**PRÁCTICA**

**OBLIGATORIA**

03179150 K

Tsolanocampos2@gmail.com

**PRÁCTICA**

**OBLIGATORIA**

**PARA EL**

**CURSO 2023/2024:**

Tomas Solano Campos



Índice:

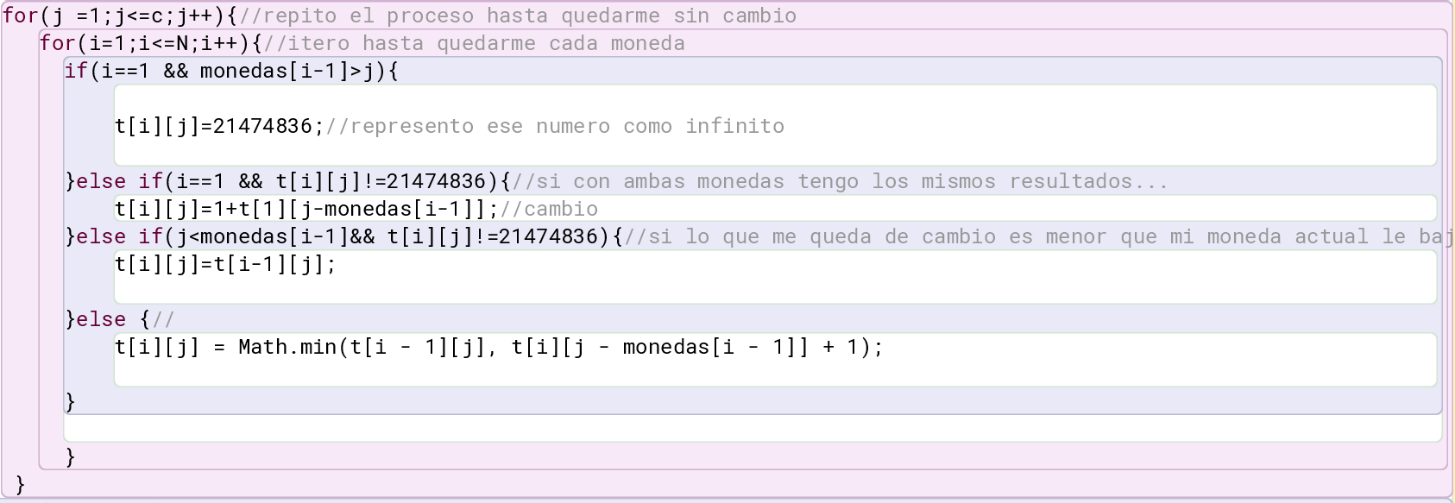
Contenido

[1. CUESTIONES TEORICAS 3](#_Toc155113425)

[2. Ejemplos de ejecución para distintos tamaños del problema 4](#_Toc155113426)

# CUESTIONES TEORICAS

1. Indica y razona sobre el coste temporal y espacial del algoritmo

Para calcular el coste tendremos que fijarnos en los métodos de darCambio. Concretamente en esta parte:

Las operaciones aritméticas tienen valor constante (se marcan en verde),estás tienen un coste de O(1),es decir son constantes. En rojo podemos ver el for de adentro ,este se repetirá hasta N es decir se repetirá hasta el numero de monedas que tengo.

Siendo el coste del problema el numero de iteraciones que pasa hasta que da el cambio j.

Por último, si nos fijamos en lo azul ,veremos otro for ,este for tiene el propósito de repetir el proceso hasta haber dado el cambio oportuno. El primer for tiene un coste de O(N) ,pero el segundo tiene un coste distinto ,ya que el numero de iteraciones no dependerán de las monedas ,sino del cambio disponible C ,a más cambio más iteraciones del programa, por ello el coste se sería O(C).Formando una matriz O(NxC).

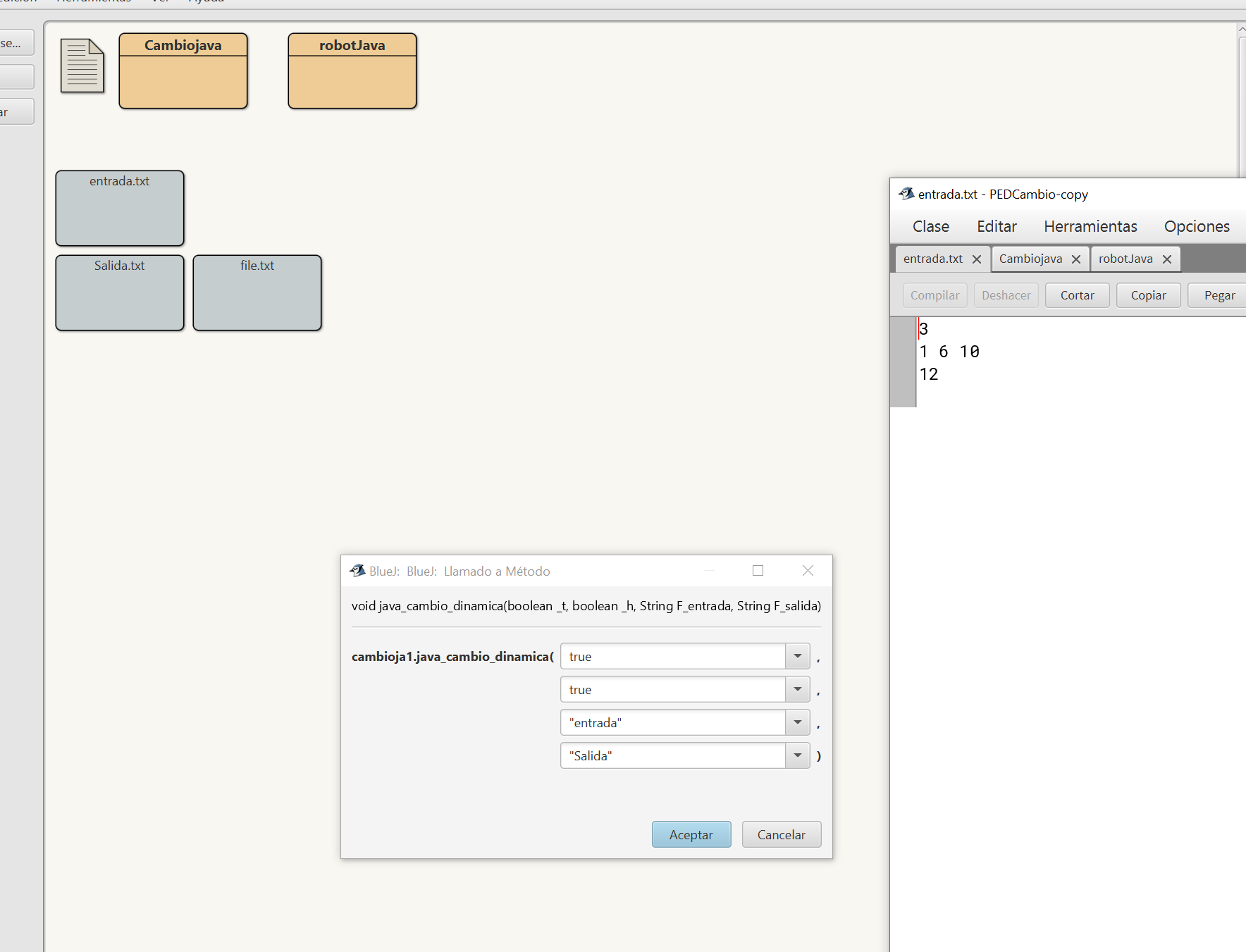
1. Explica qué otros esquemas pueden resolver el problema y razona sobre su idoneidad.

El esquema más eficiente es el usado, sin embargo, el esquema de vuelta atrás o el de Ramificación y poda también podrían sacar una solución factible.

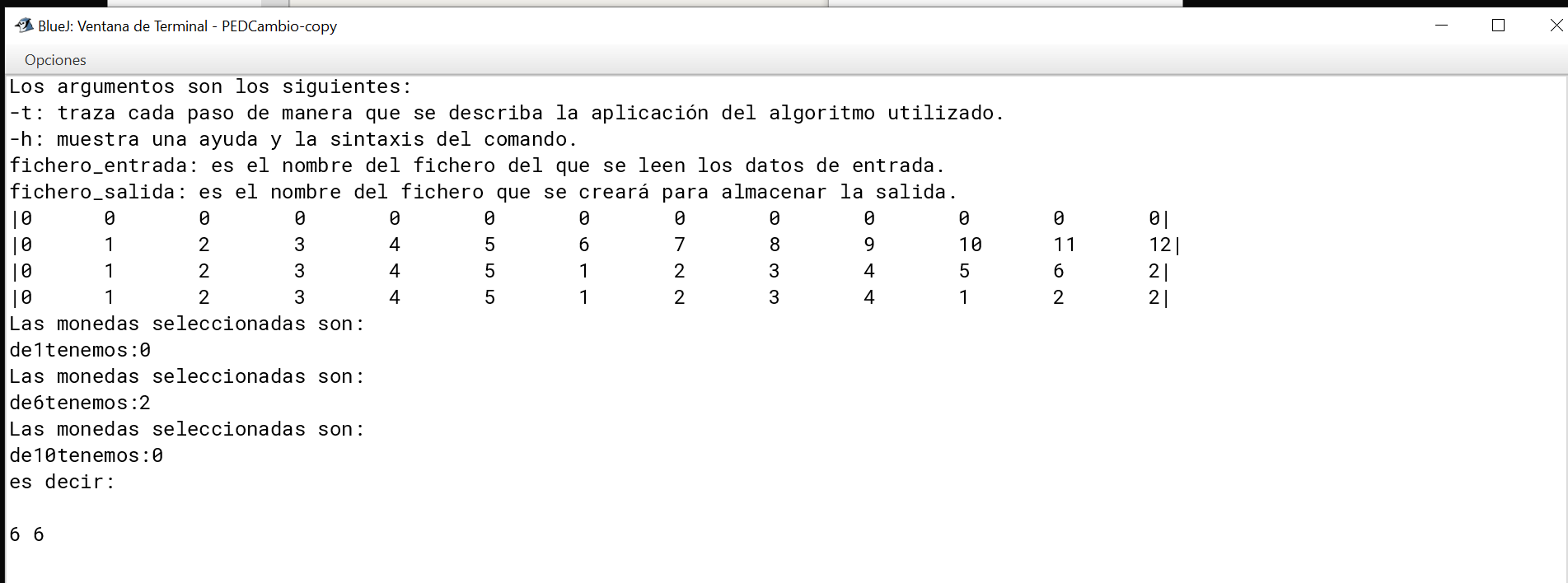
Vuelta Atrás al hacer un recorrido en profundidad, recorrerá todas las posibilidades disponibles, por ende ,acabará encontrando la solución mucho después que la programación dinámica o la ramificación y poda. Mientras que el esquema de ramificación y poda ,sería más eficiente que el de vuelta Atrás si se introducen buenas reglas .Se podrían podar todas las iteraciones que dieran infinito, por ejemplo ,poner reglas de poda más restrictivas arriesgaría la probabilidad de éxito y eso no nos interesaría.

# Ejemplos de ejecución para distintos tamaños del problema

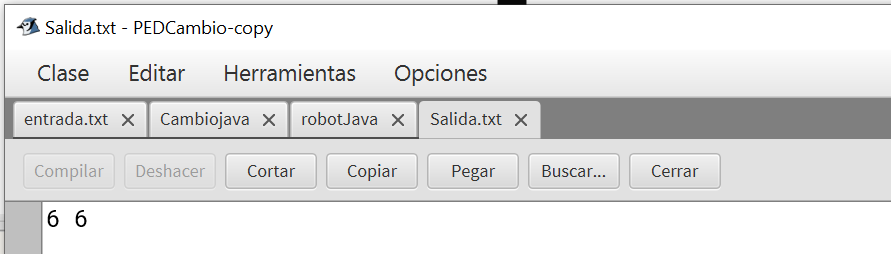
Iteración 1 (Prueba de un archivo coherente usando entradas y salidas)



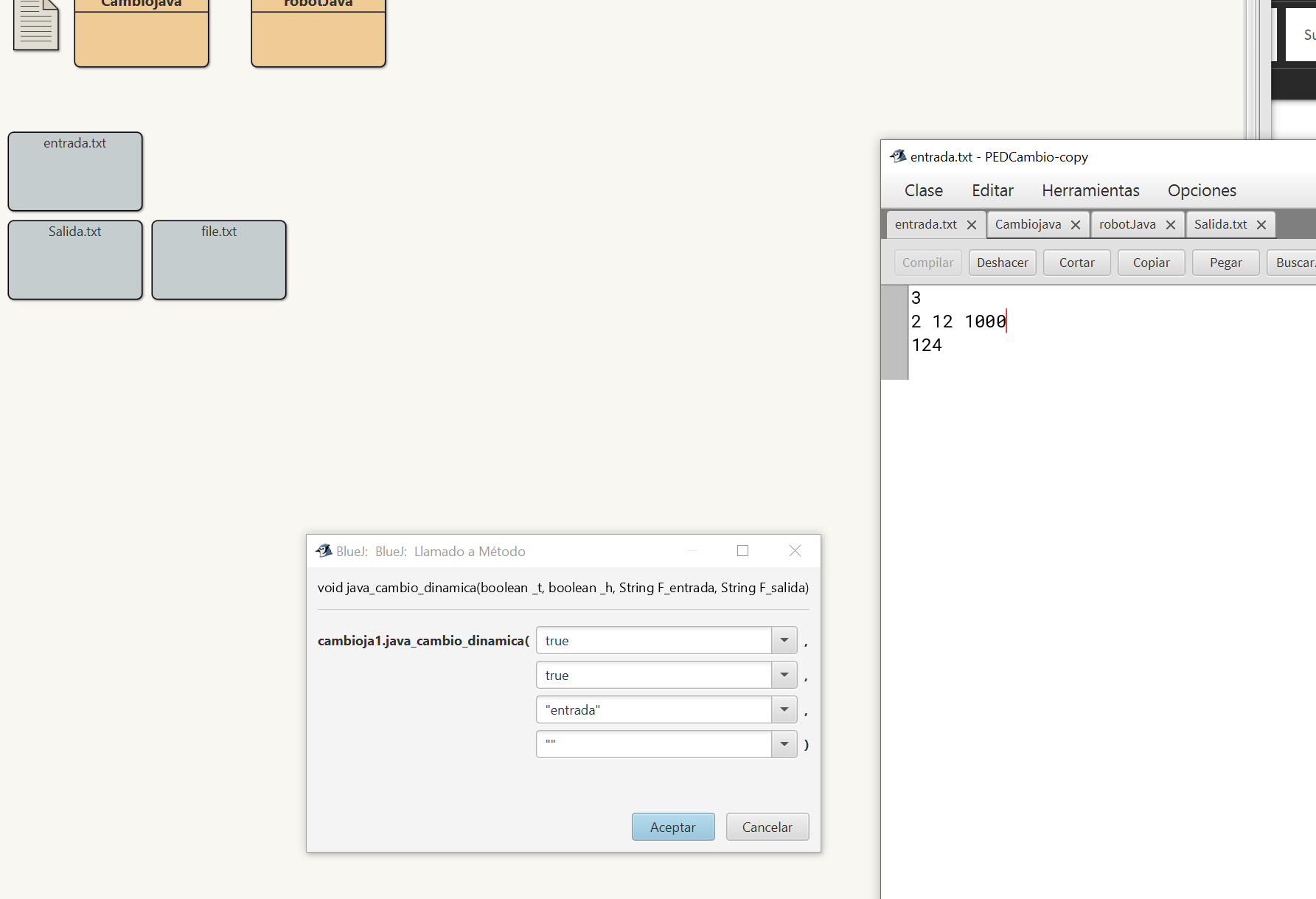
Resultado en el terminal:



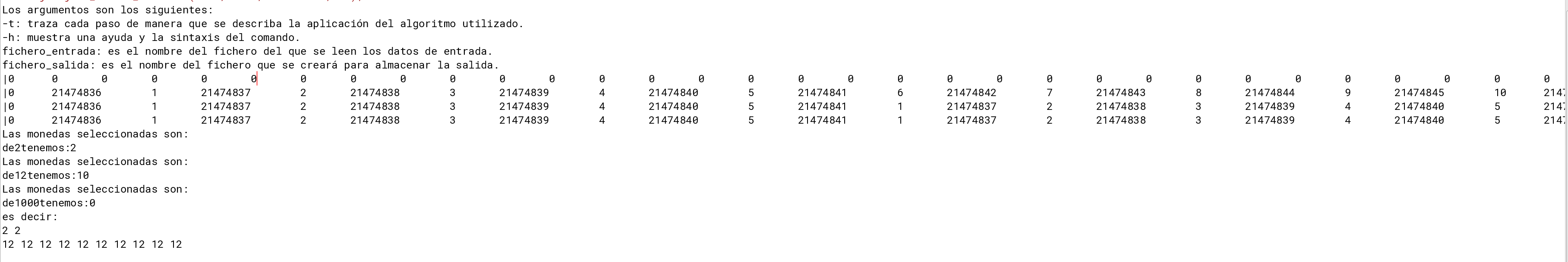
Resultado en el archivo de texto:



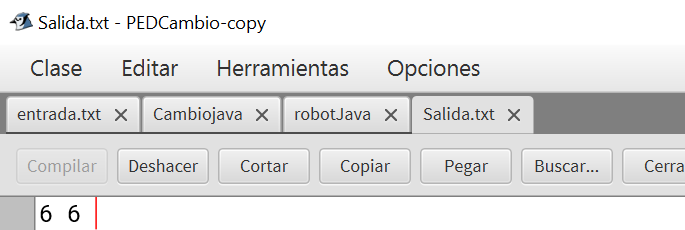
Iteración 2(Prueba de un archivo coherente usando solo el fichero de entrada)



Resultado en terminal:

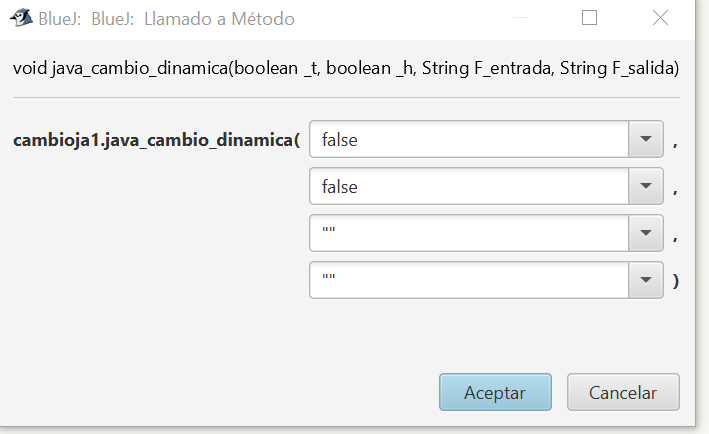


Resultado en salida:

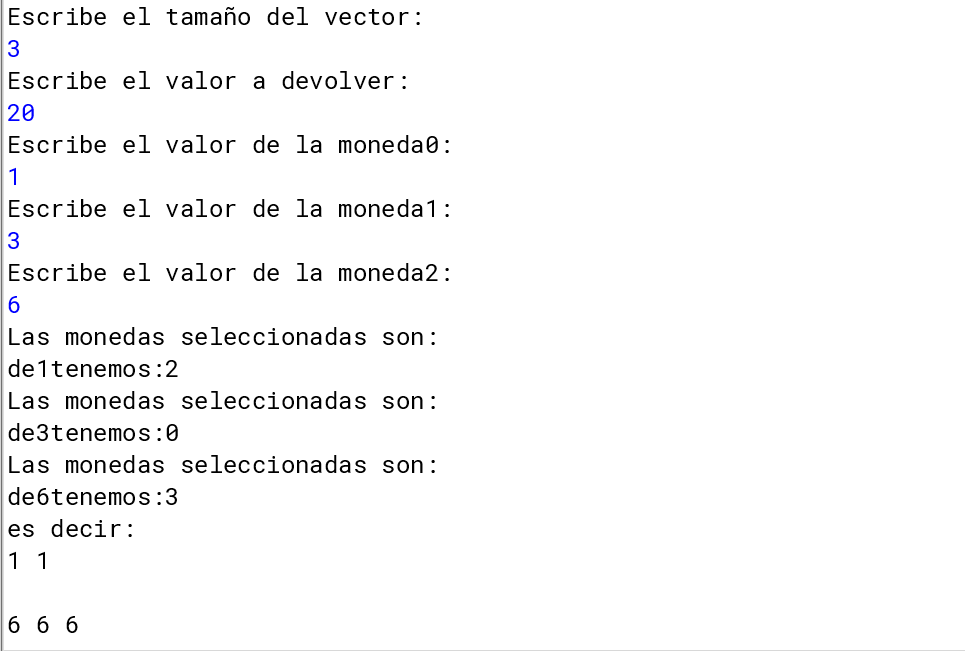


Como se puede ver en la salida queda el resultado anterior, puesto que no se ha tocado en ningún momento

Iteración 3(Prueba sin usar ningún archivo, además sin usar trazas ni ayudas)

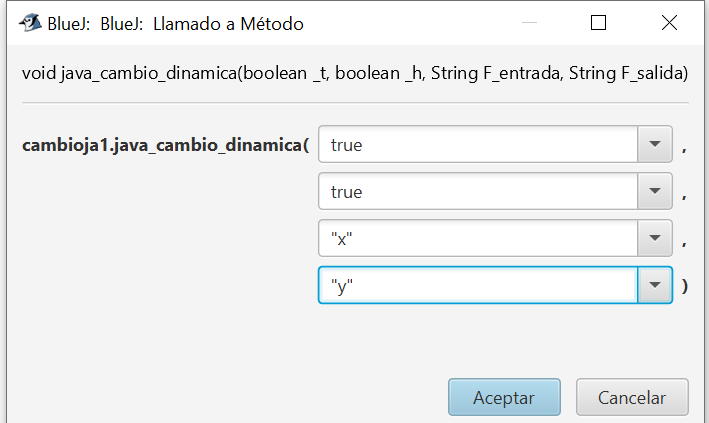


Resultado en terminal:

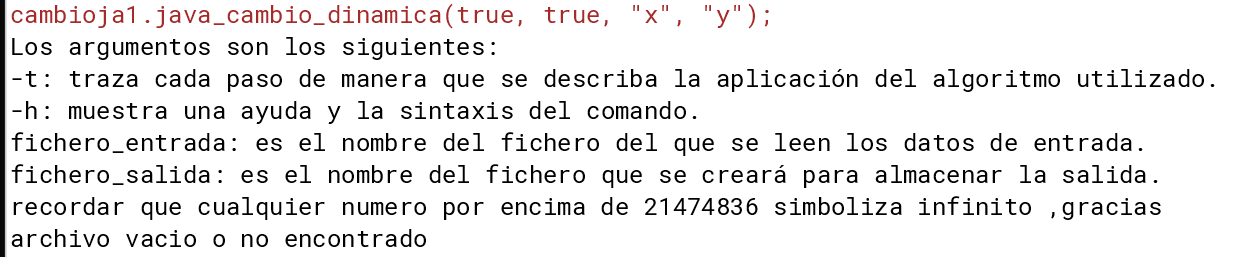


Como se puede ver la ayuda se ha desactivado ,a su vez que se ha desactivado la traza.

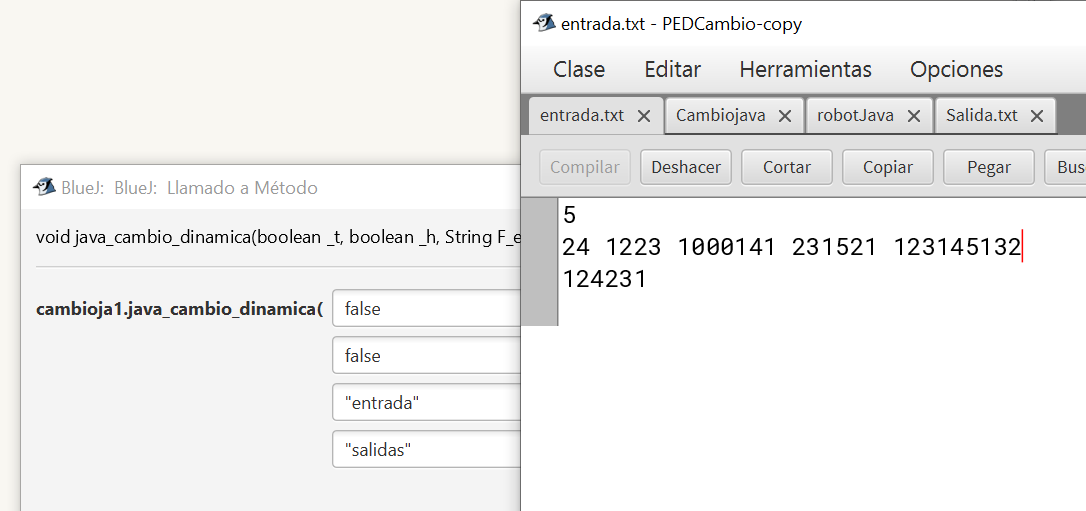
Iteración 4(Con archivo erróneo o vacío)



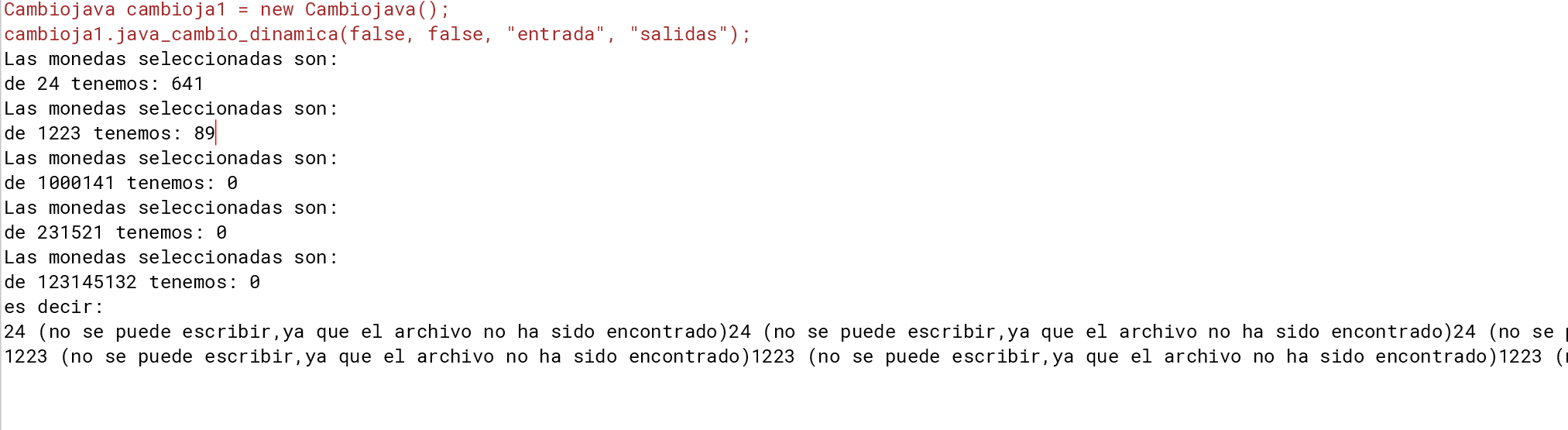
Resultado:



Iteración 5 (Con Archivo de Salida erróneo):

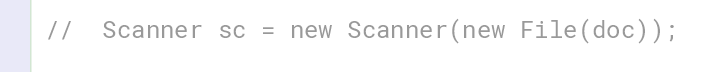


Resultado

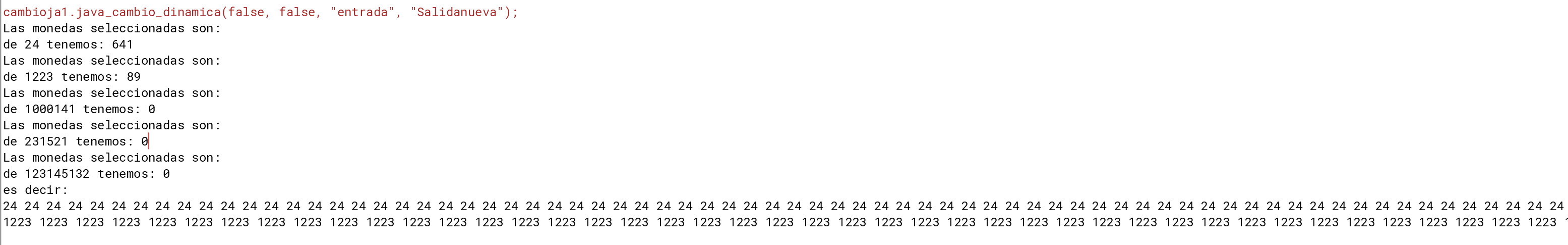


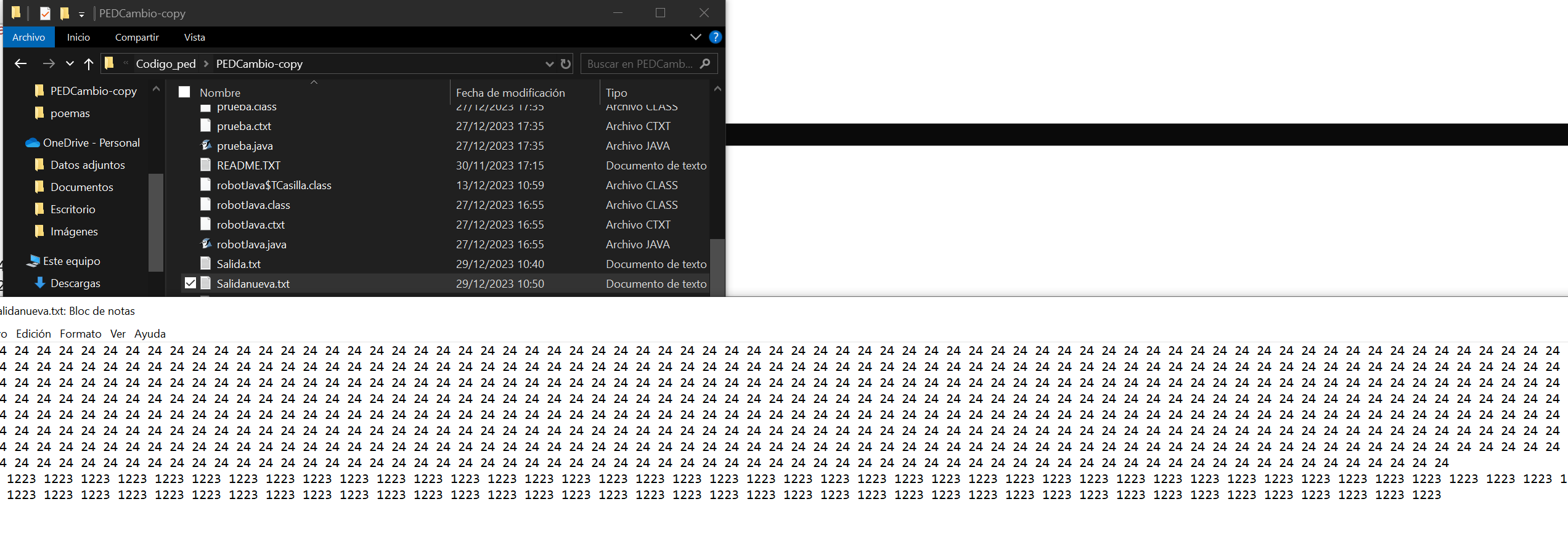
Como se puede ver al sacar los números por terminal intenta escribir en el archivo ,al no encontrarlo escribe un mensaje de error al lado de cada número que no pudo escribir.

Además, me di cuenta de que comentando está línea de código el programa crearía un archivo nuevo cada vez

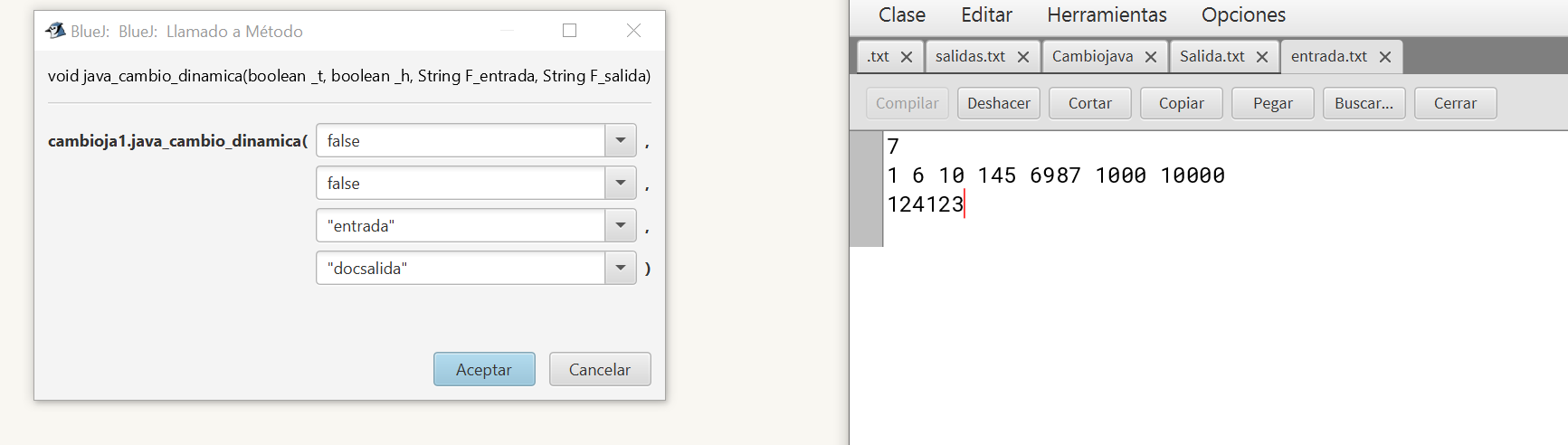


Ahora sin esa línea se puede comprobar que se crea un archivo totalmente nuevo en la carpeta:

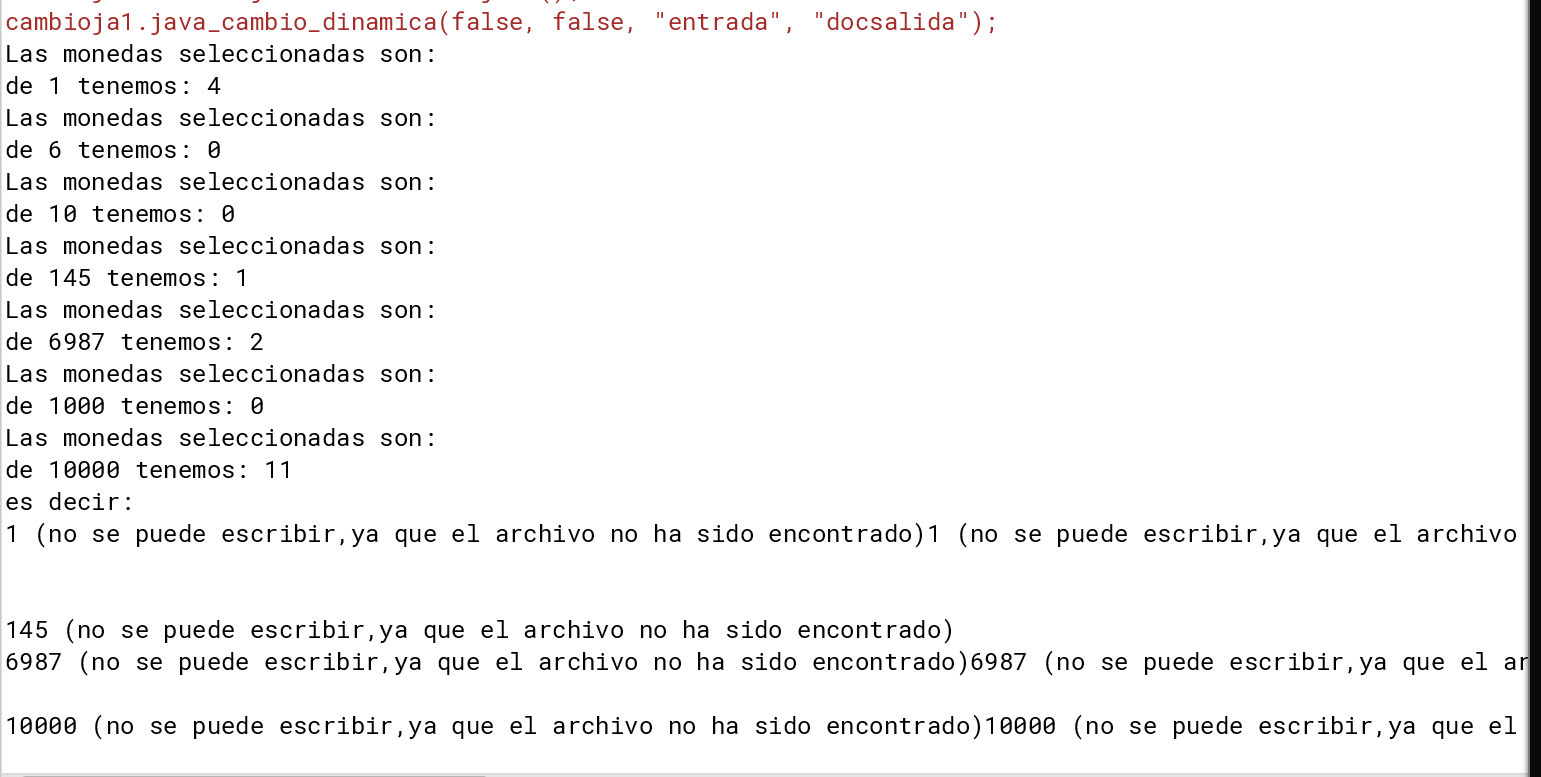


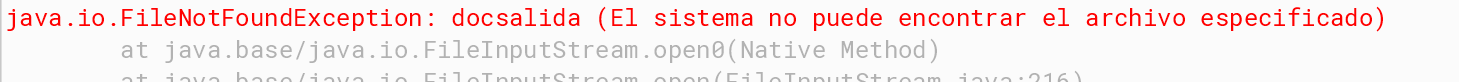


Si probamos con esa nueva línea sin comentar



Y con una salida que no existe nos dirá lo que ya se ha probado antes:





Con un montón de excepciones tratadas.

Para el código he decidido dejar esa línea sin comentar ,ya que creo que no es necesario crear nuevos archivos ,pero si me parecía interesante tratarlo.