

## Class MyPhone - Test

1/8

```
1 import java.text.DecimalFormat;
2 import javax.swing.JOptionPane;
3 import java.util.Random;
4 /*****
5  * This program simulates a cell phone with data, battery, and texting.
6  *
7  * @author Ben Parsell
8  * @version 10/1/15
9  *****/
10 public class MyPhone
11 {
12     private int numTexts; // Total number of texts
13     private double dataConsumed; // Data consumed in megabytes
14     private double batteryLife; // Battery Life in a percentage
15     private String customerName; // Customer name
16     private String phoneNumber; // Customer phone number
17     private boolean wifiOn; // Holds whether wifi is on/off
18     private final double VIDEO_DATA_PER_MIN = 250 / 60.0; // Rate for vid
19     private final double audio_data_per_min = 65 / 60.0; // Rate for audi
20     private final double audio_minutes = 720.0; // Total audio minutes on
21     private final double video_minutes = 360.0; // Total video minutes o
22     n full charge
23
24     /*****
25     Constructor for objects of class MyPhone
26     This is the default constructor
27
28     @param name customer name
29     @param num customer phone number
30     *****/
31     public MyPhone (String name, String num) {
32         customerName = name;
33         phoneNumber = num;
34         batteryLife = 0.0;
35         dataConsumed = 0.0;
36         if(phoneNumber.length() != 10){
37             phoneNumber = "9999999999";
38         }
39     }
40
41     /*****
42     Change the Customer Name.
43     This is a mutator method
```

Oct 13, 2015 12:26:33 PM

```
44
45     @param n local variable for customer name
46     *****/
47     public void setName (String n) {
48         customerName = n;
49     }
50
51
52     /*****
53     Changes the customer's phone number.
54     This is a mutator method
55
56     @param n local variable for phone number
57     *****/
58     public void setPhoneNumber (String n) {
59         phoneNumber = n;
60     }
61
62
63     /*****
64     Formats the given phone number to standard phone number
65     formatting.
66     This is a mutator method.
67     *****/
68     private String fmtPhoneNumber() {
69         phoneNumber = "(" + phoneNumber.substring(0,3) + ")" + phoneNumb
er.substring(3,6) + "-" + phoneNumber.substring(6,9);
70         return phoneNumber;
71     }
72
73
74     /*****
75     Outputs the number of total texts.
76     This is a mutator method
77
78     @return number of total texts
79     *****/
80     public int getNumTexts() {
81         return numTexts;
82     }
83
84
85     /*****
86     Outputs the battery life percentage.
87     This is an accessor method.
88
89     @return battery life percent
90     *****/
91     public double getBatteryLife() {
```

```

92         return batteryLife;
93     }
94
95
96     /*****
97     Outputs the total megabytes of data used
98     This is an accessor method
99
100     @return total data consumed in megabytes
101     *****/
102     public double getDataUsage() {
103         return dataConsumed;
104     }
105
106
107     /*****
108     Charges to battery — dependent on minutes from input
109     This is a mutator method
110
111     @param mins local variable for minutes
112     *****/
113     public void chargeBattery(int mins) {
114
115         if((((batteryLife * 120) + mins) / 120) > 1.0) {
116             batteryLife = 1.0;
117         }else if(batteryLife > 0) {
118             batteryLife = ((batteryLife * 120) + mins) / 120.0;
119         }else if(mins <= 120) {
120             batteryLife = mins / 120.0;
121         } else if(mins > 120) {
122             batteryLife = 1.0;
123         }
124         DecimalFormat fmt = new DecimalFormat("#.0");
125         fmt.format(batteryLife);
126         JOptionPane.showMessageDialog(null, "Battery Life: " + (fmt.forma
127         t((batteryLife)*100)) + "%");
128     }
129
130     /*****
131     Controls the wifi being on/off using boolean
132     This is a mutator method
133
134     *****/
135     public void setWifi (boolean wifi) {
136         wifiOn = wifi;
137     }
138
139

```

```
140  /*****
141  Determines the amount of data consumed from streaming
142  audio — dependent on minutes from parameter
143  Also handles wifi
144
145  @param mins local variable for minutes
146  @return returns only for error catch (mins <0)
147  *****/
148  public void streamAudio(int mins) {
149      if(mins < 0) {
150          return;
151      }
152
153      if(wifiOn == true) {
154          dataConsumed = dataConsumed + 0;
155          batteryLife = batteryLife - (mins/audio_minutes);
156          if(batteryLife <= 0) {
157              batteryLife = 0;
158          }
159      }
160      else if(wifiOn != true) {
161          if((mins) / audio_minutes > 1 ) {
162              dataConsumed = dataConsumed + 780;
163              batteryLife = 0.0;
164
165              } else if((mins) / audio_minutes > (batteryLife)) {
166                  dataConsumed = dataConsumed + ((audio_minutes*batteryLife) * a
167  udio_data_per_min);
168                  batteryLife = 0.0;
169                  JOptionPane.showMessageDialog(null, "Not enough battery for th
170  at many minutes. Phone will stream until battery is dead.");
171                  } else {
172                      batteryLife = batteryLife - (mins/audio_minutes);
173                      dataConsumed = dataConsumed + (mins*audio_data_per_min);
174                  }
175      }
176
177  /*****
178  Determines the amount of data consumed from streaming
179  video — dependent on minutes from parameter
180  Also handles wifi
181
182  @param mins local variable for minutes
183  @return returns only for error catch (mins <0)
184  *****/
185  public void streamVideo(int mins) {
186      if(mins < 0) {
187          return;
```

```
187     }
188
189     if(wifiOn) {
190         dataConsumed = dataConsumed;
191         batteryLife = batteryLife - (mins/video_minutes);
192         if(batteryLife <= 0) {
193             batteryLife = 0;
194         }
195     } else if(wifiOn != true) {
196         if((mins) / video_minutes > 1 ) {
197             dataConsumed = dataConsumed + 1500;
198             batteryLife = 0.0;
199             JOptionPane.showMessageDialog(null, "Not enough battery for th
at many minutes. Phone will stream until battery is dead.");
200         } else if((mins) / video_minutes > (batteryLife)) {
201             dataConsumed = dataConsumed + ((video_minutes*batteryLife) * V
IDEO_DATA_PER_MIN);
202             batteryLife = 0.0;
203             JOptionPane.showMessageDialog(null, "Not enough battery for th
at many minutes. Phone will stream until battery is dead.");
204         } else {
205             batteryLife = batteryLife - (mins/video_minutes);
206             dataConsumed = dataConsumed + (mins*VIDEO_DATA_PER_MIN);
207         }
208     }
209 }
210
211
212 /*****
213 Sends a text message and increments the total text count
214 This is a mutator method
215
216 @param text local variable for text message
217 *****/
218 public void sendText(String text) {
219     if(batteryLife > 0) {
220         numTexts += 1;
221         JOptionPane.showMessageDialog(null, "Message sent!");
222     } else if(batteryLife < 0) {
223         JOptionPane.showMessageDialog(null, "Battery is dead.");
224     }
225 }
226
227
228 /*****
229 Reads a text message randomly selected from a switch
230 statement. This uses the Random class.
231 This is a mutator method.
232
```

```
233      *****/
234      public void readText() {
235          Random rand = new Random();
236          int choice = rand.nextInt(5);
237          if(batteryLife == 0.0) {
238              JOptionPane.showMessageDialog(null, "You need to charge your
phone to get texts");
239          }else {
240              switch (choice) {
241                  case 0:
242                      JOptionPane.showMessageDialog(null, "What are you up
to?");
243                      break;
244
245                  case 1:
246                      JOptionPane.showMessageDialog(null, "How's your day g
oing?");
247                      break;
248
249                  case 2:
250                      JOptionPane.showMessageDialog(null, "Want to go grab
pizza?");
251                      break;
252
253                  case 3:
254                      JOptionPane.showMessageDialog(null, "Buddy the elf, w
hat's your favorite color?");
255                      break;
256
257                  case 4:
258                      JOptionPane.showMessageDialog(null, "What's your favo
rite movie?");
259                      break;
260              }
261              numTexts += 1;
262          }
263      }
264
265
266      /*****
267      Prints out the final customer statement.
268      MyPhoneTest uses this method to print final.
269      This is a mutator method.
270
271      *****/
272      public void printStatement() {
273          DecimalFormat fmt = new DecimalFormat("#.0");
274          System.out.println("MyPhone Monthly Statement");
275          System.out.println("");
```

```
276         System.out.println("Customer: \t\t" + customerName);
277         System.out.println("Number: \t\t" + fmtPhoneNumber());
278         System.out.println("Texts: \t\t\t" + numTexts);
279         System.out.println("Data usage: \t\t" + fmt.format(dataConsumed/1
024) + " GB");
280         System.out.println("");
281         System.out.println("2GB Plan: \t\t" + "$50.00");
282         System.out.println("Additional data fee: \t" + "$" + calcAddition
alDataFee());
283         System.out.println("Universal Usage (3%): \t" + "$" + calcUsageCh
arge());
284         System.out.println("Administrative Fee:\t$0.61");
285         System.out.println("Total Charges: \t\t" + "$" + calcTotalFee());
286     }
287
288
289     /*****
290     Resets number of texts and data consumed to start a new
291     month. This is a private helper method.
292
293     *****/
294     private void startNewMonth() {
295         dataConsumed = 0.0;
296         numTexts = 0;
297     }
298
299
300     /*****
301     Calculates any overuseage charges for data.
302     This is a private helped method.
303
304     @return extraCost returns final cost value.
305     *****/
306     private double calcAdditionalDataFee() {
307         double extraCost = 0; // Final extra cost value
308         double gbs; // Gigabytes
309         gbs = Math.ceil(dataConsumed / 1024);
310         if(dataConsumed > 2.0) {
311             extraCost = Math.ceil(gbs - 2) * 15;
312         }
313         return extraCost;
314     }
315
316
317     /*****
318     Calculates the 3% usage charge for the total bill cost.
319     This is a private helper method.
320
321     @return final value for usage charge cost.
```

```
322      *****/
323      private double calcUsageCharge() {
324          return (50.0 + calcAdditionalDataFee())* .03 ;
325      }
326
327
328      /*****
329      Calculates the total bill cost
330      This is a private helper method
331
332      @return final phone bill cost
333      *****/
334      private double calcTotalFee() {
335          double fee = 50.0 + calcAdditionalDataFee();
336          fee = fee + calcUsageCharge();
337          fee = fee + 0.61;
338          return fee;
339      }
340  }
341
342
```