**PARCIAL FINAL DE OBJETOS – LENGUAJE ORIENTADO A OBJETOS**

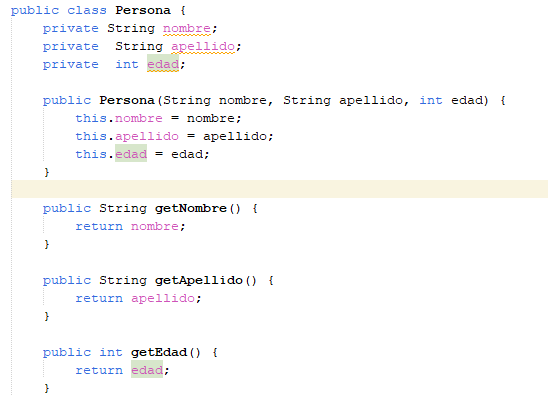
**CUARTO SEMESTRE**

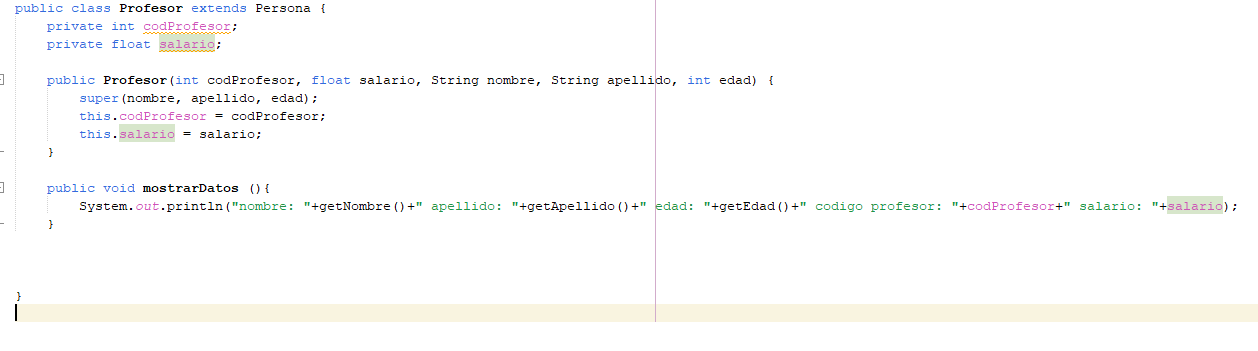
**JAMES ALEXIS MENESES**

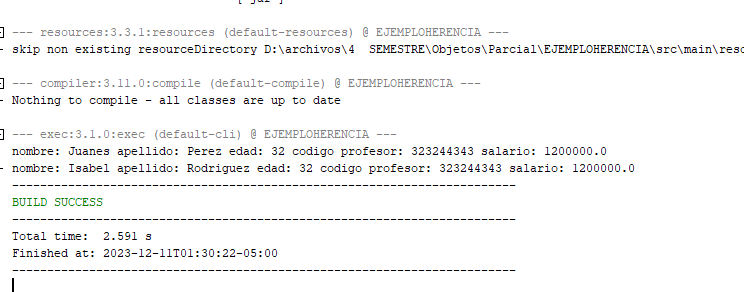
1. **Explique qué es la herencia en programación orientada a objetos y proporcione un ejemplo en Java.**

La herencia es un mecanismo que permite definir una nueva clase a partir de una clase existente, aprovechando sus atributos y métodos, y añadiendo o modificando los propios. La herencia facilita la reutilización del código, la organización de las clases en una jerarquía y el polimorfismo.

Un ejemplo de herencia en Java es el siguiente:



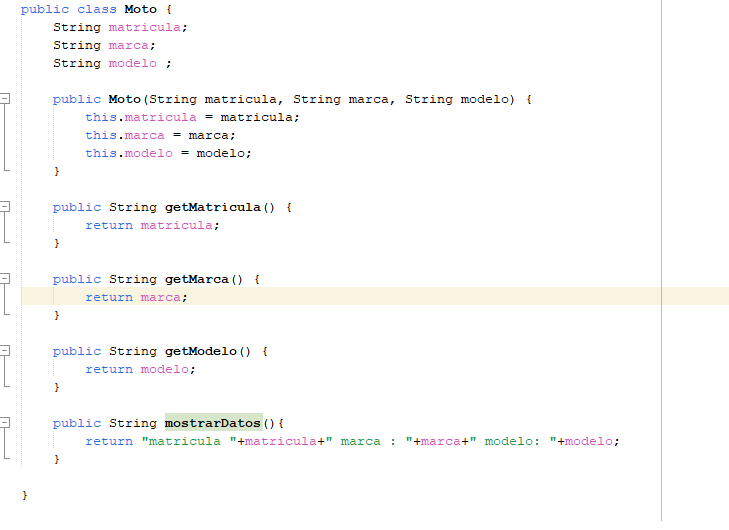


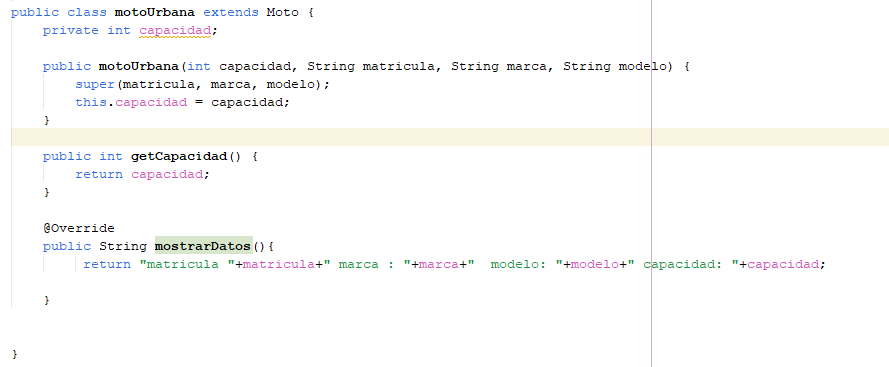
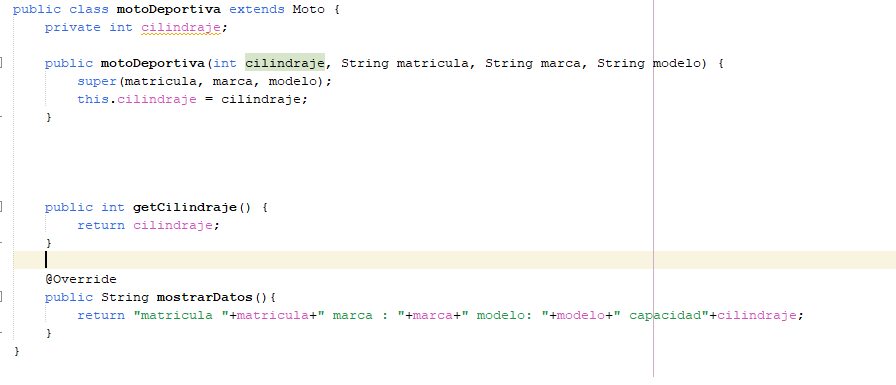


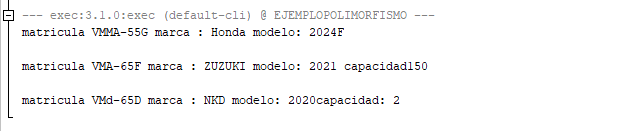
En el ejemplo anterior, la clase Profesor hereda los atributos y métodos de la clase Persona, y además define sus propios atributos (codProfesor y salario) y su propio método (mostrar Datos). La clase Profesor es una versión especializada de la clase Persona, que puede tener un comportamiento diferente o adicional al de la superclase.

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

En la programación orientada a objetos, el polimorfismo se refiere a la habilidad de una clase para tener diferentes implementaciones de un mismo método, o de una interfaz para ser implementada por diferentes clases.







En el ejemplo anterior se trata de un polimorfismo de subtipos, que se basa en la herencia. La clase Moto es una superclase que define los atributos y métodos comunes a todos los tipos de motos, como matricula, marca, modelo y mostrar Datos. Las clases moto Deportiva y moto Urbana son subclases que heredan de la clase Moto, y además añaden sus propios atributos y métodos, como cilindraje, capacidad y sobrescriben el método mostrar Datos.

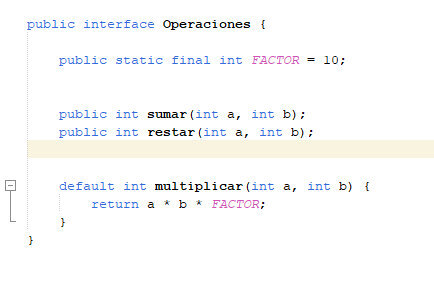
El polimorfismo se manifiesta cuando se crea un arreglo de tipo Moto, que puede contener objetos de las clases Moto, moto Deportiva y moto Urbana. Aunque todos los objetos son de tipo Moto, cada uno tiene su propia forma y comportamiento, y al invocar el método mostrar Datos, se ejecuta la versión correspondiente a cada subclase. Esto permite tratar a los objetos de forma uniforme, sin importar su tipo específico, y aprovechar la reutilización y la flexibilidad del código.

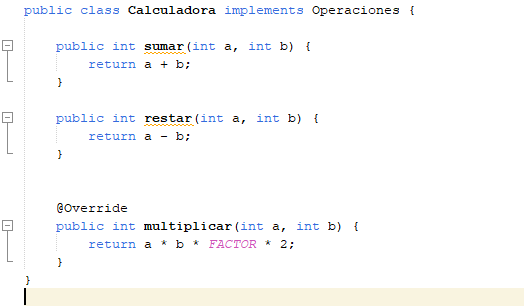
1. **¿Cuál es la diferencia entre herencia simple y herencia múltiple en Java? ¿Por qué Java no admite herencia múltiple directa de clases?**

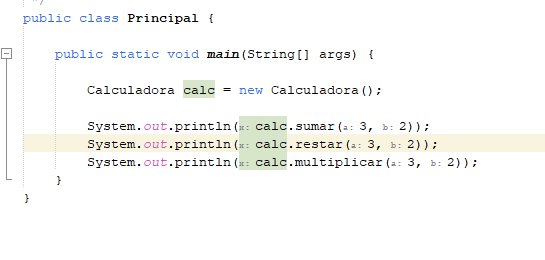
* La diferencia entre herencia simple y herencia múltiple en Java es que la herencia simple permite que una clase hija herede de una sola clase padre, mientras que la herencia múltiple permite que una clase hija herede de dos o más clases padre.
* Java no admite herencia múltiple directa de clases porque puede causar el problema del diamante, que ocurre cuando una clase hija hereda de dos clases padre que tienen un método con el mismo nombre y firma. En ese caso, no está claro cuál método debe invocar la clase hija, lo que puede provocar errores o inconsistencias. Para evitar este problema, Java solo permite herencia simple de clases, pero admite herencia múltiple de interfaces.

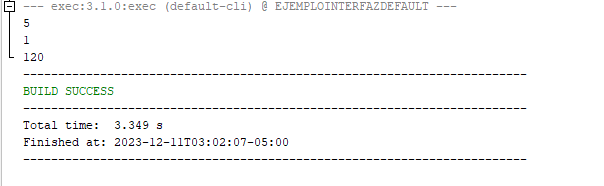
1. **Explique lo que es una interfaz en Java. Proporcione un ejemplo de una interface con un método default**

Una interfaz en Java es un tipo abstracto que contiene una colección de métodos y variables constantes. Una interfaz se usa para definir un contrato o una especificación que las clases que la implementan deben cumplir. Una interfaz también permite lograr el polimorfismo y la herencia múltiple en Java. Un ejemplo de una interfaz con un método default es el siguiente:









En el ejemplo anterior, la interfaz Operaciones define dos métodos abstractos (sumar y restar) y un método default (multiplicar). La clase Calculadora implementa la interfaz Operaciones y proporciona el cuerpo de los métodos abstractos. La clase Calculadora también sobrescribe el método default multiplicar, pero podría usar la implementación predeterminada de la interfaz si no lo hiciera. El método default permite que una interfaz tenga una implementación por defecto de un método, sin afectar a las clases que la implementan.

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

El sobreescritura de métodos es la capacidad de una clase hija de redefinir un método que hereda de una clase padre, cambiando su comportamiento o implementación. El sobreescritura de métodos se utiliza para lograr polimorfismo, que es la propiedad de un objeto de tomar diferentes formas y comportarse de diferentes maneras según el contexto.