

TECNOLÓGICO NACIONAL DE MÉXICO  
INSTITUTO TECNOLÓGICO DE TLAXIACO

INGENIERÍA EN SISTEMAS COMPUTACIONALES  
DESARROLLO DE SOFTWARE |SIE-DES-2022-01

SCD – 1003 ARQUITECTURA DE COMPUTADORAS

CÁTEDRA DEL ING. OSORIO SALINAS EDWARD

**ALUMNO:**

No	Nombre	No de Control
01	Cruz Ramírez Jaczibeth	22620233
01	Velasco López Daniel	22620076

**GRUPO:**

5BS

## REPORTE DE PRÁCTICA 7

## SIMULACIÓN DE ADMINISTRACIÓN DE MEMORIA EN LA CPU

Tlaxiaco, Oaxaca. A 05 de dic. de 24



Boulevard Tecnológico Km. 2.5, Llano Yosovee C.P. 69800. Tlaxiaco, Oaxaca. Tel. (953) 55 21322 y (953) 55 20405, e-mail: [dir\\_tlaxiaco@tecnm.mx](mailto:dir_tlaxiaco@tecnm.mx); [tecnm.mx](http://tecnm.mx) | [tlaxiaco.tecnm.mx](http://tlaxiaco.tecnm.mx)



### CONTENIDO

INTRODUCCIÓN.....	4
Simulación de administración de memoria en la CPU .....	5
2. Objetivo.....	5
3. Materiales.....	5
4. Pasos para la Simulación de administración de memoria en la CPU .....	5
4.1.1 Pasos para la elaboración del programa.....	5
5. CONCLUSIÓN.....	14
6. BIBLIOGRAFÍA.....	15

### LISTA DE FIGURAS

Ilustración 1 Software NetBeans.....	6
Ilustración 2 Nombre del proyecto .....	6
Ilustración 3 Simulación de administración p1 .....	7
Ilustración 4 Simulación de administración p2.....	7
Ilustración 5 Simulación de administración p3.....	8
Ilustración 6 Comprobación de errores p1.....	8
Ilustración 7 Comprobación de errores p2 .....	9
Ilustración 8 Ejecución del proyecto.....	9
Ilustración 9 Ejecución exitoso .....	10
Ilustración 10 Simulación del proyecto opción 2 p1.....	11
Ilustración 11 Simulación del proyecto opción 2 p2.....	11
Ilustración 12 Simulación del proyecto opción 2 p3.....	12
Ilustración 13 Ejecución del programa .....	12
Ilustración 14 Resultado Obtenido.....	13

### INTRODUCCIÓN

En esta práctica se aborda la simulación de la administración de memoria en la CPU, un proceso esencial en el funcionamiento de los sistemas operativos modernos. La administración eficiente de la memoria garantiza que los procesos y tareas se ejecuten de manera ordenada, maximizando el rendimiento del sistema. Utilizando el entorno de desarrollo NetBeans, esta actividad busca replicar las etapas fundamentales de este procedimiento mediante la implementación de un programa en lenguaje Java. A lo largo de este documento, se describirán los pasos realizados, los resultados obtenidos y las observaciones relevantes derivadas del ejercicio, proporcionando una comprensión clara y práctica del tema.

### Simulación de administración de memoria en la CPU

#### 2. Objetivo

El objetivo de esta práctica es simular el funcionamiento de un sistema operativo y realizar un resumen de los pasos que se llevan a cabo en la administración de memoria en la CPU.

#### 3. Materiales

1. Software NetBeans.
2. Equipo de cómputo.
3. Conexión a internet.

#### 4. Pasos para la Simulación de administración de memoria en la CPU

La administración de procesos o tareas en la CPU implica una serie de pasos esenciales que permiten al sistema operativo manejar y coordinar eficientemente la ejecución de procesos. Estos pasos incluyen:

1. Cargar el programa en memoria.
2. Asignar espacio de memoria a las variables.
3. Asignar espacio de memoria a las instrucciones.
4. Ejecutar el programa.

##### 4.1.1 Pasos para la elaboración del programa

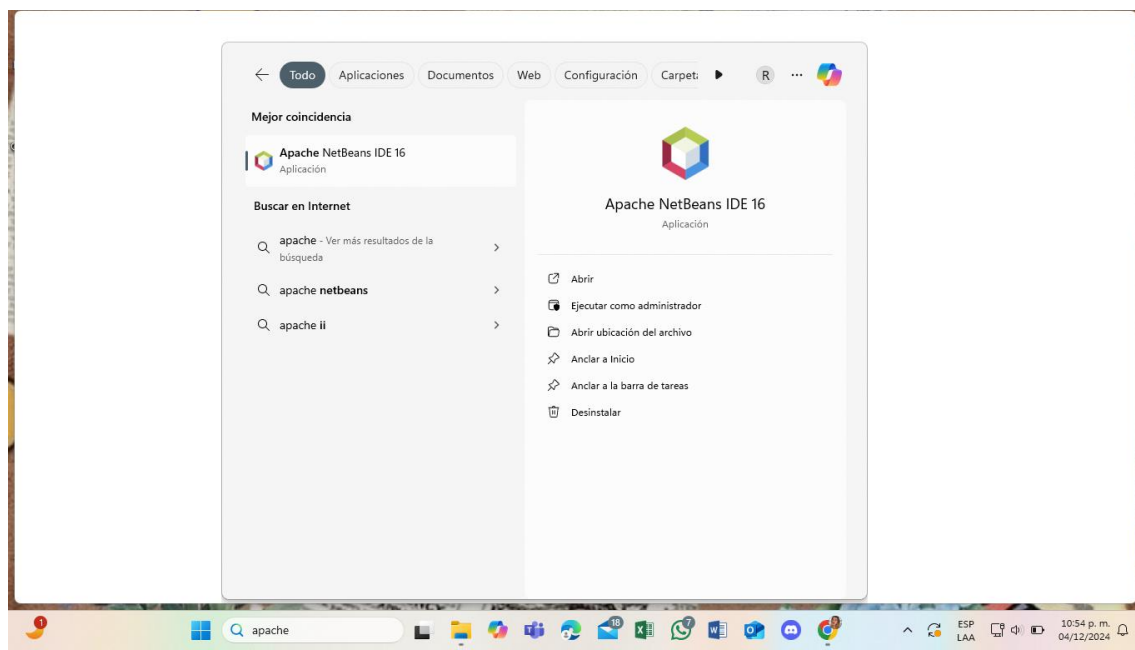
Para la elaboración de esta práctica, se hizo uso del software conocido como NetBeans, este software nos permite elaborar distintos programas, y en esta ocasión no será la excepción, la práctica, debe de ser en lenguaje de Python, sin embargo, se hizo la adaptación en lenguaje Java.

A continuación de mostrarán los códigos y resultados de dicha elaboración:

Empezamos por dirigirnos a nuestro software.

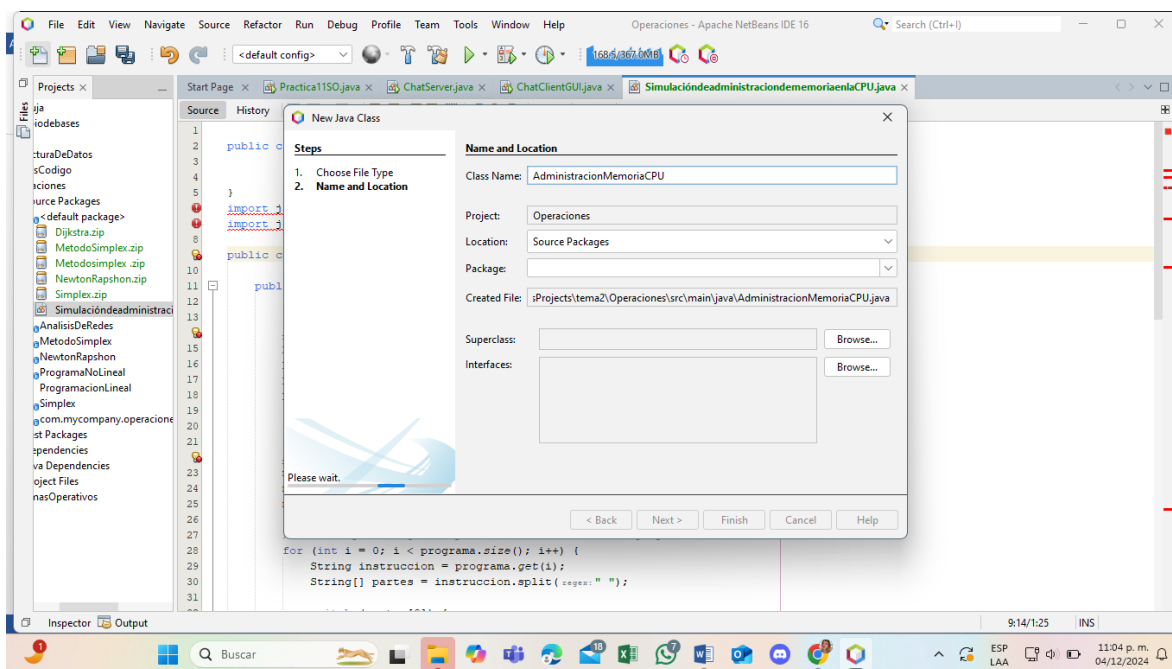
# CARACTERIZACIÓN DE EQUIPO DE CÓMPUTO

**Ilustración 1 Software NetBeans**

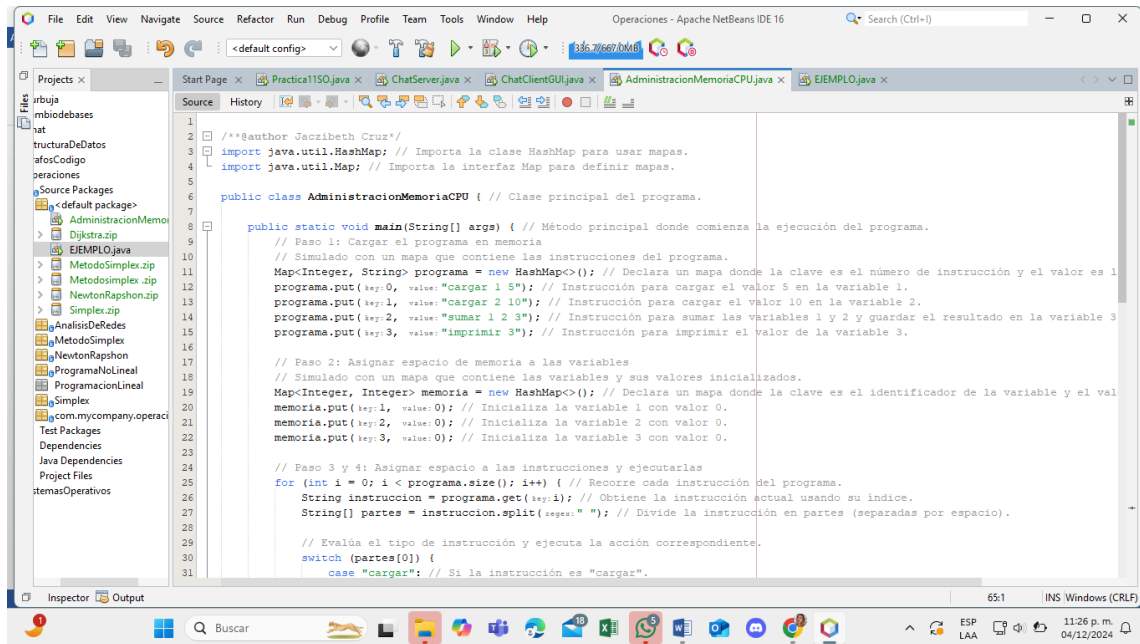


Una vez dentro de nuestro software, empezamos con la creación de nuestro proyecto, mismo que llevará por nombre de AdministracionMemoriaCPU.

**Ilustración 2 Nombre del proyecto**



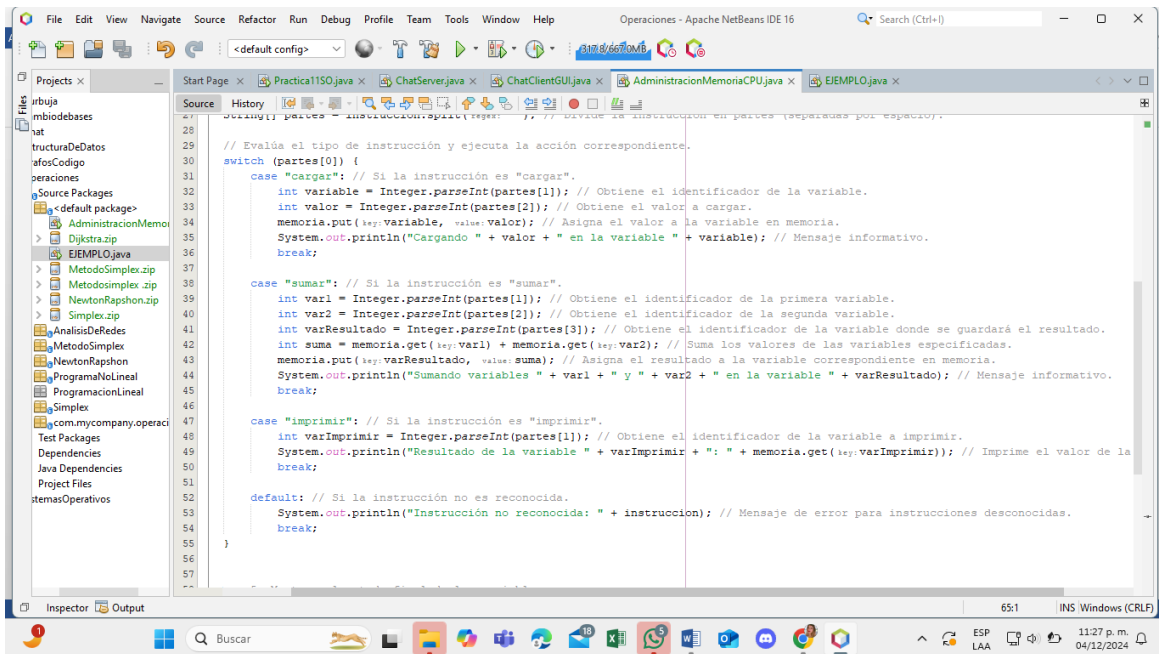
## Ilustración 3 Simulación de administración p1



The screenshot shows the Apache NetBeans IDE interface. The 'Source' tab is active, displaying the code for 'AdministracionMemoriaCPU.java'. The code is in Java and includes imports for 'HashMap' and 'Map'. It defines a 'main' method that simulates a memory management process. The code is as follows:

```
1  /**@author Jaczibeth Cruz*/
2
3  import java.util.HashMap; // Importa la clase HashMap para usar mapas.
4  import java.util.Map; // Importa la interfaz Map para definir mapas.
5
6  public class AdministracionMemoriaCPU { // Clase principal del programa.
7
8      // Paso 1: Cargar el programa en memoria
9      // Simulado con un mapa que contiene las instrucciones del programa.
10     Map<Integer, String> programa = new HashMap<>(); // Declara un mapa donde la clave es el número de instrucción y el valor es la instrucción.
11     programa.put(1, "cargar 1 5"); // Instrucción para cargar el valor 5 en la variable 1.
12     programa.put(2, "cargar 2 10"); // Instrucción para cargar el valor 10 en la variable 2.
13     programa.put(3, "sumar 1 2 3"); // Instrucción para sumar las variables 1 y 2 y guardar el resultado en la variable 3.
14     programa.put(4, "imprimir 3"); // Instrucción para imprimir el valor de la variable 3.
15
16     // Paso 2: Asignar espacio de memoria a las variables
17     // Simulado con un mapa que contiene las variables y sus valores inicializados.
18     Map<Integer, Integer> memoria = new HashMap<>(); // Declara un mapa donde la clave es el identificador de la variable y el valor es el valor inicializado.
19     memoria.put(1, 0); // Inicializa la variable 1 con valor 0.
20     memoria.put(2, 0); // Inicializa la variable 2 con valor 0.
21     memoria.put(3, 0); // Inicializa la variable 3 con valor 0.
22
23     // Paso 3 y 4: Asignar espacio a las instrucciones y ejecutarlas
24     for (int i = 0; i < programa.size(); i++) { // Recorre cada instrucción del programa.
25         String instruccion = programa.get(i); // Obtiene la instrucción actual usando su índice.
26         String[] partes = instruccion.split(" "); // Divide la instrucción en partes (separadas por espacio).
27
28         // Evalúa el tipo de instrucción y ejecuta la acción correspondiente.
29         switch (partes[0]) {
30             case "cargar": // Si la instrucción es "cargar".
```

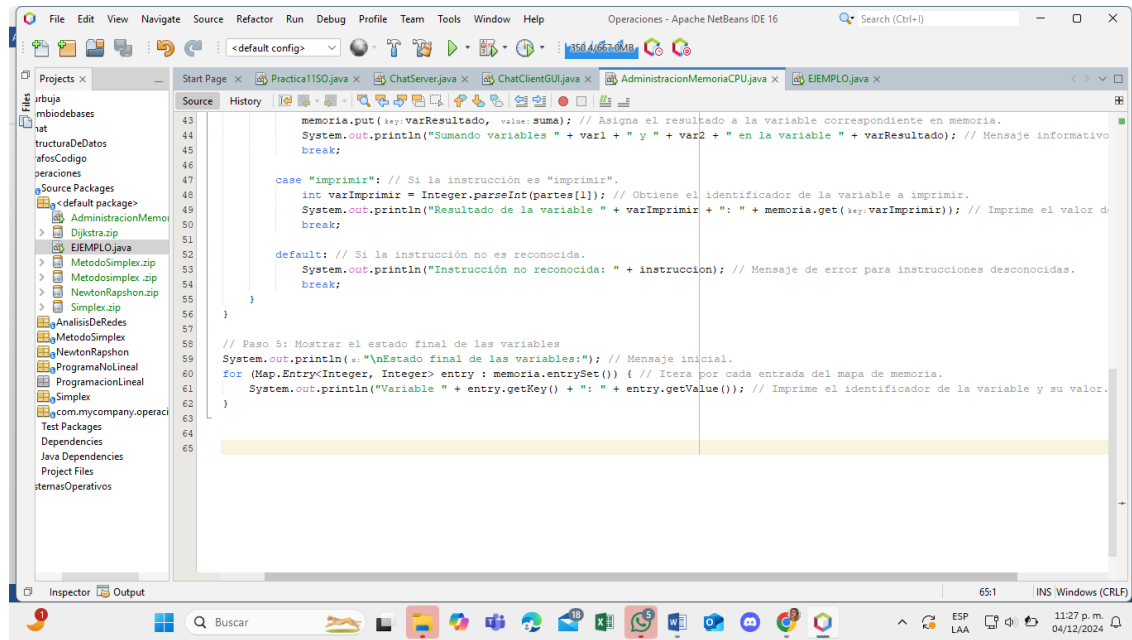
## Ilustración 4 Simulación de administración p2



The screenshot shows the Apache NetBeans IDE interface. The 'Source' tab is active, displaying the code for 'AdministracionMemoriaCPU.java'. The code continues from the previous screenshot, showing the execution of the instructions. The code is as follows:

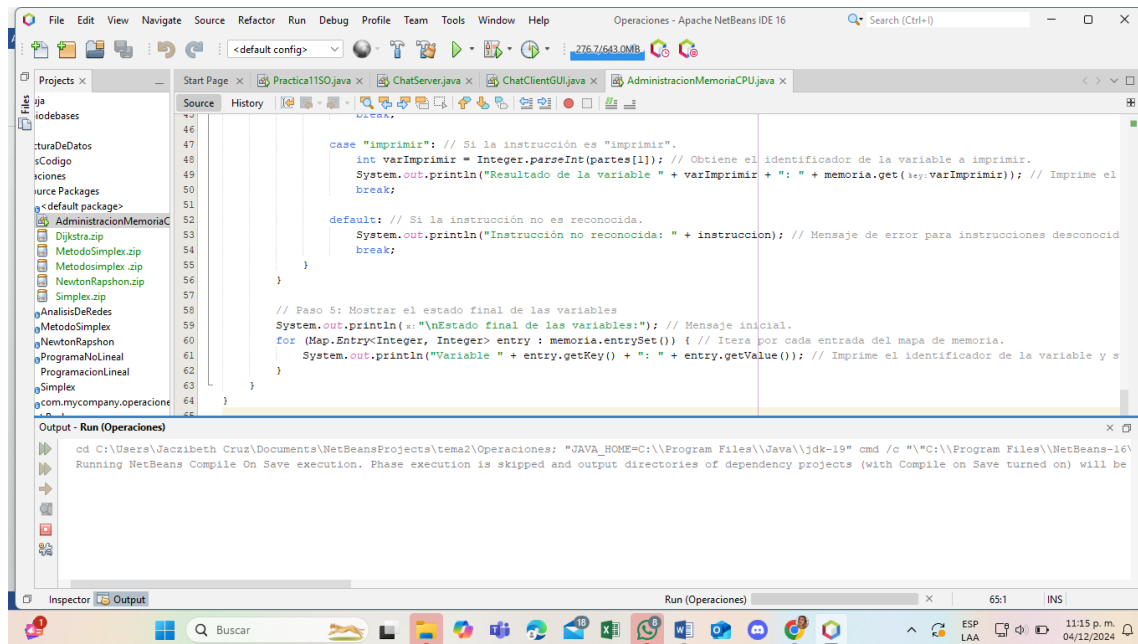
```
31             case "cargar": // Si la instrucción es "cargar".
32                 int variable = Integer.parseInt(partes[1]); // Obtiene el identificador de la variable.
33                 int valor = Integer.parseInt(partes[2]); // Obtiene el valor a cargar.
34                 memoria.put(variable, valor); // Asigna el valor a la variable en memoria.
35                 System.out.println("Cargando " + valor + " en la variable " + variable); // Mensaje informativo.
36                 break;
37
38             case "sumar": // Si la instrucción es "sumar".
39                 int var1 = Integer.parseInt(partes[1]); // Obtiene el identificador de la primera variable.
40                 int var2 = Integer.parseInt(partes[2]); // Obtiene el identificador de la segunda variable.
41                 int varResultado = Integer.parseInt(partes[3]); // Obtiene el identificador de la variable donde se guardará el resultado.
42                 int suma = memoria.get(var1) + memoria.get(var2); // Suma los valores de las variables especificadas.
43                 memoria.put(varResultado, suma); // Asigna el resultado a la variable correspondiente en memoria.
44                 System.out.println("Sumando variables " + var1 + " y " + var2 + " en la variable " + varResultado); // Mensaje informativo.
45                 break;
46
47             case "imprimir": // Si la instrucción es "imprimir".
48                 int varImprimir = Integer.parseInt(partes[1]); // Obtiene el identificador de la variable a imprimir.
49                 System.out.println("Resultado de la variable " + varImprimir + ": " + memoria.get(varImprimir)); // Imprime el valor de la variable.
50                 break;
51
52             default: // Si la instrucción no es reconocida.
53                 System.out.println("Instrucción no reconocida: " + instruccion); // Mensaje de error para instrucciones desconocidas.
54                 break;
55         }
56     }
57 }
```

## Ilustración 5 Simulación de administración p3



Una vez realizado nuestro proyecto, procedemos con la ejecución de nuestro proyecto con la finalidad de verificar que no existan errores que interrumpan con la finalidad del proyecto, esto se realiza de la siguiente manera:

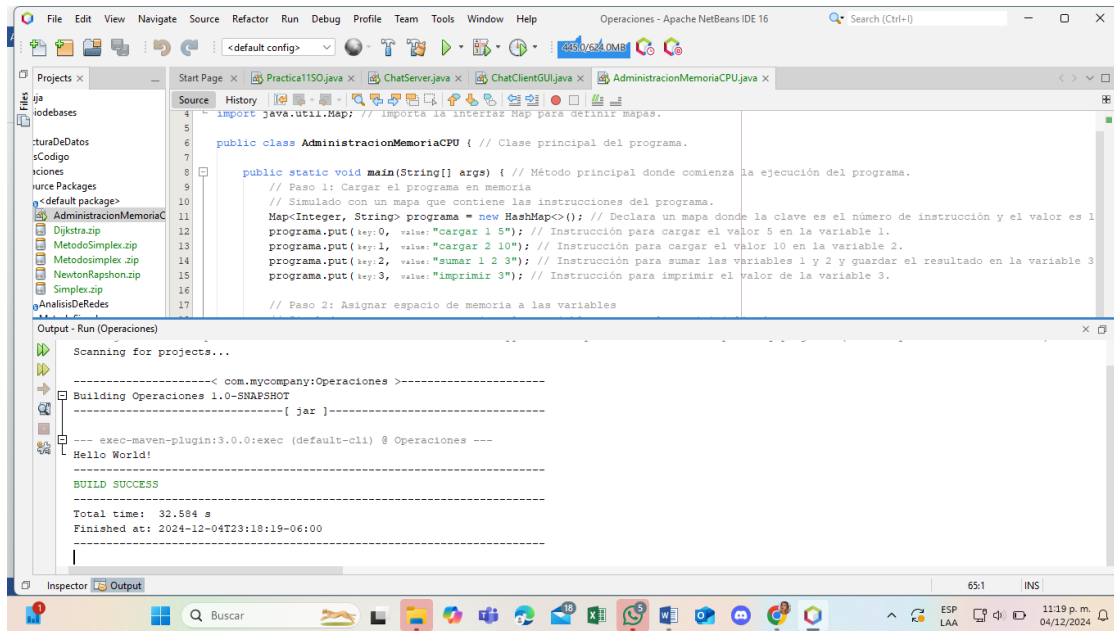
## Ilustración 6 Comprobación de errores p1





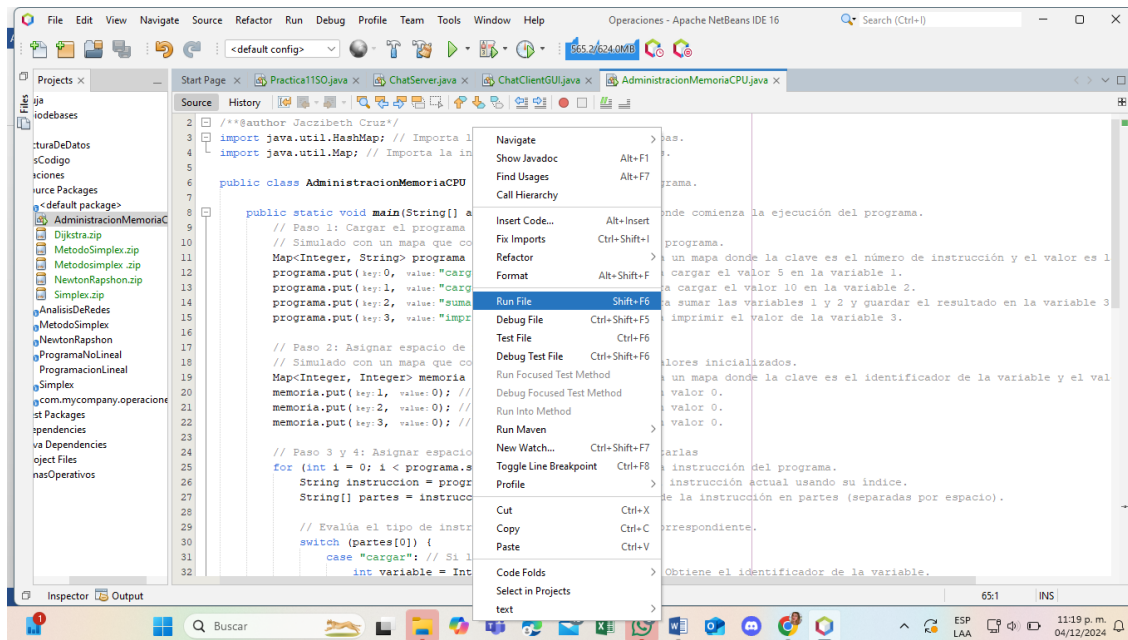
## CARACTERIZACIÓN DE EQUIPO DE CÓMPUTO

**Ilustración 7 Comprobación de errores p2**

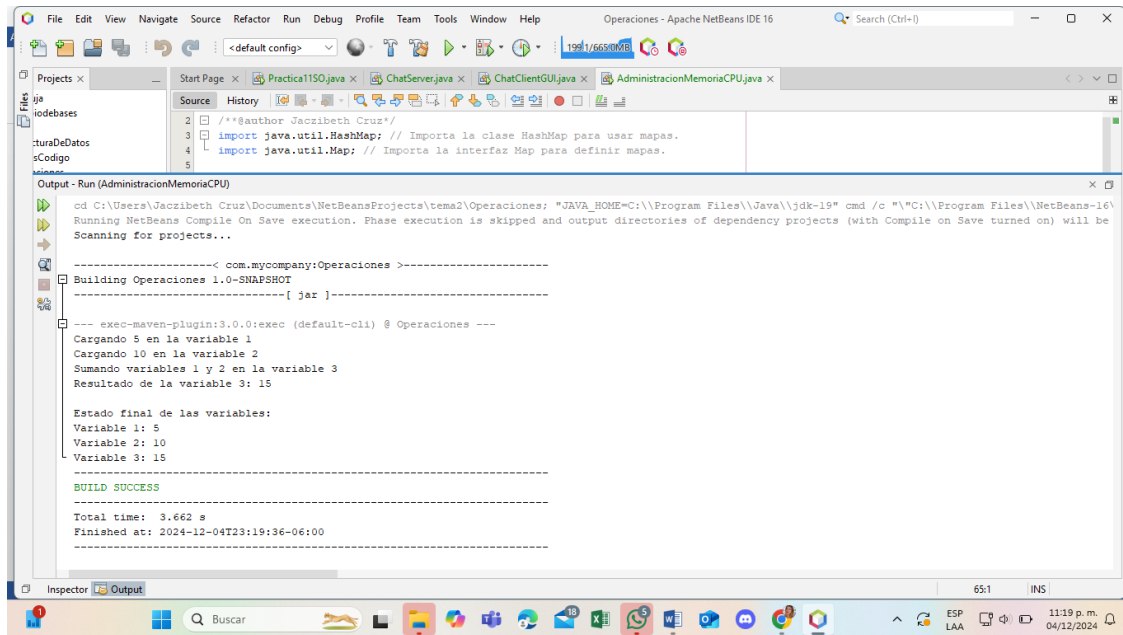


Al comprobar que no existen los errores, se procede con la verificación de funcionalidad del proyecto.

**Ilustración 8 Ejecución del proyecto**



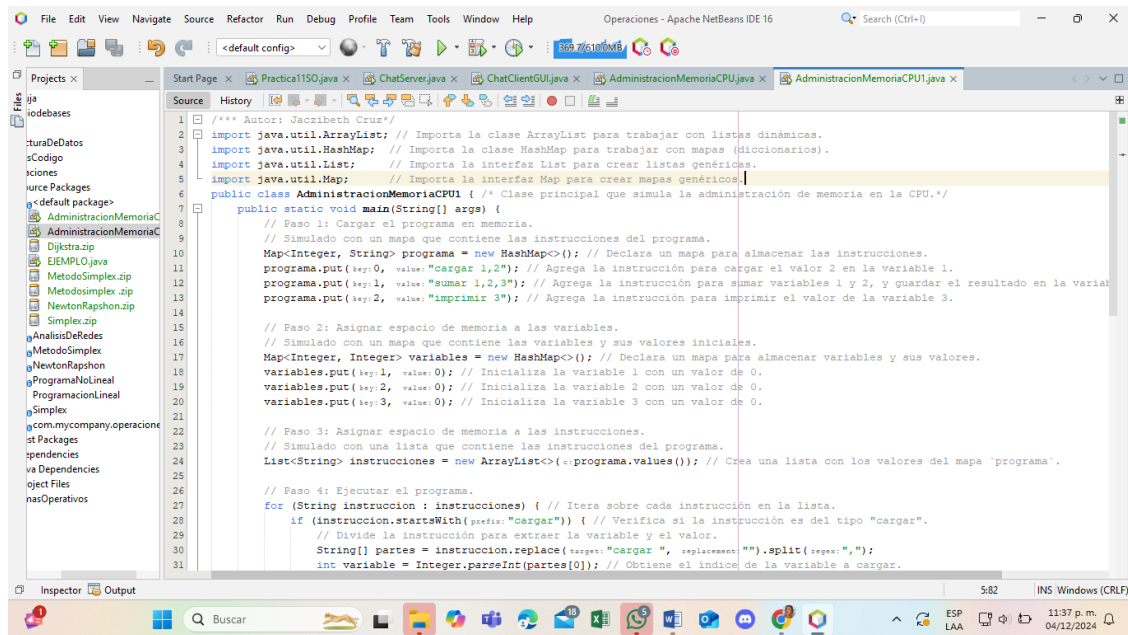
**Ilustración 9 Ejecución exitoso**



Como podemos observar, el programa funciona correctamente, por lo que podemos decir que el programa se puede exponer a nuevos desafíos con nuevas actualizaciones en sus códigos.

De igual manera, se realizó otra manera de realizar el programa, usando el lenguaje Java, tal que se muestra a continuación:

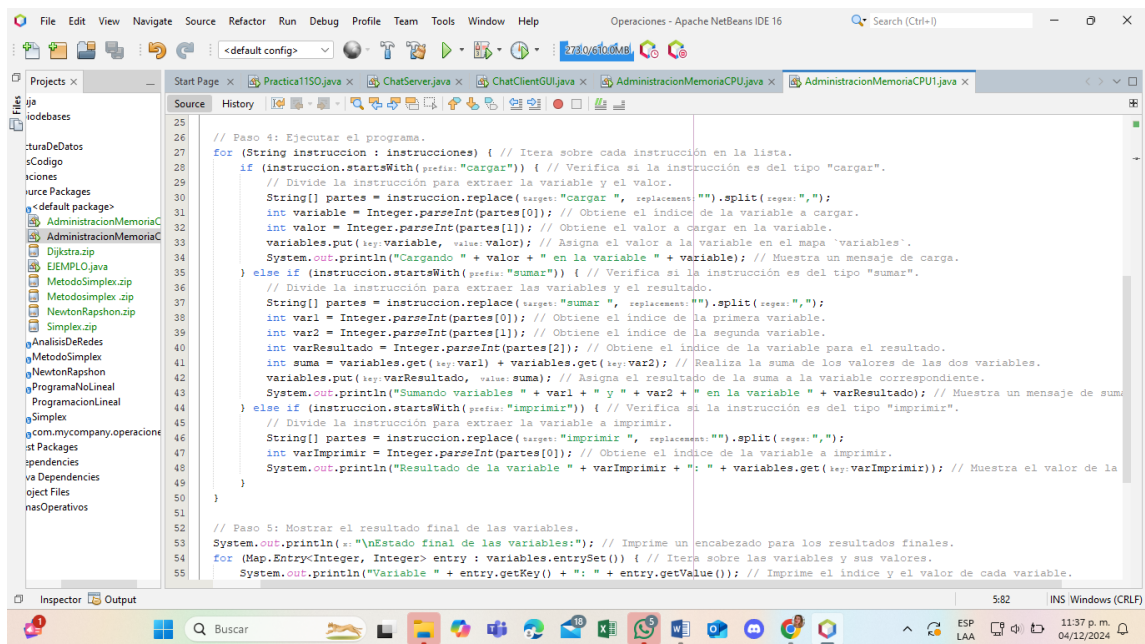
Ilustración 10 Simulación del proyecto opción 2 p1



The screenshot shows the Apache NetBeans IDE interface. The 'Source' window displays the code for 'AdministracionMemoriaCPU1.java'. The code is in Spanish and implements a memory management simulation. It includes imports for ArrayList, HashMap, List, and Map. The main method 'main' is divided into four steps: 1. Loading the program into memory (creating a HashMap with instructions), 2. Assigning memory space to variables (creating a HashMap for variables), 3. Assigning memory space to instructions (creating a List of instructions), and 4. Executing the program (iterating through instructions and performing operations like 'cargar', 'sumar', and 'imprimir').

```
1  /** Autor: Jaczibeth Cruz */
2
3  import java.util.ArrayList; // Importa la clase ArrayList para trabajar con listas dinámicas.
4  import java.util.HashMap; // Importa la clase HashMap para trabajar con mapas (diccionarios).
5  import java.util.List; // Importa la interfaz List para crear listas genéricas.
6  import java.util.Map; // Importa la interfaz Map para crear mapas genéricos.
7
8  public class AdministracionMemoriaCPU1 { /* Clase principal que simula la administración de memoria en la CPU.*/
9
10     // Paso 1: Cargar el programa en memoria.
11     // Simulado con un mapa que contiene las instrucciones del programa.
12     Map<Integer, String> programa = new HashMap<>(); // Declara un mapa para almacenar las instrucciones.
13     programa.put(0, "cargar 1,2"); // Agrega la instrucción para cargar el valor 2 en la variable 1.
14     programa.put(1, "sumar 1,2,3"); // Agrega la instrucción para sumar variables 1 y 2, y guardar el resultado en la variable 3.
15     programa.put(2, "imprimir 3"); // Agrega la instrucción para imprimir el valor de la variable 3.
16
17     // Paso 2: Asignar espacio de memoria a las variables.
18     // Simulado con un mapa que contiene las variables y sus valores iniciales.
19     Map<Integer, Integer> variables = new HashMap<>(); // Declara un mapa para almacenar variables y sus valores.
20     variables.put(0, 0); // Inicializa la variable 1 con un valor de 0.
21     variables.put(1, 0); // Inicializa la variable 2 con un valor de 0.
22     variables.put(2, 0); // Inicializa la variable 3 con un valor de 0.
23
24     // Paso 3: Asignar espacio de memoria a las instrucciones.
25     // Simulado con una lista que contiene las instrucciones del programa.
26     List<String> instrucciones = new ArrayList<>(programa.values()); // Crea una lista con los valores del mapa 'programa'.
27
28     // Paso 4: Ejecutar el programa.
29     for (String instruccion : instrucciones) { // Itera sobre cada instrucción en la lista.
30         if (instruccion.startsWith("cargar")) { // Verifica si la instrucción es del tipo "cargar".
31             // Divide la instrucción para extraer la variable y el valor.
32             String[] partes = instruccion.replace("cargar ", "").split(" ");
33             int variable = Integer.parseInt(partes[0]); // Obtiene el índice de la variable a cargar.
34
35             // Paso 5: Ejecutar el programa.
36             for (String instruccion : instrucciones) { // Itera sobre cada instrucción en la lista.
37                 if (instruccion.startsWith("cargar")) { // Verifica si la instrucción es del tipo "cargar".
38                     // Divide la instrucción para extraer la variable y el valor.
39                     String[] partes = instruccion.replace("cargar ", "").split(" ");
40                     int variable = Integer.parseInt(partes[0]); // Obtiene el índice de la variable a cargar.
41                     int valor = Integer.parseInt(partes[1]); // Obtiene el valor a cargar en la variable.
42                     variables.put(variable, valor); // Asigna el valor a la variable en el mapa 'variables'.
43                     System.out.println("Cargando " + valor + " en la variable " + variable); // Muestra un mensaje de carga.
44                 } else if (instruccion.startsWith("sumar")) { // Verifica si la instrucción es del tipo "sumar".
45                     // Divide la instrucción para extraer las variables y el resultado.
46                     String[] partes = instruccion.replace("sumar ", "").split(" ");
47                     int var1 = Integer.parseInt(partes[0]); // Obtiene el índice de la primera variable.
48                     int var2 = Integer.parseInt(partes[1]); // Obtiene el índice de la segunda variable.
49                     int varResultado = Integer.parseInt(partes[2]); // Obtiene el índice de la variable para el resultado.
50                     int suma = variables.get(var1) + variables.get(var2); // Realiza la suma de los valores de las dos variables.
51                     variables.put(varResultado, suma); // Asigna el resultado de la suma a la variable correspondiente.
52                     System.out.println("Sumando variables " + var1 + " y " + var2 + " en la variable " + varResultado); // Muestra un mensaje de suma.
53                 } else if (instruccion.startsWith("imprimir")) { // Verifica si la instrucción es del tipo "imprimir".
54                     // Divide la instrucción para extraer la variable a imprimir.
55                     String[] partes = instruccion.replace("imprimir ", "").split(" ");
56                     int varImprimir = Integer.parseInt(partes[0]); // Obtiene el índice de la variable a imprimir.
57                     System.out.println("Resultado de la variable " + varImprimir + ": " + variables.get(varImprimir)); // Muestra el valor de la variable.
58                 }
59             }
60         }
61     }
62
63     // Paso 5: Mostrar el resultado final de las variables.
64     System.out.println("\nEstado final de las variables:"); // Imprime un encabezado para los resultados finales.
65     for (Map.Entry<Integer, Integer> entry : variables.entrySet()) { // Itera sobre las variables y sus valores.
66         System.out.println("Variable " + entry.getKey() + ": " + entry.getValue()); // Imprime el índice y el valor de cada variable.
67     }
68 }
```

Ilustración 11 Simulación del proyecto opción 2 p2

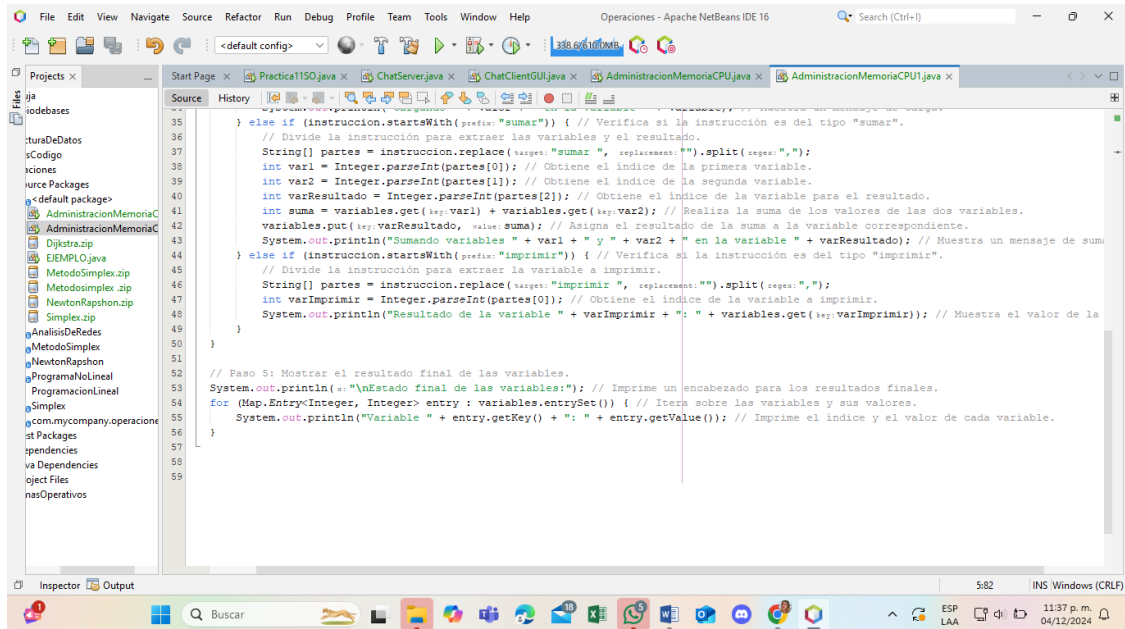


The screenshot shows the Apache NetBeans IDE interface. The 'Source' window displays the code for 'AdministracionMemoriaCPU1.java'. The code is in Spanish and implements a memory management simulation. It includes imports for ArrayList, HashMap, List, and Map. The main method 'main' is divided into four steps: 1. Loading the program into memory (creating a HashMap with instructions), 2. Assigning memory space to variables (creating a HashMap for variables), 3. Assigning memory space to instructions (creating a List of instructions), and 4. Executing the program (iterating through instructions and performing operations like 'cargar', 'sumar', and 'imprimir').

```
1  /** Autor: Jaczibeth Cruz */
2
3  import java.util.ArrayList; // Importa la clase ArrayList para trabajar con listas dinámicas.
4  import java.util.HashMap; // Importa la clase HashMap para trabajar con mapas (diccionarios).
5  import java.util.List; // Importa la interfaz List para crear listas genéricas.
6  import java.util.Map; // Importa la interfaz Map para crear mapas genéricos.
7
8  public class AdministracionMemoriaCPU1 { /* Clase principal que simula la administración de memoria en la CPU.*/
9
10     // Paso 1: Cargar el programa en memoria.
11     // Simulado con un mapa que contiene las instrucciones del programa.
12     Map<Integer, String> programa = new HashMap<>(); // Declara un mapa para almacenar las instrucciones.
13     programa.put(0, "cargar 1,2"); // Agrega la instrucción para cargar el valor 2 en la variable 1.
14     programa.put(1, "sumar 1,2,3"); // Agrega la instrucción para sumar variables 1 y 2, y guardar el resultado en la variable 3.
15     programa.put(2, "imprimir 3"); // Agrega la instrucción para imprimir el valor de la variable 3.
16
17     // Paso 2: Asignar espacio de memoria a las variables.
18     // Simulado con un mapa que contiene las variables y sus valores iniciales.
19     Map<Integer, Integer> variables = new HashMap<>(); // Declara un mapa para almacenar variables y sus valores.
20     variables.put(0, 0); // Inicializa la variable 1 con un valor de 0.
21     variables.put(1, 0); // Inicializa la variable 2 con un valor de 0.
22     variables.put(2, 0); // Inicializa la variable 3 con un valor de 0.
23
24     // Paso 3: Asignar espacio de memoria a las instrucciones.
25     // Simulado con una lista que contiene las instrucciones del programa.
26     List<String> instrucciones = new ArrayList<>(programa.values()); // Crea una lista con los valores del mapa 'programa'.
27
28     // Paso 4: Ejecutar el programa.
29     for (String instruccion : instrucciones) { // Itera sobre cada instrucción en la lista.
30         if (instruccion.startsWith("cargar")) { // Verifica si la instrucción es del tipo "cargar".
31             // Divide la instrucción para extraer la variable y el valor.
32             String[] partes = instruccion.replace("cargar ", "").split(" ");
33             int variable = Integer.parseInt(partes[0]); // Obtiene el índice de la variable a cargar.
34             int valor = Integer.parseInt(partes[1]); // Obtiene el valor a cargar en la variable.
35             variables.put(variable, valor); // Asigna el valor a la variable en el mapa 'variables'.
36             System.out.println("Cargando " + valor + " en la variable " + variable); // Muestra un mensaje de carga.
37         } else if (instruccion.startsWith("sumar")) { // Verifica si la instrucción es del tipo "sumar".
38             // Divide la instrucción para extraer las variables y el resultado.
39             String[] partes = instruccion.replace("sumar ", "").split(" ");
40             int var1 = Integer.parseInt(partes[0]); // Obtiene el índice de la primera variable.
41             int var2 = Integer.parseInt(partes[1]); // Obtiene el índice de la segunda variable.
42             int varResultado = Integer.parseInt(partes[2]); // Obtiene el índice de la variable para el resultado.
43             int suma = variables.get(var1) + variables.get(var2); // Realiza la suma de los valores de las dos variables.
44             variables.put(varResultado, suma); // Asigna el resultado de la suma a la variable correspondiente.
45             System.out.println("Sumando variables " + var1 + " y " + var2 + " en la variable " + varResultado); // Muestra un mensaje de suma.
46         } else if (instruccion.startsWith("imprimir")) { // Verifica si la instrucción es del tipo "imprimir".
47             // Divide la instrucción para extraer la variable a imprimir.
48             String[] partes = instruccion.replace("imprimir ", "").split(" ");
49             int varImprimir = Integer.parseInt(partes[0]); // Obtiene el índice de la variable a imprimir.
50             System.out.println("Resultado de la variable " + varImprimir + ": " + variables.get(varImprimir)); // Muestra el valor de la variable.
51         }
52     }
53
54     // Paso 5: Mostrar el resultado final de las variables.
55     System.out.println("\nEstado final de las variables:"); // Imprime un encabezado para los resultados finales.
56     for (Map.Entry<Integer, Integer> entry : variables.entrySet()) { // Itera sobre las variables y sus valores.
57         System.out.println("Variable " + entry.getKey() + ": " + entry.getValue()); // Imprime el índice y el valor de cada variable.
58     }
59 }
```

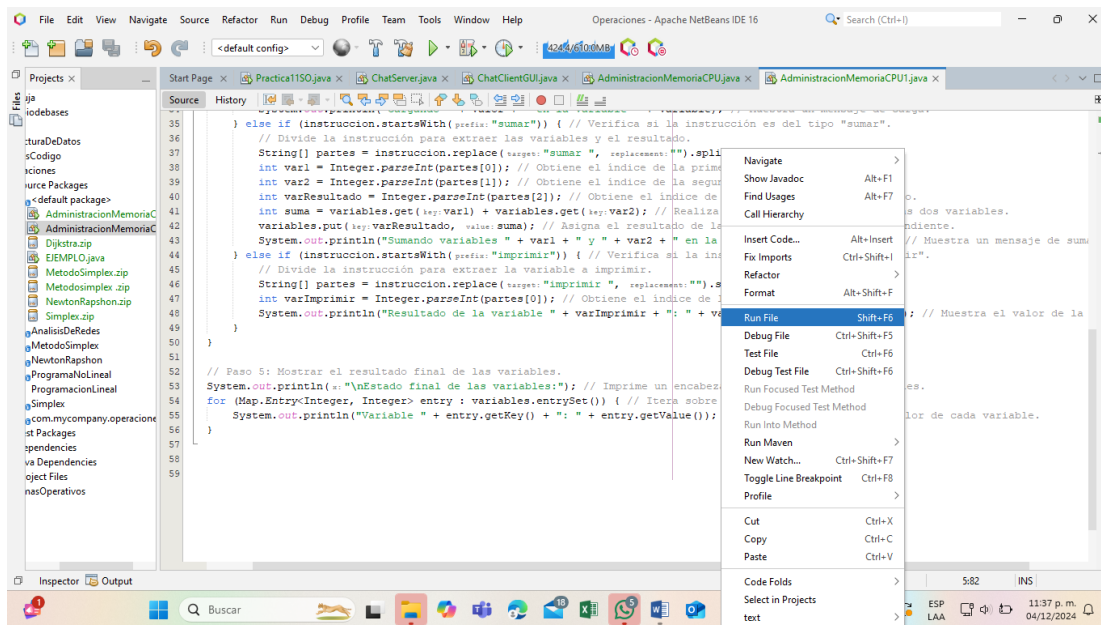
# CARACTERIZACIÓN DE EQUIPO DE CÓMPUTO

## Ilustración 12 Simulación del proyecto opción 2 p3



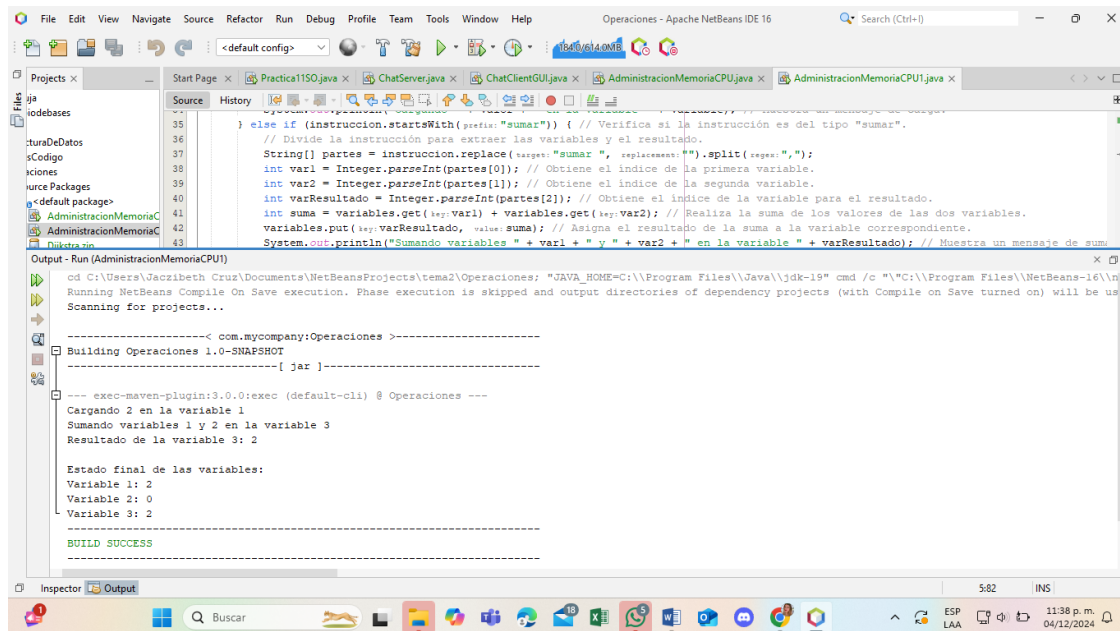
Continuamos con la ejecución del programa.

## Ilustración 13 Ejecución del programa



Una vez ejecutado, se muestra el resultado del programa.

Ilustración 14 Resultado Obtenido



The screenshot displays the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The main editor window shows a Java file named 'AdministracionMemoriaCPU.java' with the following code:

```
35 } else if (instruccion.startsWith(prefix: "sumar")) { // Verifica si la instrucción es del tipo "sumar".
36 // Divide la instrucción para extraer las variables y el resultado.
37 String[] partes = instruccion.replace(target: "sumar ", replacement: "").split(regex: ",");
38 int var1 = Integer.parseInt(partes[0]); // Obtiene el índice de la primera variable.
39 int var2 = Integer.parseInt(partes[1]); // Obtiene el índice de la segunda variable.
40 int varResultado = Integer.parseInt(partes[2]); // Obtiene el índice de la variable para el resultado.
41 int suma = variables.get(key: var1) + variables.get(key: var2); // Realiza la suma de los valores de las dos variables.
42 variables.put(key: varResultado, value: suma); // Asigna el resultado de la suma a la variable correspondiente.
43 System.out.println("Sumando variables " + var1 + " y " + var2 + " en la variable " + varResultado); // Muestra un mensaje de suma
```

The Output window at the bottom shows the execution results for 'Run (AdministracionMemoriaCPU1)'. The output includes the following text:

```
cd C:\Users\Jacquith Cruz\Documents\NetBeansProjects\tema2\Operaciones; "JAVA_HOME=C:\Program Files\Java\jdk-19" cmd /c "%C:\Program Files\NetBeans-16\
Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compile on Save turned on) will be us
Scanning for projects...

-----< com.mycompany:Operaciones >-----
Building Operaciones 1.0-SNAPSHOT
-----[ jar ]-----

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Operaciones ---
Cargando 2 en la variable 1
Sumando variables 1 y 2 en la variable 3
Resultado de la variable 3: 2

Estado final de las variables:
Variable 1: 2
Variable 2: 0
Variable 3: 2

BUILD SUCCESS
```

De esta manera, se muestra el resultado en dos maneras diferentes del programa, por lo que podemos concluir que la realización del programa se realizó correctamente.

### 5. CONCLUSIÓN

En esta práctica, se logró desarrollar y ejecutar con éxito un programa que simula la administración de memoria en la CPU, mostrando de manera efectiva las etapas de carga, asignación y ejecución de procesos. La correcta funcionalidad del programa refleja la utilidad de NetBeans como herramienta de desarrollo y destaca la flexibilidad del lenguaje Java para abordar problemas computacionales complejos. Este ejercicio no solo fortaleció los conocimientos técnicos sobre la administración de memoria, sino que también brindó una base sólida para enfrentar nuevos desafíos relacionados con el diseño e implementación de sistemas operativos.

### 6. BIBLIOGRAFÍA

GitHub. (05 de diciembre de 2024). *GitHub*. Obtenido de <https://github.com/Daniel-Velasco-Lopez/tec-nm-tlaxiaco-arqui-compu/blob/main/practices/Practica-7.md>

GitHub. (05 de diciembre de 2024). *GitHub*. Obtenido de <https://github.com/Jaczibeth-Cruz-Ramirez/tec-nm-tlaxiaco-arqui-compu/blob/main/practices/Practica-7.md>