

PARTE TEÓRICA LENGUAJE ORIENTADO A OBJETOS

STIVEN SNEIDER PATIÑO RIASCOS

INSTITUTO TECNOLOGICO DEL PUTUMAYO ITP

DESARROLLO DE SOFTWARE

MOCOA PUTUMAYO

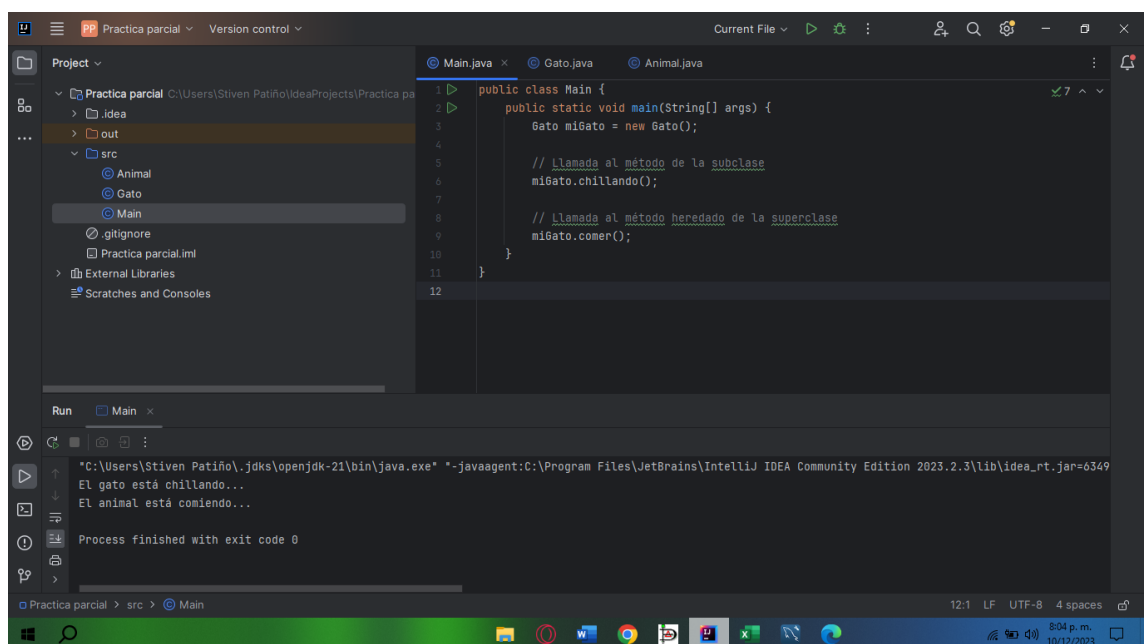
10/12/2023

Parte Teórica

1. Explique qué es la herencia en programación orientada a objetos y proporcione un ejemplo en Java.

La herencia es un principio fundamental de la programación orientada a objetos (OOP). Permite que una clase adquiera las propiedades y métodos de otra clase. En este contexto, la clase de la cual se heredan las propiedades y métodos se conoce como clase base o superclase, y la clase que hereda se conoce como subclase o clase derivada.

La herencia permite la reutilización de código y el establecimiento de relaciones jerárquicas entre clases, lo que facilita la organización y estructuración del código.



```
1 public class Main {
2     public static void main(String[] args) {
3         Gato miGato = new Gato();
4
5         // llamada al método de la subclase
6         miGato.chillando();
7
8         // llamada al método heredado de la superclase
9         miGato.comer();
10    }
11 }
12
```

Run Main x

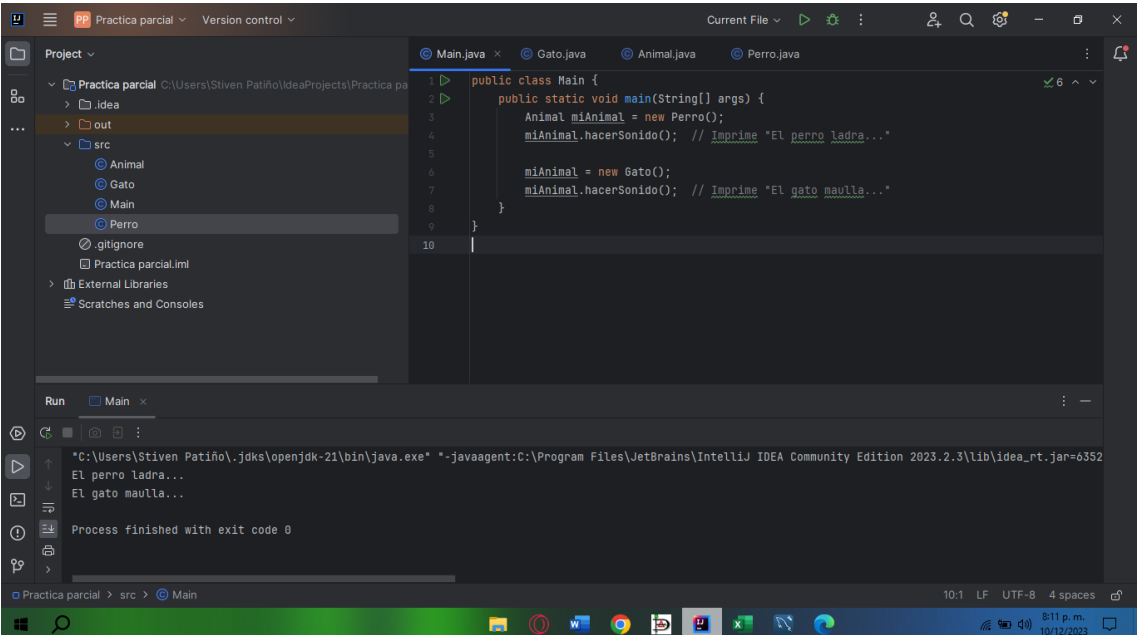
"C:\Users\Stiven Patiño\.jdk\openjdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.3\lib\idea_rt.jar=6349" El gato está chillando... El animal está comiendo... Process finished with exit code 0

En este ejemplo, la clase `Gato` hereda de la clase `Animal`. Por lo tanto, un objeto de la clase `Gato` puede llamar tanto a los métodos definidos en su propia clase (`chillando()`) como a los métodos heredados de la superclase `Animal` (`comer()`).

2. Describa el concepto de polimorfismo y proporcione un ejemplo práctico en Java.

El polimorfismo es otro concepto fundamental de la programación orientada a objetos (OOP). El término “polimorfismo” proviene del griego y significa “muchas formas”. En el contexto de la OOP, se refiere a la capacidad de un objeto para tomar muchas formas.

El polimorfismo permite que una referencia a una clase base represente a una subclase. En otras palabras, una clase puede referirse a cualquier objeto de su propia clase o de cualquier subclase. Esto permite que los métodos que operan en referencias a la clase base puedan afectar a las subclases.



The screenshot shows an IDE window with a project named "Practica parcial". The project structure on the left includes a "src" folder with files "Animal", "Gato", "Perro", and "Main". The "Main.java" file is open in the editor, showing the following code:

```
1 public class Main {
2     public static void main(String[] args) {
3         Animal miAnimal = new Perro();
4         miAnimal.hacerSonido(); // Imprime "El perro ladra..."
5
6         miAnimal = new Gato();
7         miAnimal.hacerSonido(); // Imprime "El gato maulla..."
8     }
9 }
10
```

Below the editor, the "Run" tab shows the execution output:

```
"C:\Users\Stiven Patiño\.jdk\openjdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2.3\lib\idea_rt.jar=6352
El perro ladra...
El gato maulla...
Process finished with exit code 0
```

En este ejemplo, `Perro` y `Gato` son subclases de `Animal`. Ambas subclases tienen un método `hacerSonido()`, que es una redefinición del método `hacerSonido()` en la superclase `Animal`. Esto es un ejemplo de **sobrescritura de métodos**, que es una forma de polimorfismo. En el método `main()`, la referencia `miAnimal` es de tipo `Animal`, pero su objeto actual puede ser `Perro` o `Gato`. Esto es lo que permite el polimorfismo. Dependiendo del objeto actual de `miAnimal`, se llama al método `hacerSonido()` correspondiente. Esto se conoce como **enlace dinámico** o **enlace tardío**.

**3. ¿Cuál es la diferencia entre herencia simple y herencia múltiple en Java?
¿Por qué Java no admite herencia múltiple directa de clases?**

La herencia simple es cuando una clase hereda de una sola superclase. En otras palabras, una subclase tiene una sola superclase. Java admite la herencia simple.

Por otro lado, la herencia múltiple es cuando una clase puede heredar comportamientos y características de más de una superclase. Esto significa que una subclase puede tener más de una superclase.

Java no admite la herencia múltiple directa de clases debido a problemas como la ambigüedad y la complejidad que puede surgir. Por ejemplo, si una clase hereda de dos superclases y ambas superclases tienen un método con la misma firma, entonces sería ambiguo determinar cuál método heredaría la subclase.

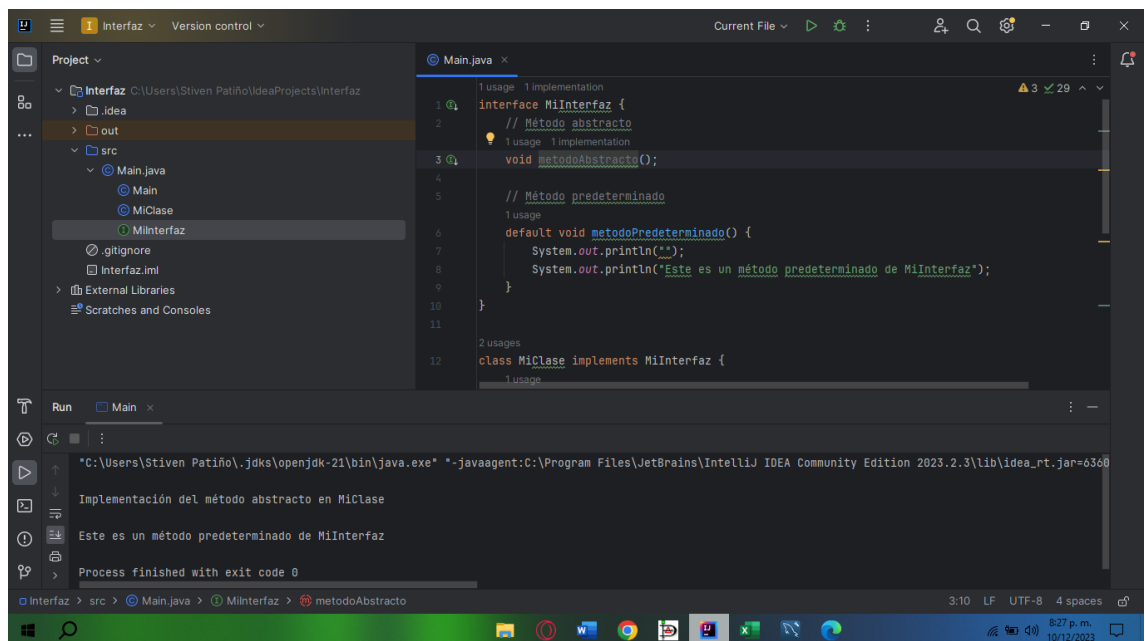
Para superar esta limitación y aun así aprovechar los beneficios de la herencia múltiple, Java proporciona un mecanismo llamado interfaces. Una interfaz es similar a una clase, pero solo contiene métodos abstractos y constantes. Una clase en Java puede implementar cualquier número de interfaces, lo que permite un tipo de herencia múltiple.

4. Explique lo que es una interfaz en Java. Proporcione un ejemplo de una interface con un método default.

Una interfaz en Java es un tipo de referencia similar a una clase y se puede definir utilizando la palabra clave `interface`. Puede contener solo métodos abstractos (métodos sin cuerpo), constantes estáticas (variables finales) y métodos predeterminados.

Las interfaces proporcionan una forma de lograr la abstracción y también pueden ser utilizadas para lograr la herencia múltiple en Java. Una clase puede implementar cualquier número de interfaces.

Un método predeterminado (o método default) es un método con una implementación predeterminada que se introdujo en Java 8. Permite agregar nuevos métodos a las interfaces sin afectar las clases que implementan estas interfaces. Esto ayuda a evitar problemas de compatibilidad cuando se necesitan mejoras en las interfaces.



En este ejemplo, `MiClase` implementa la interfaz `MiInterfaz`. Por lo tanto, debe proporcionar una implementación para el método abstracto `metodoAbstracto()`.

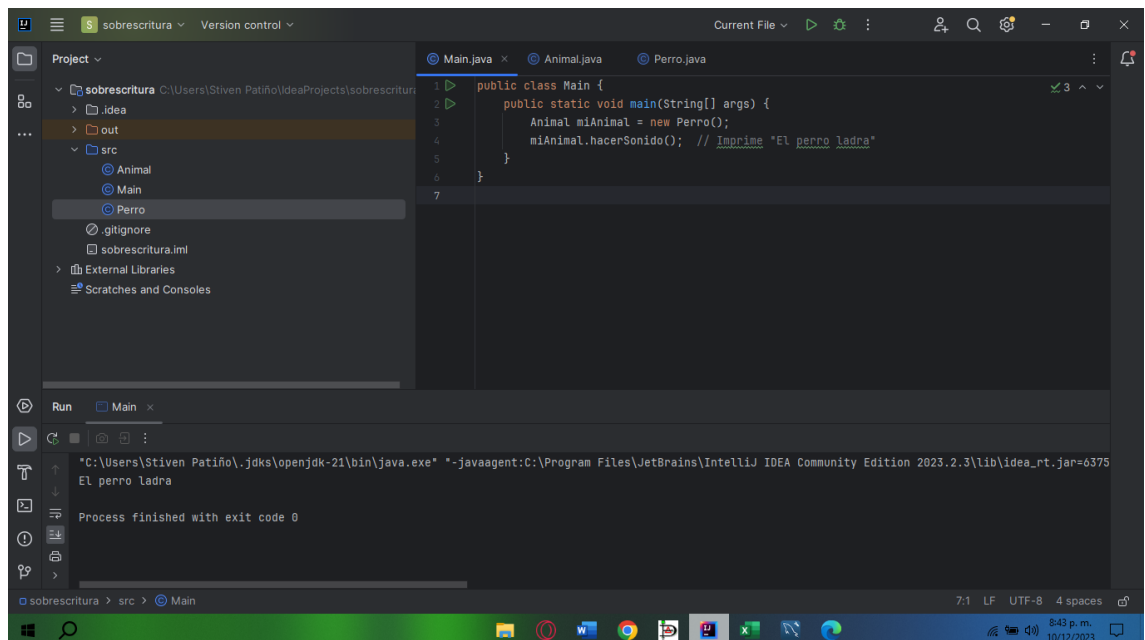
Sin embargo, no necesita proporcionar una implementación para el método predeterminado `metodoPredeterminado()`, ya que ya tiene una implementación predeterminada en la interfaz. Si `MiClase` quisiera proporcionar su propia implementación para `metodoPredeterminado()`, podría hacerlo sobrescribiendo el método. Si no lo hace, se utilizará la implementación predeterminada de la interfaz.

5. ¿Qué es la sobrescritura de métodos (method overriding) en Java y cómo se utiliza para lograr polimorfismo?

La sobrescritura de métodos, también conocida como method overriding, es un concepto en Java donde una subclase proporciona una implementación específica de un método que ya está proporcionado por su superclase. Se utiliza para lograr el polimorfismo en tiempo de ejecución.

Para sobrescribir un método, la subclase debe proporcionar un método con la misma firma que en la superclase, es decir, el mismo nombre de método y los mismos parámetros.

La sobrescritura de métodos se utiliza para proporcionar una implementación específica de un método en la subclase que es más adecuada para su tipo de objeto, en lugar de utilizar la implementación proporcionada en la superclase.



En este ejemplo, la clase `Perro` sobrescribe el método `hacerSonido()` de la clase `Animal`. Cuando se llama al método `hacerSonido()` en un objeto de tipo `Animal` que actualmente es un `Perro`, se llama al método `hacerSonido()` de la clase `Perro`. Esto es polimorfismo en tiempo de ejecución.