

Programación I

Estructuras Dinámicas

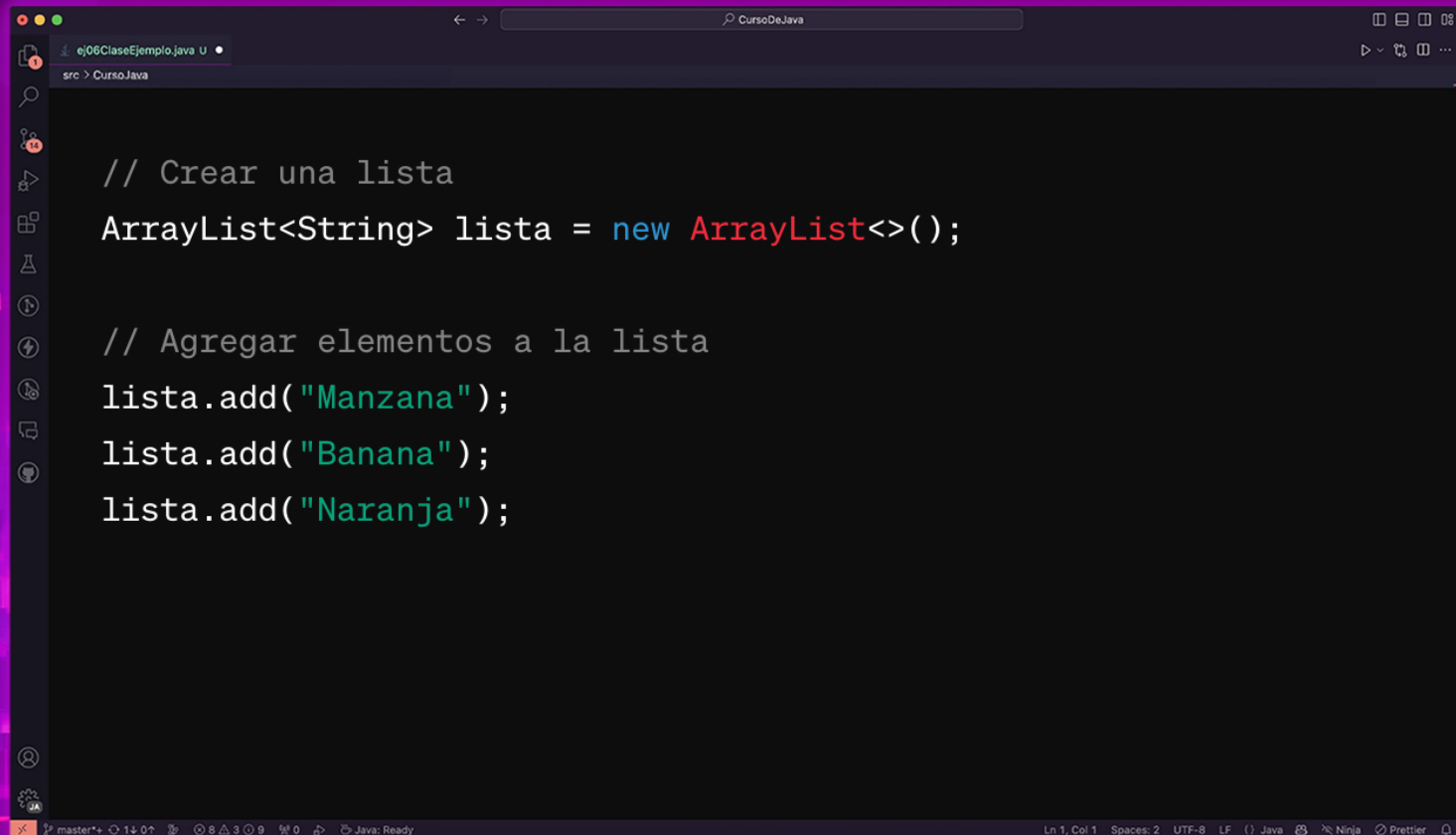
Listas

Una lista es una estructura de datos que almacena una colección ordenada de elementos. En Java, una de las implementaciones más comunes es ArrayList, que es una lista redimensionable que implementa la interfaz List.

Características

- ❑ **Tamaño Dinámico:** Puedes agregar o eliminar elementos en cualquier momento sin tener que preocuparte por el tamaño inicial del array.
- ❑ **Implementa la Interfaz List:** ArrayList implementa la interfaz List, lo que significa que hereda todas las operaciones definidas en esa interfaz, como agregar, eliminar, buscar elementos, entre otras.
- ❑ **Acceso Aleatorio Eficiente:** Puedes acceder a cualquier elemento en la lista directamente mediante su índice.
- ❑ **Permite Duplicados y Elementos Nulos:** Permiten elementos duplicados y pueden contener elementos nulos (null).
- ❑ **Iteración Eficiente:** Los ArrayList proporcionan métodos eficientes para iterar sobre los elementos de la lista, como el uso de bucles for-each o la iteración a través de índices.

Crear y Agregar elementos

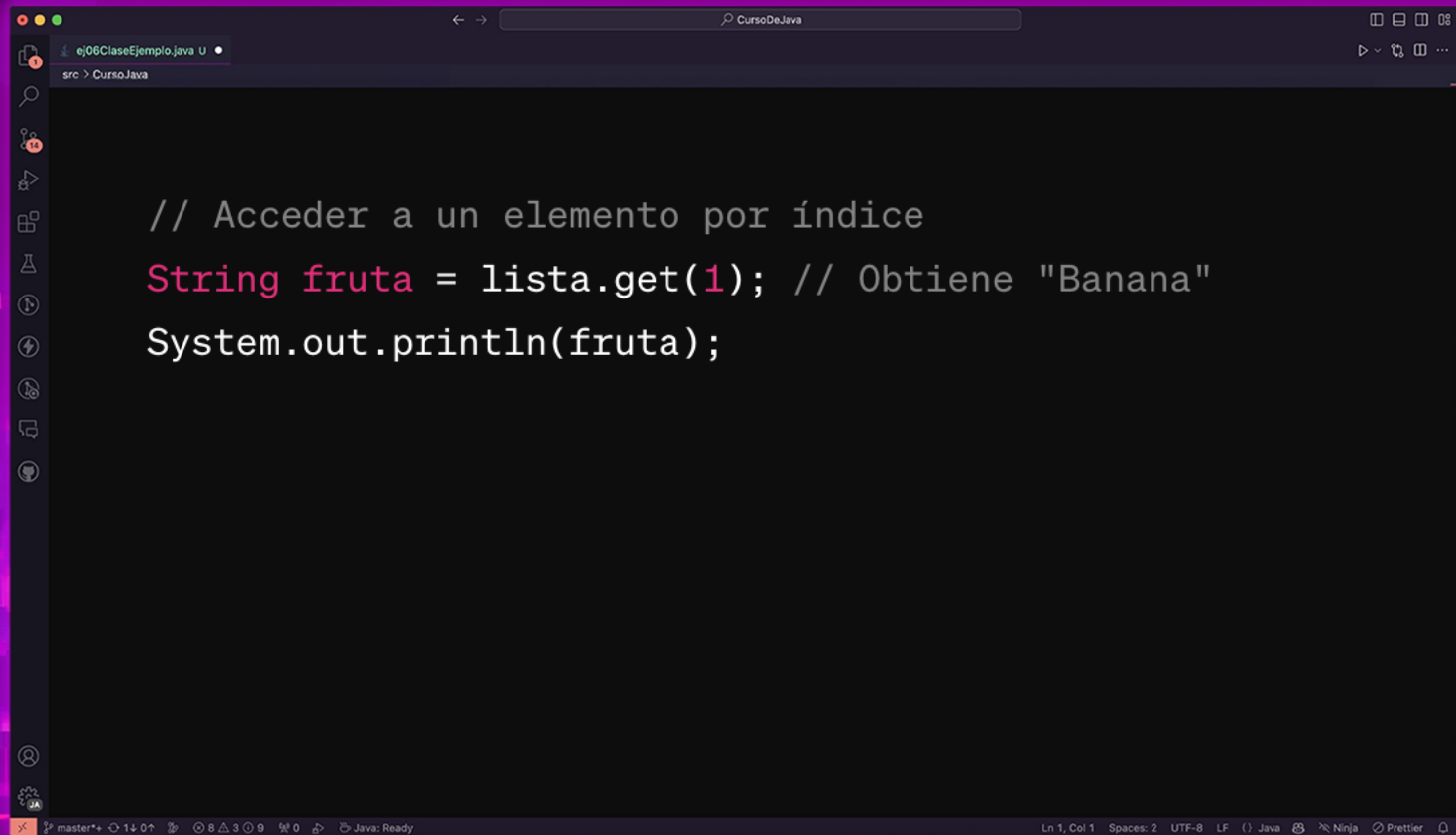


The image shows a screenshot of an IDE window with a dark theme. The title bar at the top indicates the file is 'ej06ClaseEjemplo.java' and the project is 'CursoDeJava'. The code editor contains the following Java code:

```
// Crear una lista  
ArrayList<String> lista = new ArrayList<>();  
  
// Agregar elementos a la lista  
lista.add("Manzana");  
lista.add("Banana");  
lista.add("Naranja");
```

The IDE interface includes a sidebar on the left with icons for Explorer, Search, Run and Debug, Source Control, and Extensions. The bottom status bar shows 'master' branch, 140 commits, 8 files, 3 folders, 0 errors, and 0 warnings. It also indicates 'Java: Ready' and shows the current cursor position as 'Ln 1, Col 1' with 'Spaces: 2', 'UTF-8', 'LF', and 'Java' encoding. The status bar also lists 'Ninja' and 'Prettier' as installed extensions.

Acceder a un elemento

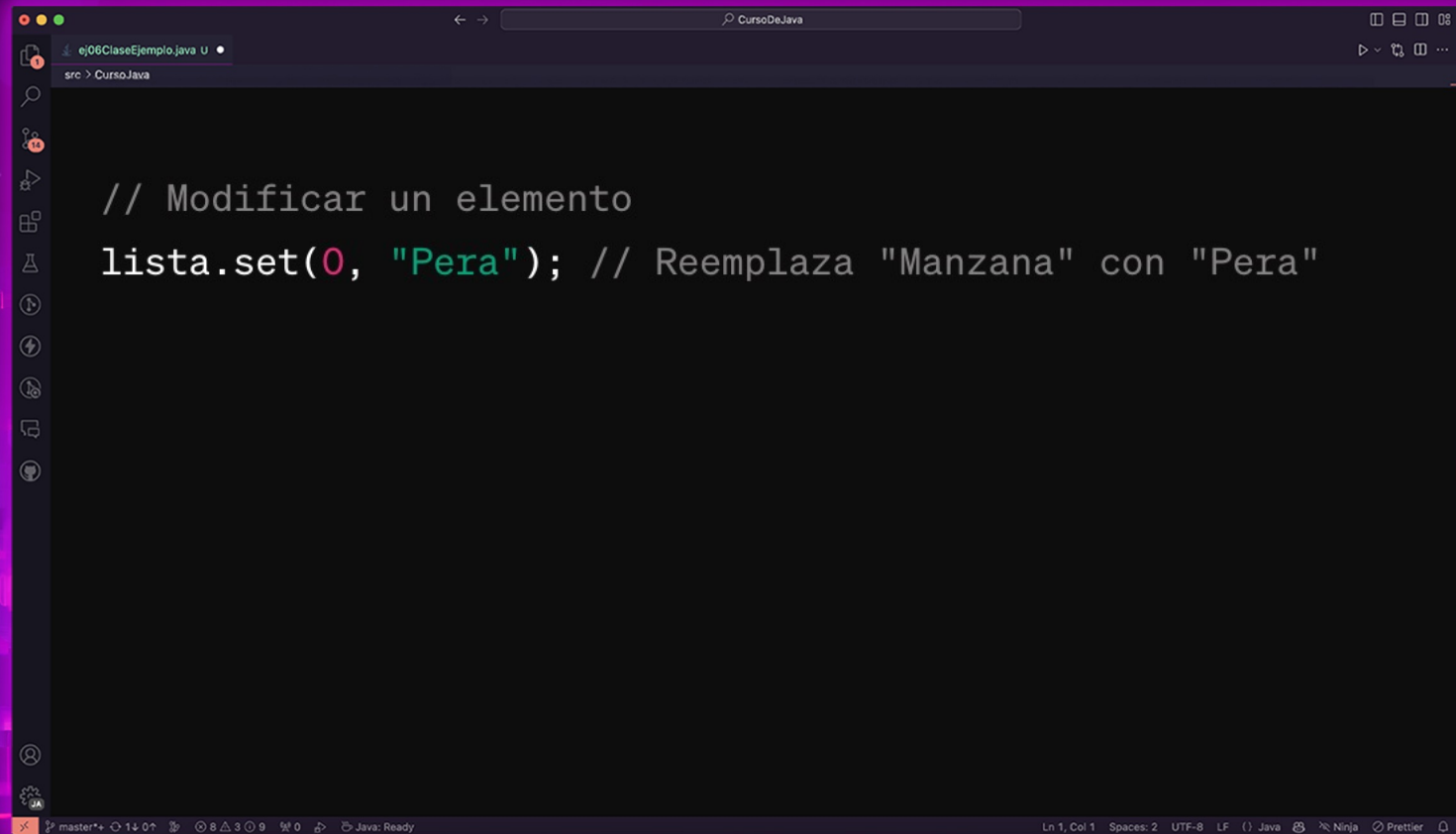


The image shows a screenshot of an IDE window with a dark theme. The title bar at the top indicates the file is 'ej06ClaseEjemplo.java' and the project is 'CursoDeJava'. The code editor contains the following Java code:

```
// Acceder a un elemento por índice  
String fruta = lista.get(1); // Obtiene "Banana"  
System.out.println(fruta);
```

The IDE interface includes a sidebar on the left with various icons for file management and a status bar at the bottom showing 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', '() Java', and 'Ninja Prettier'.

Modificar elementos

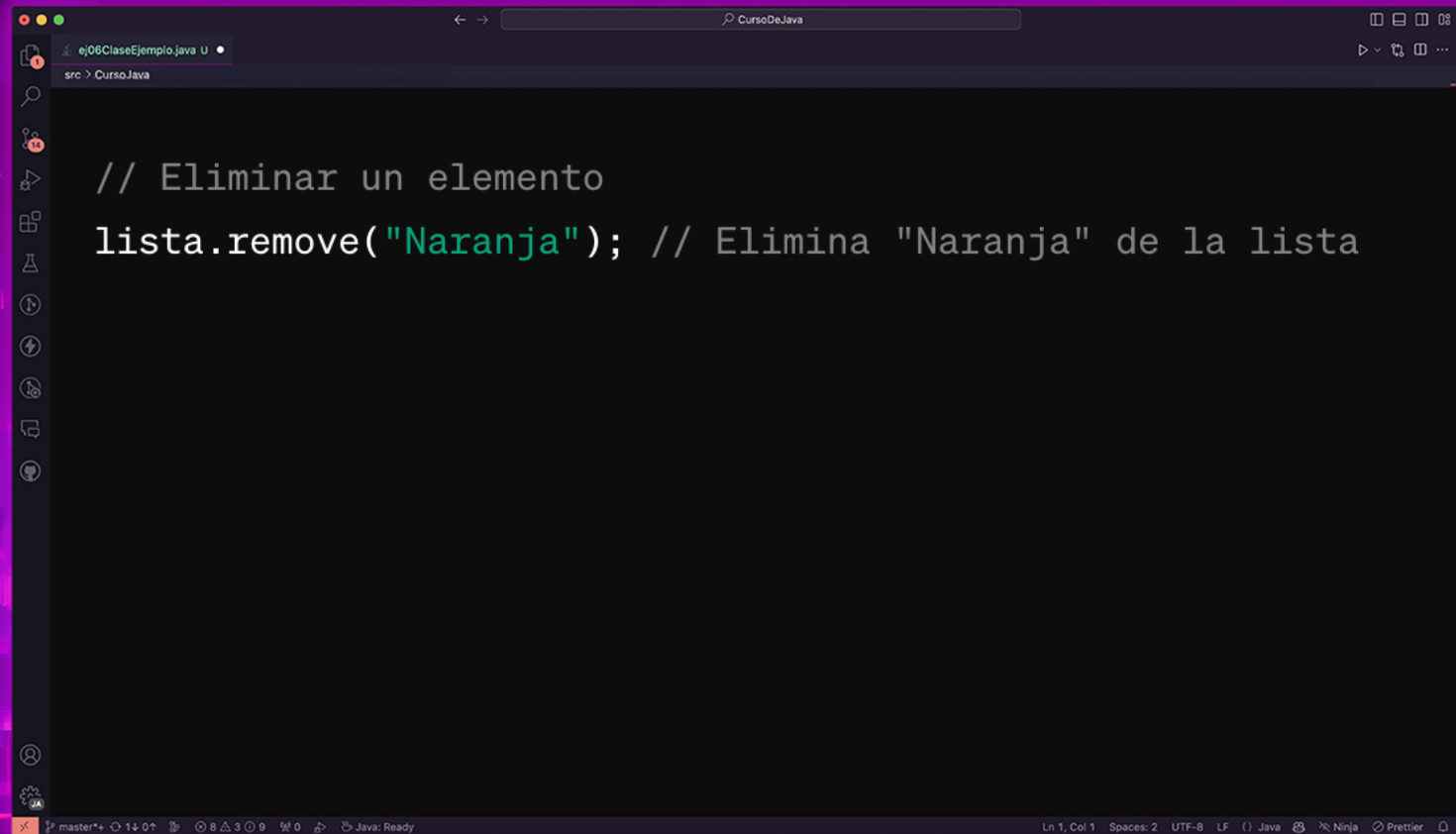


The screenshot shows an IDE window with a dark theme. The title bar at the top reads "CursoDeJava". The editor contains two lines of Java code: a comment and a list modification statement. The code is as follows:

```
// Modificar un elemento  
lista.set(0, "Pera"); // Reemplaza "Manzana" con "Pera"
```

The IDE interface includes a sidebar on the left with various icons for file management and development tools. The bottom status bar shows the current file is "master*", the cursor is at "Ln 1, Col 1", and the encoding is "UTF-8".

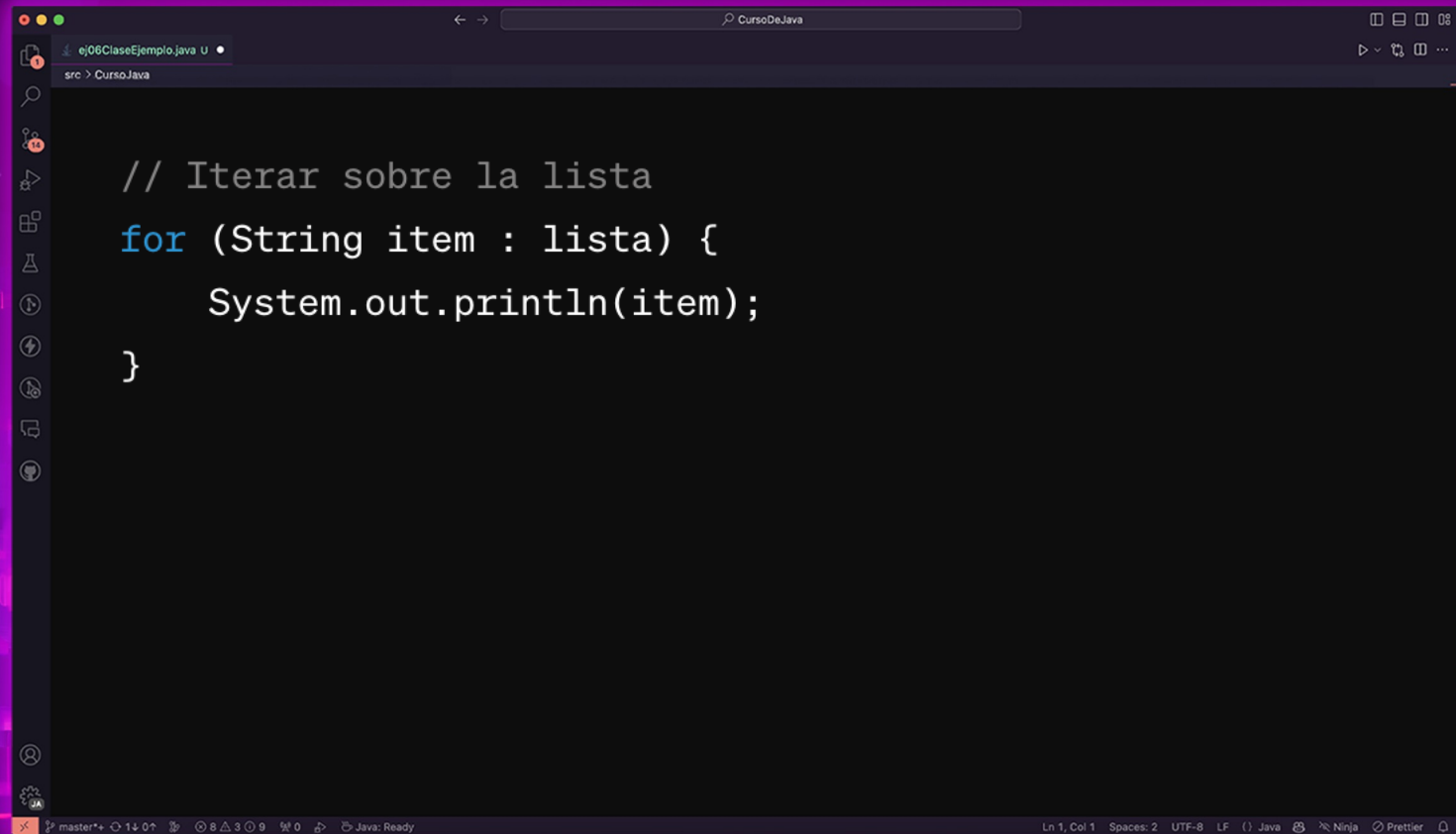
Eliminar elementos



The image shows a screenshot of an IDE window with a dark theme. The title bar at the top indicates the file is 'ej06ClaseEjemplo.java' and the project is 'CursoDeJava'. The code editor contains two lines of Java code: a comment '// Eliminar un elemento' and a line of code 'lista.remove("Naranja"); // Elimina "Naranja" de la lista'. The IDE interface includes a sidebar on the left with icons for Explorer, Search, Run and Debug, Source Control, Extensions, Testing, Remote Explorer, Docker, and Settings. The status bar at the bottom shows 'master* 14:07', a progress indicator, 'Java: Ready', and file encoding details 'Ln 1, Col 1 Spaces: 2 UTF-8 LF () Java Ninja Prettier'.

```
// Eliminar un elemento  
lista.remove("Naranja"); // Elimina "Naranja" de la lista
```

Iterar sobre la lista de elementos

A screenshot of a code editor window with a dark theme. The window title is 'ej06ClaseEjemplo.java U'. The breadcrumb shows 'src > CursoJava'. The code is in Java and demonstrates a for-each loop to iterate over a list. The status bar at the bottom shows 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', and 'Prettier'.

```
// Iterar sobre la lista
for (String item : lista) {
    System.out.println(item);
}
```


Pilas

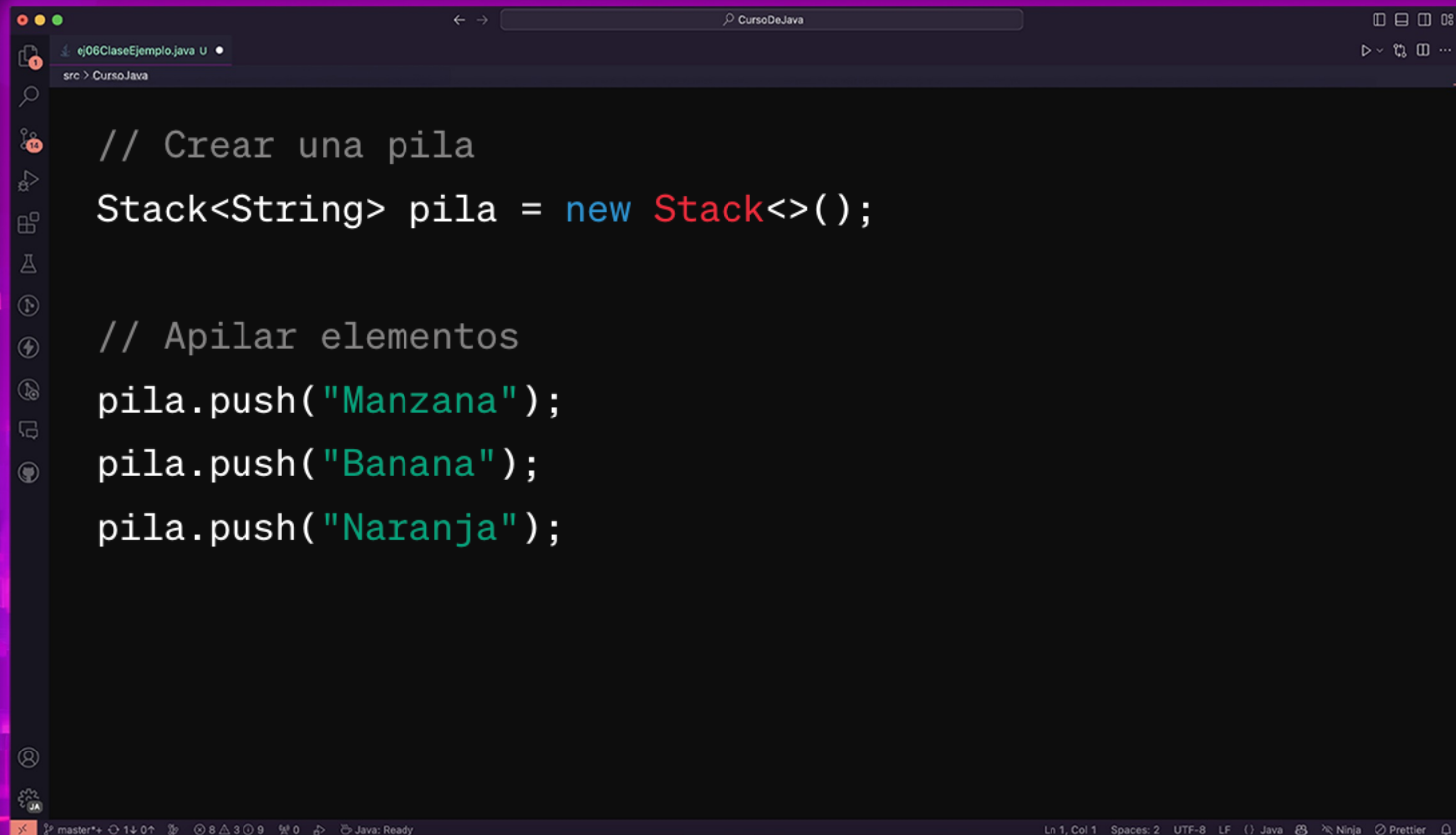
Una pila es una estructura de datos que sigue el principio LIFO (Last In, First Out), lo que significa que el último elemento en entrar es el primero en salir.

En Java, puedes implementar una pila utilizando Stack de la clase `java.util`.

Características

- ❑ **LIFO:** El último elemento agregado es el primero en ser removido. Esto lo hace útil en situaciones donde se necesita un comportamiento de apilamiento.
- ❑ **Operaciones Básicas:** Un Stack proporciona las operaciones básicas de una pila.
 - push() para agregar un elemento en la cima de la pila
 - pop() para remover y devolver el elemento en la cima de la pila
 - peek() para obtener el elemento en la cima sin removerlo.
- ❑ **Métodos Adicionales:** Además de las operaciones básicas, también proporciona otros métodos.
 - isEmpty() para verificar si la pila está vacía
 - search() para buscar un elemento y obtener su posición en la pila
 - size() para obtener el tamaño actual de la pila

Crear y Agregar elementos

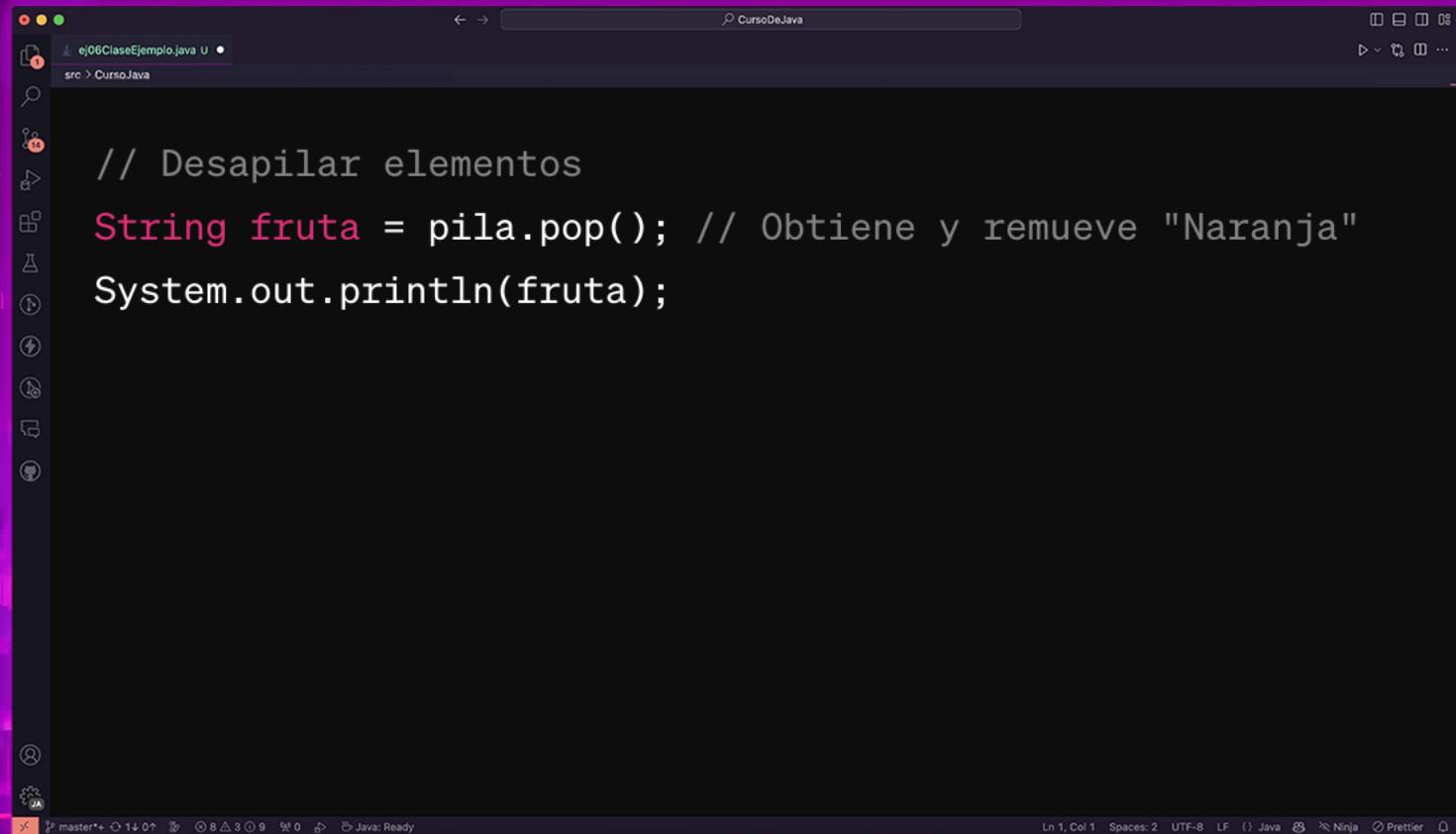
A screenshot of a code editor window with a dark theme. The window title is 'ej06ClaseEjemplo.java'. The breadcrumb navigation shows 'src > CursoJava'. The code is in Java and demonstrates creating a stack and pushing elements onto it. The code is:

```
// Crear una pila
Stack<String> pila = new Stack<>();

// Apilar elementos
pila.push("Manzana");
pila.push("Banana");
pila.push("Naranja");
```

The editor has a sidebar on the left with icons for Explorer, Search, Run and Debug, Source Control, Extensions, Testing, Remote Explorer, and User. The status bar at the bottom shows 'master* 140', '8 3 9', '0', 'Java: Ready', 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', 'Ninja', and 'Prettier'.

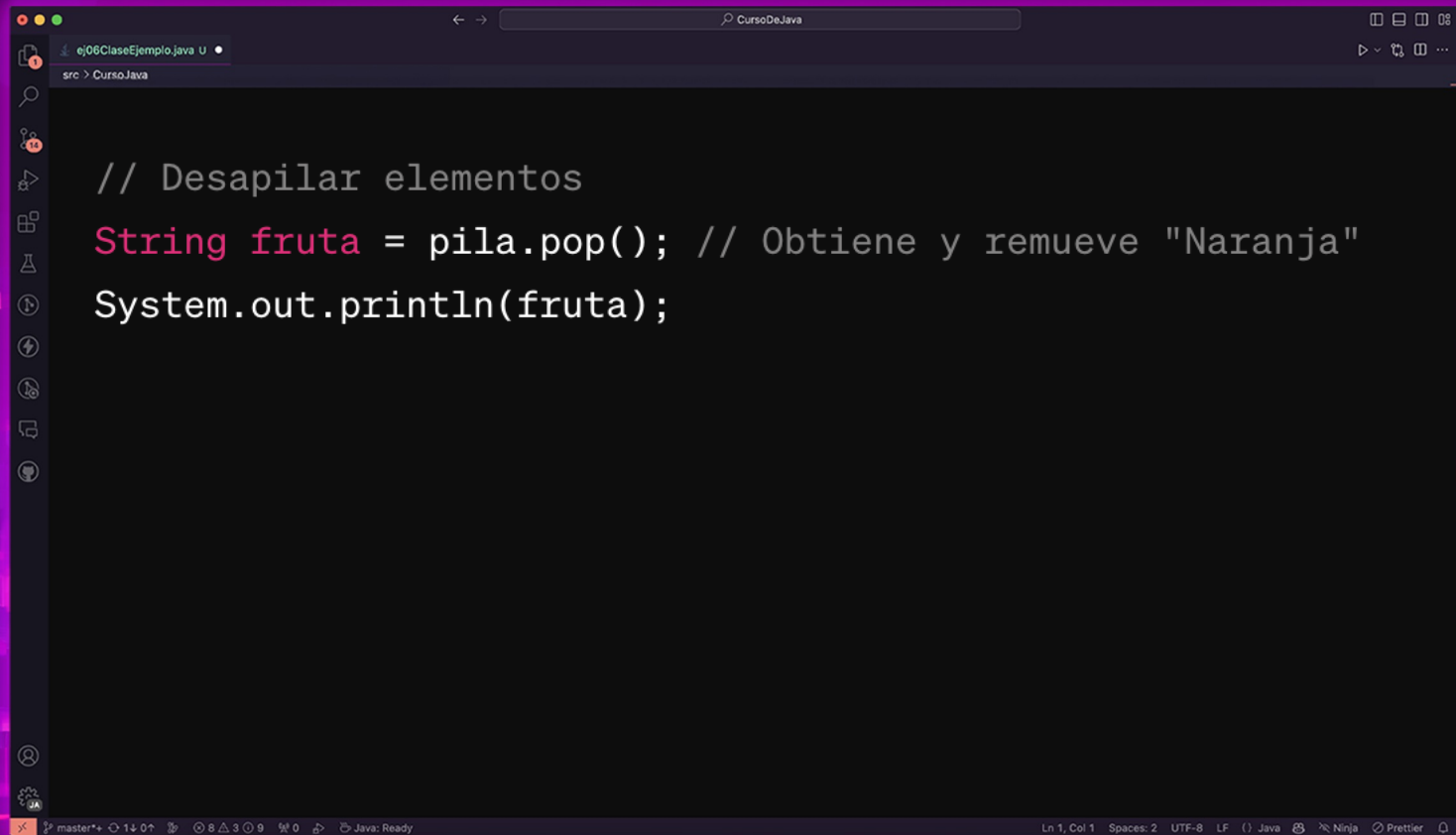
Desapilar elementos

A screenshot of a code editor window with a dark theme. The title bar shows 'ej06ClaseEjemplo.java' and 'CursoDeJava'. The code is in Java and demonstrates popping an element from a stack. The code is:

```
// Desapilar elementos  
String fruta = pila.pop(); // Obtiene y remueve "Naranja"  
System.out.println(fruta);
```

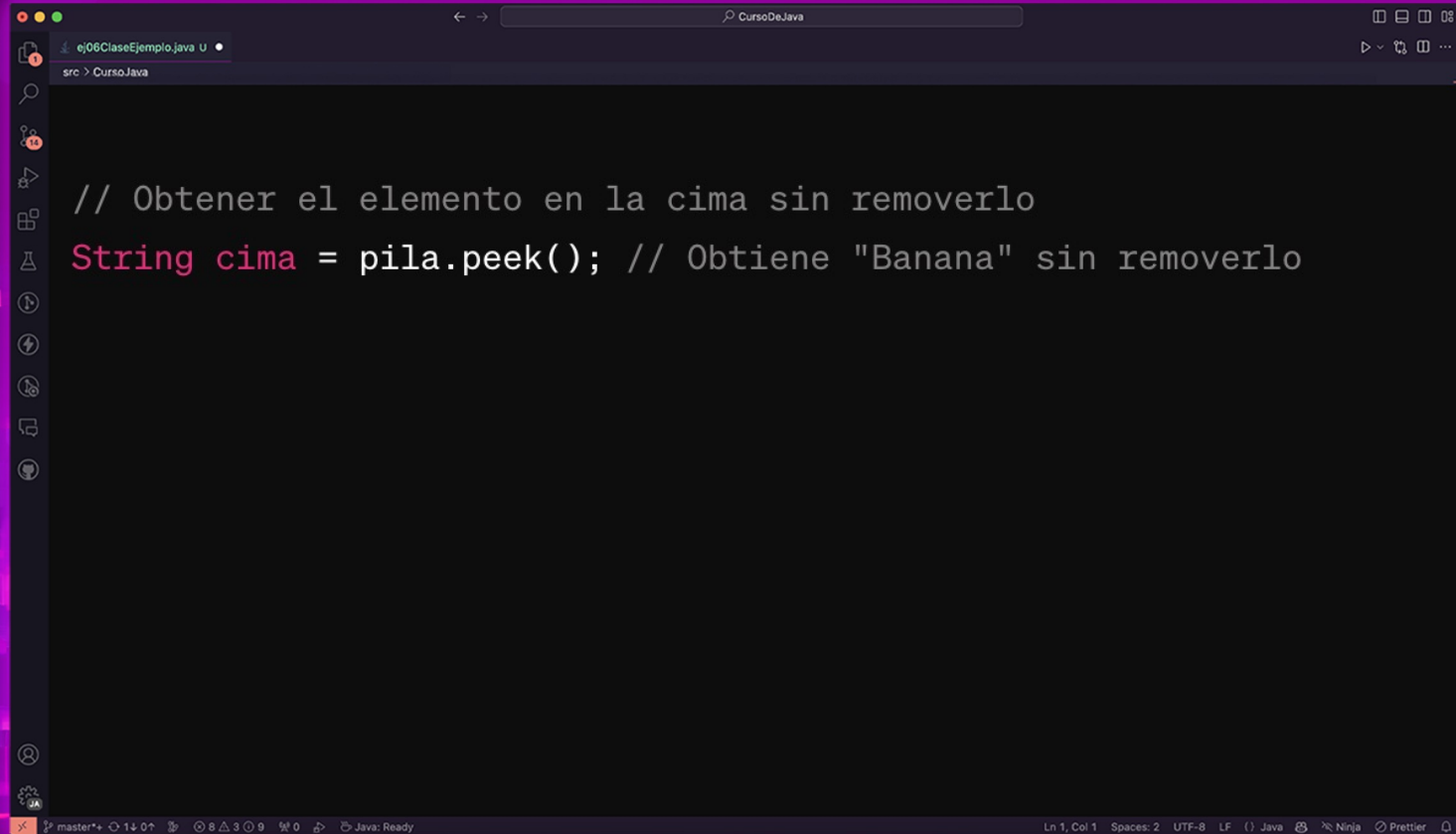
 The editor has a sidebar on the left with icons for Explorer, Search, Run and Debug, Source Control, Test Explorer, Extensions, and Settings. The status bar at the bottom shows 'master+', '140%', '8 3 9', '0', 'Java: Ready', 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', 'Ninja', and 'Prettier'.

Obtener y remover

A screenshot of a code editor window with a dark theme. The window title is 'ej06ClaseEjemplo.java U'. The breadcrumb shows 'src > CursoJava'. The code contains a comment and two lines of Java code. The variable 'fruta' is highlighted in red. The status bar at the bottom shows 'master*+ 14 0', '8 3 9', '0', 'Java: Ready', 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', 'Ninja', and 'Prettier'.

```
// Desapilar elementos
String fruta = pila.pop(); // Obtiene y remueve "Naranja"
System.out.println(fruta);
```


Obtener sin remover

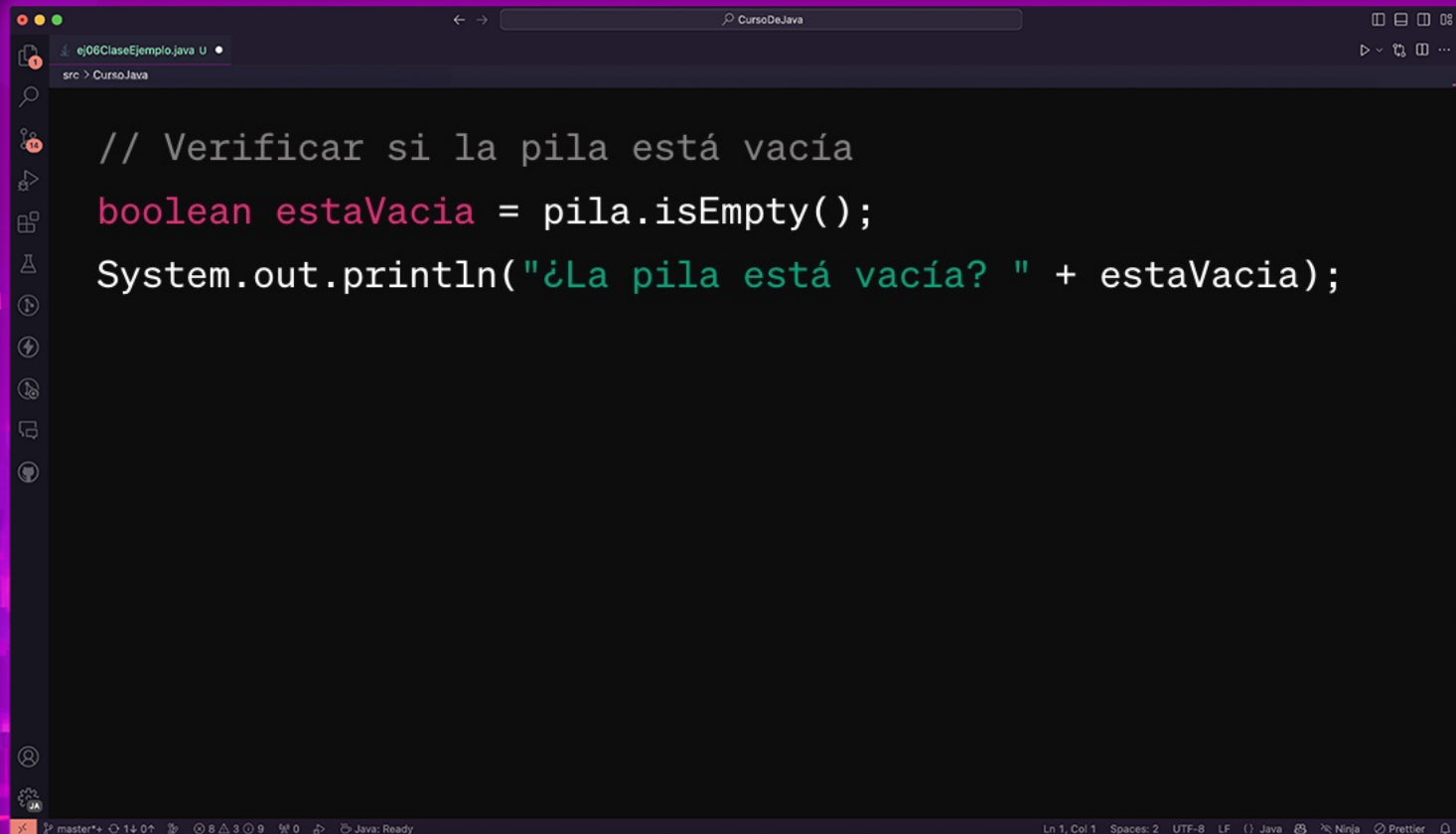


The image shows a screenshot of a code editor window. The title bar at the top reads "CursoDeJava". The editor is displaying a Java file named "ej06ClaseEjemplo.java". The code inside the editor is as follows:

```
// Obtener el elemento en la cima sin removerlo  
String cima = pila.peek(); // Obtiene "Banana" sin removerlo
```

The status bar at the bottom of the editor shows "Ln 1, Col 1", "Spaces: 2", "UTF-8", "LF", "Java", and "Prettier".

Verificar si la pila está vacía



The image shows a screenshot of a code editor window. The title bar at the top reads "CursoDeJava". The editor is displaying a Java file named "ej06ClaseEjemplo.java". The code inside the editor is as follows:

```
// Verificar si la pila está vacía  
boolean estaVacia = pila.isEmpty();  
System.out.println("¿La pila está vacía? " + estaVacia);
```

The code is color-coded: comments are in grey, keywords like "boolean" and "System.out.println" are in red, and string literals are in green. The editor's interface includes a sidebar on the left with various icons for file management and development tools. At the bottom, a status bar shows "master+ 140", "8 3 9", "0", "Java: Ready", "Ln 1, Col 1", "Spaces: 2", "UTF-8", "LF", "Java", "Ninja", and "Prettier".

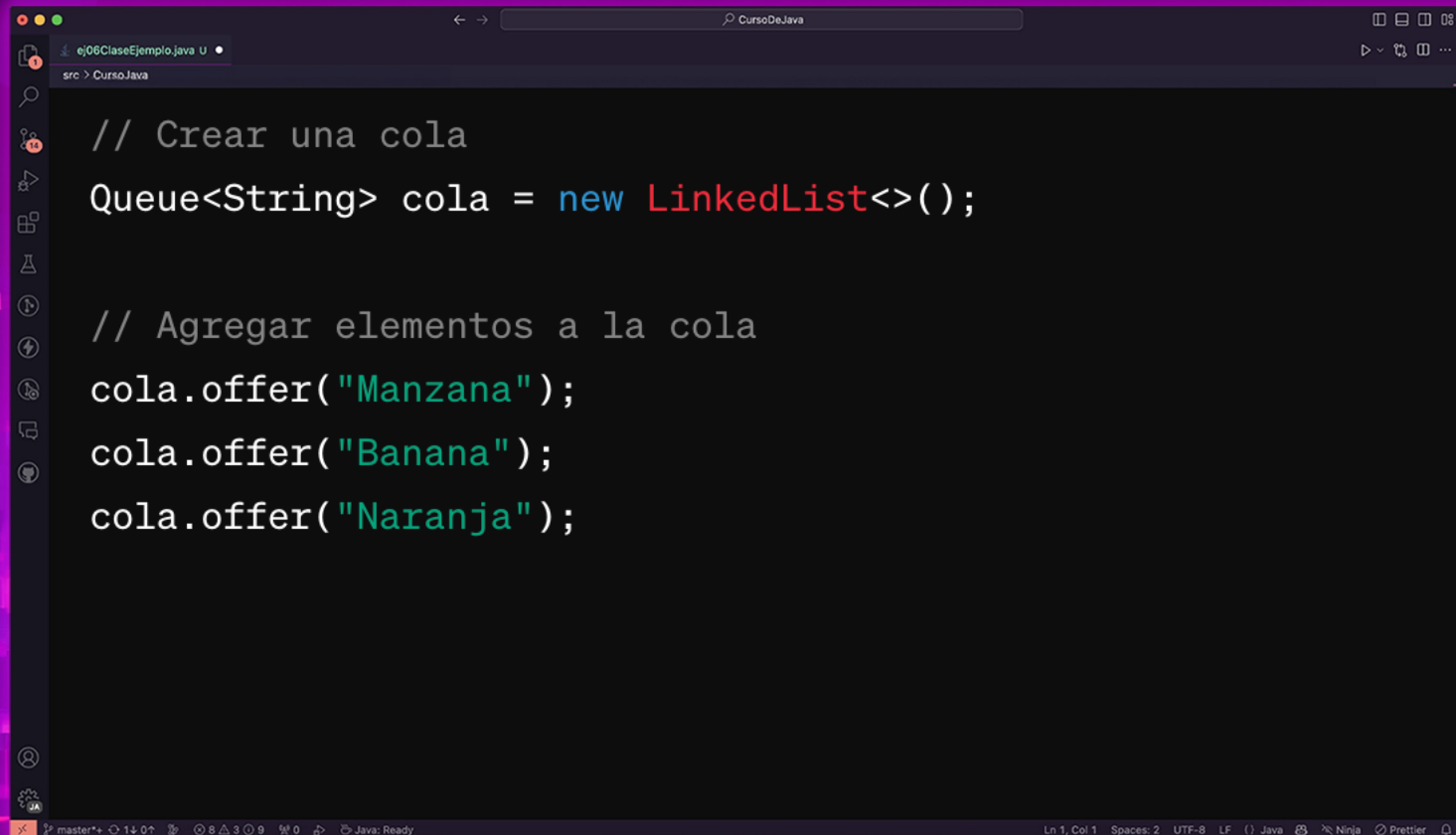
Colas

Una cola es una estructura de datos que sigue el principio FIFO (First In, First Out), lo que significa que el primer elemento en entrar es el primero en salir. En Java, puedes implementar una cola utilizando LinkedList de la clase java.util.

Características

- ❑ **Estructura de Datos de Lista Doblemente Enlazada:** Cada elemento está representado por un nodo que contiene referencias tanto al anterior como al siguiente en la lista. Esto permite un acceso eficiente tanto hacia adelante como hacia atrás en la lista.
- ❑ **Almacenamiento Dinámico:** Puede crecer o reducirse automáticamente según sea necesario.
- ❑ **Operaciones de Inserción y Eliminación Eficientes:** Debido a su estructura de lista doblemente enlazada, LinkedList es particularmente eficiente para operaciones de inserción y eliminación en cualquier posición de la lista.
- ❑ **Acceso Secuencial Eficiente:** Las operaciones como `get()` pueden ser menos eficientes que en un ArrayList. **Uso de Memoria Adicional:** Debido a la necesidad de almacenar referencias a los nodos previos y siguientes, LinkedList puede consumir más memoria que ArrayList, especialmente para listas grandes.
- ❑ **Iteración Eficiente:** LinkedList proporciona métodos eficientes para iterar sobre los elementos de la lista, ya sea mediante el uso de bucles `for-each` o iteradores.

Crear y Agregar elementos



```
// Crear una cola
Queue<String> cola = new LinkedList<>();

// Agregar elementos a la cola
cola.offer("Manzana");
cola.offer("Banana");
cola.offer("Naranja");
```

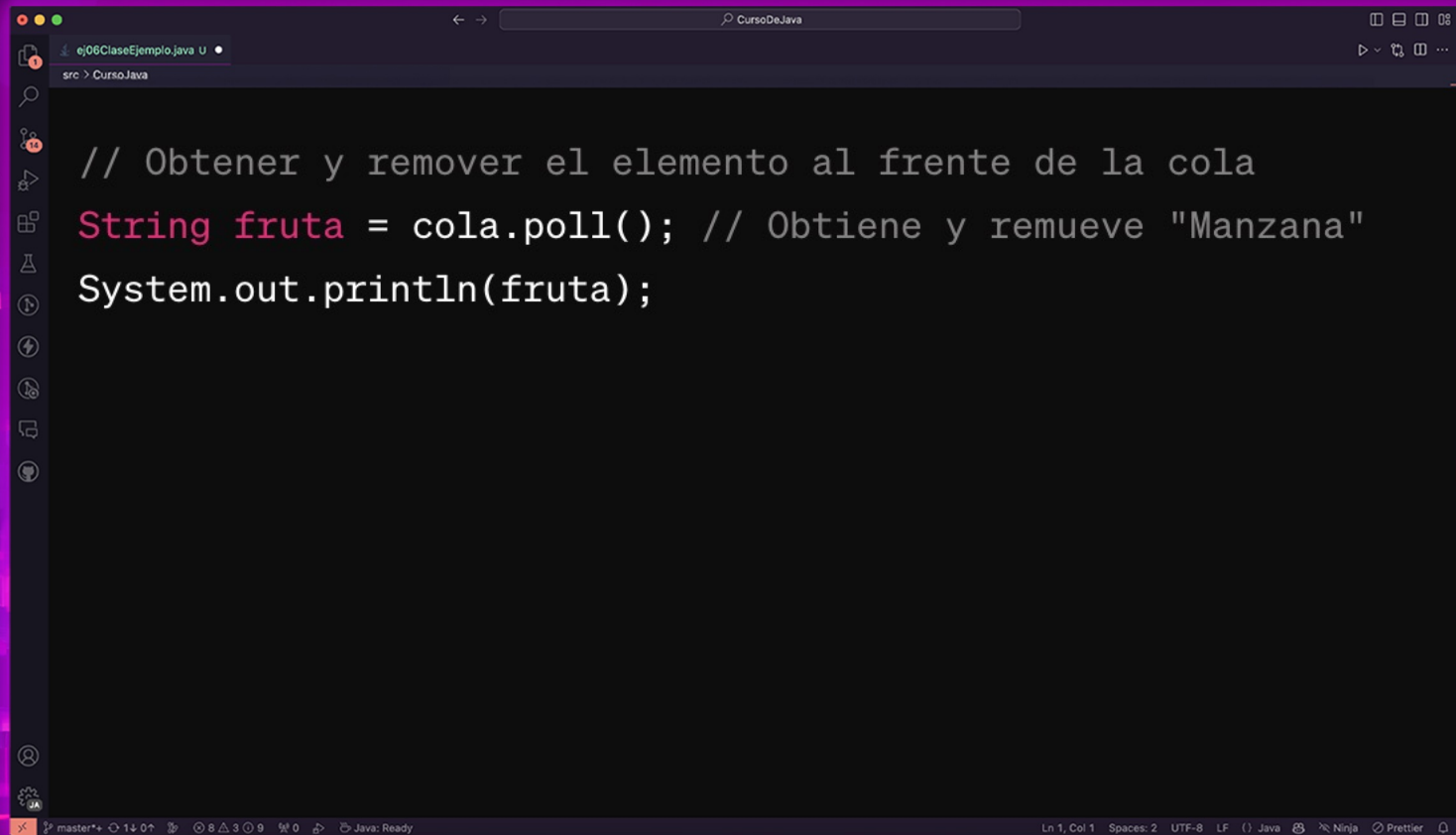
The image shows a screenshot of an IDE window titled 'CursoDeJava'. The code editor displays the following Java code:

```
// Crear una cola
Queue<String> cola = new LinkedList<>();

// Agregar elementos a la cola
cola.offer("Manzana");
cola.offer("Banana");
cola.offer("Naranja");
```

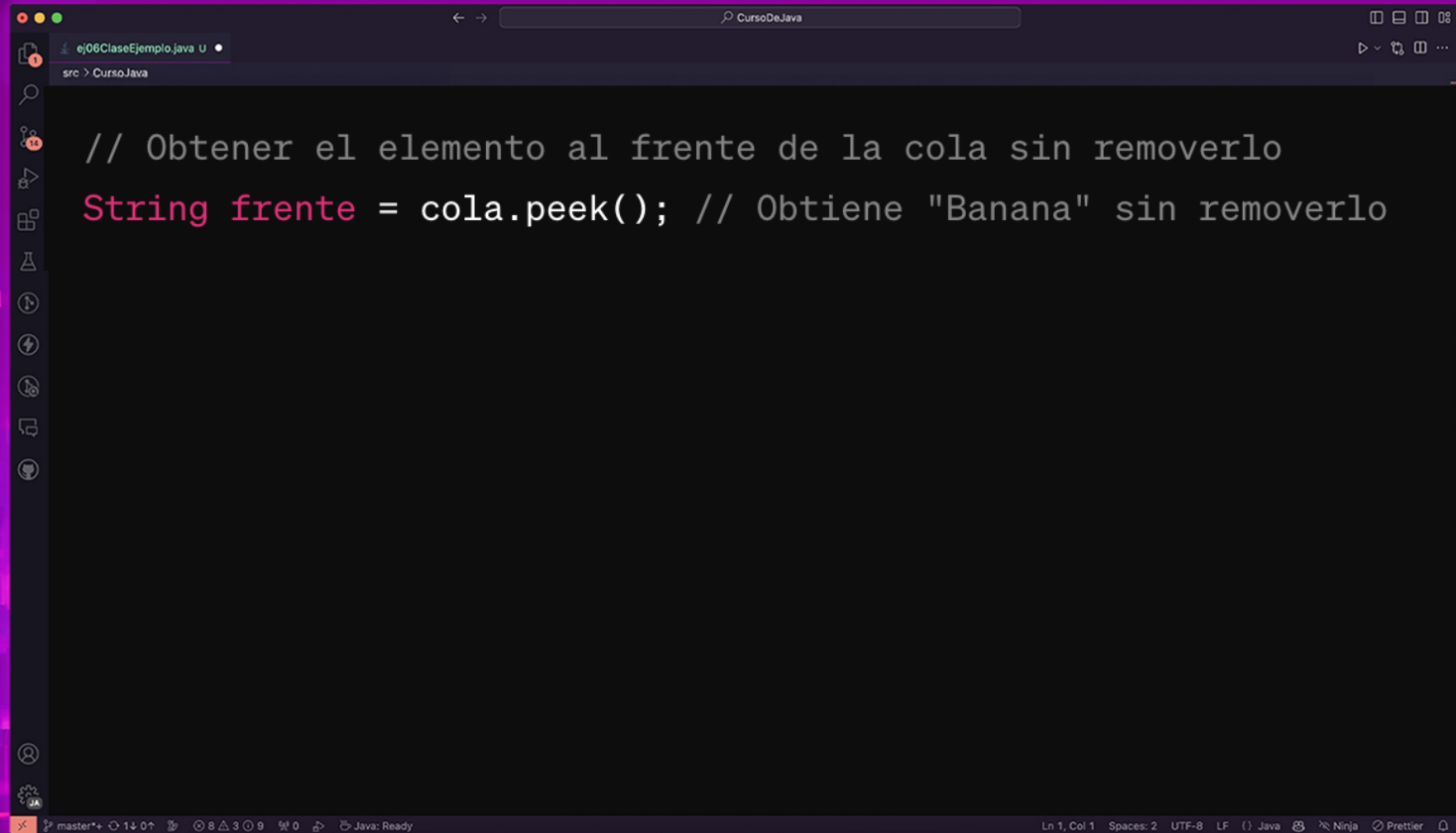
The IDE interface includes a sidebar with various icons on the left, a top toolbar with window and search icons, and a bottom status bar showing 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', and 'Prettier'.

Obtener y remover

A screenshot of a code editor window with a dark theme. The window title is 'ej06ClaseEjemplo.java U'. The breadcrumb shows 'src > CursoJava'. The code contains a comment and two lines of Java code. The first line is a comment: '// Obtener y remover el elemento al frente de la cola'. The second line is a Java statement: 'String fruta = cola.poll(); // Obtiene y remueve "Manzana"'. The third line is another Java statement: 'System.out.println(fruta);'. The status bar at the bottom shows 'master*+ 140', '8 3 9', '0', 'Java: Ready', 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', 'Ninja', and 'Prettier'.

```
// Obtener y remover el elemento al frente de la cola
String fruta = cola.poll(); // Obtiene y remueve "Manzana"
System.out.println(fruta);
```

Obtener sin remover

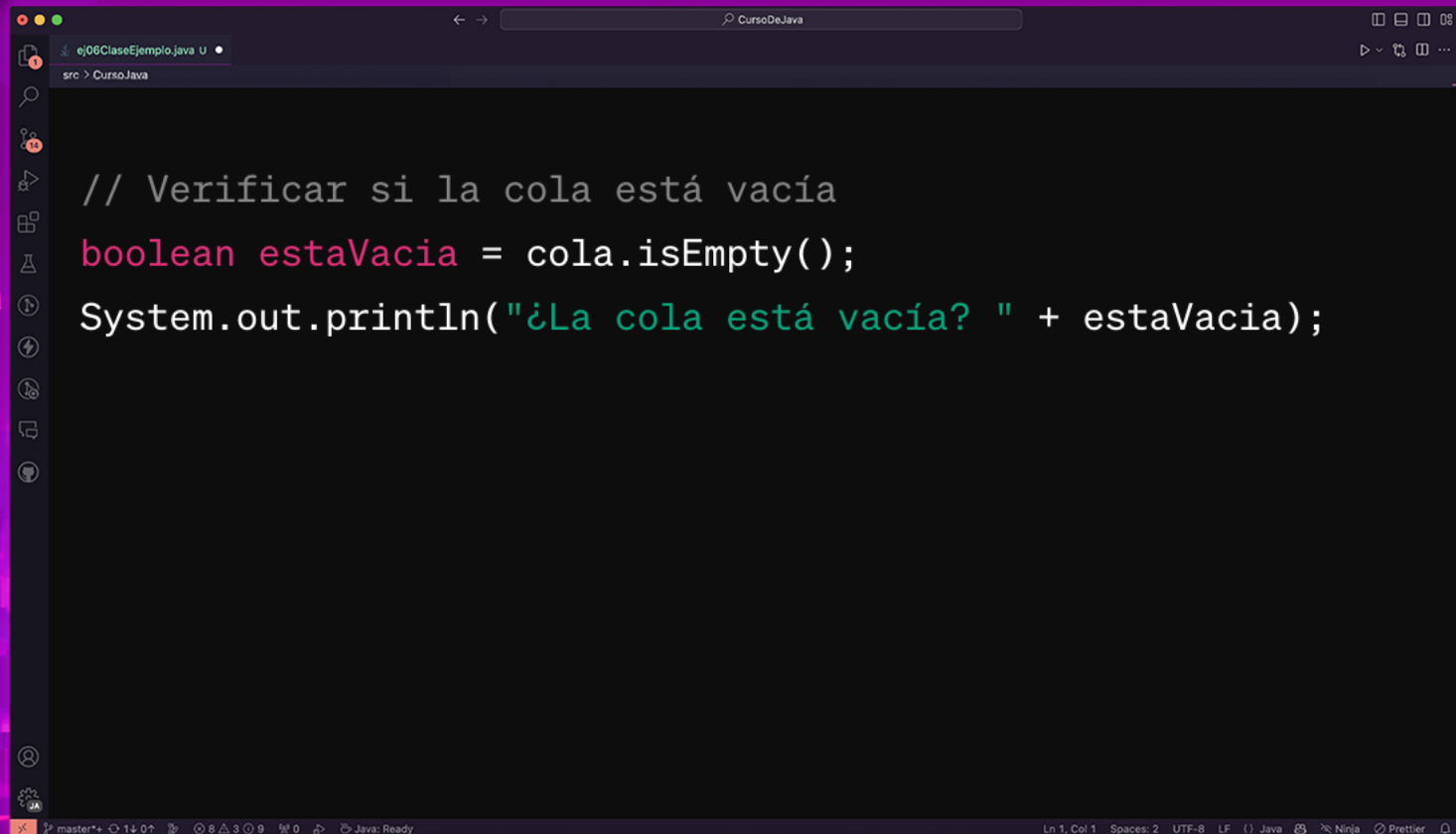


The image shows a screenshot of a code editor window. The title bar at the top indicates the file is 'ej06ClaseEjemplo.java' and the project is 'CursoDeJava'. The editor contains two lines of Java code: a comment and a line of code using the 'peek()' method. The code is as follows:

```
// Obtener el elemento al frente de la cola sin removerlo
String frente = cola.peek(); // Obtiene "Banana" sin removerlo
```

The status bar at the bottom of the editor shows 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', and 'Prettier'.

Verificar si la cola está vacía

A screenshot of a code editor window with a dark theme. The window title is 'ej06ClaseEjemplo.java U'. The address bar shows 'CursoDeJava'. The code is in Java and checks if a queue is empty. The status bar at the bottom shows 'master*+ 140', '8 3 9', '0', 'Java: Ready', 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'Java', 'Ninja', and 'Prettier'.

```
// Verificar si la cola está vacía
boolean estaVacia = cola.isEmpty();
System.out.println("¿La cola está vacía? " + estaVacia);
```