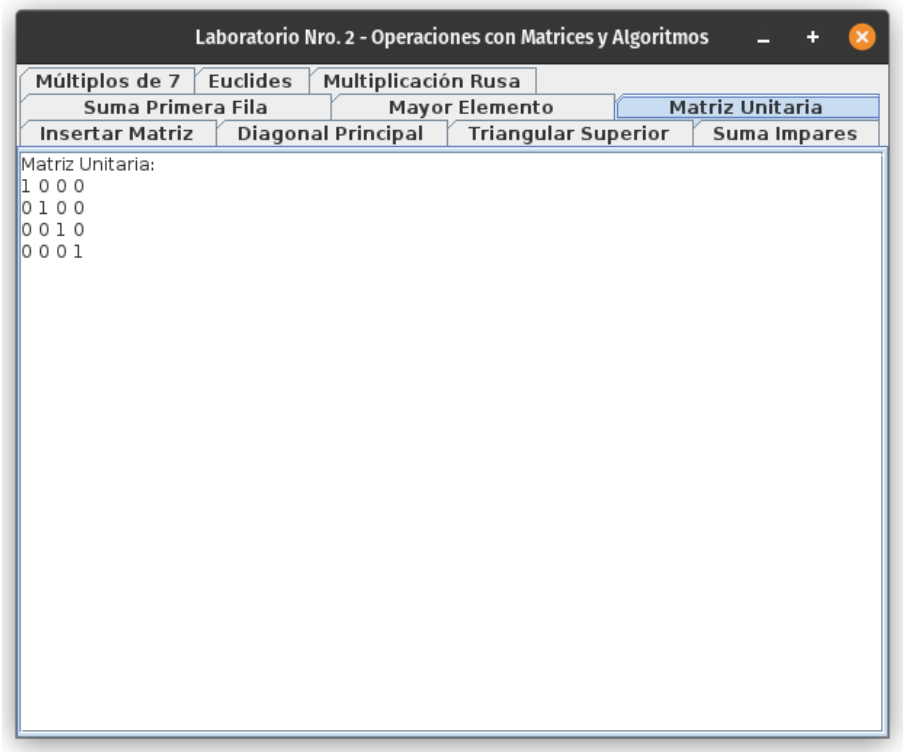
1.6 Mostrar el mayor elemento de la matriz

```
private void actualizarMayorElemento() {
    int mayor = matriz[0][0];
    for (int[] fila : matriz) {
        for (int valor : fila) {
            if (valor > mayor) {
                mayor = valor;
            }
        }
    }
    mayorElementoTextArea.setText("El mayor elemento de la matriz es: " + mayor);
}
```



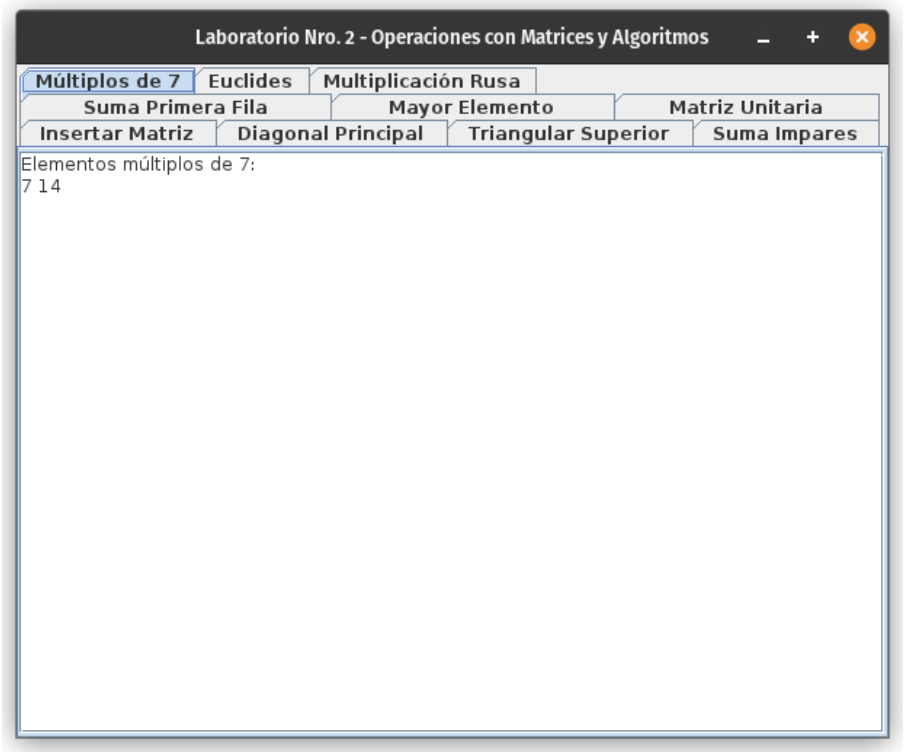
1.7 Generar la matriz unitaria

```
private void actualizarMatrizUnitaria() {
    int n = matriz.length;
    int[][] identidad = new int[n][n];
    for (int i = 0; i < n; i++) {
        identidad[i][i] = 1;
    }
    StringBuilder salida = new StringBuilder("Matriz Unitaria:\n");
    for (int[] fila : identidad) {
        for (int valor : fila) {
            salida.append(valor).append(" ");
        }
        salida.append("\n");
    }
    matrizUnitariaTextArea.setText(salida.toString());
}
```



1.8 Mostrar los elementos múltiplos de 7 de una matriz

```
private void actualizarMultiplosSiete() {
    StringBuilder salida = new StringBuilder("Elementos múltiplos de 7:\n");
    for (int[] fila : matriz) {
        for (int valor : fila) {
            if (valor % 7 == 0) {
                salida.append(valor).append(" ");
            }
        }
    }
    multiplosSieteTextArea.setText(salida.toString());
}
```

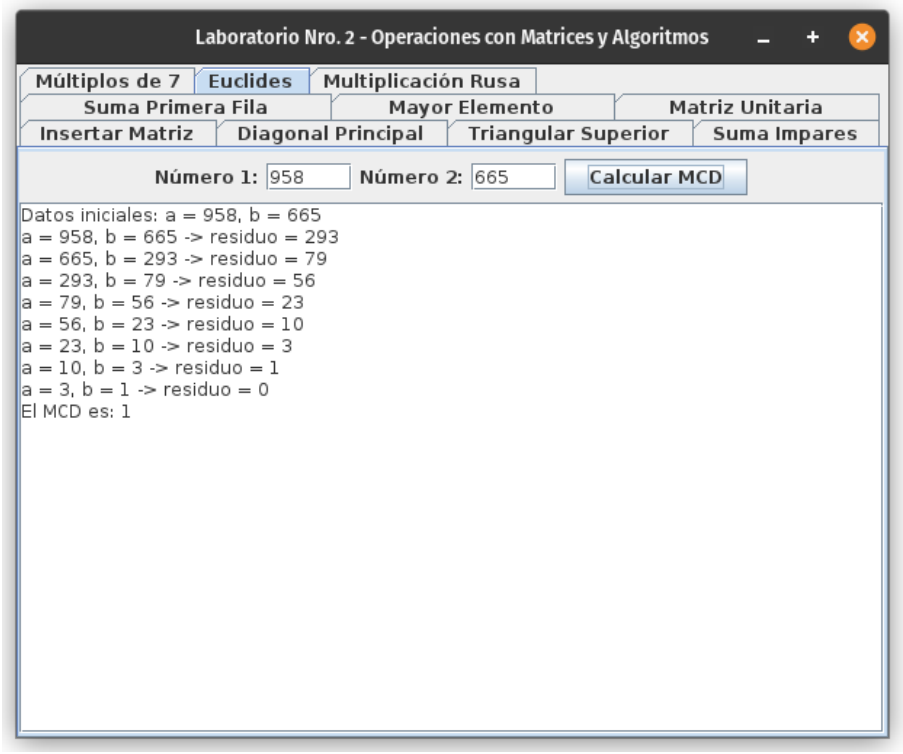


1.9 Algoritmo de Euclides

Para este algoritmo tome en consideración el siguiente caso de prueba $a = 958 \quad b = 665$

```
private String calcularMCD(int a, int b) {
    StringBuilder proceso = new StringBuilder();
    proceso.append("Datos iniciales: a = ").append(a).append(", b = ")
    .append(b).append("\n");
    while (b != 0) {
        int residuo = a % b;
        proceso.append("a = ").append(a).append(", b = ").append(b)
                .append(" -> residuo = ").append(residuo).append("\n");
        a = b;
        b = residuo;
    }
    proceso.append("El MCD es: ").append(a);
    return proceso.toString();
}
```

Knuth (1997)



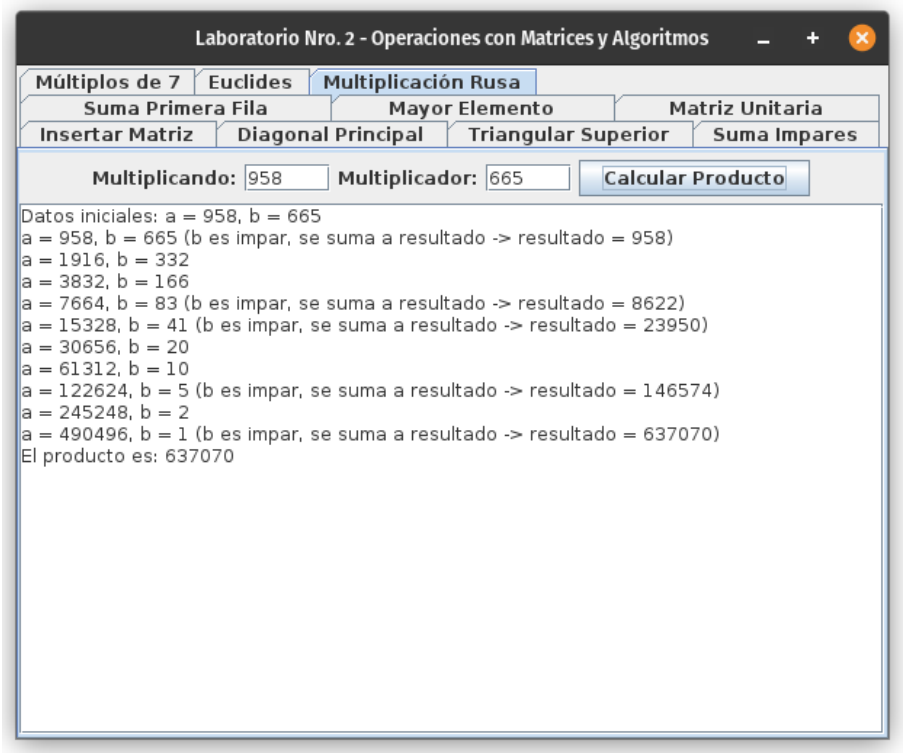
1.10 Algoritmo de la multiplicación Rusa

Para este algoritmo tome en consideración el siguiente caso de prueba $a = 958 \quad b = 665$

```
private String multiplicacionRusaConProceso(int a, int b) {
    StringBuilder proceso = new StringBuilder();
    proceso.append("Datos iniciales: a = ").append(a).append(", b = ")
    .append(b).append("\n");
    int resultado = 0;
    while (b > 0) {
        proceso.append("a = ").append(a).append(", b = ").append(b);
        if (b % 2 != 0) {
            resultado += a;
            proceso.append(" (b es impar, se suma a resultado -> resultado = ")
            .append(resultado).append(")");
        }
        proceso.append("\n");
        a *= 2;
        b /= 2;
    }
```

```
    proceso.append("El producto es: ").append(resultado);
    return proceso.toString();
}
```

Knuth (1997)



Bibliografia

Knuth, Donald E. 1997. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. 3rd ed. Addison-Wesley. https://www-cs-faculty.stanford.edu/~knuth/taocp.html.

Apendice

Los codigos usados se encuentran en el repositorio público https://github.com/andres-chirinos/notes