

Actividad 8 Estructura de Datos  
Igor Sung Min Kim Juliao  
AL02829189  
Juan Pablo Hernandez Parra  
AL02887299  
Mauricio Estrada de la Garza  
AL02976904

### Estructura del Código:

El código consta de una clase Java llamada "" que contiene los siguientes métodos:

- `imprimirSolucion(int[][] tablero)`: Este método imprime una solución del tablero de ajedrez donde las reinas están ubicadas.
- `posicion(int[][] tablero, int fila, int columna)`: Este método verifica si una reina puede ser colocada en una posición específica del tablero sin amenazar a otras reinas.
- `resolverNReinasUtil(int[][] tablero, int columna)`: Este es un método recursivo que intenta colocar las reinas en el tablero utilizando la técnica de backtracking. Explora todas las posibles combinaciones de ubicaciones para las reinas hasta encontrar una solución válida.
- `resolverNReinas()`: Este es el método principal que inicializa el tablero y llama a `resolverNReinasUtil()` para encontrar y mostrar todas las soluciones al problema de las N-Reinas.

### Funcionamiento del Código:

El código comienza llamando al método `resolverNReinas()` desde el método `main`. Este método crea un tablero de tamaño  $N \times N$  e invoca a `resolverNReinasUtil()` para encontrar todas las soluciones posibles. Si no se encuentra ninguna solución, imprime un mensaje indicando que no hay solución posible.

El método `resolverNReinasUtil()` utiliza la técnica de backtracking para explorar todas las posibles combinaciones de ubicaciones de reinas en el tablero. Se intenta colocar una reina en una columna determinada y luego se verifica si esa ubicación es válida según las reglas del problema. Si es válida, se procede recursivamente a colocar las reinas restantes. Si no es válida, se retrocede y se intenta con una ubicación diferente para la reina actual.

## Pruebas:

### 1 Reina:

The screenshot shows the NetBeans IDE interface. The 'Projects' pane on the left displays the project structure for 'EightQueens', including 'Source Packages' (eightqueens, Libraries, InteractivePriorityQueue) and 'Test Packages'. The 'Source' pane shows the 'EightQueens.java' file with the following code:

```
85     System.out.println("No existe solución.");
86     return;
87 }
88
89 printSolution(board);
90 }
91
92 public static void main(String[] args) {
93     Scanner scanner = new Scanner(System.in);
94 }
```

The 'Output' pane at the bottom shows the execution results:

```
run:
Ingrese el número de reinas: 1
1
BUILD SUCCESSFUL (total time: 1 second)
```

The Windows taskbar at the bottom shows the system clock as 8:03 PM on 3/16/2024.

### 2 Reinas:

The screenshot shows the NetBeans IDE interface. The 'Projects' pane on the left displays the project structure for 'EightQueens'. The 'Source' pane shows the 'EightQueens.java' file with the following code:

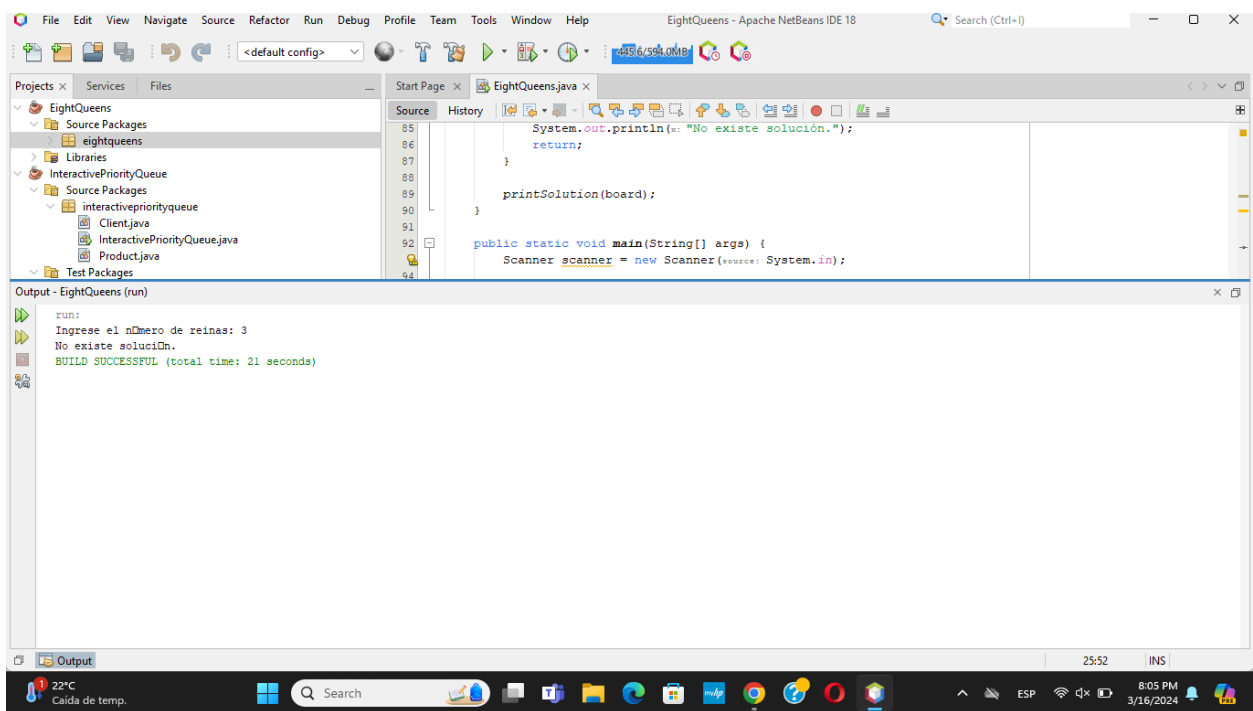
```
85     System.out.println("No existe solución.");
86     return;
87 }
88
89 printSolution(board);
90 }
91
92 public static void main(String[] args) {
93     Scanner scanner = new Scanner(System.in);
94 }
```

The 'Output' pane at the bottom shows the execution results:

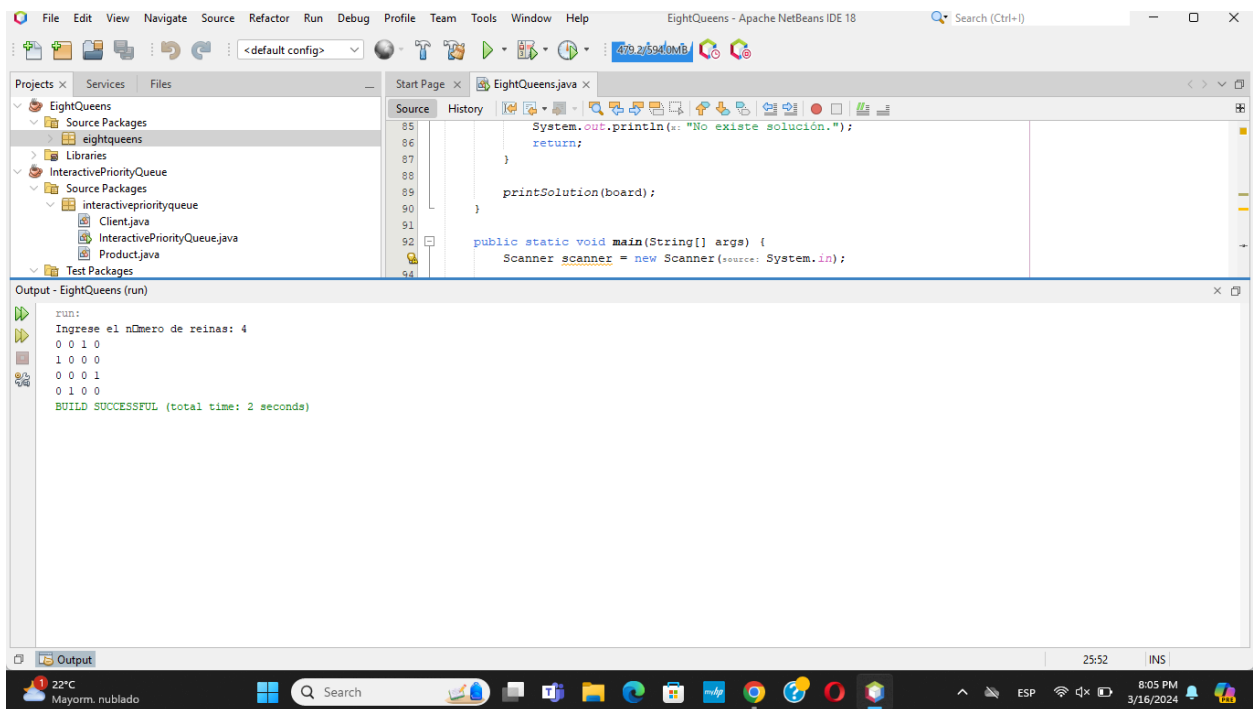
```
run:
Ingrese el número de reinas: 2
No existe solución.
BUILD SUCCESSFUL (total time: 11 seconds)
```

The Windows taskbar at the bottom shows the system clock as 8:04 PM on 3/16/2024.

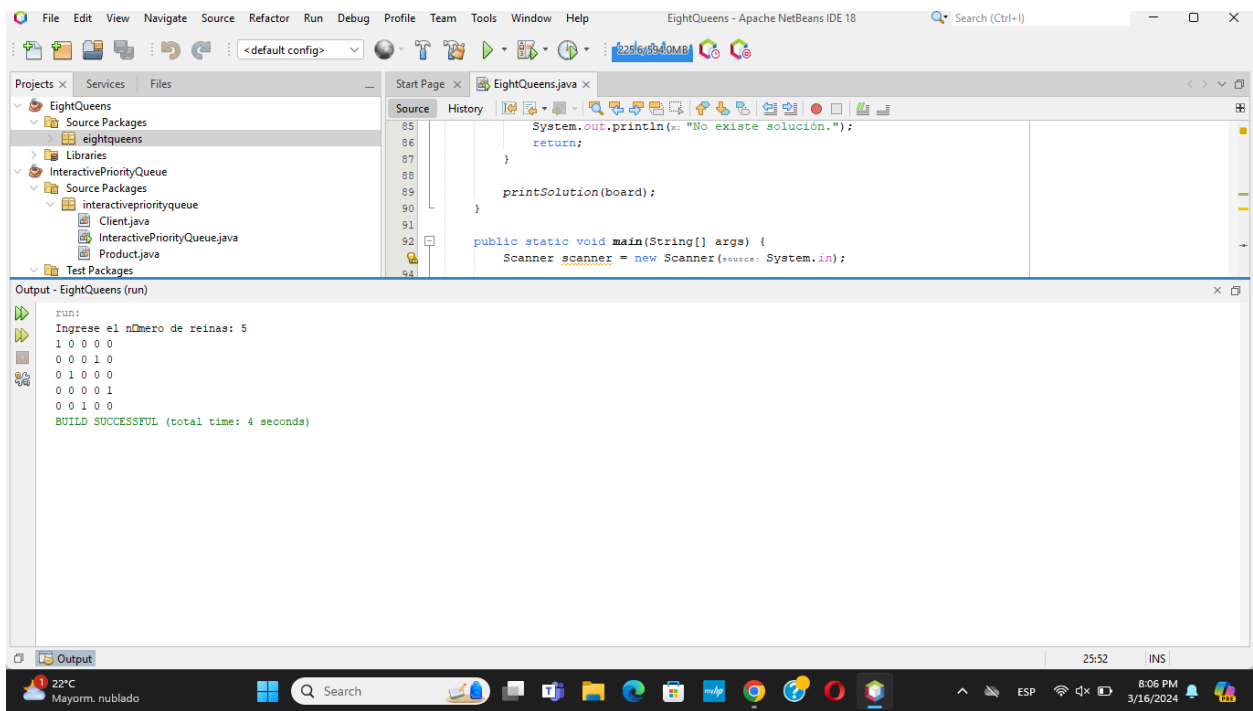
### 3 Reinas:



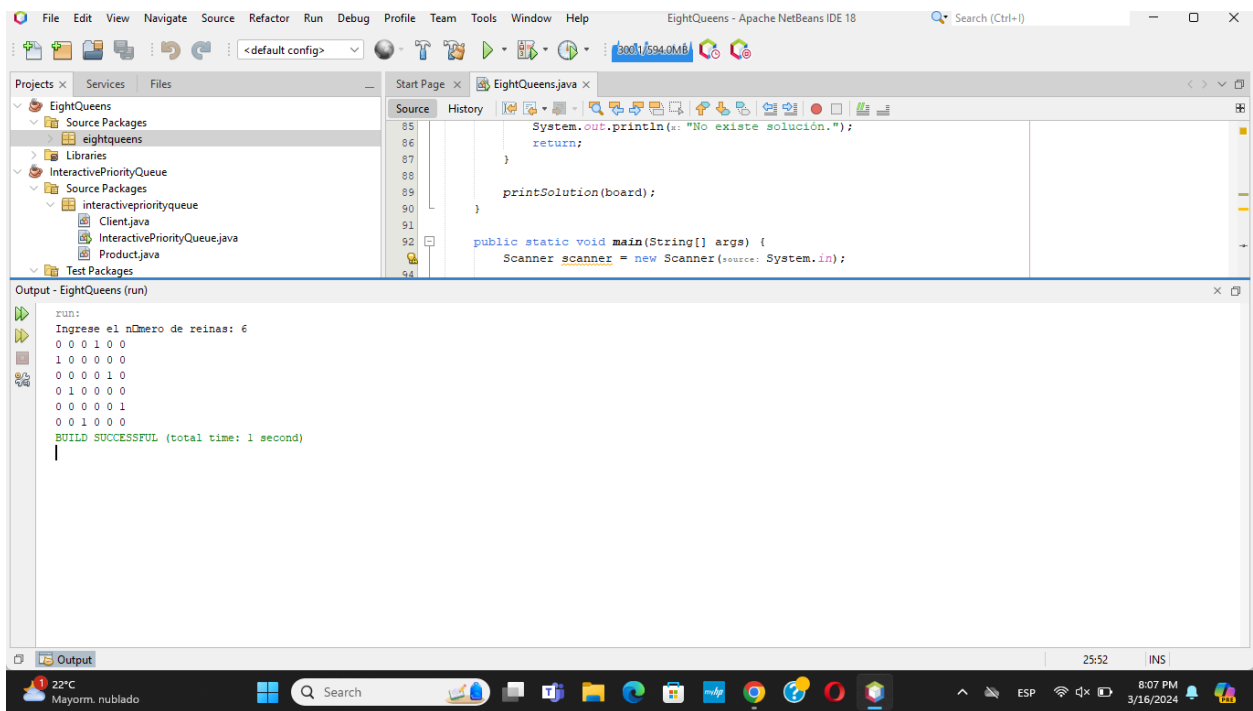
#### 4 Reinas:



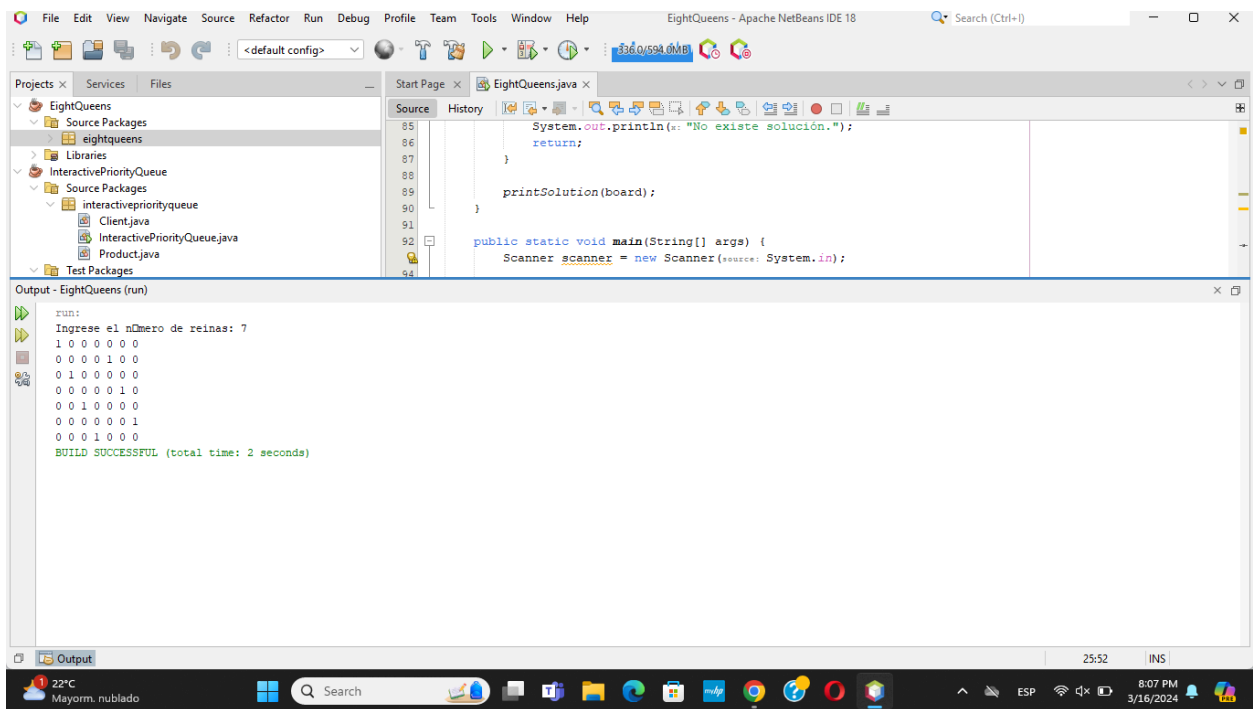
#### 5 reinas:



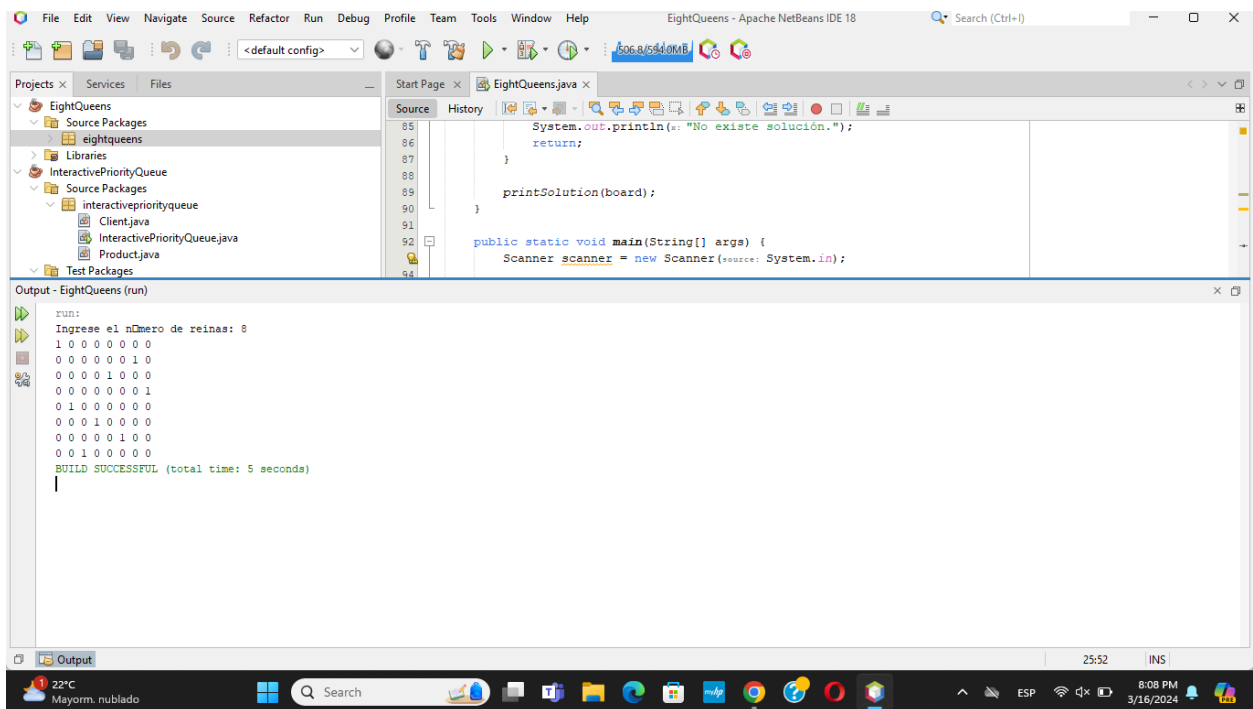
## 6 Reinas:



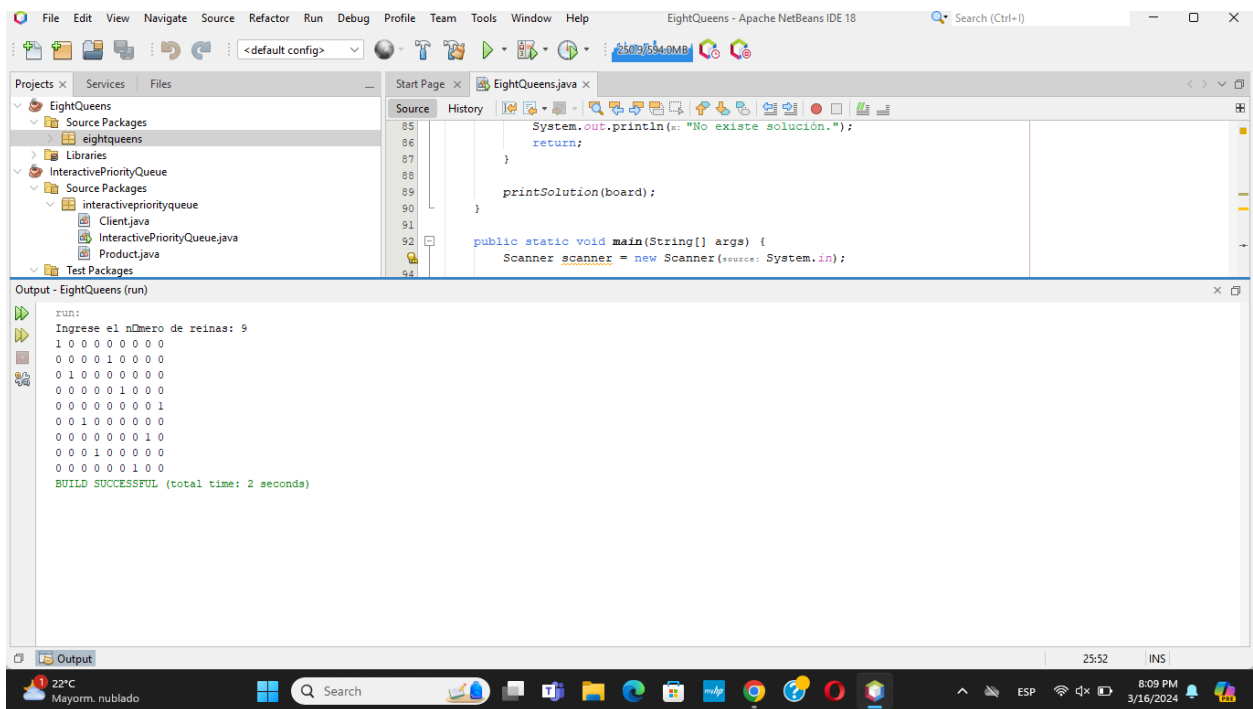
## 7 Reinas:



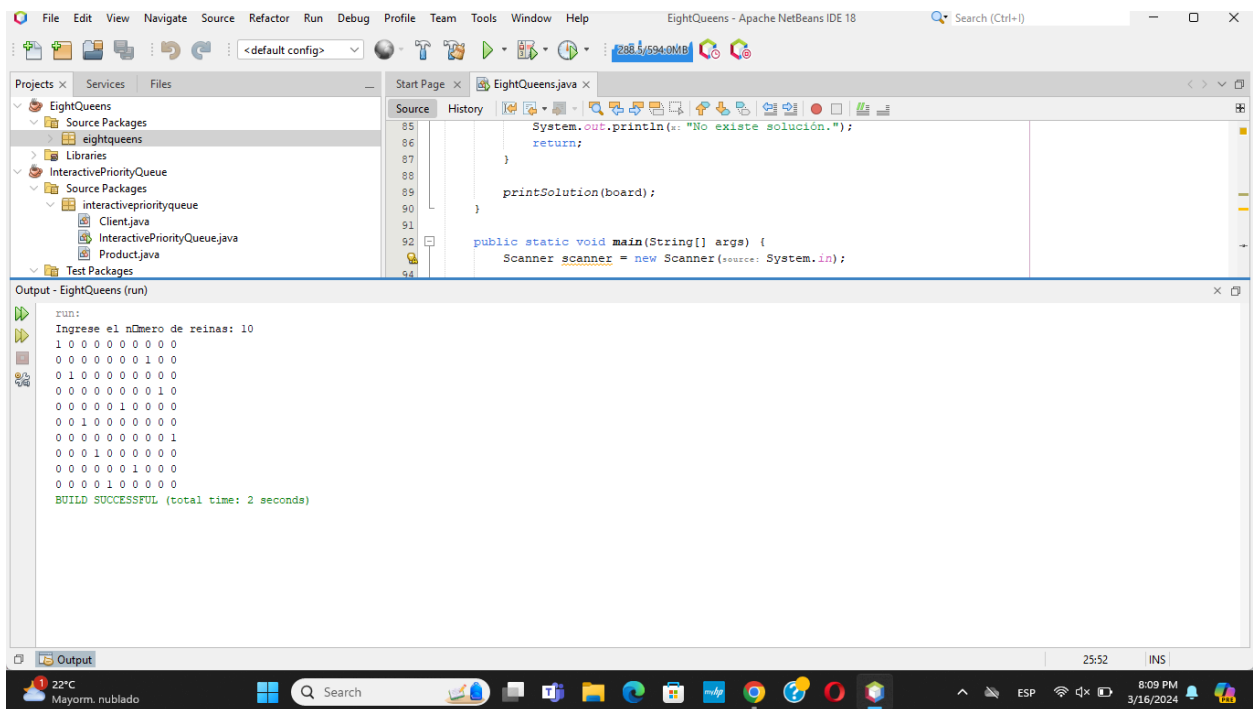
## 8 Reinas:



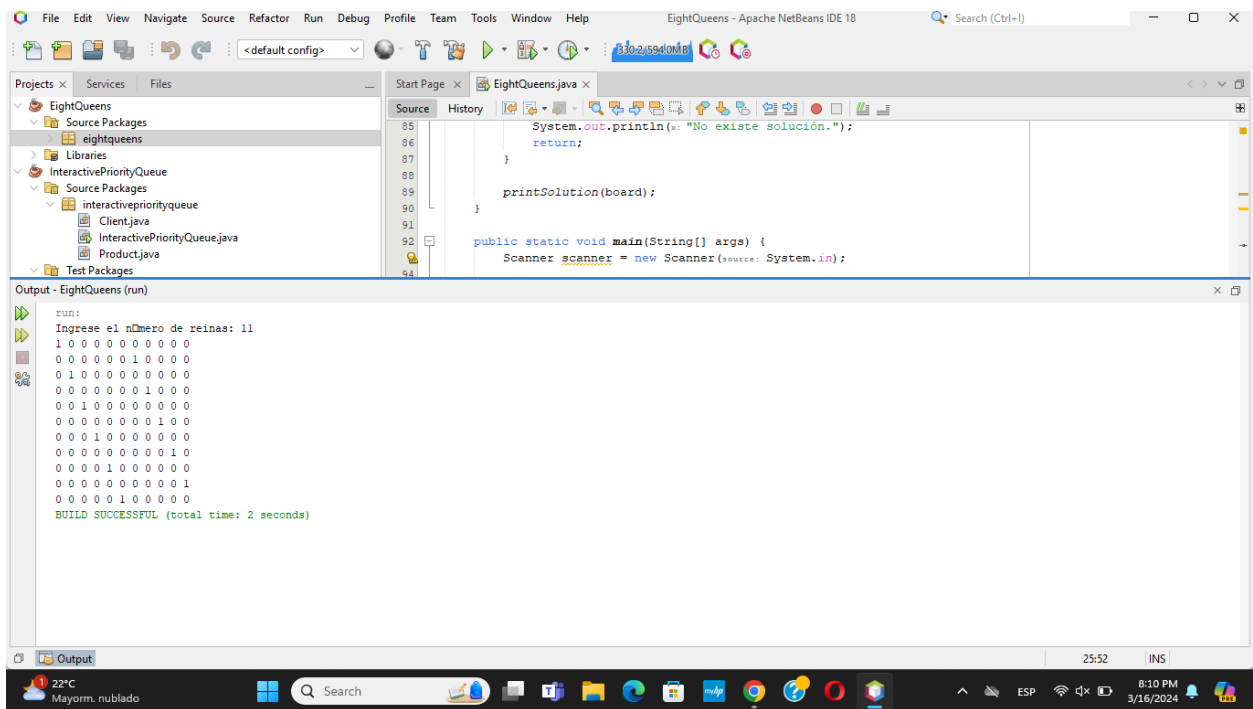
## 9 Reinas:



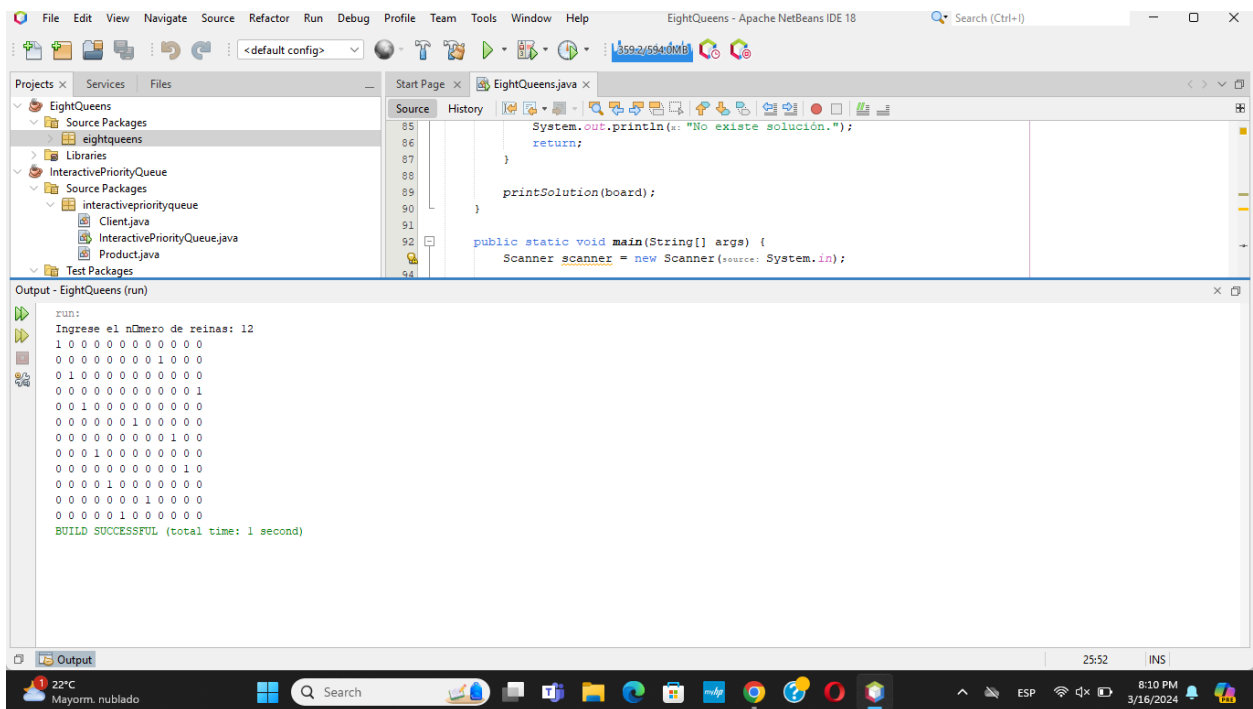
10 Reinas:



11 Reinas:



12 Reinas:



13 Reinas:





File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help EightQueens - Apache NetBeans IDE 18 Search (Ctrl+I)

Projects Services Files

- EightQueens
  - Source Packages
    - eightQueens
  - Libraries
  - InteractivePriorityQueue
    - Source Packages
      - interactivepriorityqueue
    - Client.java
    - InteractivePriorityQueue.java
    - Product.java
  - Test Packages

Start Page x EightQueens.java x

Source History

```
85 System.out.println("No existe solución.");
86 return;
87 }
88
89 printSolution(board);
90 }
91
92 public static void main(String[] args) {
93     Scanner scanner = new Scanner(System.in);
94 }
```

Output - EightQueens (run)

run:

```
Ingrese el número de reinas: 15 Reinas
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
BUILD SUCCESSFUL (total time: 5 seconds)
```

25:52 INS

22°C Mayorm. nublado Search

16 Reinas:

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help EightQueens - Apache NetBeans IDE 18 Search (Ctrl+I)

Projects Services Files

- EightQueens
  - Source Packages
    - eightQueens
  - Test Packages
  - Libraries
  - Test Libraries
  - InteractivePriorityQueue
    - Source Packages
      - interactivepriorityqueue
    - Client.java
    - InteractivePriorityQueue.java

Start Page x EightQueens.java x

Source History

```
10 */
11 import java.util.Scanner;
12
13 public class EightQueens {
14
15     // Función para imprimir la solución
16     private static void printSolution(int[][] board) {
17         for (int i = 0; i < board.length; i++) {
18             for (int j = 0; j < board.length; j++) {
```

Output - EightQueens (run)

run:

```
Ingrese el número de reinas: 16
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
BUILD SUCCESSFUL (total time: 4 seconds)
```

26:5 INS

22°C Mayorm. nublado Search

17 Reinas:



La verdad estos temas nuevos que estamos viendo de Estructura de datos, si son algo complicados, la verdad al principio puede ser algo confuso de entender, aunque este tipo de

algoritmos seguramente los tendremos que implementar en algo muy específico que nos pidan programar en el trabajo.

Igor:

Lo que aprendí de backtracking es que lo más importante es saber como hacer la que la función recursiva detecte si ya paso una vez por la instrucción o no y cómo hacer que se corran correctamente todas las posibles opciones, y para hacer esto correctamente lo más importante es hacer correctamente las condiciones de repetición y salida.

Juan Pablo:

Aunque en gran parte de este trabajo estuvo a manos de mi compañero Igor, el trabajo fue algo divertido y un verdadero reto para mí, gracias a él pude comprender el funcionamiento de este para poder redactar este reporte.