ALUMNO:

DÍAZ MEDINA JESÚS KAIMORTS

ESQUIVEL VALDEZ ALBERTO

VARGAR ROMERO ERICK EFRÁIN

MATERÍA:

REDES COMPUTACIONALES

PROFESOR:

MORENO CERVANTES AXEL ERNESTO

GRUPO:

3CM9

PRACTICA 2

**Introducción**

Las redes actuales utilizan el packet switching para la transferencia de datos. Los datos se envuelven en paquetes que se transfieren desde un origen a un destino, donde se extraen de uno en uno los datos de uno o más paquetes para reconstruir el mensaje original.

Los nodos que se comunican a través de Internet utilizan principalmente dos protocolos:

TCP - Transsmision Control Protocol

UDP - (Universal $\vert$ User) Datagram Protocol

El protocolo UDP - (User $\vert$ Universal) Datagram Protocol - se utiliza para comunicaciones en la que no se garantiza una transmisión fiable (reliable). UDP no está orientado a conexión, por lo tanto no garantiza la entrega. UDP envía paquetes de datos independientes, denominados datagramas, desde una aplicación a otra.

El envío de datagramas es similar a enviar una carta a través del servicio postal: El orden de salida no es importante y no está garantizado, y cada mensaje es independiente de cualquier otro.

**Desarrollo**

Se desarrolló una tienda en línea la cual nos permitiera de parte del servidor:

-Cargar productos con sus datos correspondientes. (Nombre, imagen, costo, existencia, etc.)

Y de parte del cliente nos va permitir:

-Visualizar los productor

-Agregarlos a un carrito de compras

-Realizar las compras.

SERVIDOR:

1. package servidortl;
2. import data.Pedinche;
3. import data.Dato;
4. import java.awt.image.BufferedImage;
5. import java.net.\*;
6. import java.io.\*;
7. import java.util.ArrayList;
8. import java.util.logging.Level;
9. import java.util.logging.Logger;
10. import javax.imageio.ImageIO;
11. public class ServidorTL {
13. public static void main (String[] args){
14. try{
15. int pto=8888;
16. ServerSocket s=new ServerSocket(pto);
17. System.out.println("Servicio iniciado... esperando clientes");
18. for(;;){
19. Socket cl=s.accept();
20. System.out.println("Cliente conectado desde"+cl.getInetAddress()+":"+cl.getLocalPort());
21. ObjectOutputStream oos= new ObjectOutputStream(cl.getOutputStream());
22. ObjectInputStream ois= new ObjectInputStream (cl.getInputStream());
23. Pedinche d1;
24. d1=(Pedinche)ois.readObject();
25. switch (d1.getTipo()) {
26. case 0:
27. {
28. int d2=0;
29. d2=getTotalProductos();
30. oos.writeObject(d2);
31. oos.flush();
32. break;
33. }
34. case 1:
35. {
36. int noProd=d1.getId();
37. Dato d2 = new Dato();
38. d2=getProducto(noProd);
39. oos.writeObject(d2);
40. oos.flush();
41. break;
42. }
43. case 2:
44. ArrayList<Integer> ids=d1.getIds();
45. ArrayList<Dato> data= new ArrayList<Dato>();
46. for (int i:ids) {
48. Dato d2 = new Dato();
49. d2=getProducto(i);
50. data.add(d2);
51. }
52. oos.writeObject(data);
53. oos.flush();
54. break;
55. case 3:
56. int d2=0;
57. d2=getTotalProductos();
58. ArrayList<Dato> dataa= new ArrayList<Dato>();
59. ArrayList<Integer> idss=d1.getIds();
60. for (int i = 1; i <= d2; i++) {
61. Dato d22 = new Dato();
62. d22=getProducto(i);
63. if (idss.contains(d22.getId())) {
64. d22.setExistance(-1);
65. }
66. dataa.add(d22);
67. }
68. updateProducts(dataa);
69. break;
70. default:
71. break;
72. }
74. System.out.println("Objeto enviado");
75. ois.close();
76. oos.close();
77. cl.close();
78. }
80. }catch (Exception e){
81. e.printStackTrace();
82. }
83. }

86. public static boolean updateProducts(ArrayList<Dato> d){
88. try {
89. File archivo = new File("fichero.txt");
90. BufferedWriter bw;
91. bw = new BufferedWriter(new FileWriter(archivo));
92. for (Dato d2:d) {
93. bw.write(d2.getId()+","+d2.getNombre()+","+d2.getDescripcion()+","+d2.getPrecio()+","+d2.getExistencia()+","+
94. + d2.getDescuento()+","+d2.getNameImagenes()+"\n");
95. }
96. bw.close();
97. return true;
99. } catch (IOException ex) {
100. return false;
101. }


105. }
107. public static int getTotalProductos(){
108. String sCadena;
109. int lNumeroLineas = 0;
111. FileReader fr;
112. try {
113. fr = new FileReader("fichero.txt");
114. BufferedReader bf = new BufferedReader(fr);
116. while ((sCadena = bf.readLine())!=null) {
117. lNumeroLineas++;
118. }
119. } catch (Exception ex) {
120. Logger.getLogger(ServidorTL.class.getName()).log(Level.SEVERE, null, ex);
121. }
123. return lNumeroLineas;
124. }
126. public static Dato getProducto(int id){
127. Dato ret= new Dato();
128. String sCadena;
129. FileReader fr;
130. try {
131. fr = new FileReader("fichero.txt");
132. BufferedReader bf = new BufferedReader(fr);
133. while ((sCadena = bf.readLine())!=null) {
134. String obj[]=sCadena.split(",");
135. if (Integer.parseInt(obj[0])==id) {
136. byte[] arrByte;
137. File f=new File(obj[6]);
138. BufferedImage o=ImageIO.read(f);
139. ByteArrayOutputStream b=new ByteArrayOutputStream();
140. ImageIO.write(o, "jpg", b);
141. arrByte=b.toByteArray();
142. ret=new Dato(Integer.parseInt(obj[0]),obj[1],obj[2],Double.parseDouble(obj[3]),Integer.parseInt(obj[4]),Double.parseDouble(obj[5]),(obj[6]),arrByte);
144. }
146. }
147. } catch (Exception ex) {
148. Logger.getLogger(ServidorTL.class.getName()).log(Level.SEVERE, null, ex);
149. }

152. return ret;
153. }
154. }

DATO:

1. package data;
2. import java.io.Serializable;
3. public class Dato implements Serializable {
4. int id;
5. String nombre;
6. String descripcion;
7. double precio;
8. int existencias;
9. double descuento;
10. String nameImagenes;
11. byte[] imagenes;
13. public Dato(int id,String n,String de,double p,int ex,double des,String imn,byte[] im)
14. {
15. this.id=id;
16. this.nombre=n;
17. this.descripcion=de;
18. this.existencias=ex;
19. this.precio=p;
20. this.descuento=des;
21. this.nameImagenes=imn;
22. this.imagenes=im;

25. }
26. public Dato() {
28. }
29. public int getId(){
30. return this.id;
31. }
32. public String getNombre(){
33. return this.nombre;
34. }
35. public String getDescripcion(){
36. return this.descripcion;
37. }
38. public double getPrecio(){
39. return this.precio;
40. }
41. public int getExistencia(){
42. return this.existencias;
43. }
44. public double getDescuento(){
45. return this.descuento;
46. }
47. public String getNameImagenes(){
48. return this.nameImagenes;
49. }
50. public byte [] getImagenes(){
51. return this.imagenes;
52. }
54. public void setExistance(int e){
55. if (this.existencias>0) {
56. this.existencias+=e;
57. }
58. }
59. }

PEDINCHE:

2. package data;
3. import java.io.Serializable;
4. import java.util.ArrayList;
5. public class Pedinche implements Serializable {
6. int tipoMov;
7. int Id;
8. ArrayList<Integer> Ids;
9. public Pedinche(int tipoMov, int Id){
10. this.tipoMov=tipoMov;
11. this.Id=Id;
12. }
13. public Pedinche(int tipoMov, ArrayList<Integer> Ids){
15. this.tipoMov=tipoMov;
16. this.Ids=Ids;
17. }
18. public Pedinche(){
19. }
20. public int getTipo(){
21. return this.tipoMov;
22. }
23. public int getId(){
24. return this.Id;
25. }
27. public ArrayList<Integer> getIds(){
28. return this.Ids;
29. }
30. }

CARRITO:

1. package tiendalinea;
2. import data.Dato;
3. import data.Pedinche;
4. import java.awt.Image;
5. import java.io.ObjectInputStream;
6. import java.io.ObjectOutputStream;
7. import java.net.Socket;
8. import java.util.ArrayList;
9. import java.util.logging.Level;
10. import java.util.logging.Logger;
11. import javax.swing.ImageIcon;
12. import javax.swing.JOptionPane;
13. import static javax.swing.JOptionPane.INFORMATION\_MESSAGE;
14. /\*\*
15. \*
16. \* @author Tempori
17. \*/
18. public class Carrito1 extends javax.swing.JDialog {
19. /\*\*
20. \* Creates new form Carrito1
21. \*/
22. ArrayList<Integer> carrito;
23. boolean isSell=false;
24. int port=8888;
25. String host= "localhost";
26. public Carrito1(java.awt.Frame parent, boolean modal,ArrayList<Integer> lista) {
27. super(parent, modal);
28. this.carrito=lista;
29. initComponents();
30. setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE\_ON\_CLOSE);
31. }
32. /\*\*
33. \* This method is called from within the constructor to initialize the form.
34. \* WARNING: Do NOT modify this code. The content of this method is always
35. \* regenerated by the Form Editor.
36. \*/
37. @SuppressWarnings("unchecked")
38. // <editor-fold defaultstate="collapsed" desc="Generated Code">
39. private void initComponents() {
40. ProductoChido = new javax.swing.JLabel();
41. PrecioChido = new javax.swing.JLabel();
42. jLabel5 = new javax.swing.JLabel();
43. jLabel4 = new javax.swing.JLabel();
44. jLabel2 = new javax.swing.JLabel();
45. jButton2 = new javax.swing.JButton();
46. Total1 = new javax.swing.JLabel();
47. jButton1 = new javax.swing.JButton();
48. jLabel1 = new javax.swing.JLabel();
49. setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE\_ON\_CLOSE);
50. addWindowListener(new java.awt.event.WindowAdapter() {
51. public void windowOpened(java.awt.event.WindowEvent evt) {
52. formWindowOpened(evt);
53. }
54. });
55. getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
56. ProductoChido.setFont(new java.awt.Font("Baskerville Old Face", 0, 18)); // NOI18N
57. ProductoChido.setVerticalAlignment(javax.swing.SwingConstants.TOP);
58. getContentPane().add(ProductoChido, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 60, 250, 190));
59. PrecioChido.setFont(new java.awt.Font("Baskerville Old Face", 0, 18)); // NOI18N
60. PrecioChido.setVerticalAlignment(javax.swing.SwingConstants.TOP);
61. getContentPane().add(PrecioChido, new org.netbeans.lib.awtextra.AbsoluteConstraints(270, 60, 120, 190));
62. jLabel5.setFont(new java.awt.Font("Baskerville Old Face", 1, 18)); // NOI18N
63. jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
64. jLabel5.setText("Precio");
65. jLabel5.setToolTipText("");
66. getContentPane().add(jLabel5, new org.netbeans.lib.awtextra.AbsoluteConstraints(240, 40, 120, -1));
67. jLabel4.setFont(new java.awt.Font("Baskerville Old Face", 1, 18)); // NOI18N
68. jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
69. jLabel4.setText("Producto");
70. jLabel4.setToolTipText("");
71. getContentPane().add(jLabel4, new org.netbeans.lib.awtextra.AbsoluteConstraints(-10, 40, 120, -1));
72. jLabel2.setFont(new java.awt.Font("Baskerville Old Face", 1, 20)); // NOI18N
73. jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
74. jLabel2.setText("Carrito de Compras ");
75. getContentPane().add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(120, 10, -1, -1));
76. jButton2.setBackground(new java.awt.Color(200, 103, 144));
77. jButton2.setForeground(new java.awt.Color(255, 255, 255));
78. jButton2.setText("Comprar");
79. jButton2.addActionListener(new java.awt.event.ActionListener() {
80. public void actionPerformed(java.awt.event.ActionEvent evt) {
81. jButton2ActionPerformed(evt);
82. }
83. });
84. getContentPane().add(jButton2, new org.netbeans.lib.awtextra.AbsoluteConstraints(50, 260, -1, -1));
85. Total1.setFont(new java.awt.Font("Baskerville Old Face", 1, 18)); // NOI18N
86. Total1.setText("Total:");
87. getContentPane().add(Total1, new org.netbeans.lib.awtextra.AbsoluteConstraints(160, 260, -1, -1));
88. jButton1.setBackground(new java.awt.Color(200, 103, 144));
89. jButton1.setForeground(new java.awt.Color(255, 255, 255));
90. jButton1.setText("Cerrar");
91. jButton1.addActionListener(new java.awt.event.ActionListener() {
92. public void actionPerformed(java.awt.event.ActionEvent evt) {
93. jButton1ActionPerformed(evt);
94. }
95. });
96. getContentPane().add(jButton1, new org.netbeans.lib.awtextra.AbsoluteConstraints(300, 260, -1, -1));
97. jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/tiendalinea/Imagenes/fondo12.jpg"))); // NOI18N
98. getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 400, 300));
99. pack();
100. }// </editor-fold>
101. private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
102. // TODO add your handling code here:
103. Pedinche pedinche= new Pedinche(3,carrito);
104. try {
105. Socket cl= new Socket(host,port);
106. ObjectOutputStream oos=new ObjectOutputStream(cl.getOutputStream());
107. ObjectInputStream ois=new ObjectInputStream(cl.getInputStream());
108. oos.writeObject(pedinche);
109. oos.flush();
110. JOptionPane.showMessageDialog(null, "Compra exitosa!!", "", INFORMATION\_MESSAGE);
111. isSell=true;
112. this.setVisible(false);
113. } catch (Exception ex) {
114. JOptionPane.showMessageDialog(null, "Compra fallida!!", "", INFORMATION\_MESSAGE);
115. }
117. }
118. private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
120. this.dispose();
121. }
122. private void formWindowOpened(java.awt.event.WindowEvent evt) {
123. // TODO add your handling code here:
124. Pedinche pedinche= new Pedinche(2,carrito);
125. try {
126. Socket cl= new Socket(host,port);
127. ObjectOutputStream oos=new ObjectOutputStream(cl.getOutputStream());
128. ObjectInputStream ois=new ObjectInputStream(cl.getInputStream());
129. oos.writeObject(pedinche);
130. oos.flush();
131. ArrayList<Dato> d2=(ArrayList<Dato>)ois.readObject();
132. Double total=0.0;
133. ProductoChido.setText("<html>");
134. PrecioChido.setText("<html>");
135. for (Dato i:d2) {
136. ProductoChido.setText(ProductoChido.getText()+i.getNombre()+"<br>");
137. PrecioChido.setText(PrecioChido.getText()+i.getPrecio()+"<br>");
138. total+=i.getPrecio();
139. }
140. ProductoChido.setText(ProductoChido.getText()+"</html>");
141. PrecioChido.setText(PrecioChido.getText()+"</html>");
142. Total1.setText("Total: "+total);
143. } catch (Exception ex) {
144. Logger.getLogger(modelos.class.getName()).log(Level.SEVERE, null, ex);
145. }
146. }
148. public boolean isSell(){
149. return isSell;
150. }
151. /\*\*
152. \* @param args the command line arguments
153. \*/
154. public static void main(String args[]) {
155. /\* Set the Nimbus look and feel \*/
156. //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
157. /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
158. \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
159. \*/
160. try {
161. for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
162. if ("Nimbus".equals(info.getName())) {
163. javax.swing.UIManager.setLookAndFeel(info.getClassName());
164. break;
165. }
166. }
167. } catch (ClassNotFoundException ex) {
168. java.util.logging.Logger.getLogger(Carrito1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
169. } catch (InstantiationException ex) {
170. java.util.logging.Logger.getLogger(Carrito1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
171. } catch (IllegalAccessException ex) {
172. java.util.logging.Logger.getLogger(Carrito1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
173. } catch (javax.swing.UnsupportedLookAndFeelException ex) {
174. java.util.logging.Logger.getLogger(Carrito1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
175. }
176. //</editor-fold>
177. /\* Create and display the dialog \*/
178. java.awt.EventQueue.invokeLater(new Runnable() {
179. public void run() {
180. Carrito1 dialog = new Carrito1(new javax.swing.JFrame(), true,null);
181. dialog.addWindowListener(new java.awt.event.WindowAdapter() {
182. @Override
183. public void windowClosing(java.awt.event.WindowEvent e) {
184. System.exit(0);
185. }
186. });
187. dialog.setVisible(true);
188. }
189. });
190. }
191. // Variables declaration - do not modify
192. private javax.swing.JLabel PrecioChido;
193. private javax.swing.JLabel ProductoChido;
194. private javax.swing.JLabel Total1;
195. private javax.swing.JButton jButton1;
196. private javax.swing.JButton jButton2;
197. private javax.swing.JLabel jLabel1;
198. private javax.swing.JLabel jLabel2;
199. private javax.swing.JLabel jLabel4;
200. private javax.swing.JLabel jLabel5;
201. // End of variables declaration
202. }

MODELOS:

2. package tiendalinea;
3. import data.Dato;
4. import data.Pedinche;
5. import java.awt.Image;
6. import java.io.IOException;
7. import java.io.ObjectInputStream;
8. import java.io.ObjectOutputStream;
9. import java.net.Socket;
10. import java.util.ArrayList;
11. import java.util.logging.Level;
12. import java.util.logging.Logger;
13. import javax.swing.ImageIcon;
14. /\*\*
15. \*
16. \* @author Yesenia
17. \*/
18. public class modelos extends javax.swing.JFrame {
20. public modelos() {
21. initComponents();
22. }
23. int pronNo=1;
24. ArrayList<Integer> carrito= new ArrayList<>();
25. int port=8888;
26. String host= "localhost";
28. @SuppressWarnings("unchecked")
29. // <editor-fold defaultstate="collapsed" desc="Generated Code">
30. private void initComponents() {
31. jLabel1 = new javax.swing.JLabel();
32. jButton1 = new javax.swing.JButton();
33. jButton2 = new javax.swing.JButton();
34. jButton3 = new javax.swing.JButton();
35. jLabel3 = new javax.swing.JLabel();
36. jCheckBox1 = new javax.swing.JCheckBox();
37. jLabel4 = new javax.swing.JLabel();
38. jLabel5 = new javax.swing.JLabel();
39. jLabel7 = new javax.swing.JLabel();
40. jLabel2 = new javax.swing.JLabel();
41. setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);
42. addWindowListener(new java.awt.event.WindowAdapter() {
43. public void windowOpened(java.awt.event.WindowEvent evt) {
44. formWindowOpened(evt);
45. }
46. });
47. getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
48. jLabel1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(255, 51, 102), 2, true));
49. getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(19, 40, 210, 160));
50. jButton1.setBackground(new java.awt.Color(200, 103, 144));
51. jButton1.setForeground(new java.awt.Color(255, 255, 255));
52. jButton1.setText("Anterior");
53. jButton1.addActionListener(new java.awt.event.ActionListener() {
54. public void actionPerformed(java.awt.event.ActionEvent evt) {
55. jButton1ActionPerformed(evt);
56. }
57. });
58. getContentPane().add(jButton1, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 210, 90, 30));
59. jButton2.setBackground(new java.awt.Color(200, 103, 144));
60. jButton2.setForeground(new java.awt.Color(255, 255, 255));
61. jButton2.setText("Siguiente");
62. jButton2.addActionListener(new java.awt.event.ActionListener() {
63. public void actionPerformed(java.awt.event.ActionEvent evt) {
64. jButton2ActionPerformed(evt);
65. }
66. });
67. getContentPane().add(jButton2, new org.netbeans.lib.awtextra.AbsoluteConstraints(140, 210, 90, 30));
68. jButton3.setBackground(new java.awt.Color(200, 103, 144));
69. jButton3.setForeground(new java.awt.Color(255, 255, 255));
70. jButton3.setText("Ver Carrito");
71. jButton3.addActionListener(new java.awt.event.ActionListener() {
72. public void actionPerformed(java.awt.event.ActionEvent evt) {
73. jButton3ActionPerformed(evt);
74. }
75. });
76. getContentPane().add(jButton3, new org.netbeans.lib.awtextra.AbsoluteConstraints(280, 280, 100, 30));
77. jLabel3.setBackground(new java.awt.Color(255, 204, 204));
78. jLabel3.setFont(new java.awt.Font("Baskerville Old Face", 0, 18)); // NOI18N
79. jLabel3.setForeground(new java.awt.Color(51, 51, 51));
80. jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
81. jLabel3.setText("Producto");
82. getContentPane().add(jLabel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 20, 400, -1));
83. jCheckBox1.setBackground(new java.awt.Color(204, 204, 255));
84. jCheckBox1.setForeground(new java.awt.Color(255, 51, 153));
85. jCheckBox1.setText("Seleccionar");
86. jCheckBox1.addActionListener(new java.awt.event.ActionListener() {
87. public void actionPerformed(java.awt.event.ActionEvent evt) {
88. jCheckBox1ActionPerformed(evt);
89. }
90. });
91. getContentPane().add(jCheckBox1, new org.netbeans.lib.awtextra.AbsoluteConstraints(250, 200, 120, 50));
92. jLabel4.setFont(new java.awt.Font("Baskerville Old Face", 0, 16)); // NOI18N
93. jLabel4.setText("Descripcion:");
94. jLabel4.setVerticalAlignment(javax.swing.SwingConstants.TOP);
95. getContentPane().add(jLabel4, new org.netbeans.lib.awtextra.AbsoluteConstraints(240, 60, 170, 140));
96. jLabel5.setFont(new java.awt.Font("Baskerville Old Face", 1, 24)); // NOI18N
97. jLabel5.setForeground(new java.awt.Color(0, 51, 51));
98. jLabel5.setText("PRECIO:");
99. getContentPane().add(jLabel5, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 280, 230, 40));
100. jLabel7.setFont(new java.awt.Font("Baskerville Old Face", 1, 24)); // NOI18N
101. jLabel7.setForeground(new java.awt.Color(0, 51, 51));
102. jLabel7.setText("EXISTENCIA: ");
103. getContentPane().add(jLabel7, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 320, 230, 40));
104. jLabel2.setBackground(new java.awt.Color(51, 102, 255));
105. jLabel2.setForeground(new java.awt.Color(0, 153, 204));
106. jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/tiendalinea/Imagenes/fondo12.jpg"))); // NOI18N
107. jLabel2.setText("v");
108. getContentPane().add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 420, 380));
109. pack();
110. }// </editor-fold>
111. private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
113. if(pronNo!=1){
114. pronNo--;
115. actualizarVentana();
116. }
118. }
119. private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
121. if(pronNo!=obtenerTotalProducto()){
122. pronNo++;
123. actualizarVentana();
124. }
125. }
126. private void jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt) {
127. if(jCheckBox1.isSelected()){
128. carrito.add(pronNo);
129. }
130. else{
131. if (!carrito.isEmpty()) {
132. carrito.remove(Integer.valueOf(pronNo));
133. }

136. }
137. }
138. private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
139. Carrito1 verCarrito= new Carrito1(this,true,carrito);
140. verCarrito.setVisible(true);
141. System.out.println(verCarrito.isSell()+"");
142. if (verCarrito.isSell()) {
143. verCarrito.dispose();
144. carrito=new ArrayList<Integer>();
145. pronNo=1;
146. actualizarVentana();
148. }
149. }
150. private void formWindowOpened(java.awt.event.WindowEvent evt) {
151. // TODO add your handling code here:
152. actualizarVentana();
153. }
155. private int obtenerTotalProducto(){
156. int total=0;
157. Pedinche pedinche= new Pedinche(0,0);
158. try {
159. Socket cl= new Socket(host,port);
160. ObjectOutputStream oos=new ObjectOutputStream(cl.getOutputStream());
161. ObjectInputStream ois=new ObjectInputStream(cl.getInputStream());
162. oos.writeObject(pedinche);
163. oos.flush();
164. total=(int)ois.readObject();
166. } catch (Exception ex) {
167. Logger.getLogger(modelos.class.getName()).log(Level.SEVERE, null, ex);
168. }
169. return total;
171. }
173. private void actualizarVentana(){
175. Pedinche pedinche= new Pedinche(1,pronNo);
176. try {
177. Socket cl= new Socket(host,port);
178. ObjectOutputStream oos=new ObjectOutputStream(cl.getOutputStream());
179. ObjectInputStream ois=new ObjectInputStream(cl.getInputStream());
180. oos.writeObject(pedinche);
181. oos.flush();
182. Dato d2=(Dato)ois.readObject();
183. jLabel4.setText("<html>Descripcion:<br><br>"+d2.getDescripcion()+"</html>");
184. jLabel5.setText("Precio: "+d2.getPrecio());
185. jLabel3.setText("Producto: "+d2.getNombre());
186. jCheckBox1.setEnabled(true);
187. jLabel7.setText("Existencia: "+d2.getExistencia());
188. if (d2.getExistencia()==0) {
189. jLabel7.setText("AGOTADO");
190. jCheckBox1.setEnabled(false);
191. }
192. ImageIcon imageIcon = new ImageIcon(new ImageIcon(d2.getImagenes()).getImage().getScaledInstance(jLabel1.getWidth(), jLabel1.getHeight(), Image.SCALE\_DEFAULT));
193. jLabel1.setIcon(imageIcon);
195. if (carrito.contains(pronNo)) {
196. jCheckBox1.setSelected(true);
197. }else{
198. jCheckBox1.setSelected(false);
199. }
200. } catch (Exception ex) {
201. Logger.getLogger(modelos.class.getName()).log(Level.SEVERE, null, ex);
202. }
203. }
204. public static void main(String args[]) {
206. try {
207. for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
208. if ("Nimbus".equals(info.getName())) {
209. javax.swing.UIManager.setLookAndFeel(info.getClassName());
210. break;
211. }
212. }
213. } catch (ClassNotFoundException ex) {
214. java.util.logging.Logger.getLogger(modelos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
215. } catch (InstantiationException ex) {
216. java.util.logging.Logger.getLogger(modelos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
217. } catch (IllegalAccessException ex) {
218. java.util.logging.Logger.getLogger(modelos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
219. } catch (javax.swing.UnsupportedLookAndFeelException ex) {
220. java.util.logging.Logger.getLogger(modelos.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
221. }
222. //</editor-fold>
223. /\* Create and display the form \*/
224. java.awt.EventQueue.invokeLater(new Runnable() {
225. public void run() {
226. new modelos().setVisible(true);
228. }
229. });
230. }
231. // Variables declaration - do not modify
232. private javax.swing.JButton jButton1;
233. private javax.swing.JButton jButton2;
234. private javax.swing.JButton jButton3;
235. private javax.swing.JCheckBox jCheckBox1;
236. private javax.swing.JLabel jLabel1;
237. private javax.swing.JLabel jLabel2;
238. private javax.swing.JLabel jLabel3;
239. private javax.swing.JLabel jLabel4;
240. private javax.swing.JLabel jLabel5;
241. private javax.swing.JLabel jLabel7;
242. // End of variables declaration
243. }

La aplicación nos muestra la siguiente pantalla en donde se muestra las siguientes pantallas:

|  |
| --- |
| FOTO 1 |
| FOTO 2 |
| FOTO 3 |
| FOTO 4 |
| FOTO 5 |
| FOTO 6 |
| FOTO 7 |

Conclusiones:

En esta práctica se hizo uso de Sockets Datagramas, que es una herramienta que nos permite mandar a distintos cliente la misma información. La práctica fue dificl puesto que el enviar las imágenes o los emoticonos tenía que ser de manera personalizada. De igual forma los emoticonos se tuvieron que mandar como imágenes para no presentar otras complicaciones extra. La práctica nos ayudo a comprender un poco más como funcionan las aplicaciones de mensajería en grupo y sin duda fue muy útil puesto que dichas herramientas son muy utilizadas en la actualidad.