

1.) Si se desea modelar un conjunto de objetos del tipo “Vendedor de sopaipillas” aplicando los conceptos de la POO.

a. ¿Qué atributos debe tener? ¿tipo de dato?

b. ¿Qué métodos debe tener? ¿tipo de retorno? ¿parámetros de entrada?

Para todas las respuestas de esta tarea hay infinidad de opciones, yo elegí:

a)

1.- Inventario - int (cantidad de sopaipillas disponibles)

2.- precio - int (precio de las sopaipilla)

3.- nombre - String (nombre del vendedor)

4.- pagar- int (valor a cancelar)

b) cabe destacar que todos estos métodos pueden transformarse en void y guardarse en los atributos si se estima conveniente

1.- crearPrecio – int – void

2.- crearInventario – int – void

3.- vender- int- int

4.- darVuelto – int – int

5.- reponer – int -int

6.- actualizarInventario – int – void

7.- todos los getters y setters para los atributos

8.- posibles métodos para completar los anteriores

2.) Ahora considere, además, la opción que los objetos de tipo vendedor de sopaipillas puedan "dar el vuelto para cada venta de sopaipillas" que realizan.

a. ¿Qué nuevos atributos debe tener? ¿tipo de dato?

b. ¿Qué nuevos métodos debe tener? ¿tipo de retorno? ¿parámetros de entrada?

A y b) se responde con la opción anterior, aunque también existe la posibilidad de crear un dinero previo como atributo, este destinado para hacer las sopaipillas y para dar posible vuelto, y los métodos para hacer esto posible

3.) Considere ahora la implementación de su diseño, creando un programa en LDP Java.

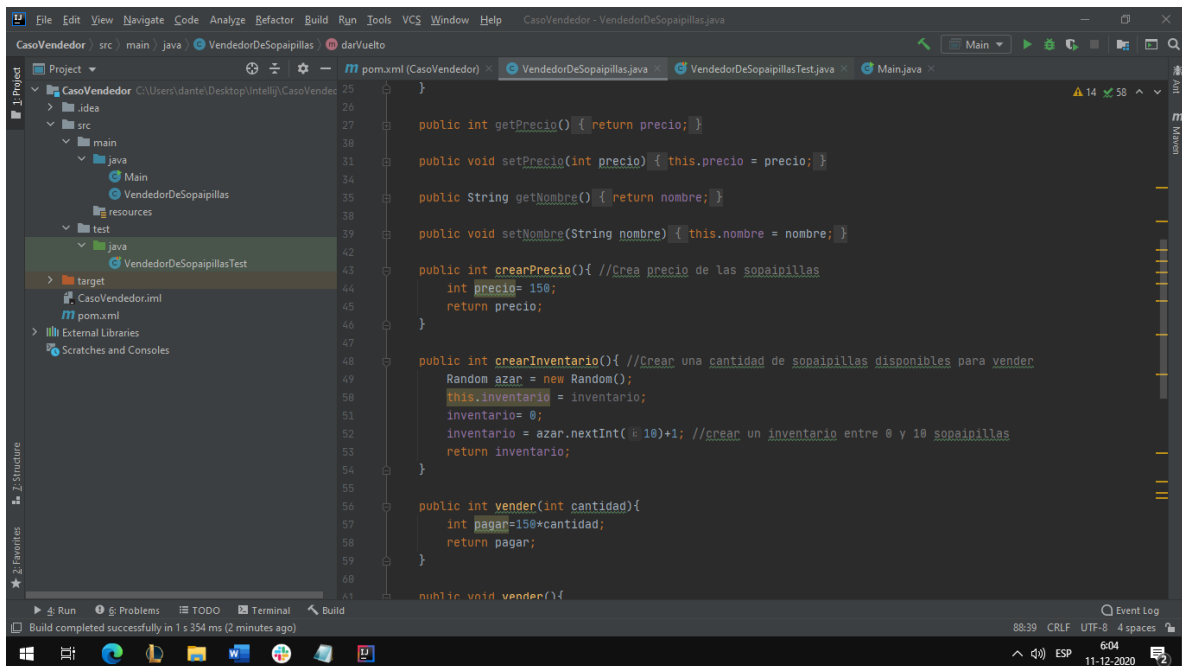
- a. Qué contenga una clase llamada VendedorSopaipillas, incluyendo los atributos y métodos de su diseño anterior.
- b. Qué contenga una clase llamada TestVendedorSopaipillas, que permita probar los principales métodos de su clase.
- c. Que contenga una clase pública llamada EjemploSopaipillas. En el método main() debe instanciar un objeto de tipo VendedorSopaipillas llamado miVendedor que use los métodos definidos en su diseño.

a)

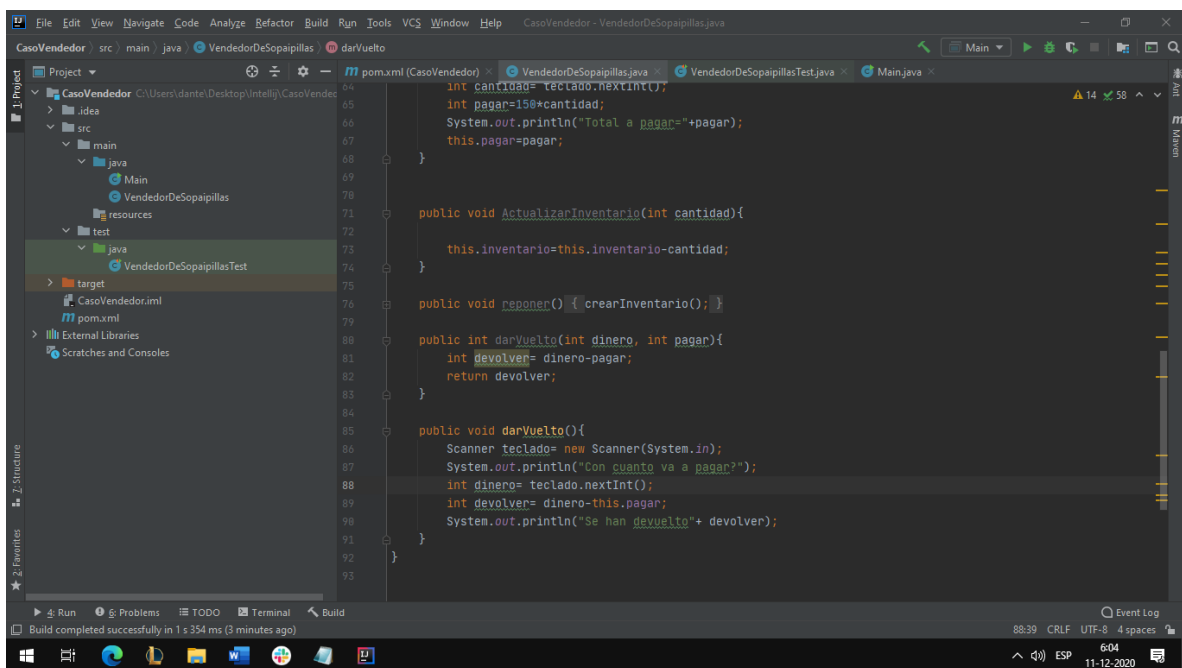
The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'CasoVendedor' with a 'src' directory containing 'main' and 'test' subdirectories. The 'main' directory contains 'Main' and 'VendedorDeSopaipillas'. The 'test' directory contains 'VendedorDeSopaipillasTest'.
- Code Editor:** Displays the implementation of the 'VendedorDeSopaipillas' class. The code includes imports for 'java.util.Random' and 'java.util.Scanner'. The class has private attributes: 'inventario' (int), 'precio' (int), 'nombre' (String), and 'pagar' (int). It includes a constructor 'VendedorDeSopaipillas(String nombre)' that initializes 'nombre', 'precio' (via 'crearPrecio()'), and 'inventario' (via 'crearInventario()'). It also has methods 'getInventario()', 'setInventario(int inventario)', 'getPrecio()', and 'setPrecio(int precio)'.
- Terminal:** Shows a successful build message: 'Build completed successfully in 1 s 354 ms (2 minutes ago)'.
- Status Bar:** Displays '88:39 CRLF UTF-8 4 spaces'.

```
1 import java.util.Random;
2 import java.util.Scanner;
3
4 public class VendedorDeSopaipillas {
5     private int inventario;
6     private int precio;
7     private String nombre;
8     private int pagar;
9
10
11     public VendedorDeSopaipillas(String nombre){
12         this.nombre = nombre;
13         this.precio=crearPrecio();
14         this.inventario=crearInventario();
15     }
16
17     public int getInventario() {
18
19         return inventario;
20     }
21
22     public void setInventario(int inventario) {
23
24         this.inventario = inventario;
25     }
26
27     public int getPrecio() { return precio; }
28
29     public void setPrecio(int precio) { this.precio = precio; }
```



```
25 }
26
27 public int getPrecio() { return precio; }
28
29 public void setPrecio(int precio) { this.precio = precio; }
30
31 public String getNombre() { return nombre; }
32
33 public void setNombre(String nombre) { this.nombre = nombre; }
34
35 public int crearPrecio() { //Crea precio de las sopaipillas
36     int precio= 150;
37     return precio;
38 }
39
40 public int crearInventario() { //Crear una cantidad de sopaipillas disponibles para vender
41     Random azar = new Random();
42     this.inventario = inventario;
43     inventario = 0;
44     inventario = azar.nextInt(10)+1; //crear un inventario entre 0 y 10 sopaipillas
45     return inventario;
46 }
47
48 public int vender(int cantidad) {
49     int pagar=150*cantidad;
50     return pagar;
51 }
52
53 public void vender() {
```

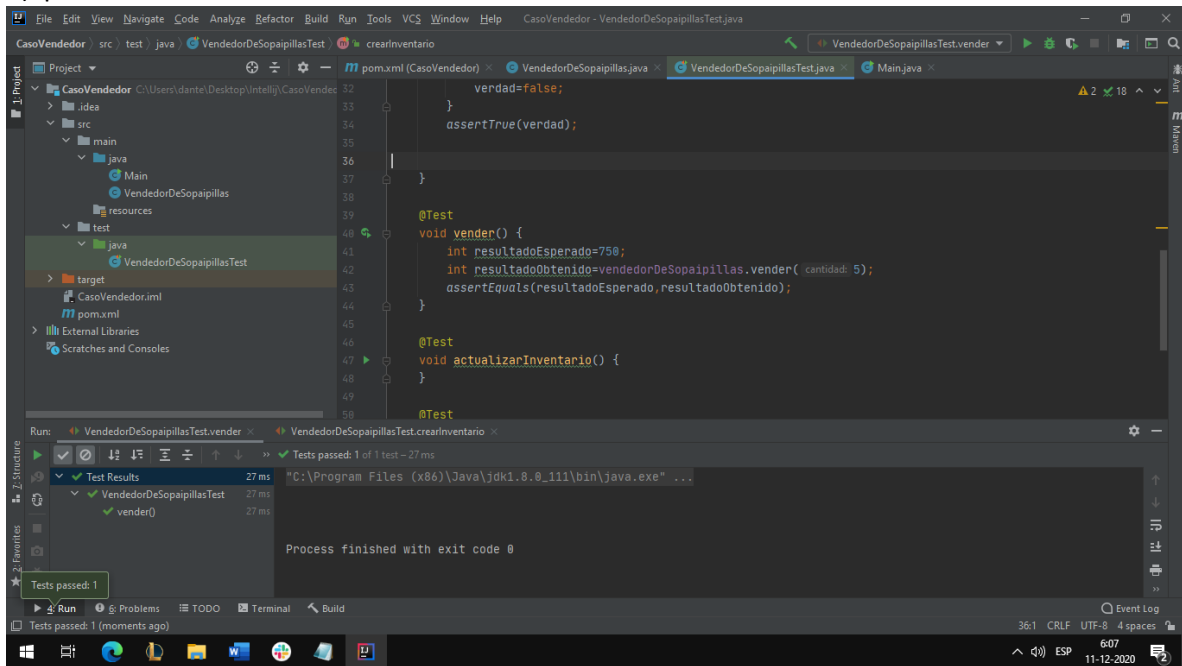


```
54     int cantidad= teclado.nextInt();
55     int pagar=150*cantidad;
56     System.out.println("Total a pagar="+pagar);
57     this.pagar=pagar;
58 }
59
60 public void ActualizarInventario(int cantidad) {
61     this.inventario=this.inventario-cantidad;
62 }
63
64 public void reponer() { crearInventario(); }
65
66 public int darVuelto(int dinero, int pagar) {
67     int devolver= dinero-pagar;
68     return devolver;
69 }
70
71 public void darVuelto() {
72     Scanner teclado= new Scanner(System.in);
73     System.out.println("Con cuanto va a pagar?");
74     int dinero= teclado.nextInt();
75     int devolver= dinero-this.pagar;
76     System.out.println("Se han devuelto"+ devolver);
77 }
78
79 }
```

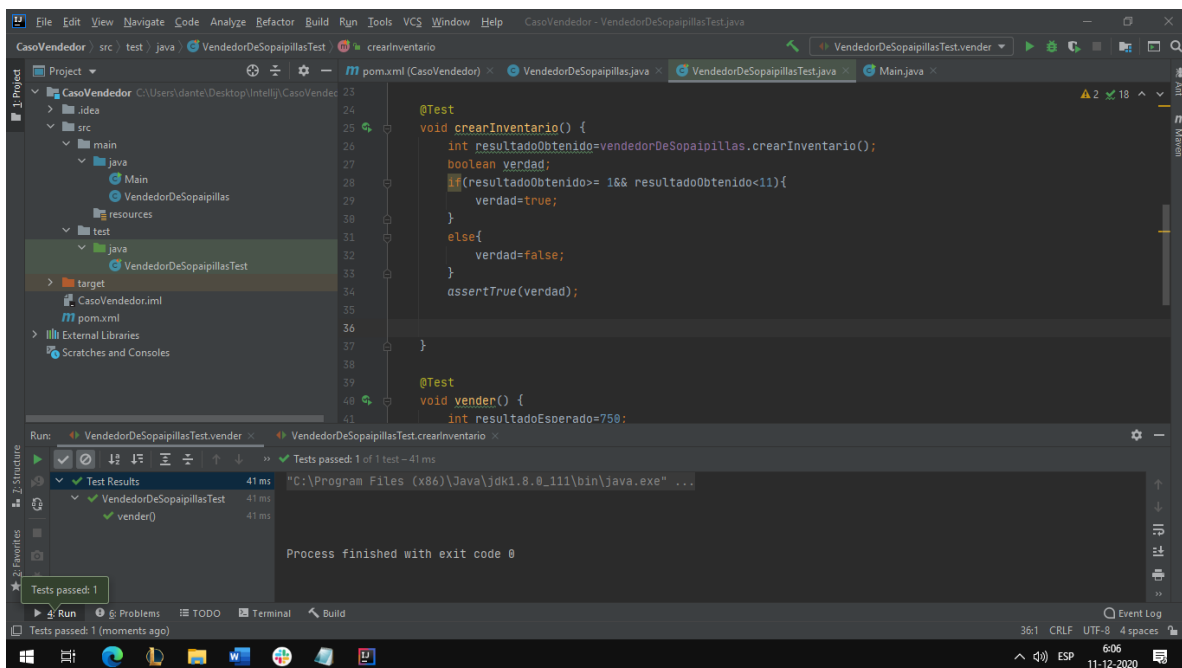
Cabe mencionar que el código siempre es mejorable, con métodos más eficientes o más realísticos como dije anteriormente agregando por ejemplo el atributo de bolsillo, con un dinero disponible para dar vuelto o crear las sopaipillas, a su vez también se puede crear un objeto sopaipillas y también mejorar el código que ya tengo agregando manejo de errores posibles, etc

O también un manejo de inventario, utilizando los métodos que cree como actualizarInventario, pero omiti su uso, para simplificar el problema y solo dar solución a lo que me pedían.

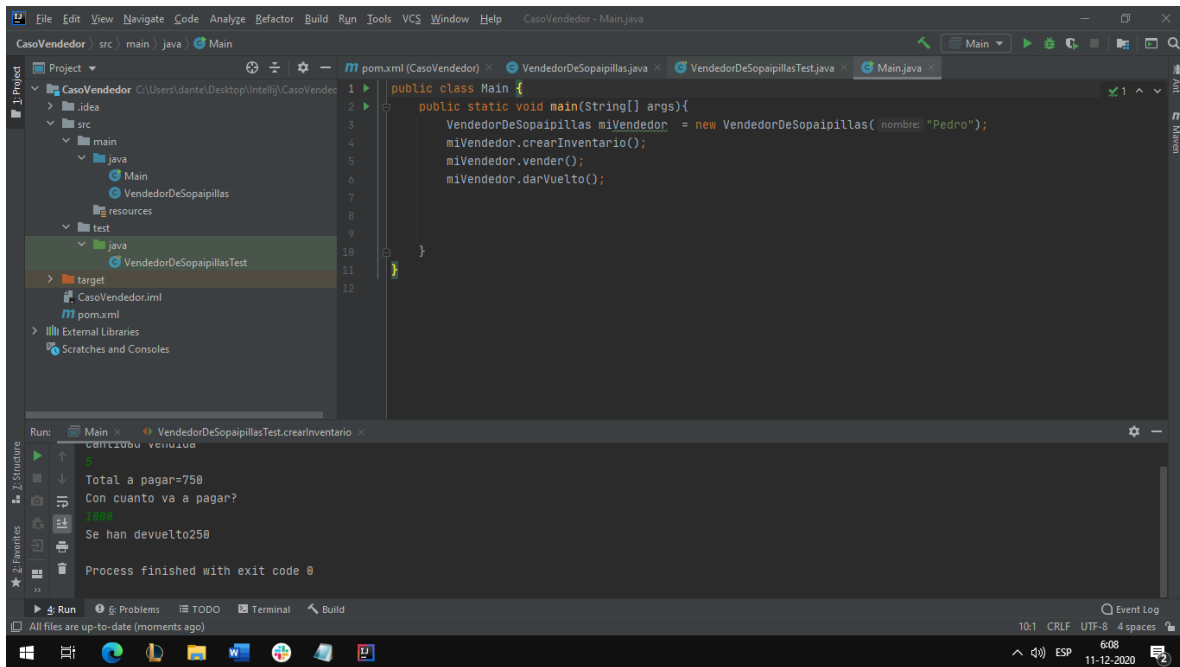
b) prueba vender



prueba crearInventario



c) ejecución del programa



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help CasoVendedor - Main.java
CasoVendedor src main java Main
pom.xml(CasoVendedor) VendedorDeSopaipillas.java VendedorDeSopaipillasTest.java Main.java
CasoVendedor
  .idea
  src
    main
      java
        Main
        VendedorDeSopaipillas
      resources
      test
        java
          VendedorDeSopaipillasTest
    target
      CasoVendedor.iml
      pom.xml
  External Libraries
  Scratches and Consoles
1 public class Main {
2     public static void main(String[] args){
3         VendedorDeSopaipillas miVendedor = new VendedorDeSopaipillas( nombre: "Pedro");
4         miVendedor.crearInventario();
5         miVendedor.vender();
6         miVendedor.darVuelto();
7     }
8 }
9
10
11
12
Run: Main VendedorDeSopaipillasTest.crearInventario
C:\Program Files\Java\jdk-11.0.2\bin\java.exe
Total a pagar=750
Con cuanto va a pagar?
1500
Se han devuelto250
Process finished with exit code 0
Run Problems TODO Terminal Build
All files are up-to-date (moments ago) 10:1 CRLF UTF-8 4 spaces 608 11-12-2020
```