

## Password.java

```
2    Chapter 9: The Password class
8
9    import java.util.*;
10
11   public class Password
12   {
13       final static int MIN_SIZE = 6;
14       final static int MAX_SIZE = 15;
15       static int maxHistory = 4;
16       static int expiresNotifyLimit = 3;
17
18       private int maxUses = 120;
19       private int remainingUses = maxUses;
20       private boolean autoExpires = true;
21       private boolean expired = false;
22
23       private ArrayList pswdHistory;
24
25
26       //Constructors for objects of class Password
27       public Password(String newPassword) throws Exception
28       {
29           pswdHistory = new ArrayList(maxHistory);
30           set(newPassword);
31       }
32
33       public Password(String newPassword, int numMaxUses) throws
Exception
34       {
35           pswdHistory = new ArrayList(maxHistory);
36           maxUses = numMaxUses;
37           remainingUses = numMaxUses;
38           set(newPassword);
39       }
40
41       public Password(String newPassword, boolean pswdAutoExpires)
throws Exception
42       {
43           pswdHistory = new ArrayList(maxHistory);
44           autoExpires = pswdAutoExpires;
45           set(newPassword);
46       }
47
48       public Password(String newPassword, int numMaxUses, boolean
pswdAutoExpires) throws Exception
```

Password.java

```
49     {
50         pswdHistory = new ArrayList(maxHistory);
51         maxUses = numMaxUses;
52         remainingUses = numMaxUses;
53         autoExpires = pswdAutoExpires;
54         set(newPassword);
55     }
56
57     public boolean getAutoExpires()
58     {
59         return autoExpires;
60     }
61
62     public void setAutoExpires(boolean autoExpires)
63     {
64         this.autoExpires = autoExpires;
65         if(autoExpires)
66             remainingUses = maxUses;
67 }
68
69     public boolean isExpired()
70     {
71         return expired;
72     }
73
74     public void setExpired(boolean newExpired)
75     {
76         expired = newExpired;
77     }
78
79     public int getExpiresNotifyLimit()
80     {
81         return expiresNotifyLimit;
82     }
83
84     public void setExpiresNotifyLimit(int newNotifyLimit)
85     {
86         if(newNotifyLimit >= 2 && newNotifyLimit <= 20)
87             expiresNotifyLimit = newNotifyLimit;
88     }
89
90     public int getMaxHistory()
91     {
92         return maxHistory;
93     }
```

Password.java

```
94
95     public void setMaxHistory(int newMaxHistory)
96     {
97         int overage = 0;
98         if(newMaxHistory >= 1 && newMaxHistory <= 10)
99         {
100             maxHistory = newMaxHistory;
101             overage = getHistorySize() - maxHistory;
102             if(overage > 0) //if size > max
allowed
103                 {
104                     do{
105                         pswdHistory.remove(0); //then remove overage
number
106                         overage --; //of oldest pswds from
list
107                     }while(overage > 0);
108                     pswdHistory.trimToSize(); //resize capacity to
max allowed
109                 }
110             }
111         }
112     }
113
114     public int getRemainingUses()
115     {
116         return remainingUses;
117     }
118
119     public int getHistorySize()
120     {
121         return pswdHistory.size();
122     }
123
124     public boolean isExpiring()
125     {
126         boolean expiring = false;
127
128         if(autoExpires && remainingUses <= expiresNotifyLimit)
129             expiring = true;
130
131         return expiring;
132     }
133
134     //Sets password to a new value ; keeps current & previous
```

Password.java

```
values in history up to max number
135     public void set(String pswd) throws Exception
136     {
137         String encryptPswd;
138         boolean pswdAdded = true;
139
140         pswd = pswd.trim();                //remove any
        leading, trailing white space
141         verifyFormat(pswd);                //verify password
        was entered correctly
142         encryptPswd = encrypt(pswd);        //convert to
        encrypted form
143
144         if(!pswdHistory.contains(encryptPswd)) //if pswd not in
        recently used list
145         {
146             if(pswdHistory.size() == maxHistory) //if list is at
        max size
147                 pswdHistory.remove(0);        //remove oldest
        password from list
148
149         pswdAdded = pswdHistory.add(encryptPswd); //add new
        pswd to end of ArrayList
150
151         if(!pswdAdded)                        //should
        never happen
152             throw new Exception("Internal list error -
        Password not accepted");
153
154         if(expired)                            //if pswd is
        expired
155             expired = false;                //reset to
        not expired
156
157         if(autoExpires)                        //if pswd
        autoexpires,
158             remainingUses = maxUses;        //reset
        uses to max
159         }
160         else
161             throw new Exception("Password recently used");
162         }
163
164     //Validates entered password against most recently saved
        value
```

Password.java

```
165     public void validate(String pswd) throws Exception
166     {
167         String encryptPswd;
168         String currentPswd;
169         int currentPswdIndex;
170
171         verifyFormat(pswd);           //verify password was
entered properly
172         encryptPswd = encrypt(pswd);  //convert to encrypted
form
173
174         if(!pswdHistory.isEmpty())    //at least one password
entry in history
175         {
176             currentPswdIndex = pswdHistory.size()-1;
177             currentPswd =
(String)pswdHistory.get(currentPswdIndex);
178
179             if(!encryptPswd.equals(currentPswd)) //if not most
recent password
180                 throw new Exception("password is invalid");
181
182             if(expired)
183                 throw new Exception("Password has expired -
please change");
184
185             if(autoExpires)
186             {
187                 --remainingUses;
188                 if(remainingUses <= 0)
189                     expired = true;
190             }
191         }
192         else
193             throw new Exception("No password on file - list
corrupted!"); //should never happen
194
195     }
196
197
198     //Verifies password has proper format
199     private void verifyFormat(String pswd) throws Exception
200     {
201         boolean numFound = false;
202
```

Password.java

```
203         if(pswd.length() == 0)
204             throw new Exception ("No password provided!");
205
206         if(pswd.length() < MIN_SIZE)
207             throw new Exception("password must be at least" +
MIN_SIZE + "charecters in length");
208
209         if(pswd.length() > MAX_SIZE)
210             throw new Exception ("Password cannot be greater the
" + MAX_SIZE + " charecters in length");
211
212         // scan through password to find if at least 1 number us
used.
213         for(int i=0;i< pswd.length() && !numFound; ++i)
214             if(Character.isDigit(pswd.charAt(i)))
215                 numFound = true;
216
217         if(!numFound)
218             throw new Exception("Password is invalid - must have
at least one number");
219     }
220
221     //Encrypts original password returning new encrypted String
222     private String encrypt(String pswd)
223     {
224         StringBuffer encryptPswd;
225         int pswdSize = 0;
226         int midpoint = 0;
227         int hashCode = 0;
228
229         //swap first and last half of password
230         pswdSize = pswd.length();
231         midpoint = pswdSize / 2;
232         encryptPswd = new
StringBuffer(pswd.substring(midpoint)           //get last half of
pswd
233             +
pswd.substring(0,midpoint));           //and
concatenate first half
234
235         encryptPswd.reverse(); //reverse order of characters in
password
236
237         for(int i = 0; i < pswdSize; ++i)
238             encryptPswd.setCharAt(i, (char)
```

# Password.java

```
(encryptPswd.charAt(i)& pswd.charAt(i)));
239
240         hashCode = pswd.hashCode(); //hash code for original
password
241         encryptPswd.append(hashCode);
242
243         return encryptPswd.toString();
244     }
245 }
246
247
```