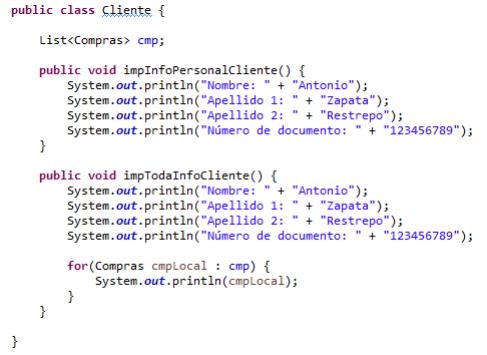
**Buenas Prácticas de Programación:**

1. Se tienen la siguiente clase cliente, la cual tiene dos métodos, uno para imprimir la información personal y otro para imprimir toda la información del cliente.



Realice 5 modificaciones que apliquen buena práctica de programación para mejorarla. Cada modificación es equivalente a un punto del examen y debe estar sustentada, de la justificación de la modificación depende si el punto es válido o no.

**RTA/**

* Se renombra la variable cmp, cmpLocal y el método impInfoPersonalCliente siguiendo las pautas de buenas prácticas, nombres con sentido, que den información de su propósito y que sean pronunciables.
* Se evita duplicidad del código, por tanto se hace el llamado al método que hace lo mismo que el código que estaba antes.

**Nota:** Las observaciones anteriores se pueden visualizar como comentario en el código para saber en cuales líneas se hizo el cambio. Es de aclarar que **el código no debe quedar con los comentarios**, que solo se hicieron para dar una explicación a este punto del taller.

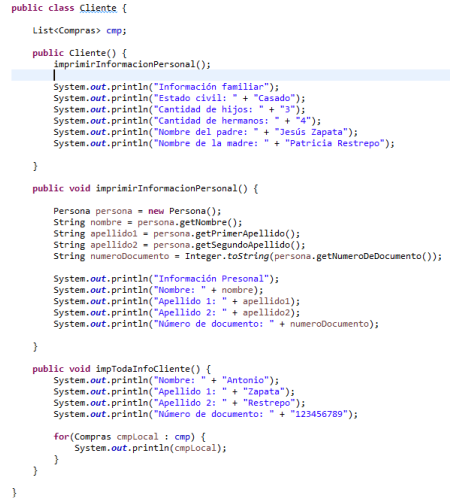
public class Cliente {  
 List<Compras> listaCompras; /\*Se renombra la variable siguiendo las pautas de buenas prácticas, nombres con sentido, que den información de su propósito y que sean pronunciables.\*/  
  
 Public void imprimirInformacionPersonal (){ /\*Se hace cambio de nombre, siguiendo las pautas de buenas prácticas usando Camel Case y un nombre diciente y evitando nomenclaturas\*/  
 System.*out*.println('Nombre: '+ 'Antonio');  
 System.*out*.println('Apellido 1: '+ 'Zapata');  
 System.*out*.println('Apellido 2: '+ 'Restrepo');  
 System.*out*.println('Numero de documento: '+ '123456789');  
 }  
  
 Public void imprimirInformacionPersonalCompleta(){ /\*Se hace cambio de nombre, siguiendo las pautas de buenas prácticas usando Camel Case y un nombre diciente y evitando nomenclaturas\*/

imprimirInformacionPersonal(); /\*Se evita duplicidad del código, por tanto se hace el llamado al metodo que hace lo mismo que el codigo que estaba antes aqui \*/

for(Compras compraLocal : listaCompras) { /\*Se hace cambio de nombre a la variable, siguiendo las pautas de buenas practicas usando Camel Case y un nombre diciente y evitando nomenclaturas\*/

system.out.printl(compraLocal);  
 }  
 }  
  
}

1. En la clase que se muestra a continuación, aplique tres métodos de refactorización para mejorarla. Cada modificación es equivalente a un punto del exámen y debe estar sustentada, de la justificación de la modificación depende si el punto es válido o no.



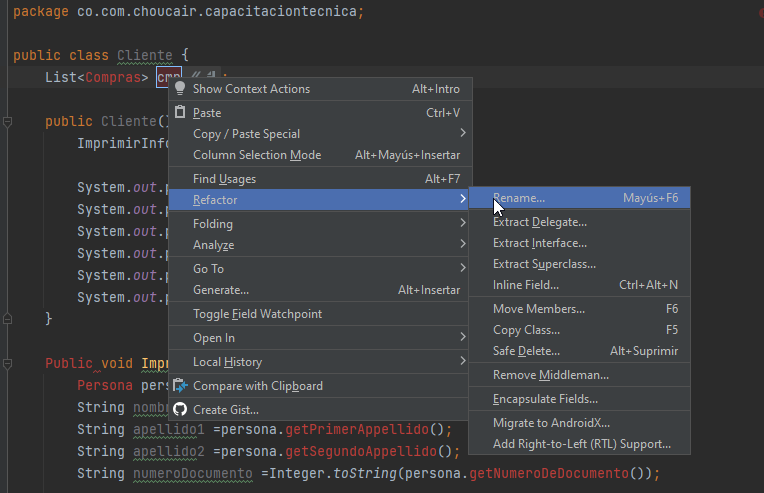
**RESPUESTA**

**REFACTORIZACION:**

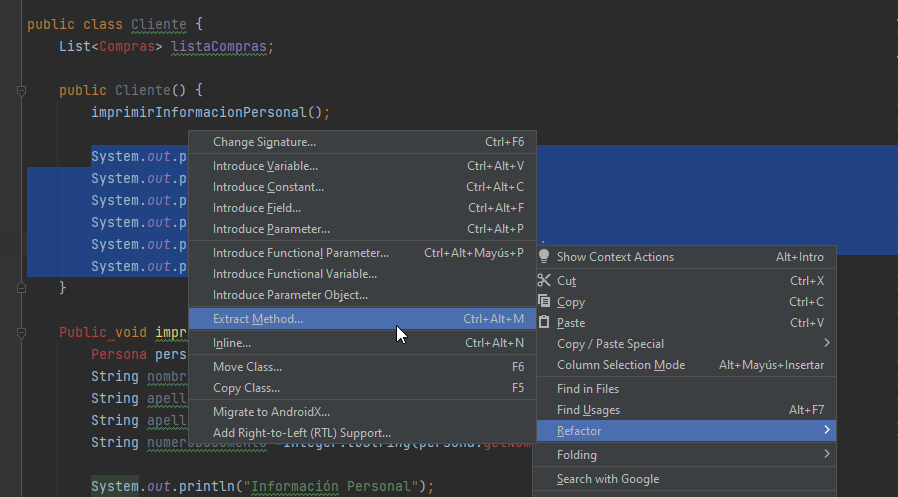
1. Se procede a usar el método de **Rename**, en intellij se debe ubicar en la variable a renombrar, dar click derecho, escoger Refactor y la opción Rename, luego se debe asignar el nombre según las pautas de buenas prácticas (el nombres debe ser fácil de pronunciar, que generen un sentido y significado que revele la intención. Se recomienda el uso de Camel Case).

Lo anterior lo tendremos en cuenta en:

* 1. El nombre de la lista cmp -> se renombra a listaCompras
  2. El nombre del método impTodaInfoCliente -> se renombra a imprimirTodaLaInformacionDelCliente
  3. El nombre de la variable cmpLocal -> se renombra compraLocal



1. Se procede a refactorizar usando el método de **introduce Method**, en intellij se debe seleccionar el código que deseamos que haga parte de un método, luego le damos click derecho y escogemos la opción Refactor luego la opción Extrad Method y luego procedemos a nombrar el nuevo método que acabamos de crear, este nombre debe cumplir con las buenas prácticas, que sean fácil de pronunciar y que genere un sentido y significado evitando desinformar de la funcionalidad del método. Refactorizamos para crear el método que tendrá una **responsabilidad única** de imprimir la información familiar, por tanto así se deja más limpio y entendible el código.

****

1. **Método de refactorización encapsulated Fields:** Se cambian los modificadores de acceso a Prívate en los métodos, debido a que estos son accedidos en la misma clase Cliente.

**El código ya refactorizado quedaría de la siguiente forma:**

public class Cliente {  
 List<Compras> listaCompras;  
  
 public Cliente() {  
 imprimirInformacionPersonal();  
  
 imprimirInformacionFamiliar();  
 }  
  
  
 private void imprimirInformacionFamiliar() {  
 System.*out*.println("Información Familiar");  
 System.*out*.println("Estado Civil: "+ "Casado");  
 System.*out*.println("Cantidad de hijos: "+ "3");  
 System.*out*.println("Cantidad de hermanos: "+ "4");  
 System.*out*.println("Nombre del padre: "+ "Jesus Zapata");  
 System.*out*.println("Nombre de la madre: "+ "Patricia Restrepo");  
 }  
  
 private void imprimirInformacionPersonal(){  
 Persona persona = new Persona();  
 String nombre = persona.getNombre();  
 String apellido1 =persona.getPrimerAppellido();  
 String apellido2 =persona.getSegundoAppellido();  
 String numeroDocumento =Integer.*toString*(persona.getNumeroDeDocumento());  
  
 System.*out*.println("Información Personal");  
 System.*out*.println("Nombre: "+ nombre);  
 System.*out*.println("Apellido 1: "+ apellido1);  
 System.*out*.println("Apellido 2: "+ apellido2);  
 System.*out*.println("Numero de documento: "+ numeroDocumento);  
 }  
  
  
 public void imprimirTodaLaInformacionDelCliente(){  
 System.*out*.println("Nombre: "+ "Antonio");  
 System.*out*.println("Apellido 1: "+ "Zapata");  
 System.*out*.println("Apellido 2: "+ "Restrepo");  
 System.*out*.println("Numero de documento: "+ "123456789");  
 for(Compras compraLocal : listaCompras) {  
 system.out.println(compraLocal);  
 }  
 }  
  
}