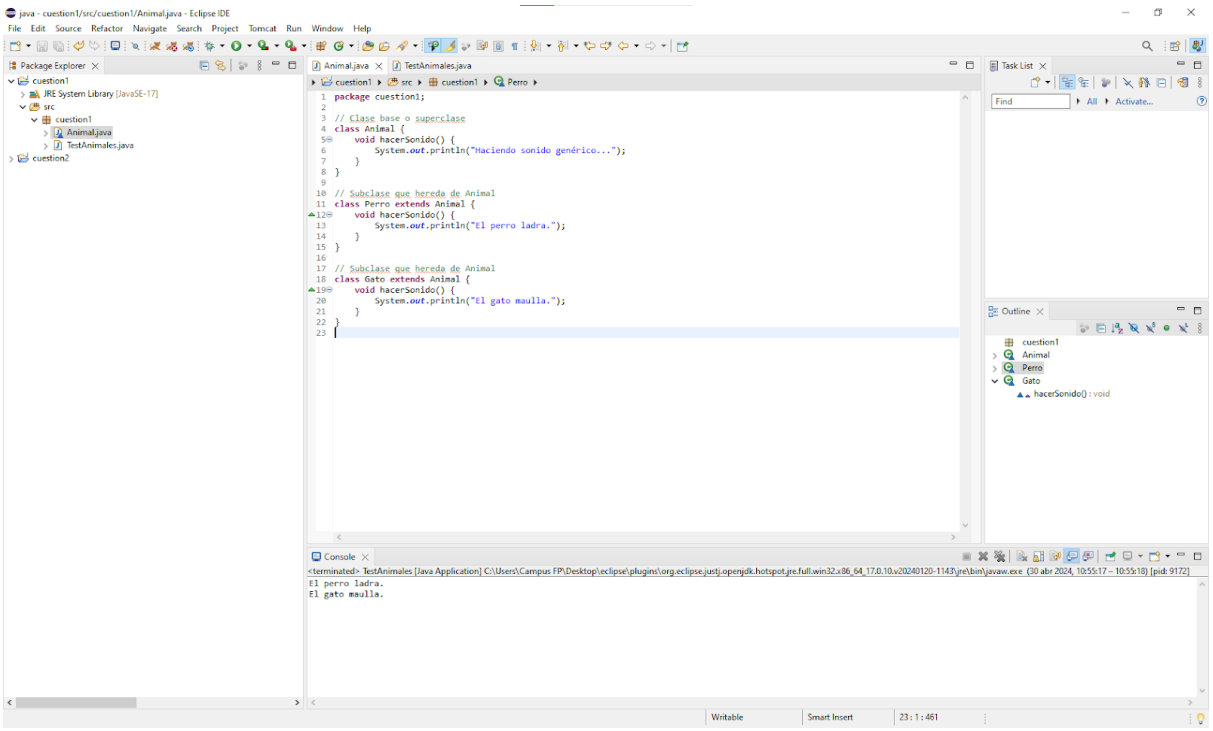
CUESTIÓN 1. POO en Java y colecciones

▪ Diseña un ejemplo de clases en donde puedas demostrar la herencia en Java. Explica si es posible realizar herencia múltiple en Java y por qué.
▪ Siguiendo el ejercicio anterior, propón un ejemplo para diferenciar sobrecarga y sobreescritura. El ejemplo funcionará en consola.
▪ En Java existen varias opciones para almacenar datos en colecciones. Explica con un ejemplo qué diferencia hay entre colecciones de tipo lista, pila, cola y vector.

RESPUESTA CUESTION 1: PDF con las respuestas planteadas, archivos correspondientes y en Github

```
package cuestion1;
// Clase base o superclase
class Animal {
  void hacerSonido() {
     System.out.println("Haciendo sonido genérico...");
  }
}
// Subclase que hereda de Animal
class Perro extends Animal {
  void hacerSonido() {
     System.out.println("El perro ladra.");
  }
}
// Subclase que hereda de Animal
class Gato extends Animal {
  void hacerSonido() {
     System.out.println("El gato maulla.");
  }
}
package cuestion1;
        public class TestAnimales {
           public static void main(String[] args) {
              Animal animal1 = new Perro();
              Animal animal2 = new Gato();
              animal1.hacerSonido(); // Salida: El perro ladra.
              animal2.hacerSonido(); // Salida: El gato maulla.
           }
        }
```



En este ejemplo, las clases Perro y Gato heredan de la clase base Animal. Cada una de ellas sobrescribe el método hacerSonido() de la superclase para proporcionar su propia implementación.

Herencia Múltiple en Java:

Java no permite la herencia múltiple de clases, lo que significa que una clase no puede extender más de una clase directamente. Esto se debe a que la herencia múltiple puede causar conflictos de nombres y ambigüedades en el código, lo que hace que el diseño sea más complejo y difícil de entender y mantener. Sin embargo, Java permite la implementación de múltiples interfaces, lo que ofrece una forma de lograr funcionalidades similares a la herencia múltiple a través de la implementación de interfaces.

Diferencia entre sobrecarga y sobreescritura:

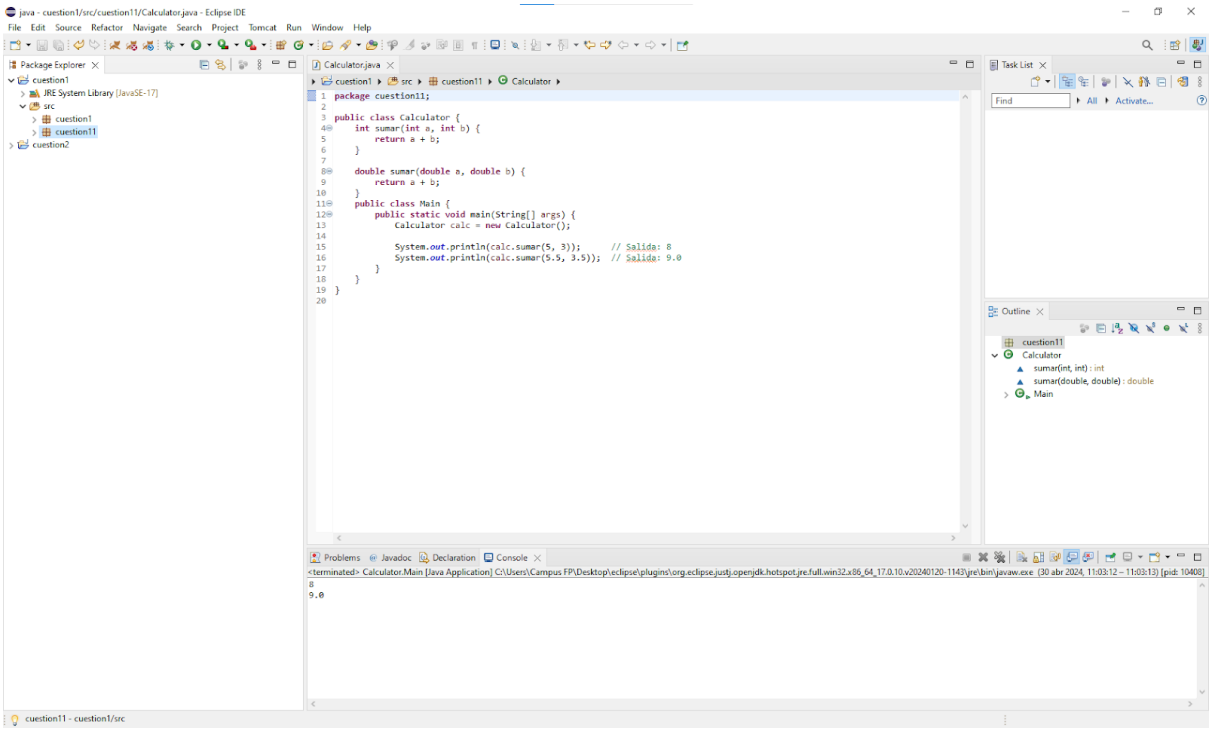
La sobrecarga ocurre cuando se tienen múltiples métodos en una clase con el mismo nombre pero con diferentes firmas (número o tipo de parámetros). La sobreescritura, por otro lado, ocurre cuando una subclase proporciona una implementación específica para un método que ya está definido en su superclase.

Un ejemplo que ilustra la diferencia entre sobrecarga y sobreescritura:

```
package cuestion11;
public class Calculator {
```

```
    int sumar(int a, int b) {
        return a + b;
    }
    double sumar(double a, double b) {
        return a + b;
    }
    public class Main {
        public static void main(String[] args) {
            Calculator calc = new Calculator();

            System.out.println(calc.sumar(5, 3));      // Salida: 8
            System.out.println(calc.sumar(5.5, 3.5));  // Salida: 9.0
        }
    }
}
```



La clase Calculadora tiene dos métodos llamados sumar(), pero con diferentes tipos de parámetros. **Esto es sobrecarga**. La decisión de cuál método usar se toma en tiempo de compilación, basándose en los tipos de los argumentos.
La sobreescritura, por otro lado, los métodos hacerSonido() en las subclases Perro y Gato proporcionan una implementación específica para ese método, reemplazando la implementación de la superclase Animal. La decisión de qué método se ejecuta se toma en tiempo de ejecución, basándose en el tipo de objeto real al que se hace referencia.