Programación Orientada a Objetos

Parcial Individual

Elaborado Por

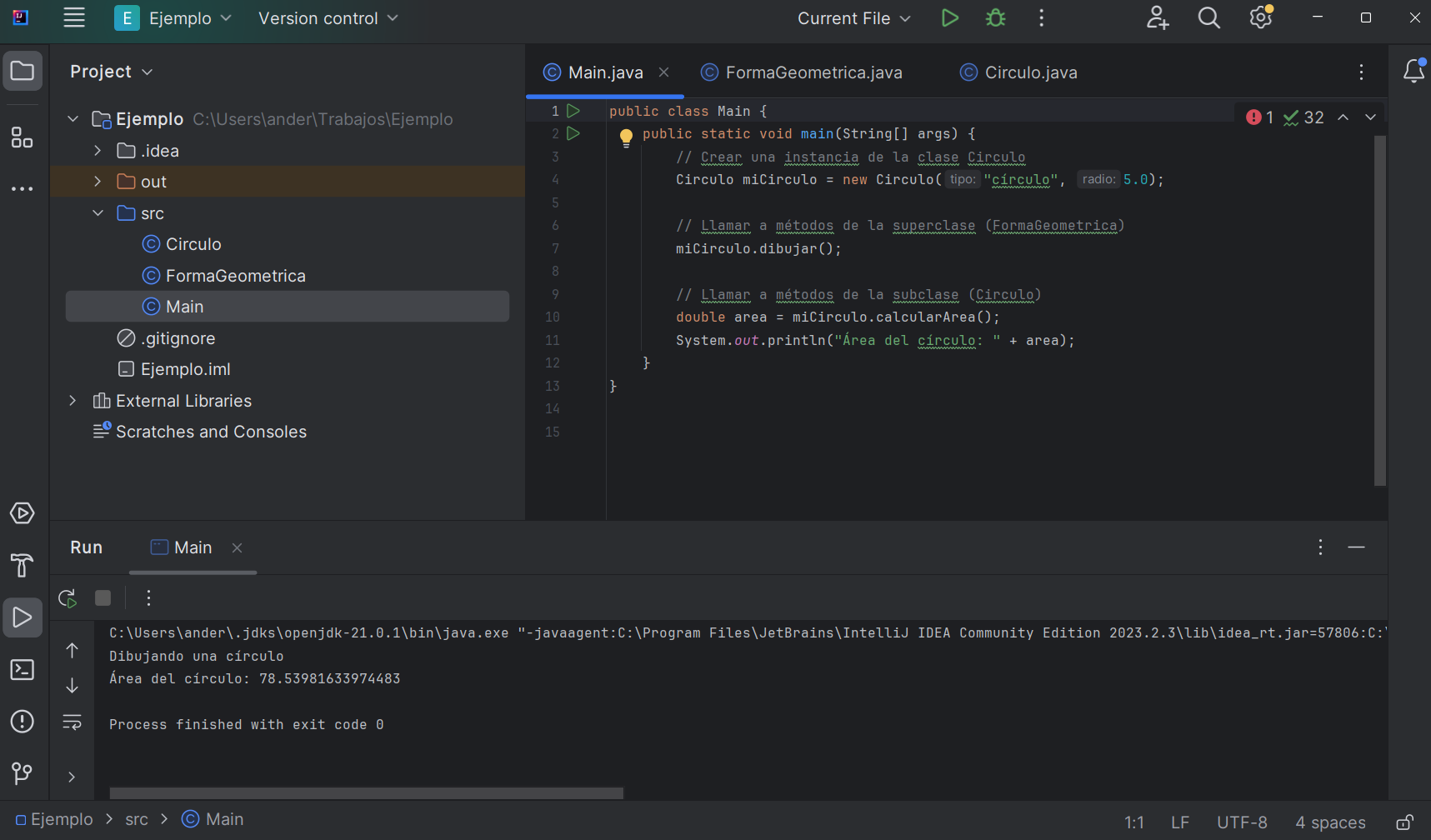
Anderson Bernal Torres

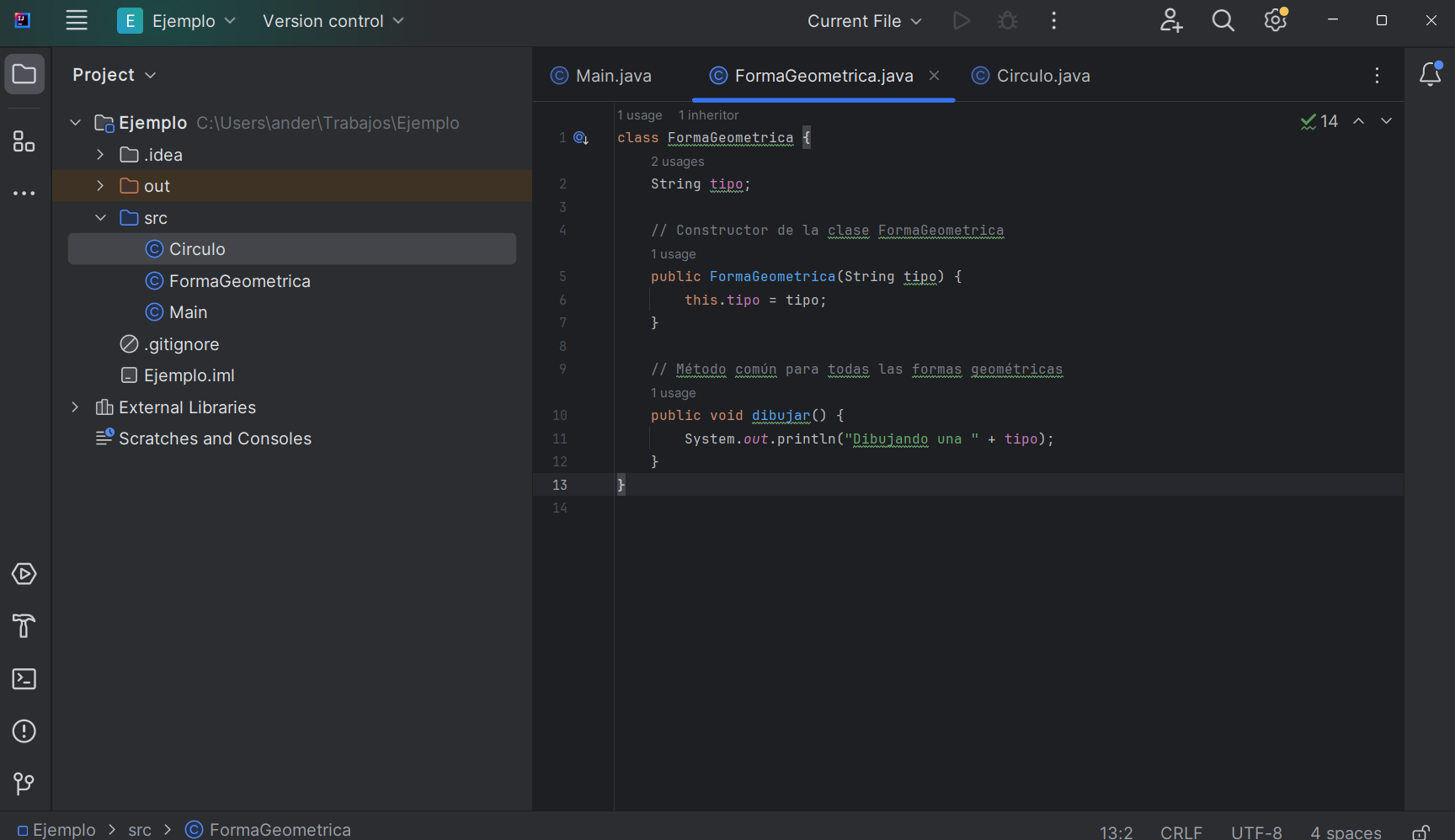
Presentado a Adrián Izquierdo

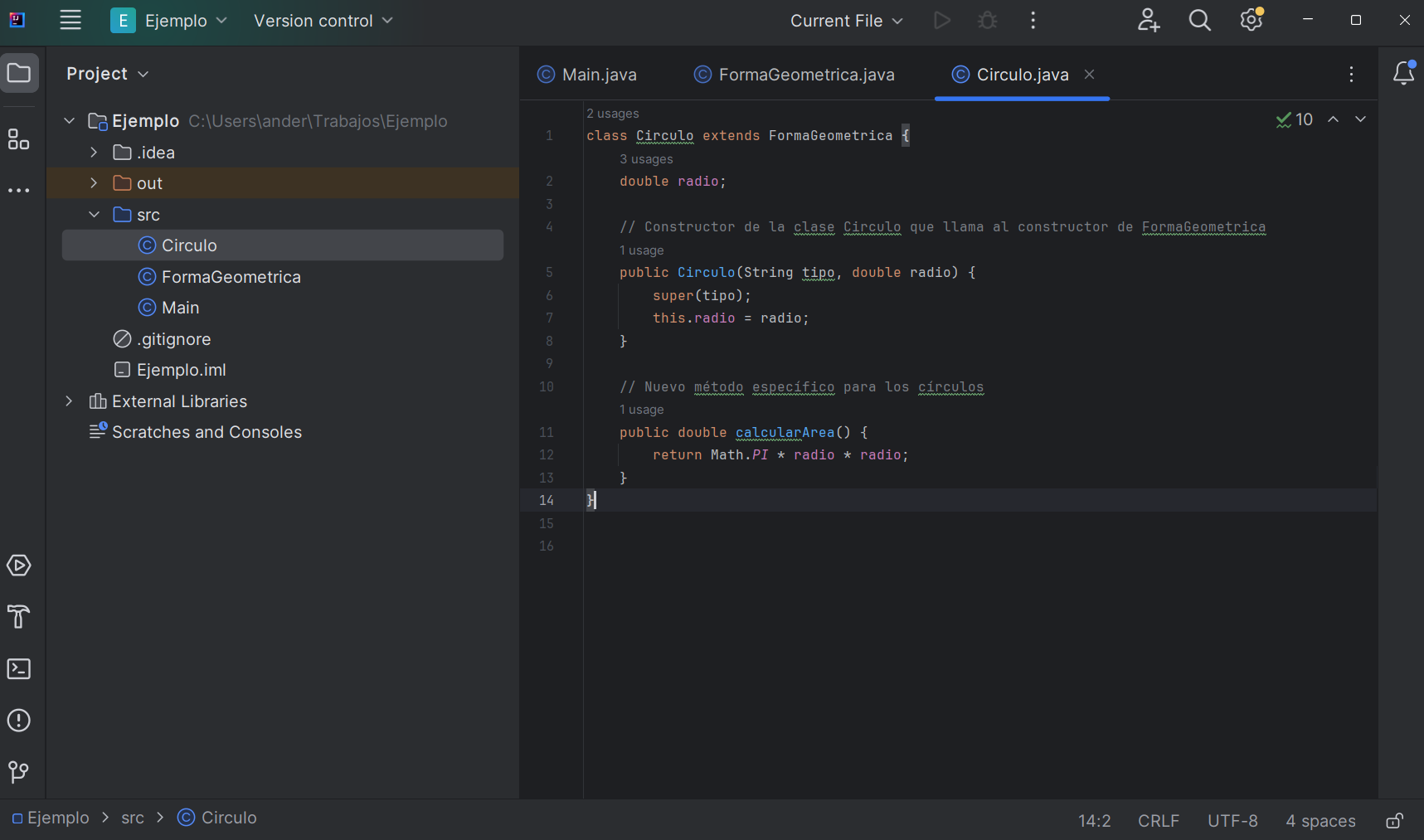
Instituto Tecnológico Del Putumayo

2023

**Herencia:** La herencia en programación orientada a objetos (POO) es un concepto fundamental que permite la creación de nuevas clases basadas en clases existentes, aprovechando y extendiendo su funcionalidad. La clase que se utiliza como base se conoce como la clase padre o superclase, mientras que la nueva clase se llama clase hija o subclase. La herencia permite la reutilización de código y la creación de jerarquías de clases la herencia se implementa mediante la palabra clave **extends**.

****

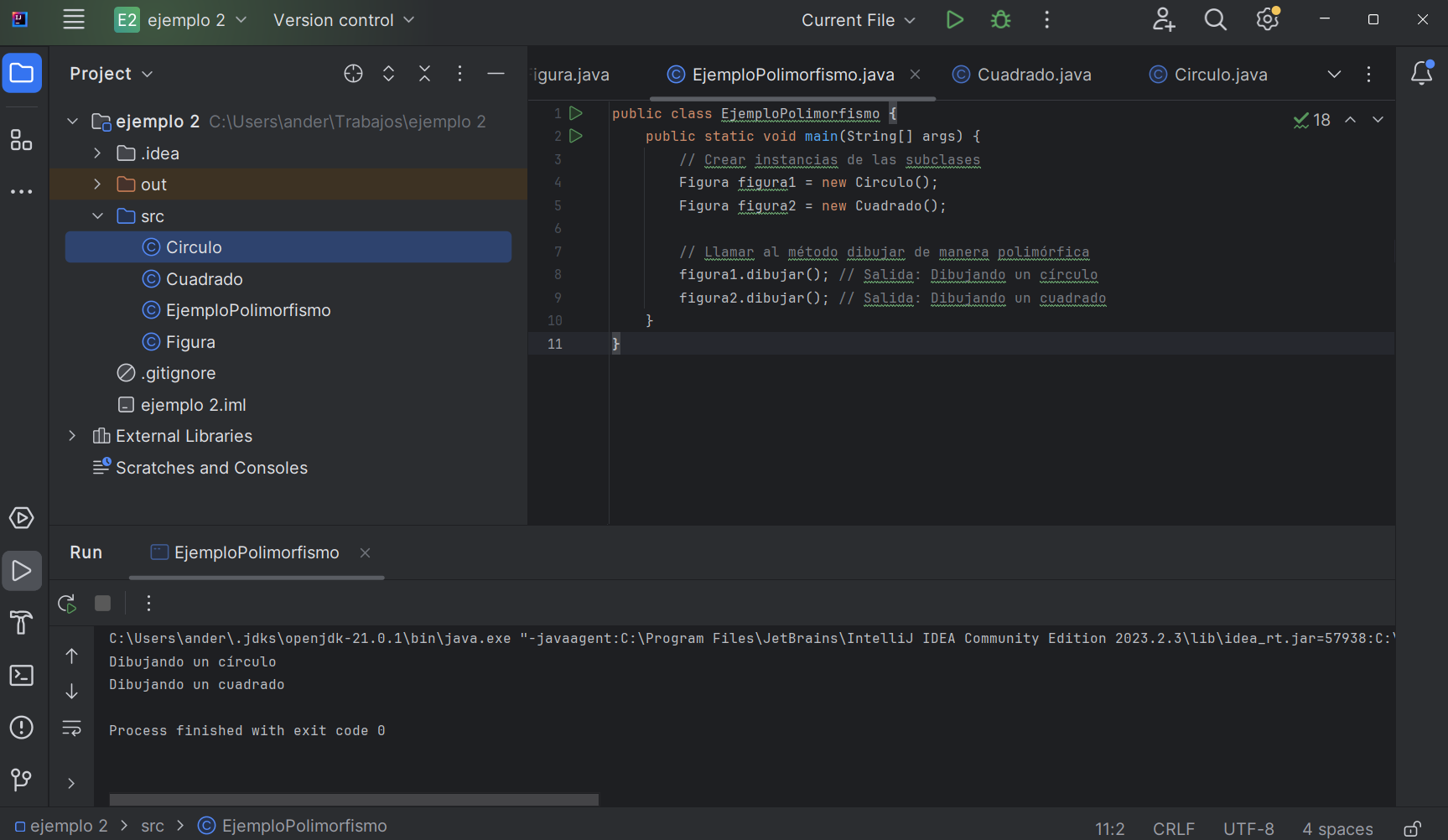
****

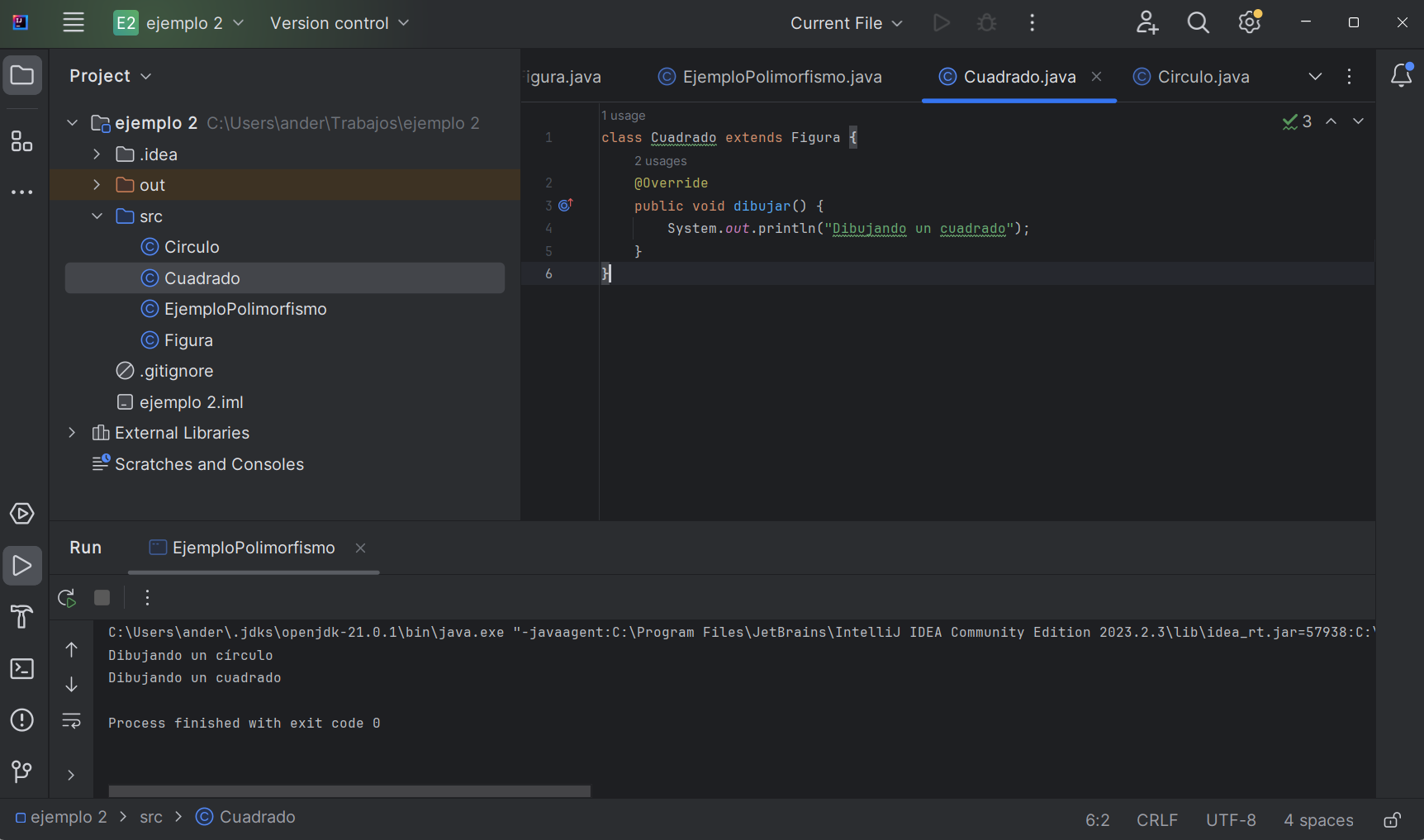
****

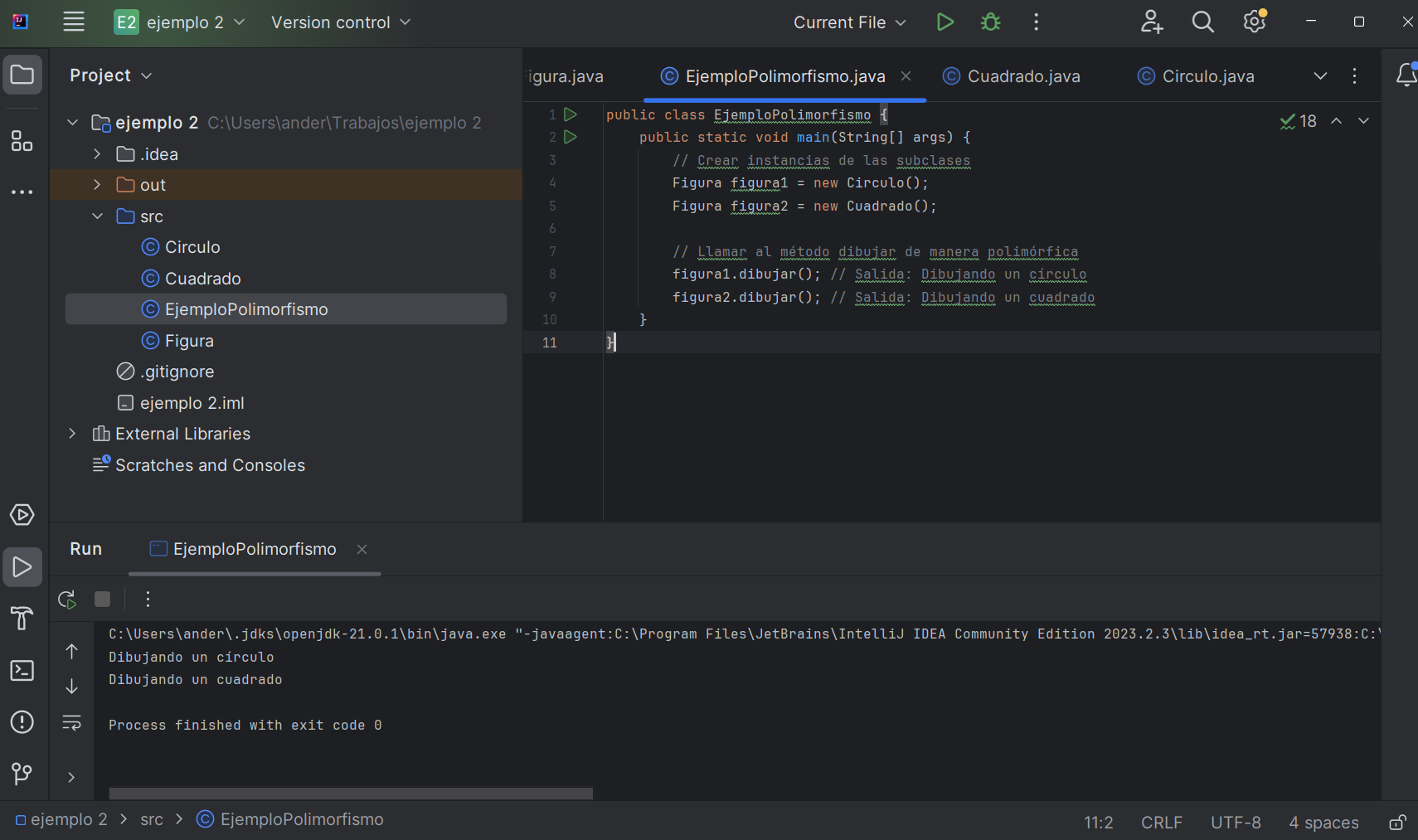
**Polimorfismo:** El polimorfismo es un concepto clave en la programación orientada a objetos (POO) que permite que un mismo método o función se comporte de manera diferente según el contexto. En otras palabras, el polimorfismo permite a un objeto tomar muchas formas. Esto se puede lograr mediante dos tipos de polimorfismo en Java: polimorfismo de compilación (también conocido como sobrecarga de métodos) y polimorfismo de ejecución (también conocido como sobrescritura de métodos).

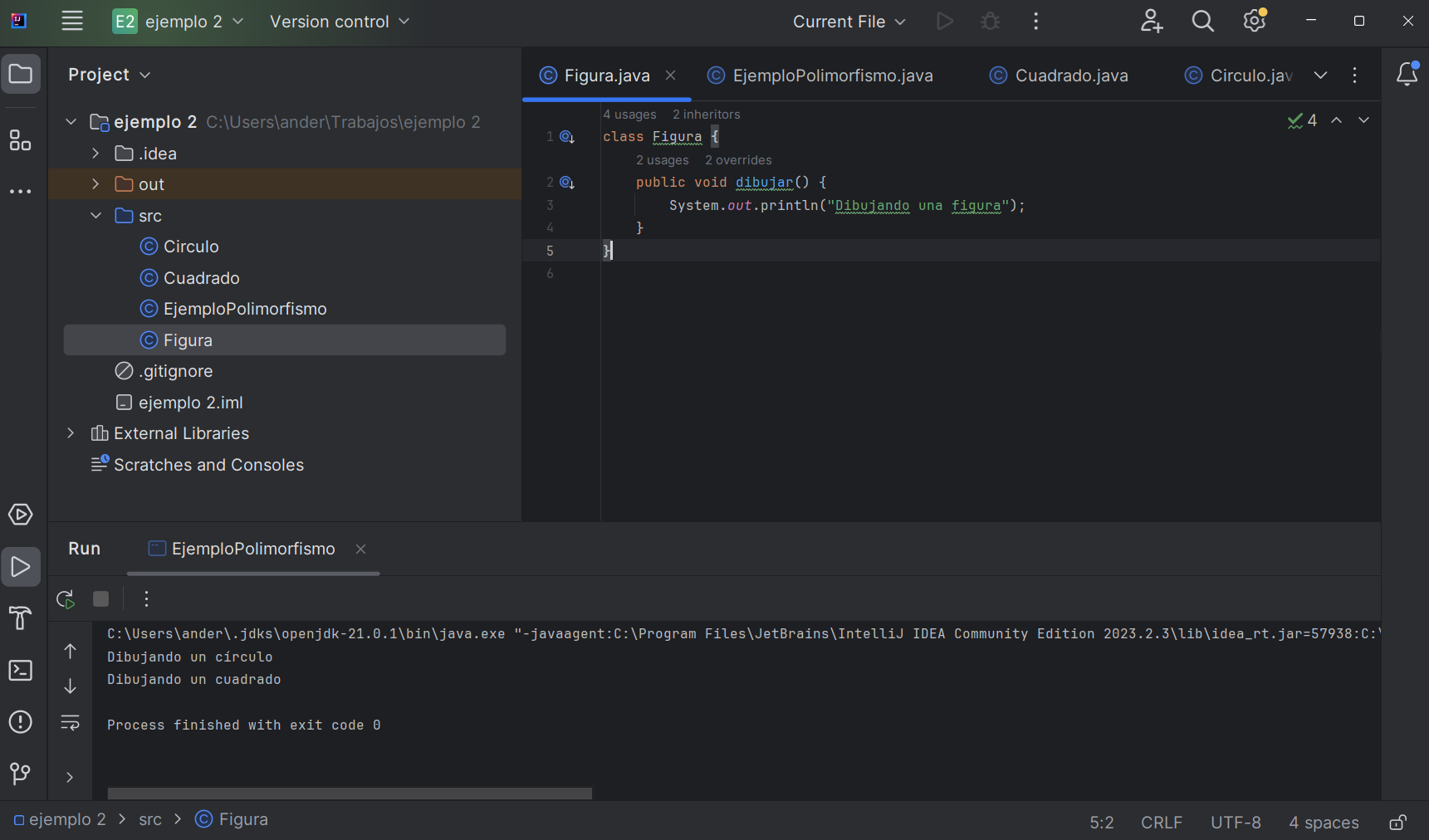
**Polimorfismo de Compilación (Sobrecarga):** Se refiere a tener múltiples métodos con el mismo nombre en la misma clase. La decisión sobre cuál método utilizar se toma en tiempo de compilación, según los tipos de parámetros.

**Polimorfismo de Ejecución (Sobrescritura):** Se refiere a la capacidad de una subclase de proporcionar una implementación específica de un método que ya está definido en su superclase.La decisión sobre cuál método utilizar se toma en tiempo de ejecución, según el tipo real del objeto.

****

****

****

****

**Herencia Simple y Herencia Múltiple:**

**Herencia Simple:**

* En la herencia simple, una clase puede heredar de otra clase única.
* En Java, una clase no puede tener más de una clase padre directa.

**Herencia Múltiple:**

* En la herencia múltiple, una clase puede heredar de varias clases.
* -Algunos lenguajes de programación admiten herencia múltiple, permitiendo que una clase tenga más de una clase padre directa.

Razones por las cuales Java no admite herencia múltiple directa:

Java fue diseñado sin soporte nativo para herencia múltiple directa de clases por varias razones:

**Ambigüedad:**

* La herencia múltiple podría dar lugar a situaciones ambiguas cuando dos clases padre tienen un método con el mismo nombre y la misma firma, lo que dificulta saber cuál implementación utilizar.

**Complejidad:**

* La herencia múltiple puede aumentar la complejidad del diseño y la resolución de conflictos, especialmente en sistemas grandes y complejos.

**Favorecer la Composición sobre la Herencia:**

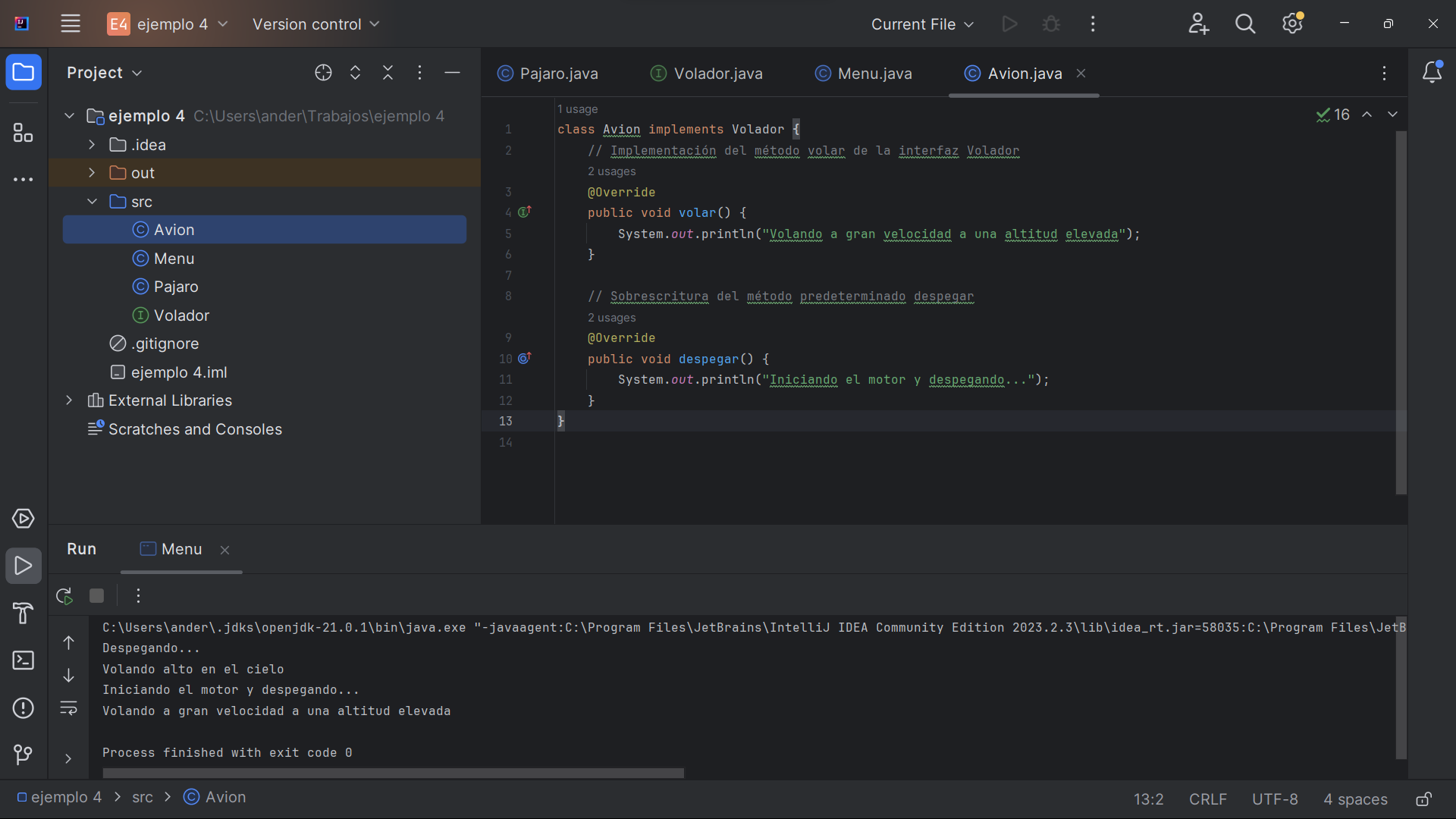
* En lugar de depender en gran medida de la herencia, Java fomenta el uso de la composición y la interfaz para lograr la reutilización de código. La composición y las interfaces proporcionan una forma más flexible y menos propensa a conflictos para compartir comportamientos entre clases.

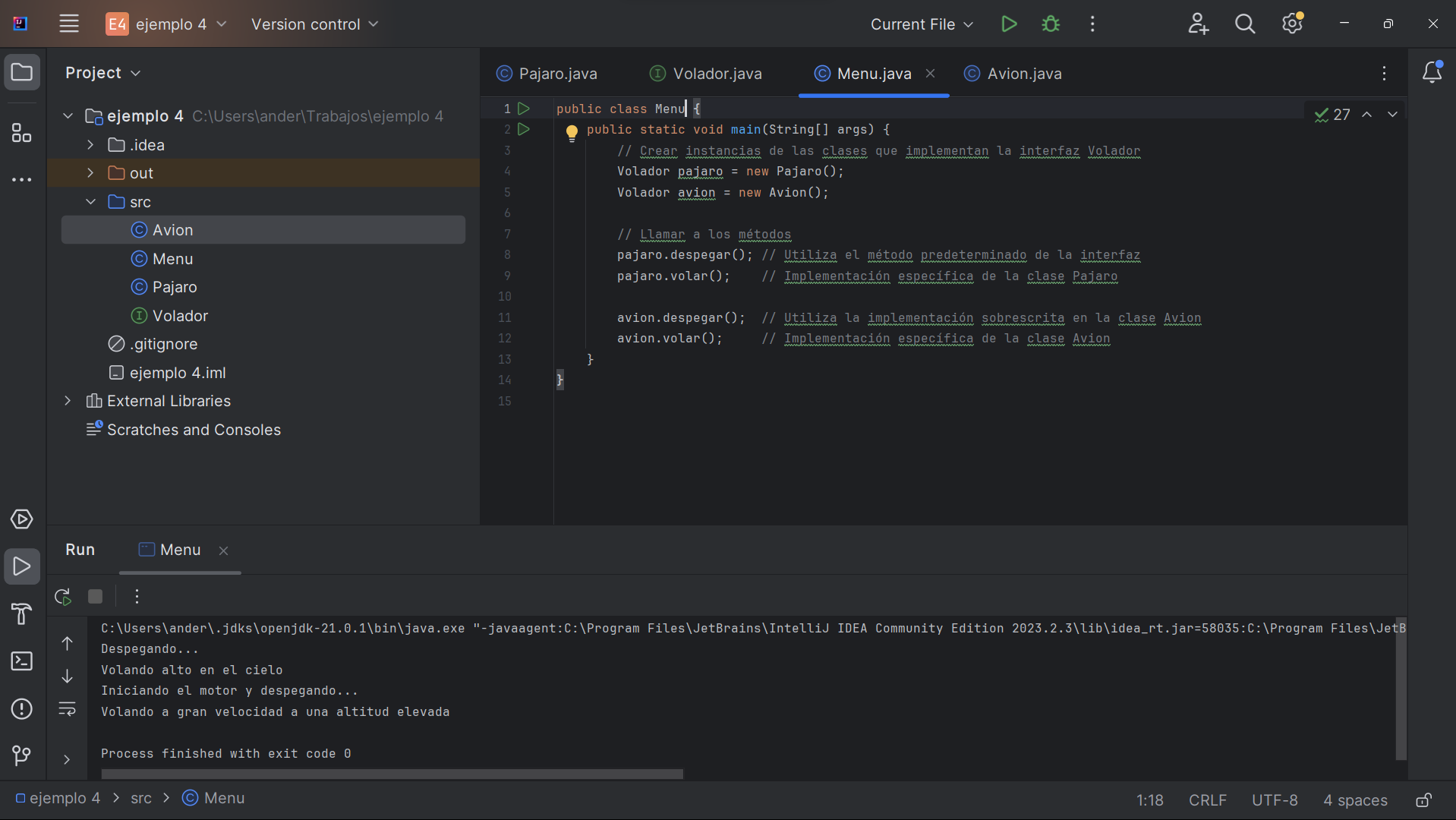
**Facilitar el Mantenimiento del Código:**

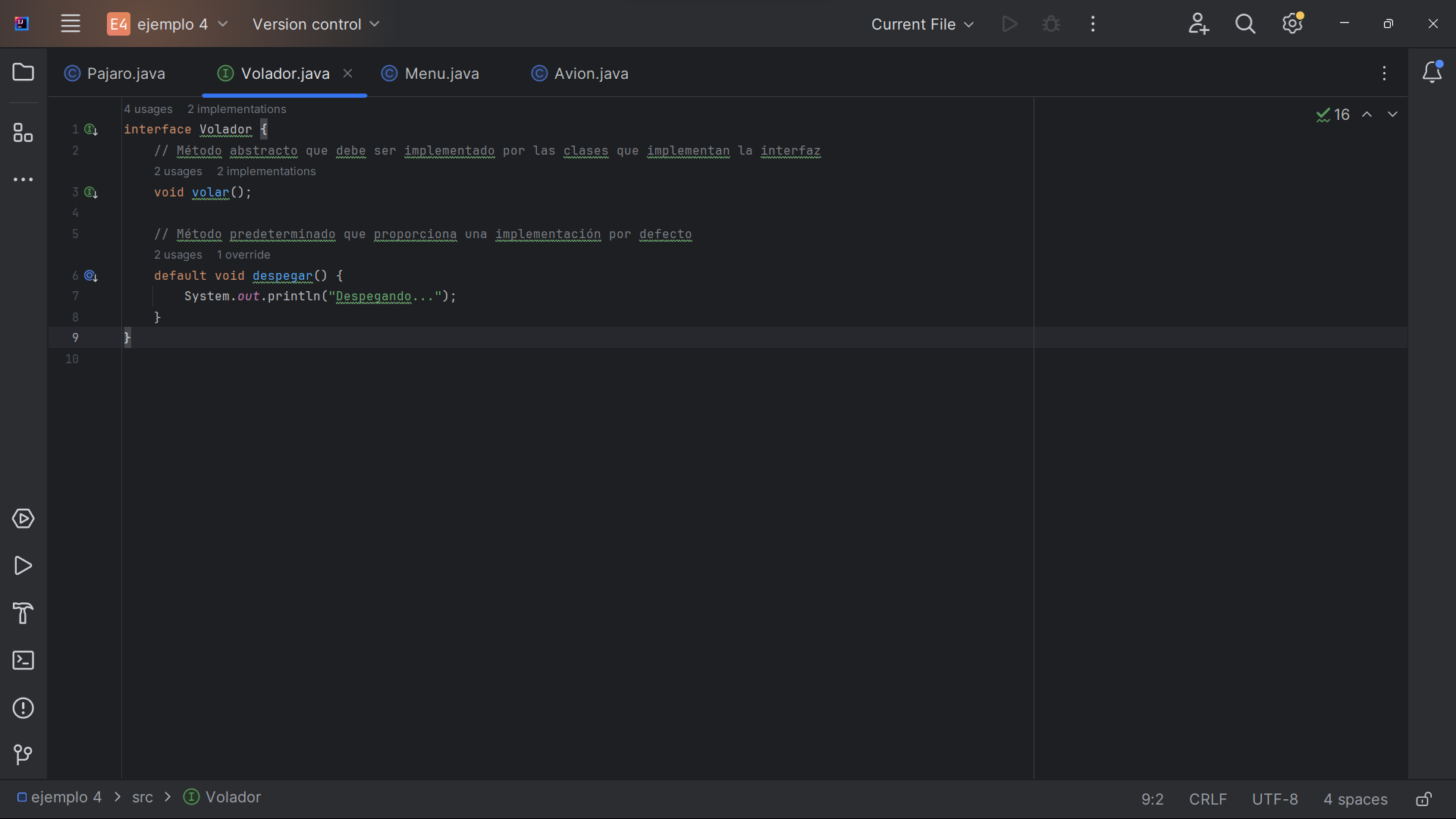
* Evitar la herencia múltiple directa hace que el código sea más fácil de entender, mantener y extender. La jerarquía de clases se vuelve más clara y menos propensa a problemas relacionados con la herencia múltiple.

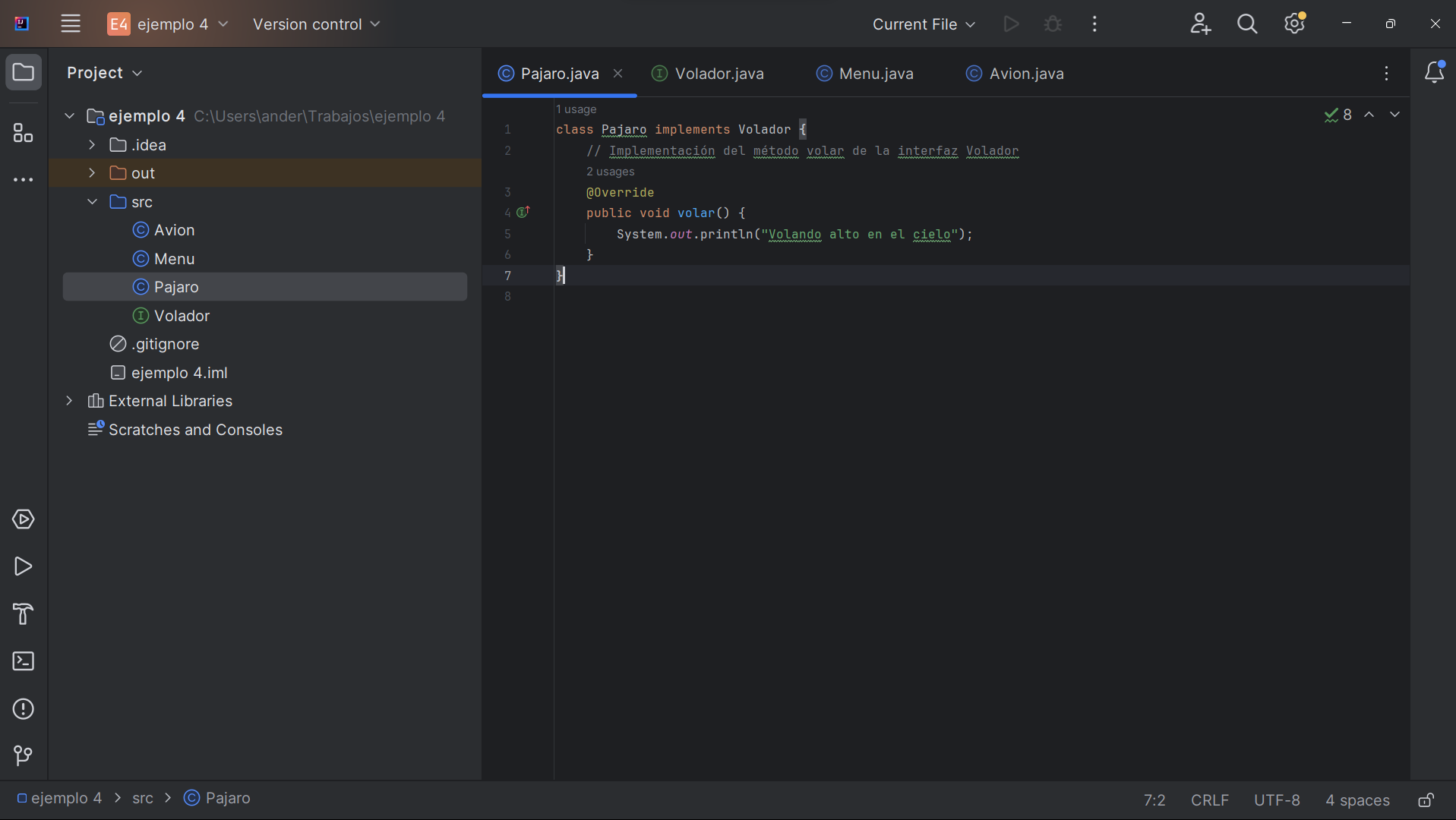
Java aborda la necesidad de compartir funcionalidades entre clases a través de interfaces, que permiten la implementación de comportamientos comunes sin heredar de múltiples clases. Además, Java 8 introdujo las interfaces con métodos predeterminados y estáticos, proporcionando una forma más flexible de compartir código sin las complicaciones asociadas con la herencia múltiple.

**Interfaz simple:** Una interfaz es una colección de métodos abstractos que pueden ser implementados por cualquier clase que lo desee. Las interfaces permiten definir contratos que las clases deben seguir, especificando qué métodos deben ser implementados por las clases que implementan la interfaz. Además de los métodos abstractos, las interfaces pueden contener métodos predeterminados y métodos estáticos a partir de Java 8.









**La sobrescritura de métodos (method overriding):** Es un concepto fundamental en la programación orientada a objetos que permite a una subclase proporcionar una implementación específica de un método que ya está definido en su superclase. La sobrescritura de métodos se utiliza para lograr polimorfismo, lo que significa que un mismo método puede comportarse de manera diferente según el tipo del objeto al que se refiere en tiempo de ejecución.

**Cómo funciona la sobrescritura de métodos:**

Una clase hija (subclase) puede proporcionar una implementación específica para un método que ya está definido en su clase padre (superclase). La firma del método en la subclase debe ser idéntica a la de la superclase, incluido el nombre del método, el tipo de retorno y los tipos de parámetros.

Cuando un objeto de la subclase se trata como un objeto de la superclase, la JVM (Máquina Virtual de Java) invocará la versión del método definida en la subclase si existe una. Esto se llama polimorfismo de ejecución.