**Instituto Politécnico Nacional.**

**Escuela Superior De Cómputo.**





**Materia:**

**Aplicaciones Para Comunicación En Red.**

**Tema:**

**Reporte Práctica 2**

**Profesor:**

**Axel Ernesto Moreno Cervantes.**

**Alumno:**

**Luis Enrique Alvarado Rojas.**

**Mario Alberto Miranda Sandoval.**

**Grupo:**

**3CM5**

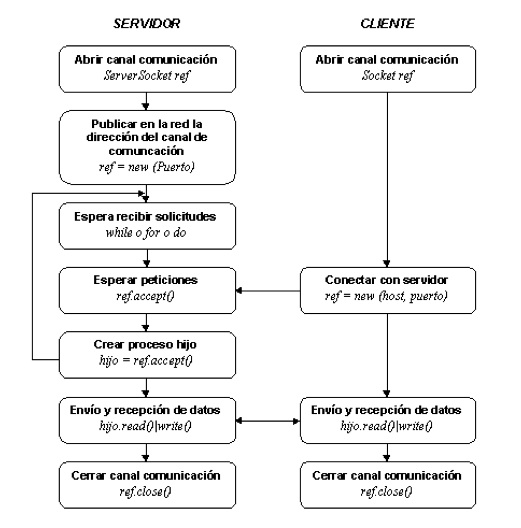
**Introducción.**

En la actualidad, muchos de los procesos que se ejecutan en una computadora requieren obtener o enviar información a otros procesos que se localizan en una computadora diferente.  Para lograr esta comunicación se utilizan los protocolos de comunicación TCP y UDP.

El protocolo TCP (Transmission Control Protocol) establece un conducto de comunicación punto a punto entre dos computadoras, es decir, cuando se requiere la transmisión de un flujo de datos entre dos equipos, el protocolo TCP establece un conducto exclusivo entre dichos equipos por el cual los datos serán transmitidos y este perdurará hasta que la transmisión haya finalizado, gracias a esto TCP garantiza que los datos enviados de un extremo de la conexión lleguen al otro extremo y en el mismo orden en que fueron enviados. Las características que posee TCP hacen que el protocolo sea conocido como un protocolo orientado a conexión.

Los sockets son una forma de comunicación entre procesos que se encuentran en diferentes máquinas de una red, los sockets proporcionan un punto de comunicación por el cual se puede enviar o recibir información entre procesos.

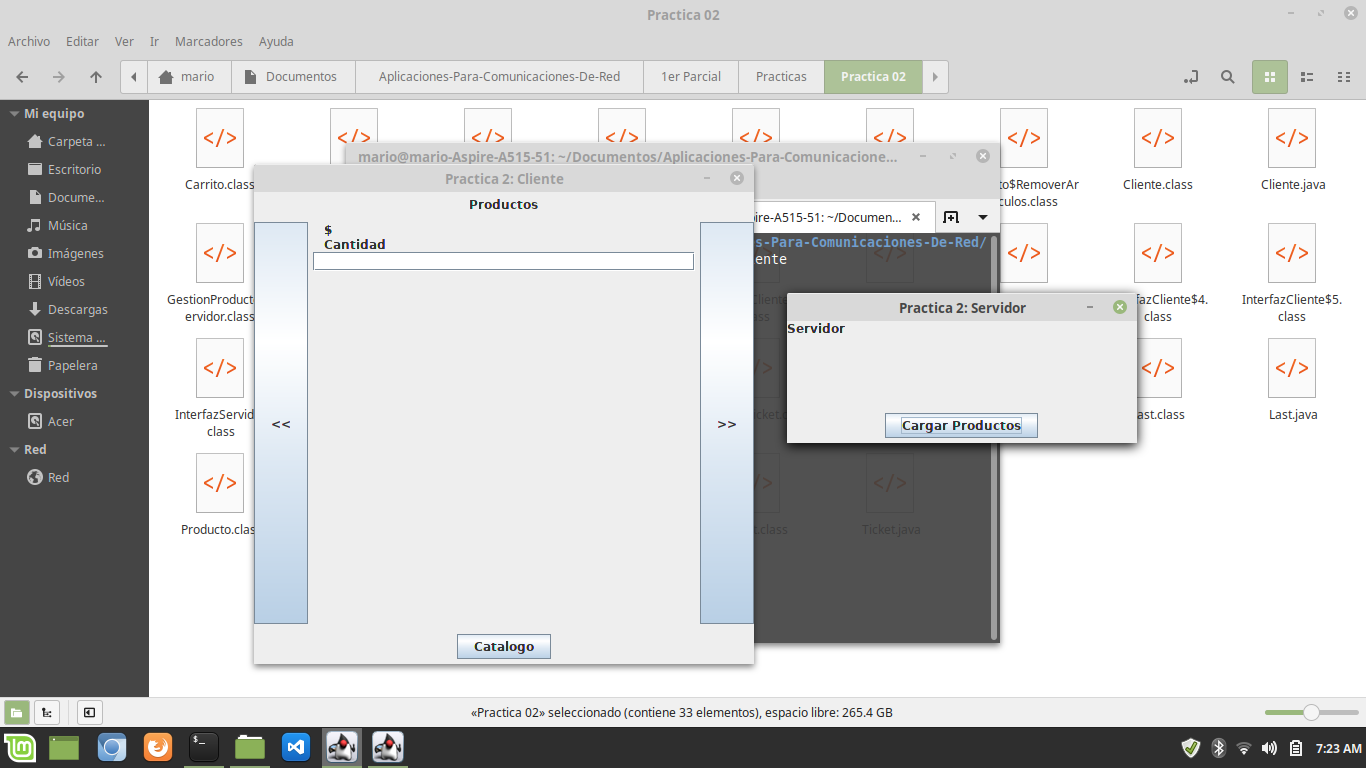
Los sockets tienen un ciclo de vida dependiendo si son sockets de servidor, que esperan a un cliente para establecer una comunicación, o socket cliente que busca a un socket de servidor para establecer la comunicación.



**Desarrollo.**

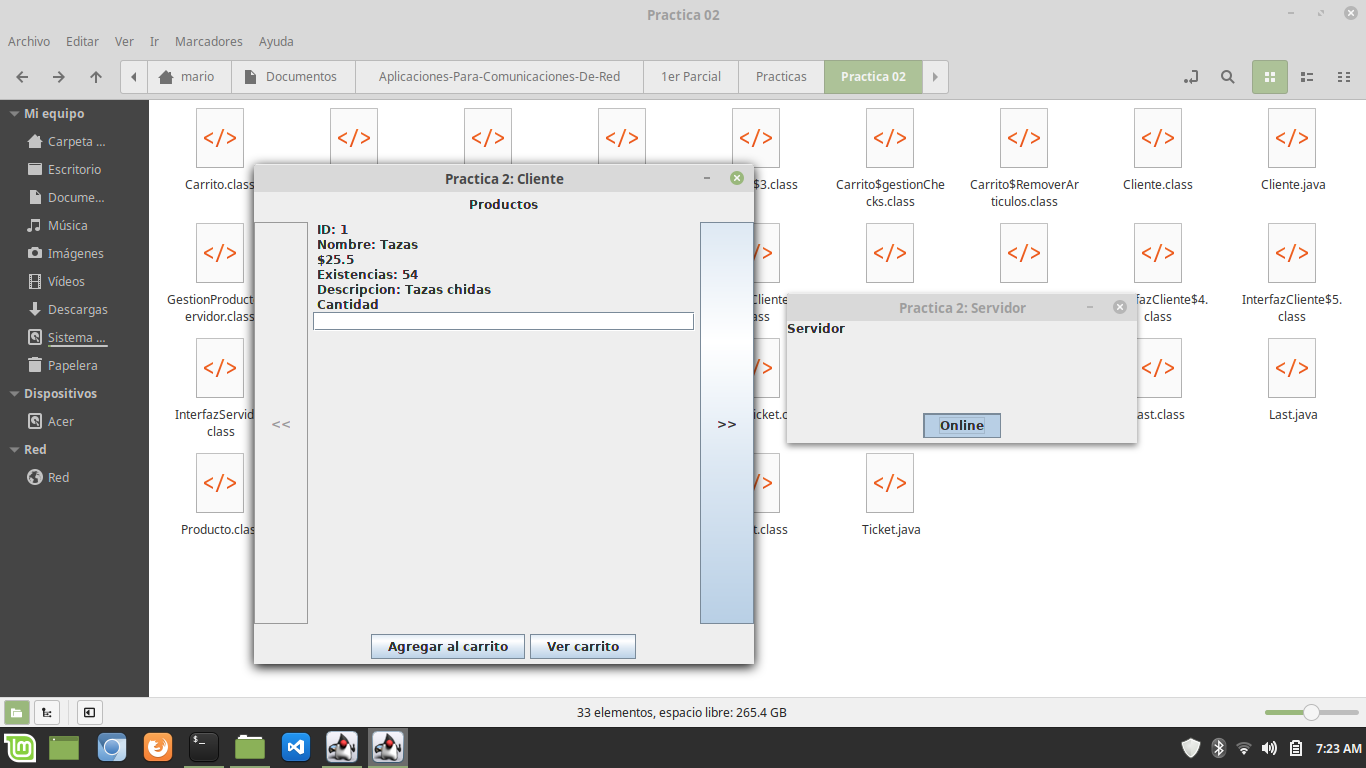
Para esta práctica se debe simular un sistema de compras en línea, donde el servidor guarde los artículos, gestione sus existencias y mande un ticket de compra, mientras que el usuario puede añadir artículos al carrito, removerlos y hacer su compra final.

Para esta práctica se genero la interfaz cliente y servidor como sigue:

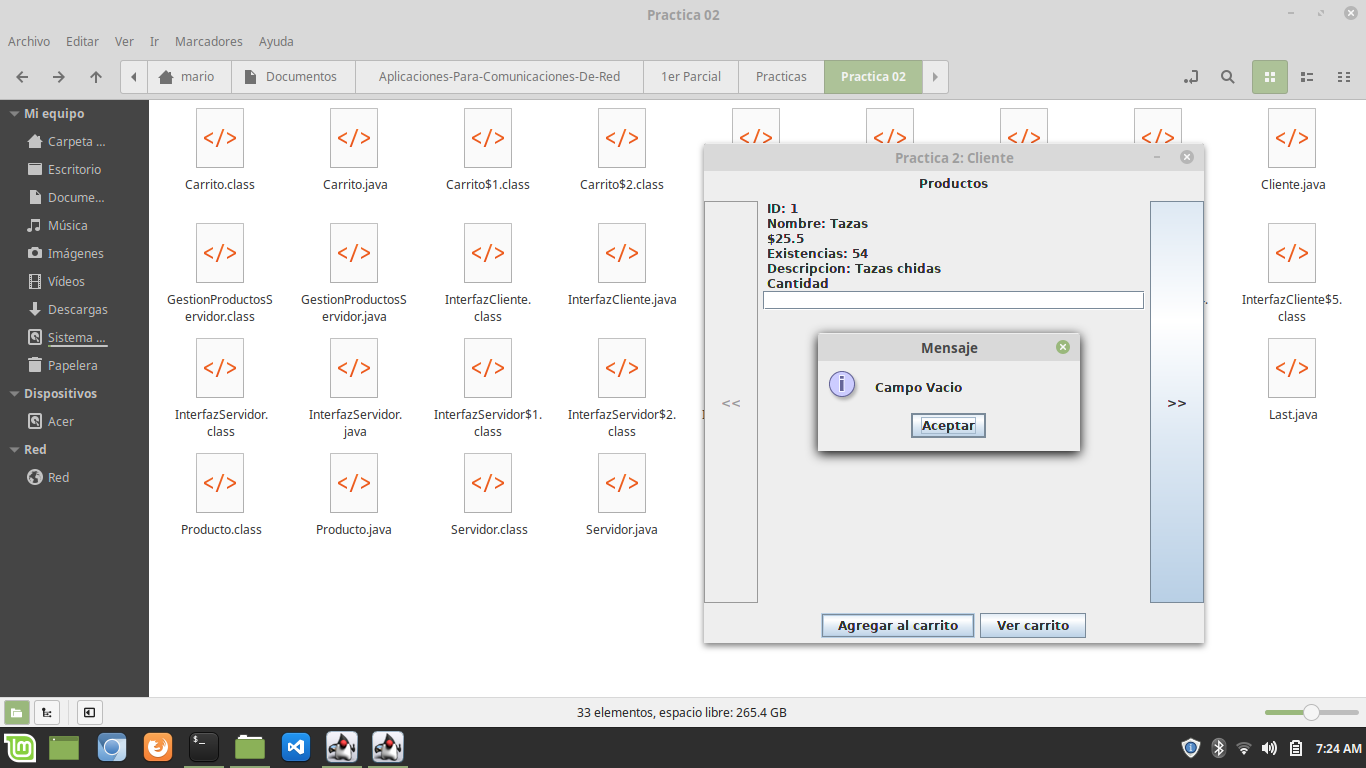


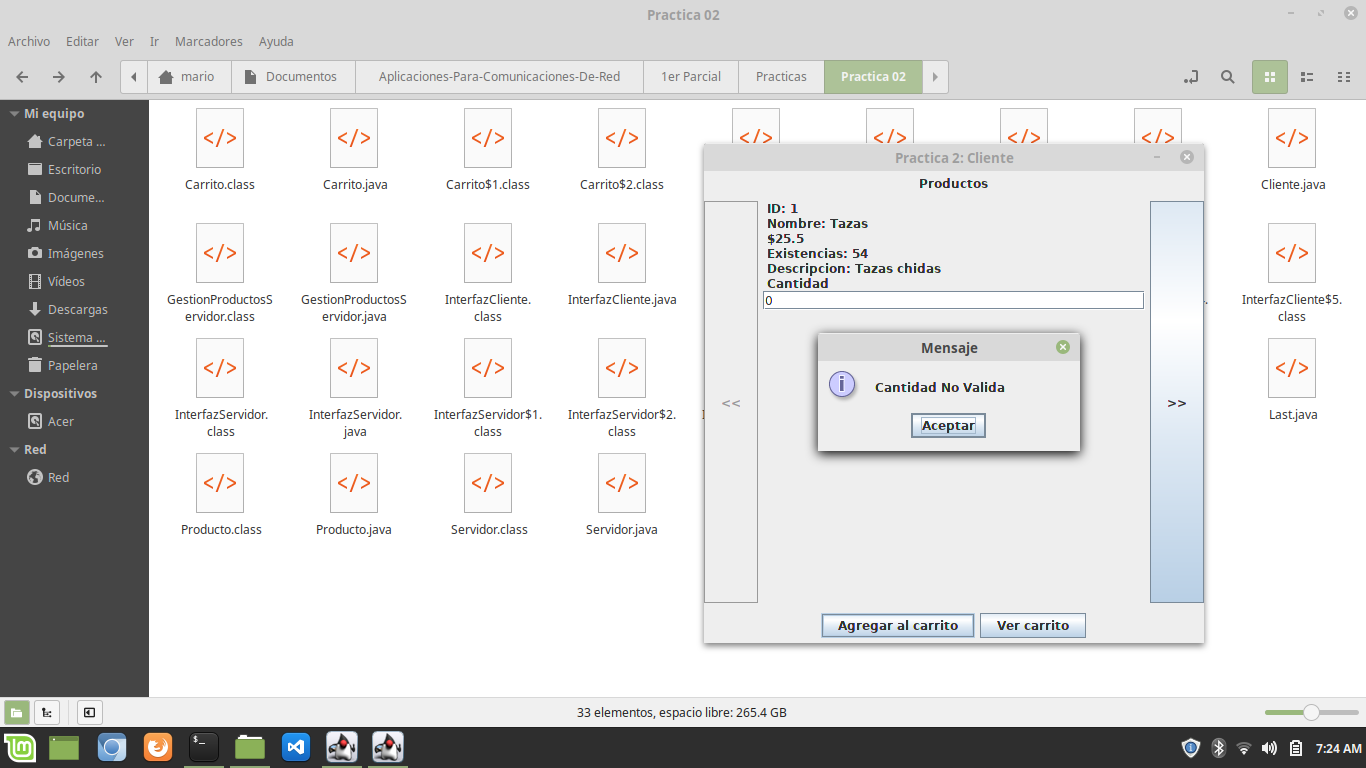
Interfaz de cliente y servidor.

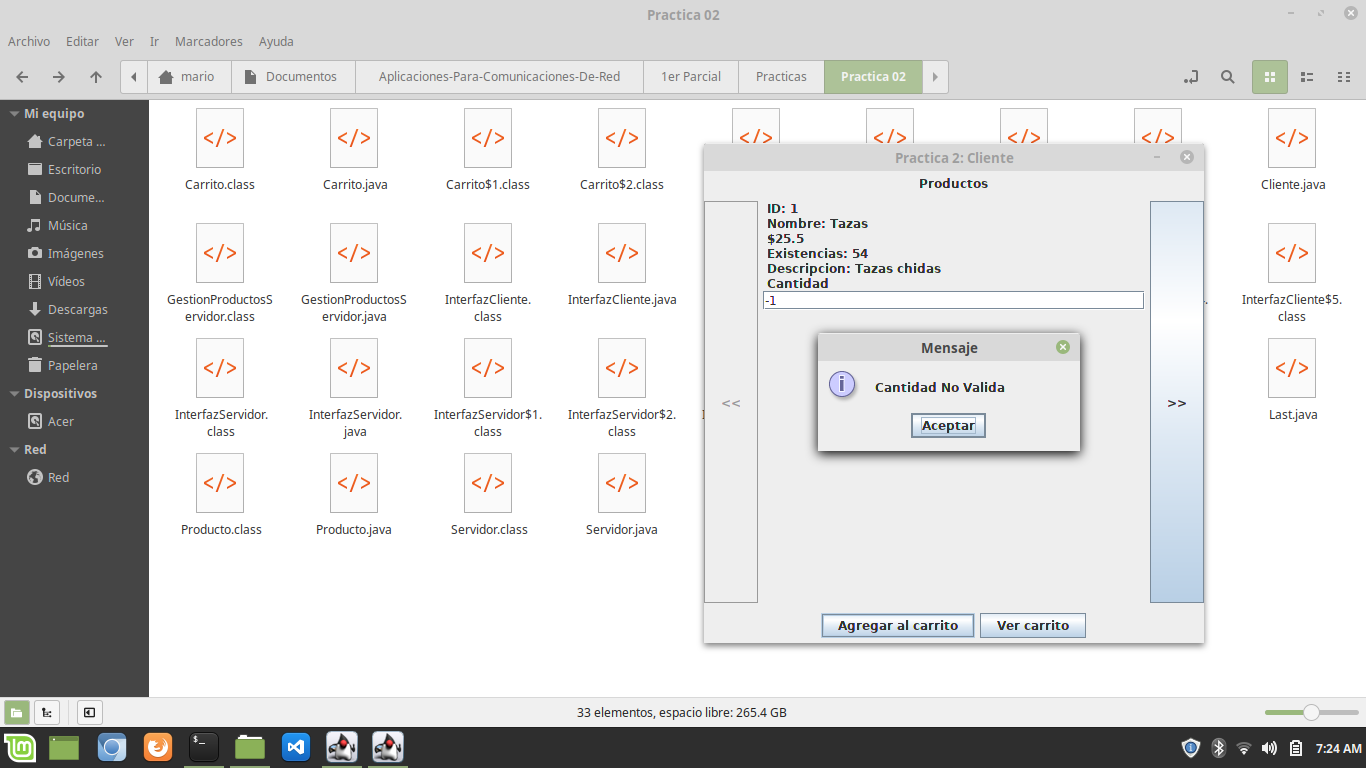
**Pruebas.**



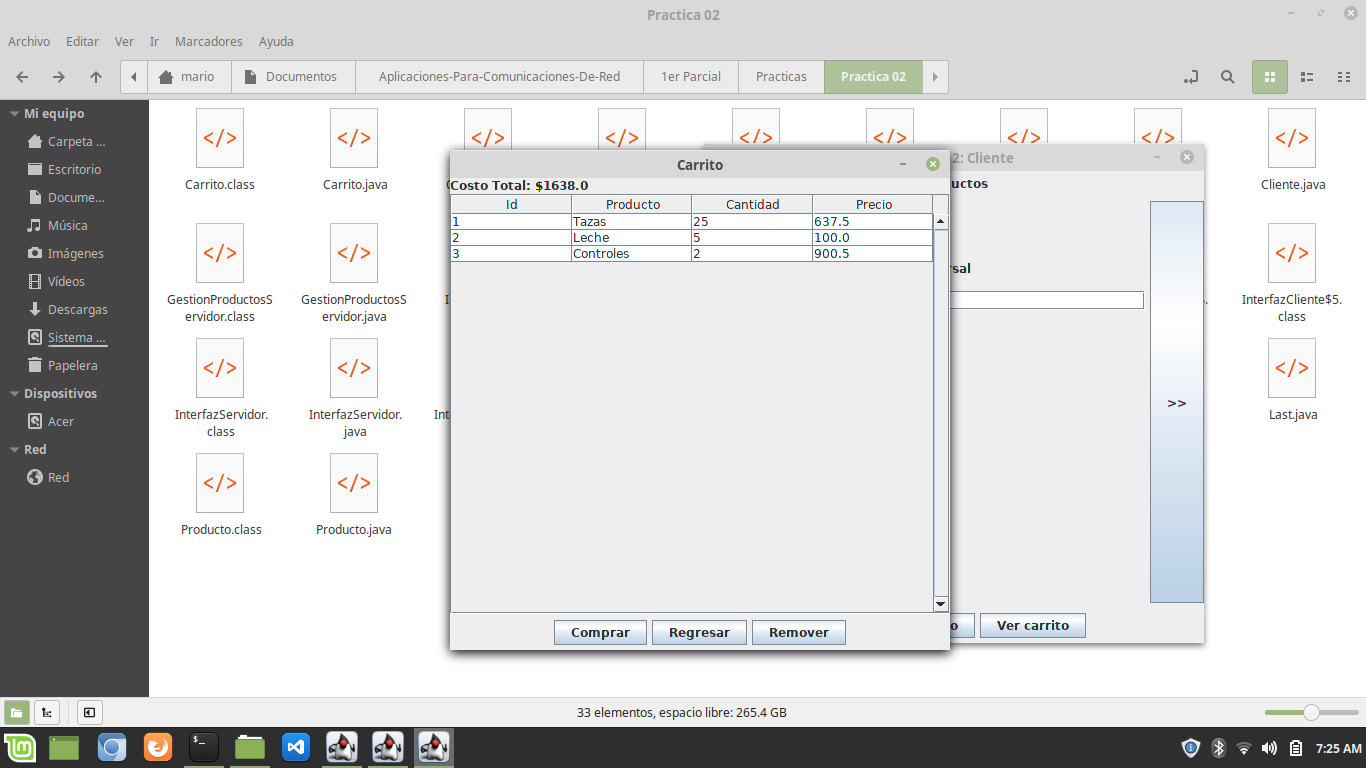
Cliente con productos cargados.

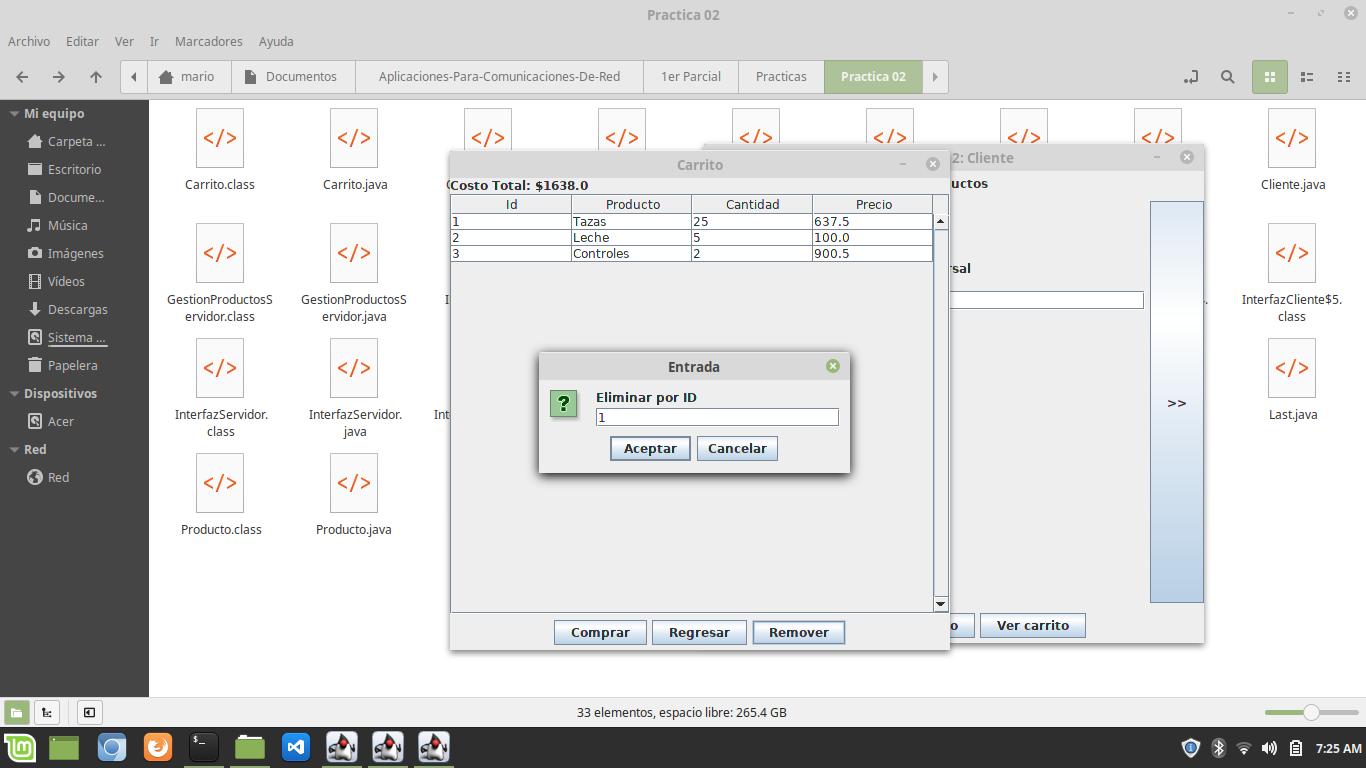


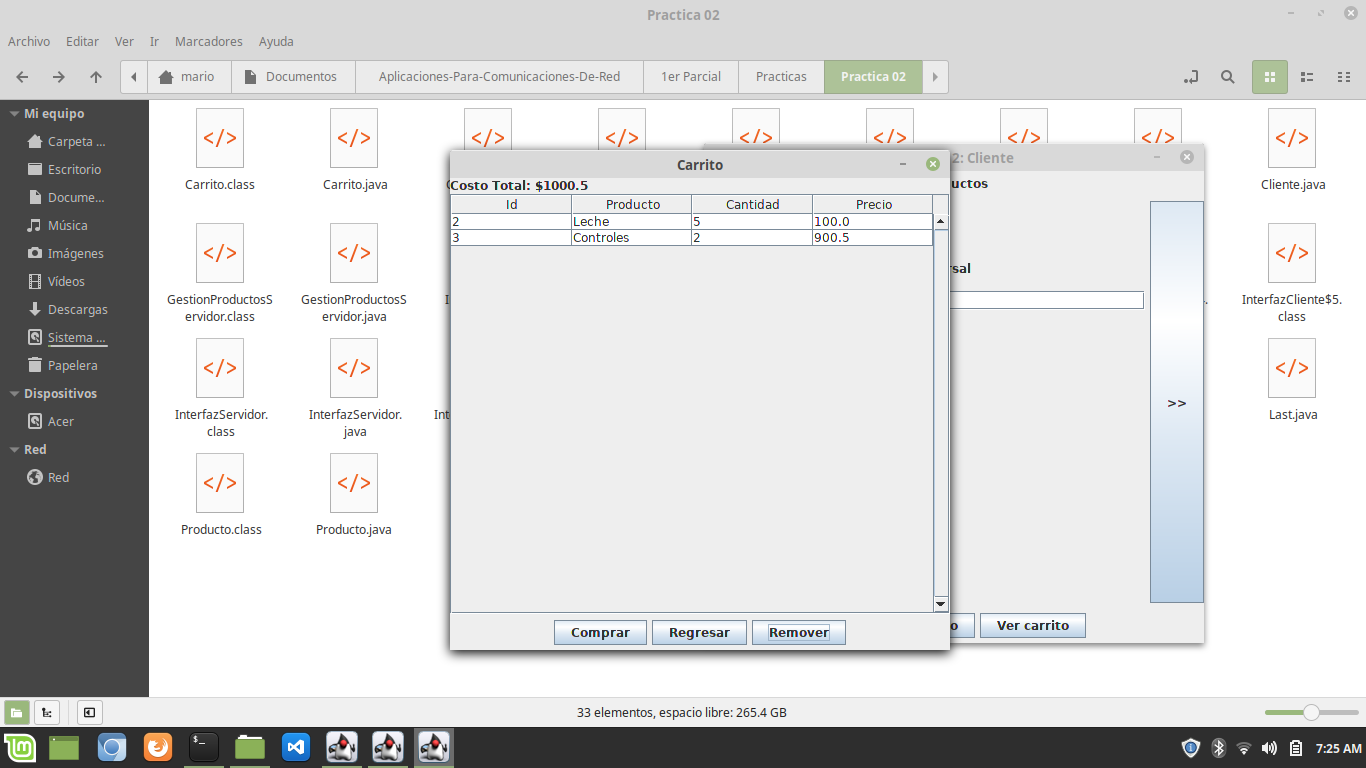




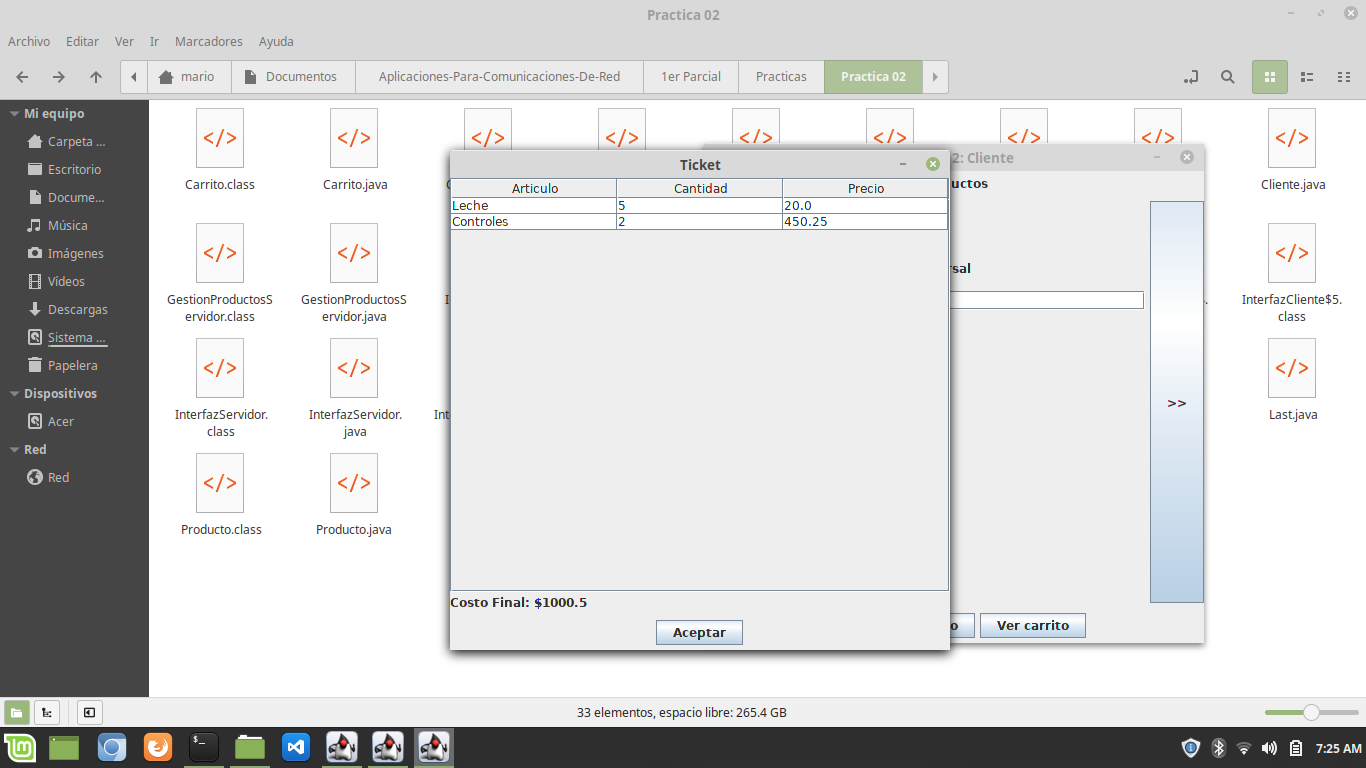
Validaciones.



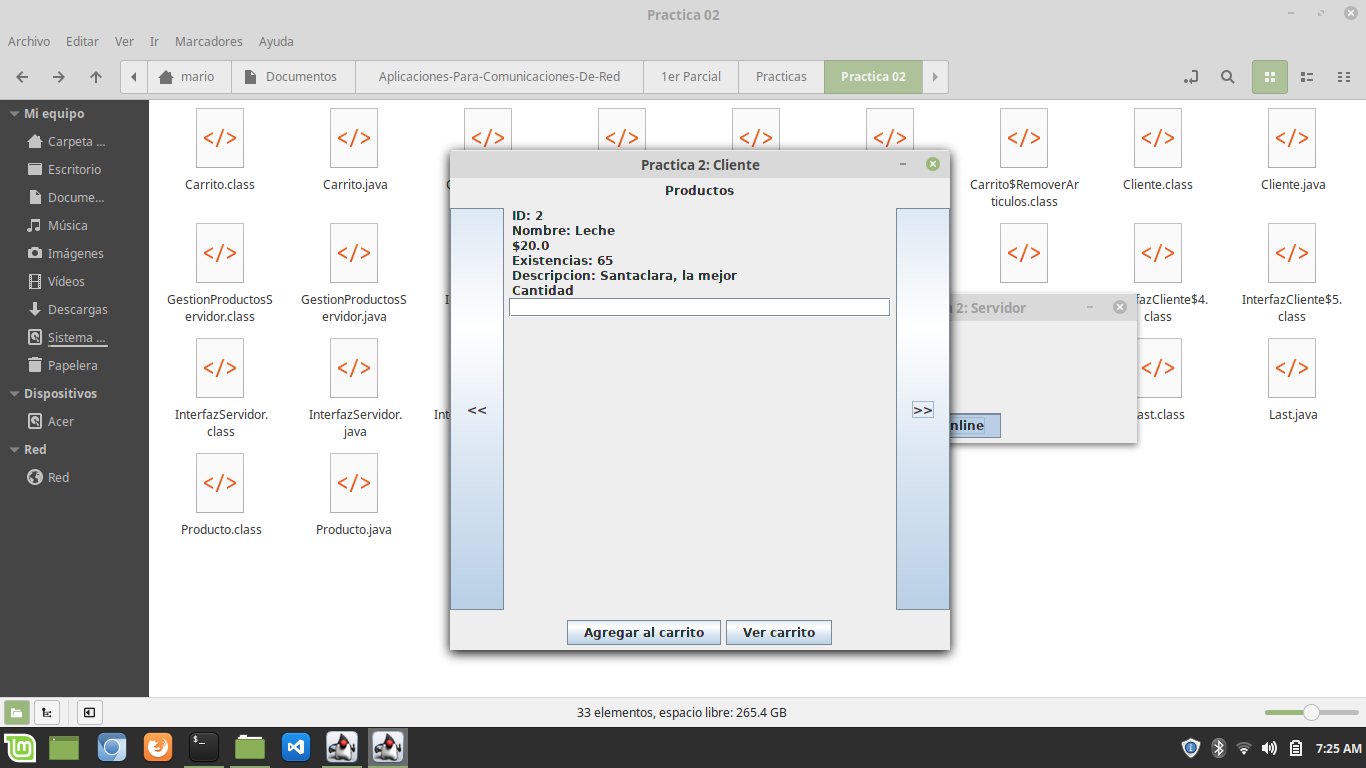


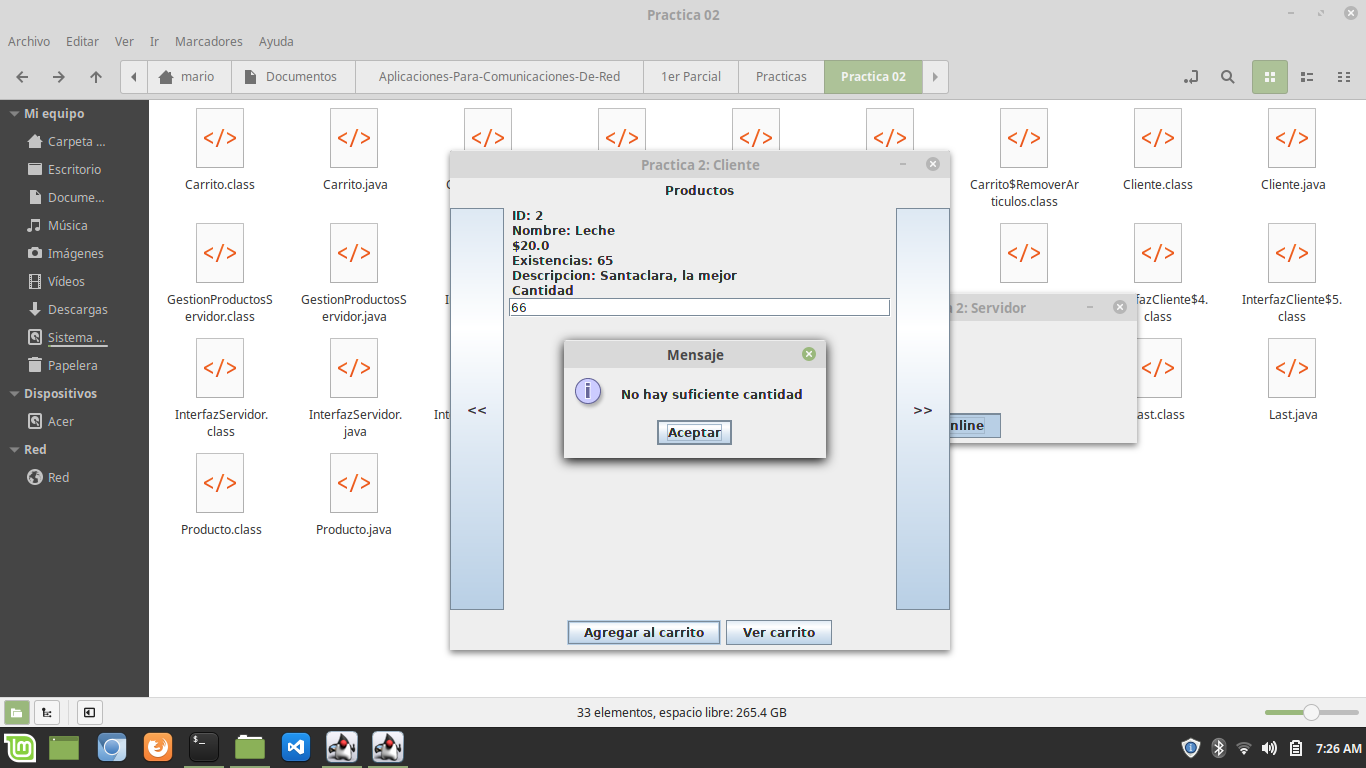


Cliente manipulando el carrito.



Ticket generado.





Un segundo cliente se conecta, las existencias se actualizan, y las validaciones continúan.

**Código**

**Producto.java**

1. **import** java.io.Serializable;
3. **public** **class** Producto **implements** Serializable {
4. **private** **static** **final** **long** serialVersionUID = 3L;
5. **public** Producto(**int** id, String nombre, **double** precio, **int** existencias, String descripcion, **boolean** promocion, String imagen) {
6. **this**.id = id;
7. **this**.nombre = nombre;
8. **this**.precio = precio;
9. **this**.existencias = existencias;
10. **this**.descripcion = descripcion;
11. **this**.promocion = promocion;
12. **this**.imagen = imagen;
13. cantidad = 0;
14. }
16. **public** **int** getID() { **return** id; }
17. **public** String getNombre() { **return** nombre; }
18. **public** **double** getPrecio() { **return** precio; }
19. **public** **int** getExistencias() { **return** existencias; }
20. **public** String getDescripcion() { **return** descripcion; }
21. **public** **boolean** getPromocion() { **return** promocion; }
22. **public** String getImagen() { **return** imagen; }
23. **public** **int** getCantidad() { **return** cantidad; }
25. **public** **void** setID(**int** id) { **this**.id = id; }
26. **public** **void** setNombre(String nombre) { **this**.nombre = nombre; }
27. **public** **void** setPrecio(**double** precio) { **this**.precio = precio; }
28. **public** **void** setExistencias(**int** existencias) { **this**.existencias = existencias; }
29. **public** **void** setDescripcion(String descripcion) { **this**.descripcion = descripcion; }
30. **public** **void** setPromocion(**boolean** promocion) { **this**.promocion = promocion; }
31. **public** **void** setImagen(String imagen) { **this**.imagen = imagen; }
32. **public** **void** setCantidad(**int** cantidad) { **this**.cantidad = cantidad; }
34. **private** **int** id;
35. **private** String nombre;
36. **private** **double** precio;
37. **private** **int** existencias;
38. **private** String descripcion;
39. **private** **boolean** promocion;
40. **private** String imagen;
41. **private** **int** cantidad;
42. }

**Last.java**

1. **import** java.io.Serializable;
2. **import** java.util.ArrayList;
4. **public** **class** Last **implements** Serializable {
5. **private** **static** **final** **long** serialVersionUID = 8L;
6. **public** Last(**int** ac, ArrayList<Producto> misProductos) {
7. **this**.ac = ac;
8. **this**.misProductos = misProductos;
9. }
11. **public** **int** getAccion() { **return** ac; }
12. **public** ArrayList<Producto> getProducto() { **return** misProductos; }
14. **public** **void** setAccion(**int** ac) { **this**.ac = ac; }
15. **public** **void** setProducto(ArrayList<Producto> misProductos) { **this**.misProductos = misProductos; }
17. **private** **int** ac;
18. **private** ArrayList<Producto> misProductos;
19. }

**Ticket.java**

1. **import** java.io.Serializable;
2. **import** java.util.ArrayList;
4. **public** **class** Ticket **implements** Serializable {
5. **private** **static** **final** **long** serialVersionUID = 9L;
6. **public** Ticket(ArrayList<Producto> productos, **double** precio) {
7. **this**.productos = productos;
8. **this**.precio = precio;
9. }//constructor
11. **public** ArrayList<Producto> getProductos() { **return** productos; }
12. **public** **double** getPrecio() { **return** precio; }
14. **private** ArrayList<Producto> productos;
15. **private** **double** precio;
16. }//clase

**InterfazCliente.java**

1. **import** javax.swing.\*;
2. **import** java.awt.\*;
3. **import** java.awt.event.\*;
5. **public** **class** InterfazCliente **extends** JFrame{
6. **private** **static** **final** **long** serialVersionUID = 1L;
7. **public** InterfazCliente() {
8. setBounds(450, 150, 500, 500);
9. setTitle("Practica 2: Cliente");
10. setResizable(**false**);
12. //Creacion de componentes
13. panelPrincipal = **new** JPanel();
14. panelInferior = **new** JPanel();
15. panelSuperior = **new** JPanel();
16. panelCentral = **new** JPanel();
17. panelCentralCentral = **new** JPanel();
18. botonAnterior = **new** JButton("<<");
19. botonSiguiente = **new** JButton(">>");
20. agregarCarrito = **new** JButton("Agregar al carrito");
21. agregarCarrito.setVisible(**false**);
22. verCarrito = **new** JButton("Ver carrito");
23. verCarrito.setVisible(**false**);
24. pedirCatalogo = **new** JButton("Catalogo");
25. cliente = **new** JLabel("Productos");
26. imagen = **new** JLabel("");
27. nombre = **new** JLabel("");
28. id = **new** JLabel("");
29. precio = **new** JLabel("$");
30. existencia = **new** JLabel("");
31. descripcion = **new** JLabel("");
32. texto = **new** JTextField(4);
33. miCarrito = **new** Carrito();
34. i = 0;
36. //Colocando a la escucha
37. pedirCatalogo.addActionListener(**new** ActionListener() {
38. **public** **void** actionPerformed(ActionEvent e) {
39. pedirCatalogo.setVisible(**false**);
40. verCarrito.setVisible(**true**);
41. agregarCarrito.setVisible(**true**);
42. miCliente = **new** Cliente(PUERTO, HOST);
43. misProductos = miCliente.recibirCatalogo();
44. botonAnterior.setEnabled(**false**);
45. dibujo(misProductos[0]);
46. }
47. });
49. botonSiguiente.addActionListener(**new** ActionListener() {
50. **public** **void** actionPerformed(ActionEvent e) {
51. **if**(contador == 9)
52. botonSiguiente.setEnabled(**false**);
53. **else** {
54. dibujo(misProductos[++contador]);
55. botonSiguiente.setEnabled(**true**);
56. botonAnterior.setEnabled(**true**);
57. }
58. }
59. });
61. botonAnterior.addActionListener(**new** ActionListener() {
62. **public** **void** actionPerformed(ActionEvent e) {
63. **if** (contador > 0) {
64. botonAnterior.setEnabled(**true**);
65. botonSiguiente.setEnabled(**true**);
66. dibujo(misProductos[--contador]);
67. } **else** **if**(contador == 9){
68. botonSiguiente.setEnabled(**false**);
69. } **else** {
70. botonAnterior.setEnabled(**false**);
71. }
72. }
73. });
75. agregarCarrito.addActionListener(**new** ActionListener() {
76. **public** **void** actionPerformed(ActionEvent e) {
77. **if**(texto.getText().equals("")){
78. JOptionPane.showMessageDialog(InterfazCliente.**this**, "Campo Vacio");
79. }
80. **else** {
81. cantidad = Integer.parseInt(texto.getText());
82. **int** ids = Integer.parseInt(id.getText().substring(4));
83. **if**(cantidad > misProductos[ids - 1].getExistencias()) {
84. JOptionPane.showMessageDialog(InterfazCliente.**this**, "No hay suficiente cantidad");
85. } **else** **if**(cantidad <= 0) {
86. JOptionPane.showMessageDialog(InterfazCliente.**this**, "Cantidad No Valida");
87. }
88. **else** {
89. misProductos[ids - 1].setExistencias(misProductos[ids - 1].getExistencias() - cantidad);
90. misProductos[ids - 1].setCantidad(misProductos[ids - 1].getCantidad() + cantidad);
91. //System.out.println(misProductos[ids - 1].getNombre() + " : " + misProductos[ids - 1].getExistencias());
92. existencia.setText("Existencias: " + misProductos[ids - 1].getExistencias());
93. miCarrito.crearCarrito(misProductos[ids - 1], ++i);
94. }
95. texto.setText("");
96. }
97. }
98. });
100. verCarrito.addActionListener(**new** ActionListener(){
101. **public** **void** actionPerformed(ActionEvent arg0) {
102. miCarrito.setVisible(**true**);
103. }
104. });
106. //Propiedades de los componentes
107. panelPrincipal.setLayout(**new** BorderLayout(5, 5));
108. panelCentral.setLayout(**new** BorderLayout(3, 3));
109. panelCentralCentral.setLayout(**new** BoxLayout(**this**.panelCentralCentral, BoxLayout.Y\_AXIS));
111. //Adicion de los componentes
112. panelCentralCentral.add(id);
113. panelCentralCentral.add(nombre);
114. panelCentralCentral.add(precio);
115. panelCentralCentral.add(existencia);
116. panelCentralCentral.add(descripcion);
117. panelCentralCentral.add(**new** JLabel("Cantidad"));
118. panelCentralCentral.add(texto);
119. panelCentral.add(imagen, BorderLayout.NORTH);
120. panelCentral.add(panelCentralCentral, BorderLayout.NORTH);
121. panelInferior.add(pedirCatalogo);
122. panelInferior.add(agregarCarrito);
123. panelInferior.add(verCarrito);
124. panelSuperior.add(cliente);
125. panelPrincipal.add(panelSuperior, BorderLayout.NORTH);
126. panelPrincipal.add(panelCentral, BorderLayout.CENTER);
127. panelPrincipal.add(botonAnterior, BorderLayout.WEST);
128. panelPrincipal.add(botonSiguiente, BorderLayout.EAST);
129. panelPrincipal.add(panelInferior, BorderLayout.SOUTH);
130. add(panelPrincipal);
132. setDefaultCloseOperation(3);
133. setVisible(**true**);
134. }//Constructor
136. **private** **void** dibujo(Producto producto) {
137. id.setText("ID: " + producto.getID());
138. imagen.setText(producto.getImagen());
139. nombre.setText("Nombre: " + producto.getNombre());
140. precio.setText("$" + producto.getPrecio());
141. existencia.setText("Existencias: " + producto.getExistencias());
142. descripcion.setText("Descripcion: " + producto.getDescripcion());
143. }//dibujo
145. **public** **static** **void** main(String[] args) {
146. **new** InterfazCliente();
147. }//main
149. //Bloque de instancias
150. **private** JPanel panelPrincipal;
151. **private** JPanel panelInferior;
152. **private** JPanel panelSuperior;
153. **private** JPanel panelCentral;
154. **private** JPanel panelCentralCentral;
155. **private** JButton botonSiguiente;
156. **private** JButton botonAnterior;
157. **private** JButton agregarCarrito;
158. **private** JButton verCarrito;
159. **private** JButton pedirCatalogo;
160. **private** JLabel cliente;
161. **private** JLabel id;
162. **private** **final** **int** PUERTO = 9999;
163. **private** **final** String HOST = "127.0.0.1";
164. **private** Cliente miCliente;
165. **private** Producto[] misProductos;
166. **private** JLabel imagen, nombre, precio, existencia, descripcion;
167. **private** **int** contador = 0;
168. **private** **int** i = 0;
169. **private** Carrito miCarrito;
170. **private** JTextField texto;
171. **private** **int** cantidad;
172. }//Clase

**InterfazServidor.java**

1. **import** java.awt.\*;
2. **import** java.awt.event.\*;
3. **import** javax.swing.\*;
5. **public** **class** InterfazServidor **extends** JFrame {
6. **private** **static** **final** **long** serialVersionUID = 2L;
8. **public** InterfazServidor() {
9. setBounds(550, 150, 350, 150);
10. setTitle("Practica 2: Servidor");
11. setResizable(**false**);
13. // Creacion de componentes
14. panelPrincipal = **new** JPanel();
15. panelInferior = **new** JPanel();
16. cargarProductos = **new** JButton("Cargar Productos");
17. online = **new** JButton("Online");
18. online.setVisible(**false**);
19. servidor = **new** JLabel("Servidor");
20. misProductos = **new** Producto[10];
21. productos = **new** GestionProductosServidor(misProductos);
23. // Propiedades de los componentes
24. panelPrincipal.setLayout(**new** BorderLayout(5, 5));
26. // Colocando a la escucha
27. cargarProductos.addActionListener(**new** ActionListener() {
28. **public** **void** actionPerformed(ActionEvent e) {
29. misProductos = productos.generarProductos();
30. online.setVisible(**true**);
31. cargarProductos.setVisible(**false**);
33. System.out.println("Productos Cargados");
34. **for**(**int** i = 0; i < misProductos.length; i += 1)
35. System.out.println(misProductos[i].getNombre() +" " +
36. misProductos[i].getExistencias());
37. }
38. });
40. online.addActionListener(**new** ActionListener() {
41. **public** **void** actionPerformed(ActionEvent e) {
42. miServidor = **new** Servidor(PUERTO, misProductos);
43. **try** {
44. miServidor.online();
45. } **catch** (ClassNotFoundException e1) {
46. e1.printStackTrace();
47. }
48. }
49. });
51. //A�adiendo componentes
52. panelInferior.add(cargarProductos);
53. panelInferior.add(online);
54. panelPrincipal.add(servidor, BorderLayout.NORTH);
55. //panelPrincipal.add(new JScrollPane(tablaProductos), BorderLayout.CENTER);
56. panelPrincipal.add(panelInferior, BorderLayout.SOUTH);
57. add(panelPrincipal);
59. setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);
60. setVisible(**true**);
61. }//Constructor
63. **public** **static** **void** main(String[] args) {
64. **new** InterfazServidor();
65. }//main
67. //Bloque de instancias
68. **private** JPanel panelPrincipal;
69. **private** JPanel panelInferior;
70. **private** JButton cargarProductos;
71. **private** JButton online;
72. **private** JLabel servidor;
73. **private** GestionProductosServidor productos;
74. **private** Producto[] misProductos;
75. **private** **final** **int** PUERTO = 9999;
76. **private** Servidor miServidor;
77. }//clase

**InterfazTicket.java**

1. **import** javax.swing.\*;
2. **import** javax.swing.table.\*;
3. **import** java.awt.event.\*;
4. **import** java.util.ArrayList;
5. **import** java.awt.\*;
7. **public** **class** InterfazTicket **extends** JFrame {
8. **private** **static** **final** **long** serialVersionUID = 11L;
9. **public** InterfazTicket() {
10. setTitle("Ticket");
11. setBounds(450, 150, 500, 500);
12. setResizable(**false**);
14. panelPrincipal = **new** JPanel();
15. panelInferior = **new** JPanel();
16. panelCostoFinal = **new** JPanel();
17. aceptar = **new** JButton("Aceptar");
18. tabla = **new** JTable();
19. costo = **new** JLabel("Costo Final: $");
20. modelo = (DefaultTableModel) tabla.getModel();
21. modelo.addColumn("Articulo");
22. modelo.addColumn("Cantidad");
23. modelo.addColumn("Precio");
25. aceptar.addActionListener(**new** ActionListener() {
26. **public** **void** actionPerformed(ActionEvent e) {
27. modelo.setRowCount(0);
28. costo.setText("Costo Final: $");
29. setVisible(**false**);
30. }
31. });
33. panelPrincipal.setLayout(**new** BorderLayout(5, 5));
34. panelCostoFinal.setLayout(**new** BorderLayout(3, 3));
35. panelInferior.add(aceptar);
36. panelCostoFinal.add(**new** JScrollPane(tabla), BorderLayout.CENTER);
37. panelCostoFinal.add(costo, BorderLayout.SOUTH);
38. panelPrincipal.add(panelCostoFinal, BorderLayout.CENTER);
39. panelPrincipal.add(panelInferior, BorderLayout.SOUTH);
40. add(panelPrincipal);
42. setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);
43. }
45. **public** **void** crear(Ticket t) {
46. ArrayList <Producto> miProducto = **new** ArrayList<>();
47. miProducto = t.getProductos();
48. **for**(Producto p : miProducto) {
49. modelo.addRow(**new** Object[]{p.getNombre(), p.getCantidad(), p.getPrecio()});
50. }
52. costo.setText("Costo Final: $" + t.getPrecio());
53. }
55. **private** JPanel panelPrincipal;
56. **private** JPanel panelInferior;
57. **private** JPanel panelCostoFinal;
58. **private** JTable tabla;
59. **private** DefaultTableModel modelo;
60. **private** JButton aceptar;
61. **private** JLabel costo;
62. }

**GestionProductosServidor.java**

1. **public** **class** GestionProductosServidor {
2. **public** GestionProductosServidor(Producto[] misProductos) {
3. **this**.misProductos = misProductos;
4. }
6. **public** Producto[] generarProductos() {
7. **for**(**int** i = 0; i < misProductos.length; i += 1)
8. misProductos[i] = **new** Producto(id[i], nombre[i], precio[i], generarExistencias(), descripcion[i], **false**, imagen[i]);
10. **return** misProductos;
11. }
13. **public** **int** generarExistencias() {
14. **return** (**int**)((Math.random() \* ((100 - 50) + 1)) + 50);
15. }
17. **private** Producto[] misProductos;
18. **private** **int**[] id = {1,2,3,4,5,6,7,8,9,10};
19. **private** String[] nombre = {"Tazas", "Leche", "Audifonos", "Controles", "Juguetes", "Tortillas", "Gorras", "Celulares", "Pan", "Cafe"};
20. **private** **double**[] precio = {25.50, 20.0, 850.0, 450.25, 360.95, 15.25, 200.56, 1500.58, 6.70, 29.99};
21. **private** String[] descripcion = {
22. "Tazas chidas",
23. "Santaclara, la mejor",
24. "Skullcandy ink'd wireless 2.0",
25. "Control universal",
26. "Juguete variado",
27. "Tortillas frescas, no existe duras",
28. "Gorras diversos modelos",
29. "Celulares pa'l kevin",
30. "Pan hecho en casa",
31. "Cafe barato pero bueno"
32. };
33. **private** String[] imagen = {
34. "Tazas chidas",
35. "Leche Santaclara",
36. "Audifonos Skullcandy",
37. "Control universal",
38. "Juguetes diversos",
39. "Tortillas",
40. "Gorras bordadas",
41. "Celulares Chinos",
42. "Pan de casa",
43. "Cafe"
44. };
45. }

**Cliente.java**

1. **import** java.io.\*;
2. **import** java.net.\*;
3. **import** java.util.\*;
5. **public** **class** Cliente {
6. **public** Cliente(**int** puerto, String host) {
7. **this**.puerto = puerto;
8. **this**.host = host;
9. }
11. **public** Producto[] recibirCatalogo() {
12. **try** {
13. cliente = **new** Socket(**this**.host, **this**.puerto);
14. Last l = **new** Last(1, **null**);
15. ObjectOutputStream oos = **new** ObjectOutputStream(cliente.getOutputStream());
16. oos.writeObject(l);
17. oos.flush();
18. ObjectInputStream ois = **new** ObjectInputStream(cliente.getInputStream());
19. misProductos = (Producto[]) ois.readObject();
20. ois.close();
21. oos.close();
23. **for**(Producto p : misProductos)
24. System.out.println(p.getNombre() + "," + p.getExistencias());
26. **return** misProductos;
27. } **catch**(Exception e) {
28. e.printStackTrace();
29. **return** **null**;
30. }//try/catch
31. }//recibirCatalogo
33. **public** **void** hacerCompra(ArrayList<Producto> productosFinal) {
34. **try** {
35. cliente = **new** Socket(**this**.host, **this**.puerto);
36. Last l = **new** Last(2, productosFinal);
37. ObjectOutputStream oos = **new** ObjectOutputStream(cliente.getOutputStream());
39. oos.writeObject(l);
40. oos.flush();
42. ObjectInputStream ois = **new** ObjectInputStream(cliente.getInputStream());
43. Ticket t = (Ticket) ois.readObject();
45. //System.out.println(t.getPrecio());
47. ois.close();
48. oos.close();
49. cliente.close();
51. intTicket = **new** InterfazTicket();
52. intTicket.crear(t);
53. intTicket.setVisible(**true**);
54. } **catch**(Exception e) { e.printStackTrace(); }
55. }//hacerCompra
57. **private** **int** puerto;
58. **private** String host;
59. **private** Producto[] misProductos;
60. **private** Socket cliente;
61. **private** InterfazTicket intTicket;
62. }

**Carrito.java**

1. **import** java.util.\*;
2. **import** javax.swing.\*;
3. **import** java.awt.\*;
4. **import** java.awt.event.\*;
5. **import** javax.swing.table.\*;

8. **public** **class** Carrito **extends** JFrame {
9. **private** **static** **final** **long** serialVersionUID = 4L;
11. **public** Carrito() {
12. setTitle("Carrito");
13. setBounds(450, 150, 500, 500);
14. setResizable(**false**);
16. //Creacion Componentes
17. panelPrincipal = **new** JPanel();
18. panelBotones = **new** JPanel();
19. comprar = **new** JButton("Comprar");
20. regresar = **new** JButton("Regresar");
21. remover = **new** JButton("Remover");
22. costo = **new** JLabel("Costo Total: $");
23. miCarrito = **new** ArrayList<>();
24. tablaProductos = **new** JTable();
25. modelo = (DefaultTableModel) tablaProductos.getModel();
26. modelo.addColumn("Id");
27. modelo.addColumn("Producto");
28. modelo.addColumn("Cantidad");
29. modelo.addColumn("Precio");
31. regresar.addActionListener(**new** ActionListener() {
32. **public** **void** actionPerformed(ActionEvent e) {
33. setVisible(**false**);
34. }
35. });
37. comprar.addActionListener(**new** ActionListener() {
38. **public** **void** actionPerformed(ActionEvent e) {
39. miCliente = **new** Cliente(PUERTO, HOST);
40. miCliente.hacerCompra(miCarrito);
41. miCarrito.clear();
42. modelo.setRowCount(0);
43. costo.setText("Costo Total: $");
44. setVisible(**false**);
45. }
46. });
48. remover.addActionListener(**new** ActionListener() {
49. **public** **void** actionPerformed(ActionEvent e) {
50. **int** idFila = Integer.parseInt(JOptionPane.showInputDialog(panelPrincipal, "Eliminar por ID"));
51. **double** precioRestar = (**double**) modelo.getValueAt(idFila - 1, 3);
52. //System.out.println(precioRestar);
53. costo.setText("Costo Total: $" + (precio - precioRestar));
54. miCarrito.remove(idFila - 1);
55. modelo.removeRow(idFila - 1);
56. }
57. });
59. panelPrincipal.setLayout(**new** BorderLayout(1, 1));
61. panelBotones.add(comprar);
62. panelBotones.add(regresar);
63. panelBotones.add(remover);
64. panelPrincipal.add(costo, BorderLayout.NORTH);
65. panelPrincipal.add(**new** JScrollPane(tablaProductos, JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS, JScrollPane.HORIZONTAL\_SCROLLBAR\_NEVER), BorderLayout.CENTER);
66. panelPrincipal.add(panelBotones, BorderLayout.SOUTH);
67. add(panelPrincipal);
69. setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);
70. setVisible(**false**);
71. }//Constructor
73. **public** **void** crearCarrito(Producto p, **int** i) {
74. modelo.addRow(**new** Object[]{i, p.getNombre(), p.getCantidad(), (p.getCantidad() \* p.getPrecio())});
75. calcularPrecio(p);
76. miCarrito.add(p);
77. }//crearCarrito
79. **private** **void** calcularPrecio(Producto producto) {
80. precio += (producto.getPrecio() \* producto.getCantidad());
81. costo.setText("Costo Total: $" + precio);
82. }

85. **private** JPanel panelPrincipal;
86. **private** JPanel panelBotones;
87. **private** JButton comprar;
88. **private** JButton regresar;
89. **private** JButton remover;
90. **private** JLabel costo;
91. **private** **double** precio = 0;
92. **private** Cliente miCliente;
93. **private** **final** **int** PUERTO = 9999;
94. **private** **final** String HOST = "127.0.0.1";
95. **private** ArrayList<Producto> miCarrito;
96. **private** DefaultTableModel modelo;
97. **private** JTable tablaProductos;
98. }//clase

**Servidor.java**

1. **import** java.io.\*;
2. **import** java.net.\*;
3. **import** java.util.ArrayList;
5. **public** **class** Servidor {
6. **public** Servidor(**int** puerto, Producto[] misProductos) {
7. **this**.puerto = puerto;
8. **this**.misProductos = misProductos;
9. }// Constructor
11. **public** **void** online() **throws** ClassNotFoundException {
12. **try** {
13. servidor = **new** ServerSocket(**this**.puerto);
14. System.out.println("Servidor Activo");
15. **for** (;;) {
16. cliente = servidor.accept();
17. ObjectInputStream ois = **new** ObjectInputStream(cliente.getInputStream());
18. Last l = (Last) ois.readObject();
19. **int** accion = l.getAccion();
20. **if** (accion == 1) {
21. //System.out.println("Mandando catalogo");
22. mandarCatalogo();
23. } **else** **if** (accion == 2) {
24. //System.out.println("Procesando Compra");
25. registrarCompra(l);
26. }
27. ois.close();
28. cliente.close();
29. } // for
30. } **catch** (IOException e) {
31. e.printStackTrace();
32. }
33. }// online
35. **private** **void** mandarCatalogo() {
36. **try** {
37. ObjectOutputStream oos = **new** ObjectOutputStream(cliente.getOutputStream());
38. oos.writeObject(misProductos);
39. oos.flush();
40. oos.close();
41. } **catch** (IOException e) {
42. e.printStackTrace();
43. } // try/catch
44. }// mandarCatalogo
46. **private** **void** registrarCompra(Last l) {
47. ArrayList<Producto> productoRecibido = l.getProducto();
48. /\*for (Producto p : productoRecibido)
49. System.out.println(p.getNombre());\*/
51. actualizarExistencia(productoRecibido);
52. }// registrarCompra
54. **private** **void** actualizarExistencia(ArrayList<Producto> productoRecibido) {
55. **for**(Producto p : productoRecibido) {
56. **if**(p.getID() == 1) {
57. misProductos[0].setExistencias(misProductos[0].getExistencias() - p.getCantidad());
58. } **else** **if**(p.getID() == 2) {
59. misProductos[1].setExistencias(misProductos[1].getExistencias() - p.getCantidad());
60. } **else** **if**(p.getID() == 3) {
61. misProductos[2].setExistencias(misProductos[2].getExistencias() - p.getCantidad());
62. } **else** **if**(p.getID() == 4) {
63. misProductos[3].setExistencias(misProductos[3].getExistencias() - p.getCantidad());
64. } **else** **if**(p.getID() == 5) {
65. misProductos[4].setExistencias(misProductos[4].getExistencias() - p.getCantidad());
66. } **else** **if**(p.getID() == 6) {
67. misProductos[5].setExistencias(misProductos[5].getExistencias() - p.getCantidad());
68. } **else** **if**(p.getID() == 7) {
69. misProductos[6].setExistencias(misProductos[6].getExistencias() - p.getCantidad());
70. } **else** **if**(p.getID() == 8) {
71. misProductos[7].setExistencias(misProductos[7].getExistencias() - p.getCantidad());
72. } **else** **if**(p.getID() == 9) {
73. misProductos[8].setExistencias(misProductos[8].getExistencias() - p.getCantidad());
74. } **else** **if**(p.getID() == 10) {
75. misProductos[9].setExistencias(misProductos[9].getExistencias() - p.getCantidad());
76. }
77. }
79. generarTicket(productoRecibido);
81. System.out.println("Productos Actualizados");
82. **for**(Producto p : misProductos)
83. System.out.println(p.getNombre() + " : " + p.getExistencias());
84. }// actualizarExistencia
86. **private** **void** generarTicket(ArrayList<Producto> productoRecibido) {
87. **double** precio = 0;
88. **for** (Producto p : productoRecibido)
89. precio += p.getPrecio() \* p.getCantidad();
91. Ticket t = **new** Ticket(productoRecibido, precio);
92. **try** {
93. ObjectOutputStream oos = **new** ObjectOutputStream(cliente.getOutputStream());
94. oos.writeObject(t);
95. oos.flush();
96. oos.close();
97. } **catch** (IOException e) {
98. e.printStackTrace();
99. }
100. //System.out.println(t.getPrecio());
101. }//generarTicket
103. **private** **int** puerto;
104. **private** ServerSocket servidor;
105. **private** Socket cliente;
106. **private** Producto[] misProductos;
107. }//Clase

**Dificultades Encontradas.**

Sobre esta práctica la mayor dificultad, fue gestionar y sincronizar las interfaces de acuerdo a los sockets, mientras que la implementación en los sockets fue relativamente sencilla, el único inconveniente era mandar más de un solo objeto, por lo que se creo la clase Last, para poder mandar un objeto dentro de esta y a su vez poder mandar nuestro objeto con distintos descriptores.

**Conclusiones.**

**Luis Enrique Rojas Alvarado.**

En esta práctica pudimos observar en una aplicación más del "uso cotidiano" las diferentes aplicaciones de los sockets y la importancia de ellos, puesto que saber cómo funcionan y qué hacen internamente las aplicaciones de este uso.

Personalmente pienso que los sockets no solo son de uso académico, si no que en realidad se usan en el mundo real para hacer sistemas en red que puedan ayudar a una pequeña empresa a administrar sus productos o cualquier otra cosa. Sin mencionar que parece lo más natural posible.

**Miranda Sandoval Mario Alberto.**

Sobre esta práctica, a diferencia de la anterior pude obtener demasiadas conclusiones, acerca del como funcionan los sockets y además ver como meter objetos dentro de otros objetos para mandarlos dentro de un solo paquete es un caso practico y funcional, solo con la desventaja del uso de más memoria.

El mover interfaces mientras se sincroniza con el servidor, mientras a su vez mantener las validaciones respectivas en el servidor fueron las que conllevaron más tiempo, mientras que la implementación de los sockets fue fácil.