**COMSATS UNIVERSITY ISLAMABAD**

**COURSE: OOP (OBJECT ORIENTED PROGRAMING)**

**OOP PROJECT SCOPE DOCUMENT.**

***SUBMITTED TO:* Ma’am. SANEEHA AMIR.**

***SUBMITTED BY*: NIMRA ZAFAR (FA19-BCS-062-3B)**

**SHAMEEN SHAHID (FA19-BCS-074-3B).**

**THIRD SEMMESTER IN BSCS**



**DATE: 30TH December, 2020.**

***Daraz clone:***

***Introduction to the project:***

This project is the clone of the Daraz app. Daraz is one of the leading website and apps available on the play store for online buying and selling.

In this project, like Daraz the customers are available with features to update their information and can shop according to their choices. The choices provided are Home appliances, Dresses, Sports and Cosmetics. The order will be delivered in time. Furthermore, to use this functionality the customers are first required to register themselves and then they can use the app.

The administration is provided with their email and password who can add, update ,delete or search their product according to their choices.

On invalid entry of information either customer or admin, a message will be displayed of invalid access. Both are required to follow the formats of information required by the app

***Code of Admin class:.***

1. package oop\_project;
2. import java.io.FileInputStream;
3. import java.io.FileOutputStream;
4. import java.io.IOException;
5. import java.io.ObjectInputStream;
6. import java.io.ObjectOutputStream;
7. import java.io.Serializable;
8. import java.util.ArrayList;
9. import java.util.Objects;
11. public class Admin implements Serializable
12. {
14. static String email="admin1567@daraz.pk";
15. static String pswd="Addprod123\_class";
16. Dresses d;
17. Cosmetics c;
18. Sports s;
19. HomeAppliances a;
20. public Admin(String email, String pswd) {
21. this.email = email;
22. this.pswd = pswd;
23. }
25. public Admin() {
26. }
27. Admin( Admin ad)
28. {
29. this.email=ad.email;
30. this.pswd=ad.pswd;
31. }
33. public String getEmail() {
34. return this.email;
35. }
37. public String getPswd() {
38. return this.pswd;
39. }
41. @Override
42. public int hashCode() {
43. int hash = 5;
44. return hash;
45. }

48. public void setEmail(String email) {
49. this.email = email;
50. }
52. public void setPswd(String pswd) {
53. this.pswd = pswd;
54. }
55. public static boolean search(String e, String p)
56. {
57. if(email.equals(e) && pswd.equals(p))
58. {
59. return true;
60. }
61. else
62. {
63. return false;
64. }
65. }
67. public static void writeToFile( Admin c )
68. {
69. ArrayList <Admin> list = readobj();
70. list.add(c);
71. try
72. {
73. ObjectOutputStream out = new ObjectOutputStream( new FileOutputStream("Customer Record"));
74. out.writeObject(list);
75. }
76. catch (IOException e)
77. {
78. }
79. }
81. public static ArrayList<Admin> readobj()
82. {
83. ArrayList<Admin> list = new ArrayList<Admin>();
84. try
85. {
86. ObjectInputStream in = new ObjectInputStream (new FileInputStream("Admin Record"));
87. list = (ArrayList<Admin>) in.readObject();
88. }
89. catch(ClassNotFoundException c)
90. {
92. }
93. catch (IOException e)
94. {
95. }
97. return list;
98. }
100. }

***Code of Admin class GUI:.***

1. package oop\_project;
3. import java.awt.BorderLayout;
4. import java.awt.GridLayout;
5. import java.awt.event.ActionEvent;
6. import java.awt.event.ActionListener;
7. import javax.swing.JButton;
8. import javax.swing.JFrame;
9. import javax.swing.JLabel;
10. import javax.swing.JOptionPane;
11. import javax.swing.JPanel;
12. import javax.swing.JTextField;
14. public class ADMIN\_MENU extends JFrame
15. {
16. JButton b1,b2;
17. JLabel l1,l2;
18. JTextField t1,t2;
19. ADMIN\_MENU()
20. {
21. setSize(400,500);
23. setLayout(new BorderLayout());
24. b1= new JButton("SUBMIT");
25. b2= new JButton("HOME");
26. l1= new JLabel("ENTER YOUR E-MAIL");
27. l2= new JLabel("ENTER YOUR PASSWORD");
28. t1= new JTextField(20);
29. t2= new JTextField(20);
30. JPanel p= new JPanel();
31. p.add(l1);
32. p.add(t1);
33. p.add(l2);
34. p.add(t2);
35. p.setLayout(new GridLayout(2,2));
36. add(p,BorderLayout.NORTH);
37. MyActionListener a = new MyActionListener();
38. JPanel p1= new JPanel();
39. p1.add(b1);
40. p1.add(b2);
41. add(p1,BorderLayout.SOUTH);
42. b1.addActionListener(a);
43. b2.addActionListener(a);
45. setVisible(true);
47. }
49. public class MyActionListener implements ActionListener
50. {
51. public void actionPerformed(ActionEvent ae)
52. {
53. if(ae.getActionCommand()=="SUBMIT")
54. {
55. if(t1.getText().isEmpty()||t2.getText().isEmpty())
56. {
57. JOptionPane.showMessageDialog(null, "Please fill all the Text Fields");
58. }
59. else
60. {
61. if(t1.getText().contains("@") && t1.getText().endsWith(".pk"))
62. {
63. if(Admin.search(t1.getText(), t2.getText()))
64. {
65. dispose();
66. JOptionPane.showMessageDialog(null, "VALID ACCESS");
67. Product o= new Product();
68. }
69. else
70. {
71. JOptionPane.showMessageDialog(null, " THAT ID NOT FOUND");
72. dispose();
73. SIGN\_IN c= new SIGN\_IN();
74. }
75. }
76. else
77. {
78. JOptionPane.showMessageDialog(null, " Invalid Email");
79. dispose();
80. SIGN\_IN c= new SIGN\_IN();
81. }
82. }
83. }
84. if(ae.getActionCommand()=="HOME")
85. {
86. dispose();
87. SIGN\_IN c= new SIGN\_IN();
88. }
89. }
90. }
92. }

***CODE for Customer:***

1. package oop\_project;
3. import java.io.Serializable;
4. import java.util.Scanner;
5. import java.io.FileInputStream;
6. import java.io.FileOutputStream;
7. import java.io.IOException;
8. import java.io.ObjectInputStream;
9. import java.io.ObjectOutputStream;
10. import java.util.ArrayList;
12. public class Customer implements Serializable
13. {
14. static Scanner input=new Scanner(System.in);
15. protected String name,contact,email,password;
16. Address ad;
18. public Customer(String name, String contact,String email, String password, Address ad) {
19. this.name = name;
20. this.contact = contact;
21. this.email=email;
22. this.password = password;
23. this.ad = new Address(ad);
24. }
26. public Customer() {
27. }
29. public String getName() {
30. return name;
31. }
33. public String getContact() {
34. return contact;
35. }
37. public String getEmail() {
38. return this.email;
39. }
41. public String getPassword() {
42. return this.password;
43. }

46. public Address getAd() {
47. return ad;
48. }
50. public void setName(String name) {
51. this.name = name;
52. }
54. public void setContact(String contact) {
55. this.contact = contact;
56. }
58. public void setEmail(String email) {
59. this.email = email;
60. }
62. public void setPassword(String password) {
63. this.password = password;
64. }


68. public void setAd(Address ad)
69. {
70. this.ad=new Address(ad);
71. }
72. public void display()
73. {
74. System.out.println("The name is " + name);
75. System.out.println("The email is " + email);
76. System.out.println("The password is " + password);
77. System.out.println("The password is " + contact);
78. System.out.println("The addresss is " + ad.getHomeno() + ad.getStreetno() +ad.getCity());
79. }
80. public static boolean searchcust(String n,String p)
81. {
82. ArrayList<Customer> list=readobj();
83. boolean search=false;
84. for(int i=0;i<list.size();i++)
85. {
86. if(list.get(i).getEmail().equals(n) && list.get(i).getPassword().equals(p))
87. {
88. search=true;
89. break;
90. }
91. else
92. search=false;
94. }
95. return search;
97. }
99. public static boolean chekemail(String e)
100. {
101. boolean esearch=true;
102. ArrayList<Customer> list=readobj();
103. for(int i=0;i<list.size();i++)
104. {
105. if(!(list.get(i).getEmail().toLowerCase().equals(e.toLowerCase())))
106. {
107. esearch=true;
108. }
109. else
110. {
111. esearch=false;
112. break;
114. }
116. }
117. return esearch;

120. }
121. public static boolean checkname(String n)
122. {
123. boolean found=false;
124. char cust\_check=0;
125. for(int i=0;i<n.length();i++)
126. {
127. cust\_check=n.charAt(i);
128. if(Character.isLetter(cust\_check))
129. {
130. found=true;
131. }
132. else
133. {
134. found=false;
135. break;
136. }
137. }
138. return found;
139. }
140. public static void writeToFile( Customer c )
141. {
142. ArrayList <Customer> list = readobj();
143. list.add(c);
144. try
145. {
146. ObjectOutputStream out = new ObjectOutputStream( new FileOutputStream("Customer Record"));
147. out.writeObject(list);
148. }
149. catch (IOException e)
150. {
151. }
152. }
153. public static ArrayList<Customer> readobj()
154. {
155. ArrayList<Customer> list = new ArrayList<Customer>();
156. try
157. {
158. ObjectInputStream in = new ObjectInputStream (new FileInputStream("Customer Record"));
159. list = (ArrayList<Customer>) in.readObject();
160. }
161. catch(ClassNotFoundException c)
162. {
164. }
165. catch (IOException e)
166. {
167. }
169. return list;
170. }
171. public static void displayAllRecords()
172. {
173. ArrayList <Customer> list = readobj();
174. for( Customer c:list)
175. {
176. System.out.println("\nThe name is "+c.getName()+"\nThe contact number is "+c.getContact()+"\nThe email is " + c.getEmail() +"\nthe password is " + c.getPassword()+"\nthe address is" +c.getAd());
177. }
178. }
179. public static void update\_password(String n,String pass)
180. {
181. ArrayList<Customer> list=readobj();
182. boolean id\_check=true;
184. if(id\_check==true)
185. {
186. for(int i=0;i<list.size();i++)
187. {
188. if(list.get(i).getEmail().equals(n))
189. {
190. list.get(i).setPassword(pass);
191. break;
192. }
193. }
194. }
195. try
196. {
197. ObjectOutputStream Output=new ObjectOutputStream(new FileOutputStream("Customer Record"));
198. Output.writeObject(list);
199. }
200. catch (IOException e)
201. {
202. System.out.println("Error");
203. }
205. }
206. public static boolean check\_contact(String c)
207. {
208. boolean check\_validation=false;
209. String format = ("\\d{3}-\\d{7}");
210. String format\_2 = ("\\d{4}-\\d{7}");
211. if (c.matches(format) || c.matches(format\_2))
212. {
213. check\_validation= true;
214. }
215. else
216. {
217. check\_validation = false;
218. }
219. return check\_validation;
220. }
221. public static void update\_Address(String n,Address a)
222. {
223. ArrayList<Customer> list=readobj();
224. boolean id\_check=true;
226. if(id\_check==true)
227. {
228. Address a2=new Address(a);
229. for(int i=0;i<list.size();i++)
230. {
231. if(list.get(i).getEmail().equals(n))
232. {
233. list.get(i).setAd(a2);
234. break;
235. }
236. }
237. }
239. try
240. {
241. ObjectOutputStream Output=new ObjectOutputStream(new FileOutputStream("Customer Record"));
242. Output.writeObject(list);
243. }
244. catch (IOException e)
245. {
246. System.out.println("Error");
247. }
249. }
250. public static void update\_cont(String n,String con)
251. {
252. ArrayList<Customer> list=readobj();
253. boolean id\_check=true;
254. if(id\_check==true)
255. {
256. for(int i=0;i<list.size();i++)
257. {
258. if(list.get(i).getEmail().equals(n))
259. {
260. list.get(i).setContact(con);
261. break;
262. }
263. }
264. }
265. try
266. {
267. ObjectOutputStream Output=new ObjectOutputStream(new FileOutputStream("Customer Record"));
268. Output.writeObject(list);
269. }
270. catch (IOException e)
271. {
272. System.out.println("Error");
273. }
275. }
276. public static void deletecustomer(String n,String p)
277. {
278. ArrayList <Customer> list = readobj();
279. int index =0;
280. for (int i =0; i < list.size() ; i++)
281. {
282. if (list.get(i).getPassword().toLowerCase()==p.toLowerCase() && list.get(i).getEmail().toLowerCase()==n.toLowerCase())
283. {
284. //System.out.println("VALUE FOUND NOW DELETING IT");
285. index = i;
286. break;
287. }
288. }
289. if(list.size()!=0)
290. {
291. System.out.println("RECORD DELETED");
292. list.remove(index);
294. }
295. else
296. {
297. System.out.println("NO RECORD FOUND ");
299. }
300. try
301. {
302. ObjectOutputStream in=new ObjectOutputStream(new FileOutputStream("Customer Record"));
303. in.writeObject(list);
304. }
305. catch(IOException e)
306. {
307. System.out.println("Exception Generated");
308. }
309. }


313. }

***Code For Customer GUI:***

1. package oop\_project;
3. import java.awt.BorderLayout;
4. import java.awt.GridLayout;
5. import java.awt.event.ActionEvent;
6. import java.awt.event.ActionListener;
7. import javax.swing.JButton;
8. import javax.swing.JFrame;
9. import javax.swing.JLabel;
10. import javax.swing.JOptionPane;
11. import javax.swing.JPanel;
12. import javax.swing.JTextField;
14. public class CUSTOMER\_MENU extends JFrame
15. {
16. JButton b1,b2;
17. JLabel l1,l2;
18. JTextField t1,t2;
19. CUSTOMER\_MENU()
20. {
21. setSize(400,500);
23. setLayout(new BorderLayout());
24. b1= new JButton("SUBMIT");
25. b2= new JButton("HOME");
26. l1= new JLabel("ENTER YOUR Email");
27. l2= new JLabel("ENTER YOUR Password");
28. t1= new JTextField(20);
29. t2= new JTextField(20);
30. JPanel p= new JPanel();
31. p.add(l1);
32. p.add(t1);
33. p.add(l2);
34. p.add(t2);
35. p.setLayout(new GridLayout(2,2));
36. add(p,BorderLayout.NORTH);
37. MyActionListener a = new MyActionListener();
38. JPanel p1= new JPanel();
39. p1.add(b1);
40. p1.add(b2);
41. add(p1,BorderLayout.SOUTH);
42. b1.addActionListener(a);
43. b2.addActionListener(a);
44. setVisible(true);
46. }
47. public class MyActionListener implements ActionListener
48. {
49. @Override
50. public void actionPerformed(ActionEvent ae)
51. {
52. if(ae.getActionCommand()=="SUBMIT")
53. {
54. if(t1.getText().isEmpty()||t2.getText().isEmpty())
55. {
56. JOptionPane.showMessageDialog(null, "Please fill all the Text Fields");
57. }
58. else
59. {
60. if(t1.getText().contains("@") && t1.getText().endsWith(".com"))
61. {
62. if(Customer.searchcust(t1.getText(), t2.getText()))
63. {
64. dispose();
65. JOptionPane.showMessageDialog(null, " Vlid access");
66. Myaccount m=new Myaccount();
67. }
68. else
69. {
70. JOptionPane.showMessageDialog(null, " THAT ID NOT FOUND");
71. dispose();
72. SIGN\_IN c= new SIGN\_IN();
73. }
74. }
75. else
76. {
77. JOptionPane.showMessageDialog(null, " Invalid Email Address");
78. dispose();
79. SIGN\_IN c= new SIGN\_IN();
80. }
81. }
82. }
83. if(ae.getActionCommand()=="HOME")
84. {
85. dispose();
86. SIGN\_IN c= new SIGN\_IN();
87. }
88. }
89. }
91. }

***Code For Add to Cart:***

1. package oop\_project;
2. import java.io.Serializable;
3. import java.util.ArrayList;
4. import java.util.Scanner;
6. public class AddToCart
7. {
8. String customer\_name;
9. static double bill;
10. String quant;
11. order o;
12. AddToCart()
13. {
14. o= new order();
15. }
16. public boolean orderDress(String code, String q)
17. {
18. boolean found=Dresses.searchDataFromFile(code);
19. int quant=0;
20. String price;
21. int index=0;
22. boolean check=false;
23. int q1=Integer.parseInt(q);
24. bill=0;
25. Scanner input= new Scanner(System.in);
26. if(found==true)
27. {
28. ArrayList<Dresses> list= Dresses.readAllDataFromFile();
29. for(int i=0;i<list.size();i++)
30. {
32. if(list.get(i).getCode().toLowerCase().equals(code.toLowerCase()))
33. {
34. System.out.println("ITEM FOUND IN THE RECORD");
35. quant=Integer.parseInt(list.get(i).getQuantity());
36. price=list.get(i).getPrice();
37. index=i;
38. }
39. }
40. if(q1>quant)
41. {
42. System.out.println("ITEM OUT OF STROKE");
43. check=false;
44. return check;
45. }
46. else
47. {
48. String k=Integer.toString(quant-q1);
49. list.get(index).setQuantity(k);
50. price=list.get(index).getPrice();
51. bill=q1\*Integer.parseInt(price);
52. o.add\_order(bill);
53. check=true;
54. }
55. }
56. return check;
57. }
59. public boolean orderSports(String code, String q)
60. {
61. boolean found=Sports.searchDataFromFile(code);
62. int quant=0;
63. String price;
64. int index=0;
65. int q1= Integer.parseInt(q);
66. bill=0;
67. boolean check=false;
68. Scanner input= new Scanner(System.in);
69. if(found==true)
70. {
71. ArrayList<Sports> list= Sports.readAllDataFromFile();
72. for(int i=0;i<list.size();i++)
73. {
75. if(list.get(i).getCode().toLowerCase().equals(code.toLowerCase()))
76. {
77. System.out.println("ITEM FOUND IN THE RECORD");
78. quant=Integer.parseInt(list.get(i).getQuantity());
79. price=list.get(i).getPrice();
80. index=i;
81. }
82. }
83. if(q1>quant)
84. {
85. System.out.println("ITEM OUT OF STROKE");
86. check=false;
87. return check;
88. }
89. else
90. {
91. String k=Integer.toString(quant-q1);
92. list.get(index).setQuantity(k);
93. price=list.get(index).getPrice();
94. bill=q1\*Integer.parseInt(price);
95. check=true;
96. }
97. }
98. o.add\_order(bill);
99. return check;
100. }
102. public boolean orderCosmetics(String code, String q)
103. {
104. boolean found=Cosmetics.searchDataFromFile(code);
105. int quant=0;
106. String price;
107. int index=0;
108. boolean check=false;
109. int q1=Integer.parseInt(q);
110. bill=0;
111. Scanner input= new Scanner(System.in);
112. if(found==true)
113. {
114. ArrayList<Cosmetics> list= Cosmetics.readAllDataFromFile();
115. for(int i=0;i<list.size();i++)
116. {
118. if(list.get(i).getCode().toLowerCase().equals(code.toLowerCase()))
119. {
120. System.out.println("ITEM FOUND IN THE RECORD");
121. quant=Integer.parseInt(list.get(i).getQuantity());
122. price=list.get(i).getPrice();
123. index=i;
124. }
125. }
126. if(q1>quant)
127. {
128. System.out.println("ITEM OUT OF STROKE");
129. check=false;
130. return check;
131. }
133. else
134. {
135. String k=Integer.toString(quant-q1);
136. list.get(index).setQuantity(k);
137. price=list.get(index).getPrice();
138. bill=q1\*Integer.parseInt(price);
139. check=true;
140. }
141. }
142. o.add\_order(bill);
143. return check;
144. }
145. public boolean orderHomeAppliances(String code, String q)
146. {
147. boolean found=HomeAppliances.searchDataFromFile(code);
148. int quant=0;
149. String price;
150. int index=0;
151. bill=0;
152. boolean check=false;
153. int q1=Integer.parseInt(q);
154. Scanner input= new Scanner(System.in);
155. if(found==true)
156. {
157. ArrayList<HomeAppliances> list= HomeAppliances.readAllDataFromFile();
158. for(int i=0;i<list.size();i++)
159. {
161. if(list.get(i).getCode().toLowerCase().equals(code.toLowerCase()))
162. {
163. System.out.println("ITEM FOUND IN THE RECORD");
164. quant=Integer.parseInt(list.get(i).getQuantity());
165. price=list.get(i).getPrice();
166. index=i;
167. }
168. }
169. if(q1>quant)
170. {
171. System.out.println("ITEM OUT OF STROKE");
172. check=false;
173. return check;
174. }
175. else
176. {
177. String k=Integer.toString(quant-q1);
178. list.get(index).setQuantity(k);
179. price=list.get(index).getPrice();
180. bill=q1\*Integer.parseInt(price);
182. }
183. }
184. o.add\_order(bill);
185. return check;
186. }
187. }

***Code For add To Cart GUI:***

1. package oop\_project;
2. import java.awt.BorderLayout;
3. import java.awt.GridLayout;
4. import java.awt.event.ActionEvent;
5. import java.awt.event.ActionListener;
6. import java.util.ArrayList;
7. import javax.swing.JButton;
8. import javax.swing.JFrame;
9. import javax.swing.JLabel;
10. import javax.swing.JOptionPane;
11. import javax.swing.JPanel;
12. import javax.swing.JTextField;
14. public class ADD\_TO\_CART\_GUI extends JFrame
15. {
16. JButton c1;
17. JButton c2;
18. JButton c3;
19. JButton c4;
20. JLabel l1,l2;
21. JTextField t1,t2;
22. JButton b1,b2;
23. JButton o1,o2,o3,o4;
24. ADD\_TO\_CART\_GUI()
25. {
26. setSize(400,600);
27. setLayout(new BorderLayout());
28. b1= new JButton("HOME");
29. b2=new JButton("SHOW BILL");
30. o1= new JButton("ORDER DRESS");
31. o2= new JButton("ORDER SPORTS");
32. o3= new JButton("ORDER COSMETICS");
33. o4= new JButton("ORDER HOME APPLIANCES");
34. l1= new JLabel("ENTER THE CODE");
35. l2= new JLabel("ENTER QUANTITY");
36. JPanel p4= new JPanel();
38. p4.setLayout(new GridLayout(2,2));
39. t1= new JTextField(20);
40. t2= new JTextField(20);
41. c1=new JButton ("SHOP DRESSES");
42. c2=new JButton ("SHOP SPOTS ITEM");
43. c3= new JButton ("SHOP COSMETICS");
44. c4= new JButton ("SHOP HOME APPLIANCES");
45. JPanel p1= new JPanel();
46. p1.add(l1);
47. p1.add(t1);
48. p1.add(l2);
49. p1.add(t2);
50. p1.setLayout(new GridLayout(2,2));
51. JPanel p2= new JPanel();
52. p2.add(c1);
53. p2.add(c2);
54. p2.add(c3);
55. p2.add(c4);
56. p2.add(o1);
57. p2.add(o2);
58. p2.add(o3);
59. p2.add(o4);
60. JPanel p3= new JPanel();
61. p3.add(b1);
62. p3.add(b2);
63. add(p3,BorderLayout.SOUTH);
64. add(p2,BorderLayout.CENTER);
65. add(p1,BorderLayout.NORTH);
66. c1.addActionListener(new MyActionListener() );
67. c2.addActionListener(new MyActionListener() );
68. c3.addActionListener(new MyActionListener() );
69. c4.addActionListener(new MyActionListener() );
70. b1.addActionListener(new MyActionListener() );
71. b2.addActionListener(new MyActionListener());
72. o1.addActionListener(new MyActionListener());
73. o2.addActionListener(new MyActionListener());
74. o3.addActionListener(new MyActionListener());
75. o4.addActionListener(new MyActionListener());
76. setVisible(true);
77. }
79. public class MyActionListener implements ActionListener
80. {
81. public void actionPerformed(ActionEvent ae)
82. {
83. if(ae.getActionCommand()=="SHOP DRESSES")
84. {
85. ArrayList<Dresses> d= Dresses.readAllDataFromFile();
86. for(int i=0;i<d.size();i++)
87. {
88. JOptionPane.showMessageDialog(null, d.get(i).getCode()+" "+d.get(i).getBrand()+" "+d.get(i).getPrice()+" "+d.get(i).getType());
89. }

92. }
93. if(ae.getActionCommand()=="ORDER DRESS")
94. {
95. AddToCart c= new AddToCart();
96. if(t1.getText().isEmpty() && t2.getText().isEmpty() )
97. {
98. JOptionPane.showMessageDialog(null,"PLEASE FILL THE TEXT FIELDS FIRST");
99. }
100. if(t1.getText().isEmpty() || t2.getText().isEmpty())
101. {
102. JOptionPane.showMessageDialog(null,"PLEASE FILL THE TEXT FIELDS FIRST");
103. }
104. else
105. {
106. if(c.orderDress(t1.getText(),(t2.getText()))==true)
107. {
108. JOptionPane.showMessageDialog(null,"YOUR ORDER IS TAKEN");
109. JOptionPane.showMessageDialog(null,"WE WILL DELIEVER YOUR ODER IN 3 T 4 DAYS ");
110. }
111. else
112. {
113. JOptionPane.showMessageDialog(null, "ITEM OUT OF STROKE");
114. }
116. }
117. }
118. if(ae.getActionCommand()=="ORDER SPORTS")
119. {
120. AddToCart c= new AddToCart();
121. if(t1.getText().isEmpty() && t2.getText().isEmpty() )
122. {
123. JOptionPane.showMessageDialog(null,"PLEASE FILL THE TEXT FIELDS FIRST");
124. }
125. if(t1.getText().isEmpty() || t2.getText().isEmpty())
126. {
127. JOptionPane.showMessageDialog(null,"PLEASE FILL THE TEXT FIELDS FIRST");
128. }
129. else
130. {
131. if(c.orderSports(t1.getText(), (t2.getText()))==true)
132. {
133. JOptionPane.showMessageDialog(null,"YOUR ORDER IS TAKEN");
134. JOptionPane.showMessageDialog(null,"WE WILL DELIEVER YOUR ODER IN 3 T 4 DAYS ");
135. }
136. else
137. {
138. JOptionPane.showMessageDialog(null, "ITEM OUT OF STROKE");
139. }
141. }
142. }
143. if(ae.getActionCommand()=="ORDER COSMETICS")
144. {
145. AddToCart c= new AddToCart();
146. if(t1.getText().isEmpty() && t2.getText().isEmpty() )
147. {
148. JOptionPane.showMessageDialog(null,"PLEASE FILL THE TEXT FIELDS FIRST");
149. }
150. if(t1.getText().isEmpty() || t2.getText().isEmpty())
151. {
152. JOptionPane.showMessageDialog(null,"PLEASE FILL THE TEXT FIELDS FIRST");
153. }
154. else
155. {
156. if(c.orderCosmetics(t1.getText(), (t2.getText()))==true)
157. {
158. JOptionPane.showMessageDialog(null,"YOUR ORDER IS TAKEN");
159. JOptionPane.showMessageDialog(null,"WE WILL DELIEVER YOUR ODER IN 3 T 4 DAYS ");
160. }
161. else
162. {
163. JOptionPane.showMessageDialog(null,"ITEM OUT OF STROKE");
164. }
165. }
166. }
167. if(ae.getActionCommand()=="ORDER HOME APPLIANCES")
168. {
169. AddToCart c= new AddToCart();
170. if(t1.getText().isEmpty() && t2.getText().isEmpty() )
171. {
172. JOptionPane.showMessageDialog(null,"PLEASE FILL THE TEXT FIELDS FIRST");
173. }
174. if(t1.getText().isEmpty()|| t2.getText().isEmpty())
175. {
176. JOptionPane.showMessageDialog(null,"PLEASE FILL THE TEXT FIELDS FIRST");
177. }
178. else
179. {
180. if(c.orderHomeAppliances(t1.getText(),(t2.getText()))==true)
181. {
182. JOptionPane.showMessageDialog(null,"YOUR ORDER IS TAKEN");
183. JOptionPane.showMessageDialog(null,"WE WILL DELIEVER YOUR ODER IN 3 T 4 DAYS ");
184. }
185. else
186. {
187. JOptionPane.showMessageDialog(null,"ITEM OUT OF STROKE");
188. }
189. }
190. }
192. if(ae.getActionCommand()=="SHOP SPORTS ITEMS")
193. {
194. ArrayList<Sports> d= Sports.readAllDataFromFile();
195. for(int i=0;i<d.size();i++)
196. {
197. JOptionPane.showMessageDialog(null, d.get(i).getCode()+" "+d.get(i).getBrand()+" "+d.get(i).getPrice()+" "+d.get(i).getName());
198. }
199. }
200. if(ae.getActionCommand()=="SHOP COSMETICS")
201. {
202. ArrayList<Cosmetics> d= Cosmetics.readAllDataFromFile();
203. for(int i=0;i<d.size();i++)
204. {
205. JOptionPane.showMessageDialog(null, d.get(i).getCode()+" "+d.get(i).getBrand()+" "+d.get(i).getPrice()+" "+d.get(i).getName());
206. }
207. }
208. if(ae.getActionCommand()=="SHOP HOME APPLIANCES")
209. {
210. ArrayList<HomeAppliances> d= HomeAppliances.readAllDataFromFile();
211. for(int i=0;i<d.size();i++)
212. {
213. JOptionPane.showMessageDialog(null, d.get(i).getCode()+" "+d.get(i).getBrand()+" "+d.get(i).getPrice()+" "+d.get(i).getName());
214. }
215. }
216. if(ae.getActionCommand()=="HOME")
217. {
218. dispose();
219. SIGN\_IN C= new SIGN\_IN();
220. }
221. if(ae.getActionCommand()=="SHOW BILL")
222. {
223. order o = new order();
224. JOptionPane.showMessageDialog(null, "YOUR TOTAL BILL IS "+" "+o.get\_bill());
225. JOptionPane.showMessageDialog(null, "WE WILL DELIEVER YOUR ORDER IN 3 TO 4 DAYS");
226. JOptionPane.showMessageDialog(null,"THANKYOU FOR SHOPPING");
227. o.set\_bill(0);
228. }
229. }
230. }
231. }

***CODE FOR SIGN IN:***

1. package oop\_project;
2. import java.awt.Color;
3. import java.awt.Font;
4. import java.awt.GridLayout;
5. import java.awt.event.ActionEvent;
6. import java.awt.event.ActionListener;
7. import javax.swing.ImageIcon;
8. import javax.swing.JButton;
9. import javax.swing.JFrame;
11. public class SIGN\_IN extends JFrame
12. {
13. JButton b1,b2,b6;
14. ImageIcon i1,i2,i3;
15. JFrame frame;
16. SIGN\_IN()
17. {
18. frame=new JFrame();
19. i1=new ImageIcon("person.png");
20. i2=new ImageIcon("admin.png");
21. i3=new ImageIcon("home1.png");
22. frame.setSize(400,800);
23. frame.setLayout(new GridLayout(3,3));
24. frame.getContentPane().setBackground(new Color(30,20,40,40));
25. b1= new JButton("CUSTOMER");
26. b1.setFont(new Font("Courgette",Font.BOLD,16));
27. b1.setIcon(i1);
28. b1.setHorizontalTextPosition(JButton.RIGHT);
29. b1.setBackground(Color.white);
30. b2= new JButton("ADMIN");
31. b2.setFont(new Font("Courgette",Font.BOLD,16));
32. b2.setIcon(i2);
33. b2.setHorizontalTextPosition(JButton.RIGHT);
34. b2.setBackground(Color.white);
35. b6= new JButton("HOME");
36. b6.setFont(new Font("Courgette",Font.BOLD,16));
37. b6.setIcon(i3);
38. b6.setHorizontalTextPosition(JButton.RIGHT);
39. b6.setBackground(Color.white);
40. MyActionListener a = new MyActionListener();
41. frame.add(b1);
42. frame.add(b2);
43. frame.add(b6);
44. b1.addActionListener(a);
45. b2.addActionListener(a);
46. b6.addActionListener(a);
47. frame.setVisible(true);
49. }
50. public class MyActionListener implements ActionListener
51. {
52. public void actionPerformed(ActionEvent ae)
53. {
54. if(ae.getActionCommand()=="CUSTOMER")
55. {
56. frame.dispose();
57. CUSTOMER\_MENU C= new CUSTOMER\_MENU();
58. }
59. if(ae.getActionCommand()=="ADMIN")
60. {
61. frame.dispose();
62. ADMIN\_MENU d= new ADMIN\_MENU();
63. }
64. if(ae.getActionCommand()=="HOME")
65. {
66. frame.dispose();
67. WELCOME\_PAGE w=new WELCOME\_PAGE();
68. }
69. }
71. }
72. }

***CODE FOR SIGN UP:***

1. package oop\_project;
2. import java.awt.BorderLayout;
3. import java.awt.FlowLayout;
4. import java.awt.GridLayout;
5. import java.awt.event.ActionEvent;
6. import java.awt.event.ActionListener;
7. import javax.swing.JButton;
8. import javax.swing.JFrame;
9. import javax.swing.JLabel;
10. import javax.swing.JOptionPane;
11. import javax.swing.JPanel;
12. import javax.swing.JTextField;
13. public class SIGN\_UP extends JFrame
14. {
15. JButton b1,b2;
16. JLabel l1,l2,l3,l4,l8,l9,l10;
17. JTextField t1,t2,t3,t4,t8,t9,t10;
18. SIGN\_UP(){
19. setLayout(new BorderLayout());
20. setSize(400,500);
21. l1=new JLabel("Name: ");
22. l2=new JLabel(" Contact: ");
23. l3=new JLabel("Email: ");
24. l4=new JLabel("Password: ");
25. l8=new JLabel("Home no: ");
26. l9=new JLabel("Street no: ");
27. l10=new JLabel("City: ");
28. t1= new JTextField(20);
29. t2= new JTextField(20);
30. t3=new JTextField(20);
31. t4=new JTextField(20);
32. t8=new JTextField(20);
33. t9=new JTextField(20);
34. t10=new JTextField(20);
35. b1=new JButton("Home");
36. b2=new JButton("Submit");
37. JPanel p1=new JPanel();
38. p1.setLayout(new GridLayout(20,10));
39. p1.add(l1);
40. p1.add(t1);
41. p1.add(l2);
42. p1.add(t2);
43. p1.add(l3);
44. p1.add(t3);
45. p1.add(l4);
46. p1.add(t4);
47. p1.add(l8);
48. p1.add(t8);
49. p1.add(l9);
50. p1.add(t9);
51. p1.add(l10);
52. p1.add(t10);
53. add(p1,BorderLayout.NORTH);
54. JPanel p2=new JPanel();
55. p2.setLayout(new FlowLayout());
56. p2.add(b1);
57. p2.add(b2);
58. add(p2, BorderLayout.SOUTH);
59. MyActionListener a=new MyActionListener();
60. b1.addActionListener(a);
61. b2.addActionListener(a);
62. setVisible(true);
63. }
64. public class MyActionListener implements ActionListener
65. {
66. Customer c1;
67. Address a1;
68. @Override
69. public void actionPerformed(ActionEvent ae)
70. {
71. if(ae.getActionCommand()=="Submit")
72. {
73. a1=new Address(t8.getText(),t9.getText(),t10.getText());
74. c1=new Customer(t1.getText(),t2.getText(),t3.getText(),t4.getText(),a1);
75. if(t1.getText().isEmpty()||t2.getText().isEmpty()||t3.getText().isEmpty()||t4.getText().isEmpty()||t8.getText().isEmpty()||t9.getText().isEmpty()||t10.getText().isEmpty())
76. {
77. JOptionPane.showMessageDialog(null, "Please fill all the Text Fields");
78. }
79. else
80. {
81. if(Customer.checkname(t1.getText()))
82. {
83. if(Customer.check\_contact(t2.getText()))
84. {
85. if(Customer.chekemail(t3.getText()))
86. {
87. if(t3.getText().contains("@") && t3.getText().endsWith(".com"))
88. {
89. if(Address.homecheck(t8.getText()))
90. {
91. if(Address.streetcheck(t9.getText()))
92. {
93. if(Address.citycheck(t10.getText()))
94. {
95. Customer.writeToFile(c1);
96. JOptionPane.showMessageDialog(null, "Successful");
97. dispose();
98. DarazClone d= new DarazClone();
99. }
100. else
101. {
102. JOptionPane.showMessageDialog(null, "Invalid city name");
103. dispose();
104. DarazClone d= new DarazClone();
105. }
106. }
107. else
108. {
109. JOptionPane.showMessageDialog(null, "Invalid street name");
110. dispose();
111. DarazClone d= new DarazClone();
112. }
113. }
114. else
115. {
116. JOptionPane.showMessageDialog(null, "Invalid home entry");
117. dispose();
118. DarazClone d= new DarazClone();
120. }
122. }
123. else
124. {
125. JOptionPane.showMessageDialog(null, "Invalid format");
126. dispose();
127. DarazClone d= new DarazClone();
128. }
129. }
130. else
131. {
132. JOptionPane.showMessageDialog(null, "This Email already exist");
133. dispose();
134. DarazClone d= new DarazClone();
135. }
136. }
137. else
138. {
139. JOptionPane.showMessageDialog(null, "Invalid format:Follow Proper format of phone");
140. dispose();
141. DarazClone d= new DarazClone();
143. }
145. }
146. else
147. {
148. JOptionPane.showMessageDialog(null, "Invalid Name");
149. dispose();
150. DarazClone d= new DarazClone();
151. }
152. }
153. }
154. if(ae.getActionCommand()=="Home")
155. {
156. dispose();
157. DarazClone d= new DarazClone();
158. }

161. }
162. }
163. }

***CODE FOR Some Important Function:***

***Validation code for Customer:***

1. public static boolean checkname(String n)
2. {
3. boolean found=false;
4. char cust\_check=0;
5. for(int i=0;i<n.length();i++)
6. {
7. cust\_check=n.charAt(i);
8. if(Character.isLetter(cust\_check))
9. {
10. found=true;
11. }
12. else
13. {
14. found=false;
15. break;
16. }
17. }
18. return found;
19. }
20. Contact format:
21. public static boolean check\_contact(String c)
22. {
23. boolean check\_validation=false;
24. String format = ("\\d{3}-\\d{7}");
25. String format\_2 = ("\\d{4}-\\d{7}");
26. if (c.matches(format) || c.matches(format\_2))
27. {
28. check\_validation= true;
29. }
30. else
31. {
32. check\_validation = false;
33. }
34. return check\_validation;
35. }
36. public static void update\_Address(String n,Address a)
37. {
38. ArrayList<Customer> list=readobj();
39. boolean id\_check=true;
41. if(id\_check==true)
42. {
43. Address a2=new Address(a);
44. for(int i=0;i<list.size();i++)
45. {
46. if(list.get(i).getEmail().equals(n))
47. {
48. list.get(i).setAd(a2);
49. break;
50. }
51. }
52. }
54. try
55. {
56. ObjectOutputStream Output=new ObjectOutputStream(new FileOutputStream("Customer Record"));
57. Output.writeObject(list);
58. }
59. catch (IOException e)
60. {
61. System.out.println("Error");
62. }
64. }
65. Email Existence check:
66. public static boolean chekemail(String e)
67. {
68. boolean esearch=true;
69. ArrayList<Customer> list=readobj();
70. for(int i=0;i<list.size();i++)
71. {
72. if(!(list.get(i).getEmail().toLowerCase().equals(e.toLowerCase())))
73. {
74. esearch=true;
75. }
76. else
77. {
78. esearch=false;
79. break;
81. }
83. }
84. return esearch;
86. }

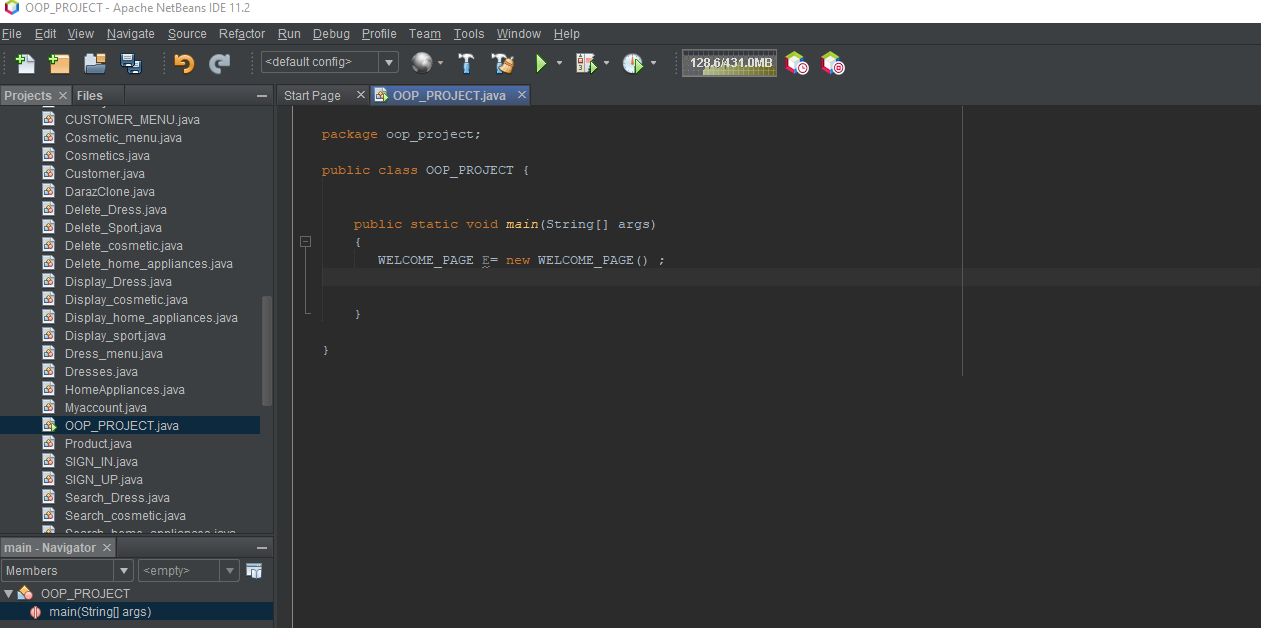
***Code for the Cosmetics Functions Update, Search and delete:***

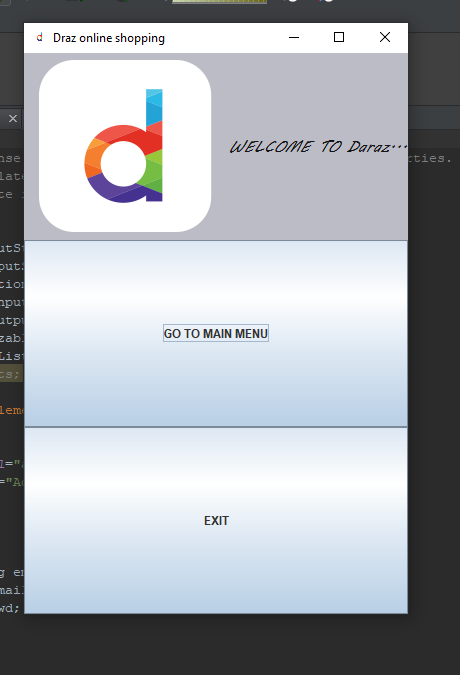
1. public static void UpdatePrice(String code,String Updated\_price)
2. {
3. ArrayList<Cosmetics> list=readAllDataFromFile();
4. Scanner input= new Scanner(System.in);
5. String Updated;
6. boolean check\_price=true;
7. while(check\_price==true)
8. {
9. for(int h=0;h<Updated\_price.length();h++)
10. {
11. char p=Updated\_price.charAt(h);
12. if (Character.isDigit(p))
13. {
14. check\_price=false;
15. }
16. else
17. {
18. check\_price=true;
19. }
20. }
21. if(check\_price==true)
22. {
23. System.out.println("PRICE MUST BE IN DIGIT FORM");
24. System.out.println("ENTER AGAIN THE UPDATED PRICE");
25. Updated=input.next();
26. Updated\_price=Updated;
27. }
28. }
29. for(int i=0;i<list.size();i++)
30. {
31. if(list.get(i).getCode().toLowerCase().equals(code.toLowerCase()))
32. {
33. list.get(i).setPrice(Updated\_price);
34. break;
35. }
36. else
37. {
38. System.out.println("NO RECORD OF DRESS FOUND AGAINST THIS CODE");
39. }
40. }
41. try
42. {
43. ObjectOutputStream Output=new ObjectOutputStream(new FileOutputStream("Cosmetics Record"));
44. Output.writeObject(list);
45. }
46. catch (IOException e)
47. {
48. System.out.println("Error");
49. }
50. }

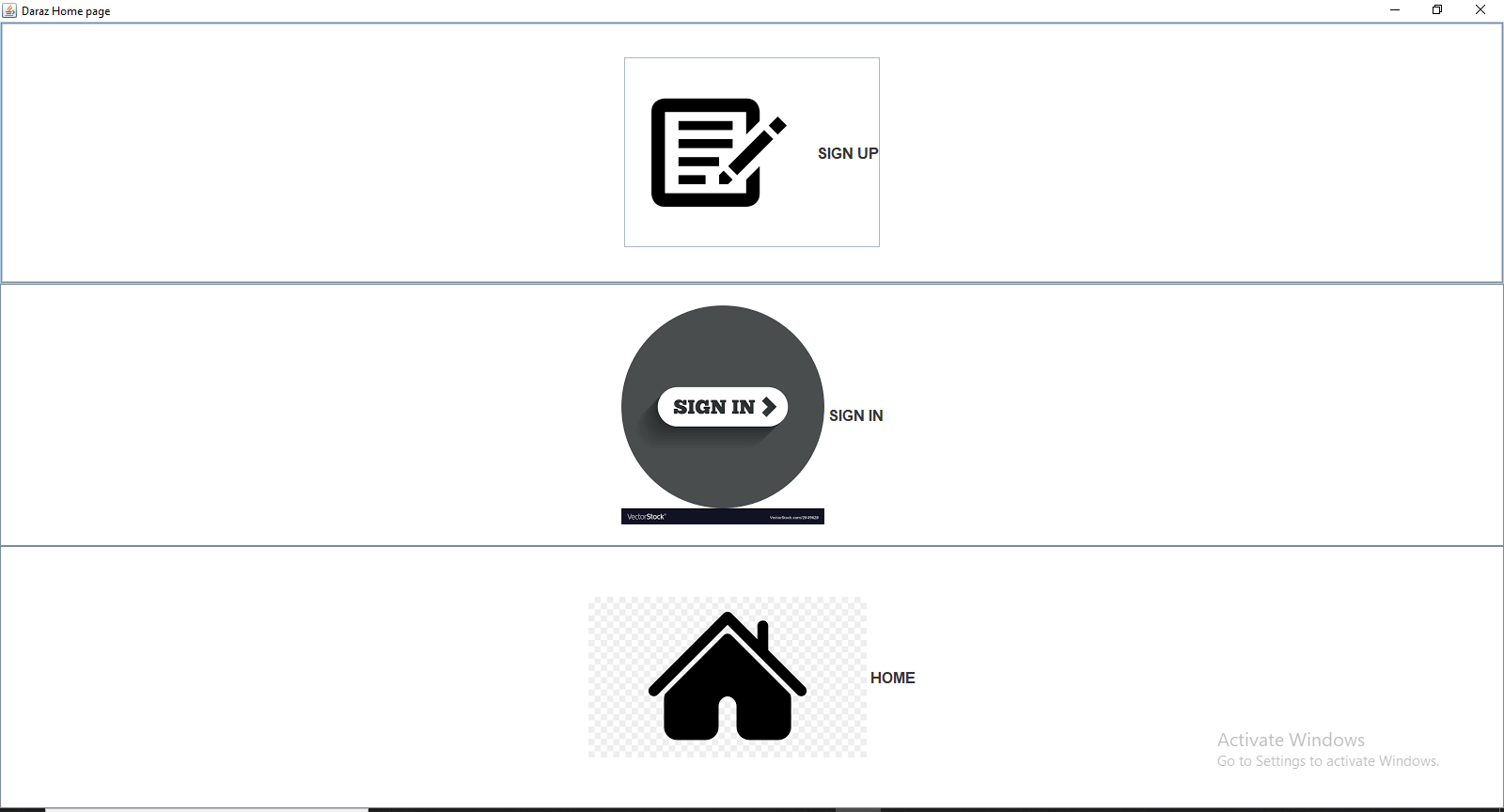



56. public static boolean searchDataFromFile(String code)
57. {
58. ArrayList <Cosmetics> list = readAllDataFromFile();
59. for(int i=0;i<list.size();i++)
60. {
61. if(list.get(i).getCode().toLowerCase().equals(code.toLowerCase()))
62. {
63. System.out.println(list.get(i).getCode()+" "+list.get(i).getBrand()+" "+list.get(i).getName()+" "+list.get(i).getQuantity()+" "+list.get(i).getPrice());
64. return true;
65. }
66. }
67. return false;
68. }
69. public static void UpdateQuantity(String code,String Updated\_quantity)
70. {
71. ArrayList<Cosmetics> list=readAllDataFromFile();
72. boolean check\_quantity=true;
73. Scanner input= new Scanner(System.in);
74. String Updated;
76. while(check\_quantity==true)
77. {
78. for(int h=0;h<Updated\_quantity.length();h++)
79. {
80. char q=Updated\_quantity.charAt(h);
81. if (Character.isDigit(q))
82. {
83. check\_quantity=false;
84. }
85. else
86. {
87. check\_quantity=true;
88. }
89. }
90. if(check\_quantity==true)
91. {
92. System.out.println("QUANTITY MUST BE IN DIGIT FORM");
93. System.out.println("ENTER AGAIN THE UPDATED QUANTITY");
94. Updated=input.next();
95. Updated\_quantity=Updated;
96. }
97. }
98. for(int i=0;i<list.size();i++)
99. {
100. if(list.get(i).getCode().toLowerCase().equals(code.toLowerCase()))
101. {
102. list.get(i).setQuantity(Updated\_quantity);
103. break;
104. }
105. else
106. {
107. System.out.println("NO RECORD OF DRESS FOUND AGAINST THIS CODE");
108. }
109. }
110. try
111. {
112. ObjectOutputStream Output=new ObjectOutputStream(new FileOutputStream("Cosmetics Record"));
113. Output.writeObject(list);
114. }
115. catch (IOException e)
116. {
117. System.out.println("Error");
118. }
119. }
120. public static void deleteProduct(String code)
121. {
122. ArrayList <Cosmetics> list = readAllDataFromFile();
123. int index =0;
124. for (int i =0; i< list.size() ; i++)
125. {
126. if (list.get(i).getCode()==code)
127. {
128. System.out.println("VALUE FOUND NOW DELETING IT");
129. index = i;
130. break;
131. }
132. }
133. if(list.size()!=0)
134. {
135. list.remove(index);
136. }
137. else
138. {
139. System.out.println("NO RECORD FOUND ");
140. }
141. try
142. {
143. ObjectOutputStream output = new ObjectOutputStream( new FileOutputStream("Cosmetics Record"));
144. output.writeObject(list);
145. }
146. catch (IOException e)
147. {
149. }
150. }

**OOP PROJECT SCOPE DOCUMENT:**









***ADMIN LOGIN:***

