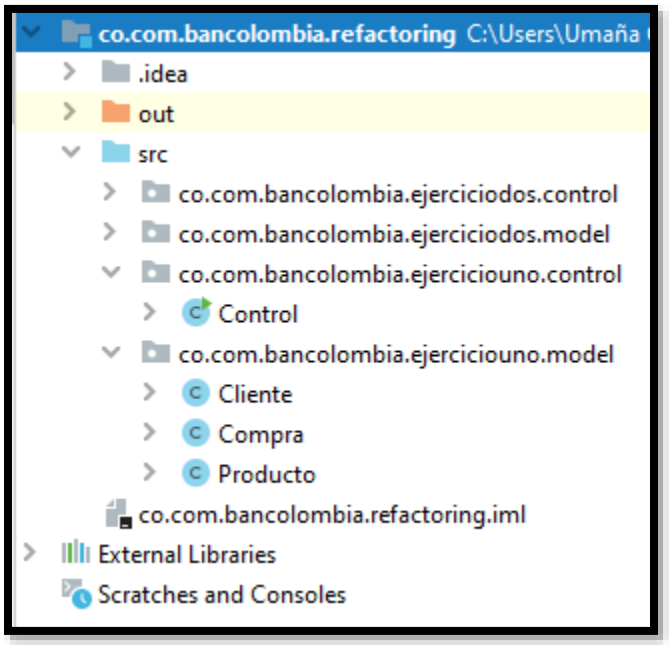
TALLER BUENAS PRACICAS DE PROGRAMACIÓN

https://github.com/bryanumana/Refactoring

Ejercicio Uno:

1. Crear varias clases adicionales como lo son: Control, Cliente, Producto y Compra; con el fin de visualizar mejor el modelo de la aplicación.



2. Añadir los atributos correctamente con su respectiva nomenclatura y separándolos de la clase principal:

```java
package co.com.bancolombia.ejerciciouno.model;

public class Cliente {

    private int cedulaCliente;
    private String nombreCliente;
    private String primerApellidoCliente;
    private String segundoApellidoCliente;
```

```java
package co.com.bancolombia.ejerciciouno.model;

public class Producto {

    private String nombreProducto;
```

3. Renombrar los métodos, ya que en el ejercicio propuesto contenían abreviaciones:

```java
    public void imprimirInformacionCliente(){
        System.out.println("Número de cedula:" + this.getCedulaCliente());
        System.out.println("Nombre: " + this.getNombreCliente());
        System.out.println("Primer Apellido: " + this.getPrimerApellidoCliente());
        System.out.println("Segundo Apellido: " + this.getSegundoApellidoCliente());
        System.out.println("");
    }
```

```java
public void mostrarOrden() {
    System.out.println("PRODUCTOS COMPRADOS POR EL CLIENTE");
    System.out.println("Orden:" + this.idOrden);

    for(int i = 0; i < this.contadorProductos; ++i) { System.out.println("Producto:" + this.compras[i].getNombreProducto()); }

}
```

4. Crear una clase aparte para apreciar las compras llamada Compra y así posteriormente mostrarlas junto con la información del cliente, con el fin de que cada método tenga una sola responsabilidad.

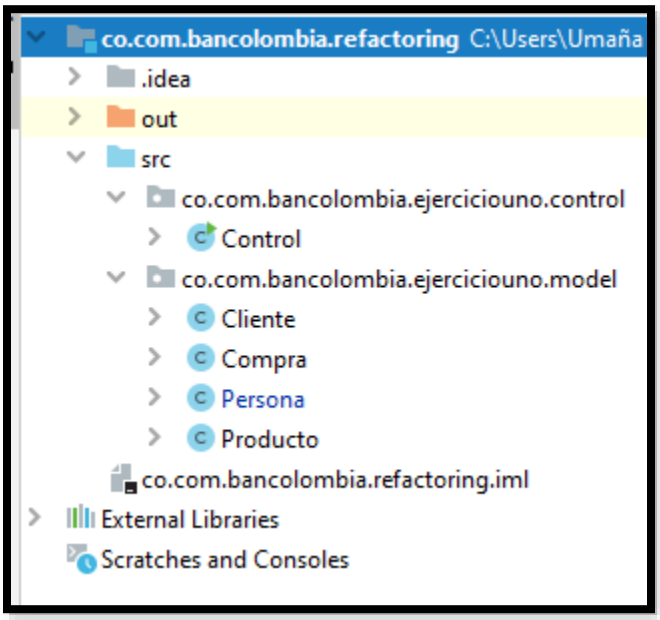
```java
public class Compra {

    private final int idOrden;
    private Producto[] compras;
    private static int contadorCompras;
    private int contadorProductos;

    public Compra() {
        this.idOrden = ++contadorCompras;
        this.compras = new Producto[10];
    }

    public void agregarComputadora(Producto computadora) {
        if (this.contadorProductos < 10) {
            this.compras[this.contadorProductos++] = computadora;
        } else {
            System.out.println("Has superado el limite: 10");
        }

    }

    public void mostrarOrden() {
        System.out.println("PRODUCTOS COMPRADOS POR EL CLIENTE");
        System.out.println("Orden:" + this.idOrden);

        for(int i = 0; i < this.contadorProductos; ++i) { System.out.println("Producto:" + this.compras[i].getNombreProducto()); }

    }
```

5. En la clase Control, aplicando la POO, es donde se crean los objetos para poder ejecutarse de una forma eficiente, para esto se hace uso de los sets gets y los métodos de cada clase especificada anteriormente.

```java
public class Control {
    public static void main(String[] args) {



        Cliente objCliente = new Cliente();

        objCliente.setCedulaCliente(123456789);
        objCliente.setNombreCliente("Antonio");
        objCliente.setPrimerApellidoCliente("Restrepo");
        objCliente.setSegundoApellidoCliente("Zapata");

        Producto cafe = new Producto();
        Producto maiz = new Producto();

        cafe.setNombreProducto("cafe");
        maiz.setNombreProducto("maiz");

        Compra orden1 = new Compra();
        orden1.agregarComputadora(cafe);
        orden1.agregarComputadora(maiz);

        System.out.println("_____INFORMACIÓN DEL CLIENTE_____");
        objCliente.imprimirInformacionCliente();

        System.out.println("_____INFORMACIÓN COMPLETA DEL CLIENTE_____");
        objCliente.imprimirInformacionCliente();
        orden1.mostrarOrden();
    }
```

Ejercicio Dos:

Move Class: Se dividió la clase principal en cinco clases, con el fin de independizar a los objetos; cabe resaltar que cada clase tiene set y get, para que no genere sobrecarga y se puedan crear objetos con mayor eficiencia.



Encapsulate Fields and Rename: En el ejercicio propuesto a la hora de crear la clase Persona, se hizo sin ningún tipo de encapsulamiento, así que al crear los atributos y métodos de cada nueva clase respectivamente, se dieron un nombramiento adecuado y una encapsulación óptima, además que esta clase hereda, por lo que todos los encapsulamientos son de tipo protected.

```
public class Persona {
    protected String estadoCivil;
    protected int cantidadHijos;
    protected int cantidadHermanos;
    protected String nombreDelPadre;
    protected String nombreDeLaMadre;
```

```java
public class Cliente extends Persona{

    private int cedulaCliente;
    private String nombreCliente;
    private String primerApellidoCliente;
    private String segundoApellidoCliente;
```

```java
public class Producto {

    private String nombreProducto;
```

Introduce Methods: Se independizaron los métodos a cada clase correspondiente, al igual que se agregó un nuevo método llamado: agregarProducto, porque en el ejercicio anterior no se sabía muy bien de dónde o cómo funcionaba esa lista de compras, y en esa misma intancia se observa que el for y ocupa una sola línea.

```java
    public void imprimirInformacionFamiliar(){
        System.out.println("Estado civil:" + this.getEstadoCivil());
        System.out.println("Número de hijos:" + this.getCantidadHijos());
        System.out.println("Número de hermanos: " + this.getCantidadHermanos());
        System.out.println("Nombre del padre: " + this.getNombreDelPadre());
        System.out.println("Nombre de la madre: " + this.getNombreDeLaMadre());
        System.out.println("");
    }
    public void imprimirInformacionCliente(){
        System.out.println("Número de cedula:" + this.getCedulaCliente());
        System.out.println("Nombre: " + this.getNombreCliente());
        System.out.println("Primer Apellido: " + this.getPrimerApellidoCliente());
        System.out.println("Segundo Apellido: " + this.getSegundoApellidoCliente());
        System.out.println("");
    }
```

```java
public void agregarComputadora(Producto computadora) {
    if (this.contadorProductos < 10) {
        this.compras[this.contadorProductos++] = computadora;
    } else {
        System.out.println("Has superado el limite: 10");
    }

}

public void mostrarOrden() {
    System.out.println("PRODUCTOS COMPRADOS POR EL CLIENTE");
    System.out.println("Orden:" + this.idOrden);

    for(int i = 0; i < this.contadorProductos; ++i) { System.out.println("Producto: " + this.compras[i].getNombreProducto()); }

}
```