**Método de ingeniería**

Algoritmos y Estructuras de Datos

# Identificación del problema

## Contextualización

Venus y Marte están en guerra. La Tierra, como aliada de Marte, ha sido elegida para ayudarle en esta guerra. Se sabe que Marte tiene naves escondidas, cuya ubicación (en X,Y) solo puede revelarse por medio de una técnica matemática llamada “Multiplicación de matrices”.

## Problema

Se requiere encontrar una forma de poder multiplicar una secuencia de matrices eficientemente.

## Necesidades

1. Se debe tener una interfaz que permita ingresar las dimensiones de las matrices (El campo de batalla).
2. Se necesita que haya una forma de generar valores aleatorios en cada posición de la matriz.
3. Se requiere que sea permitido crear una secuencia de matrices para multiplicarse.
4. Se solicita que haya una interfaz que muestre el resultado de dicha multiplicación para ver la ubicación de las tropas de Marte.

# Recopilación de información

Con el objetivo de comprender como cumplir con las necesidades expuestas, se requiere de una investigación de conceptos. De esta forma, proporcionar una mejor visión del problema, logrando entender matemáticamente lo que hay que hacer, cómo hacerlo, y las limitaciones que se deben respetar/cumplir para lograrlo.

* Matriz: “En general, una matriz es un conjunto ordenado en una estructura de filas y columnas” (Definición de Matriz, 2004) La cantidad de filas y columnas pueden variar. Sin embargo, “Decimos que una matriz es de orden m x n (o de dimensión m x n) cuando tiene m filas y n columnas.” (Matemáticas IES, s.f.)
* Multiplicación de matrices: Es una operación matemática donde cogemos las filas de una matriz A y las multiplicamos por las columnas de una matriz B. Es muy importante entender que “Usted solo puede multiplicar dos matrices si sus dimensiones son compatibles , lo que significa que el número de columnas en la primera matriz es igual al número de renglones en la segunda matriz” (Varsity Tutors , s.f.).

# Búsqueda de soluciones creativas

## Lluvia de ideas

Dado a que nuestro problema es un problema estrictamente matemático, se realizaron diferentes búsquedas por la web sobre las diferentes técnicas para multiplicación de matrices y aspectos para tener en cuenta, y se llegó a que:

* Se plante el uso de una interfaz gráfica intuitiva que muestre de manera gráfica cada una de las matrices a ser multiplicadas simultáneamente, esto es; la matriz de “Batalla Pasada” por la matriz de “Coeficientes”, igual a la matriz de “Batalla Actual”.
* Se propone el uso de dos métodos que se encarguen del problema por aparte para luego juntarlos y dar una solución. El primer método, recibirá las matrices a multiplicar y comprobará si es posible o no multiplicarlas aplicando la propiedad matemática de la multiplicación de matrices mencionada anteriormente. El segundo, efectuara la operación matemática multiplicando cada fila por cada columna respectivamente.
* Se propone la implementación de un método adicional que sea utilizado únicamente para el caso especial de multiplicación de matrices por escalar, este debe ser optimizado en términos de su complejidad temporal dado a la simplicidad el caso.
* Para la generación de números aleatorios que conformaran cada termino de cada una de las matrices se encontraron las siguientes alternativas:
  + La clase *java.util.Random*
  + La clase *java.security.SecureRandom*
* Para el uso de números aleatorios sin repetir, se propones dos alternativas:
  + Almacenar los números previamente generados en una estructura de datos, creando así una base de números para que al momento de la generación de un nuevo entero este sea verificado y se descarte o utilice respectivamente.
  + Tener un rango de números enteros ordenados para luego desordenarlos y poder utilizarlos de manera aleatoria.
* Para reducir la cantidad de métodos utilizados, se podría llevar a cabo la comprobación en el mismo método que las multiplica y la multiplicación misma en este método. Sin embargo, la idea es tratar de usar la menor cantidad de líneas de código posible así que la comprobación debe ser eficiente.

# Transición de las Ideas a los Diseños Preliminares

## Alternativas rechazadas

Lo primero que logramos decidir fue rechazar la idea de hacer todo en un mismo método. Esto se debe a que, si termináramos usando un solo método para resolverlo todo, no estaríamos empleando la división de responsabilidades. Por esta razón, la idea de hacer la comprobación y otras funciones en el mismo algoritmo queda rechazada.

Adicionalmente, decidimos no llevar a cabo la idea de tener un rango de números enteros ordenados para luego desordenarlos y poder utilizarlos de manera aleatoria. Esto, como consecuencia de una revisión del gasto que conlleva tener un arreglo que deba ser desordenado antes de ser utilizado.

Por otra parte, descartamos también la idea de utilizar *java.security.SecureRandom* porque no es necesario generar números criptográficamente seguros. El problema no requiere del mismo tipo de seguridad que requeriría una contraseña o cosas de ese calibre. Por lo anterior, decidimos que no hacía falta usar dicha clase.

## Alternativas aceptadas

* Habla de los pro y contras de la a) en la fase 3 (NICOLÁS)
* El uso de dos métodos para la multiplicación nos ayuda a cumplir con el objetivo de la división de responsabilidades. Sin embargo, podría generar algunos contratiempos si no son declarados eficientemente. Esto se debería a la cantidad de instrucciones que manejaría cada método y su respectivo gasto de memoria.
* Encontramos que para lograr obtener nuestros números aleatorios sería correcto usar la clase *java.util.Random,* pues ya sabemos cómo funciona y lleva a cabo la tarea requerida sin ningún inconveniente. Esto se debe a que necesitamos únicamente números enteros positivos para llevar a cabo la solución.

# Evaluación y Selección de la Mejor Solución

# Preparación de Informes y Especificaciones

# Implementación del Diseño