1 import java.util.Calendar;  
 2 import java.util.GregorianCalendar;  
 3 import java.util.Random;  
 4   
 5 import javafx.application.Application;  
 6 import javafx.geometry.Pos;  
 7 import javafx.scene.Scene;  
 8 import javafx.scene.control.Label;  
 9 import javafx.scene.layout.BorderPane;  
 10 import javafx.scene.layout.GridPane;  
 11 import javafx.scene.layout.Pane;  
 12 import javafx.scene.layout.StackPane;  
 13 import javafx.scene.paint.Color;  
 14 import javafx.scene.shape.Arc;  
 15 import javafx.scene.shape.ArcType;  
 16 import javafx.scene.shape.Circle;  
 17 import javafx.scene.shape.Line;  
 18 import javafx.scene.text.Text;  
 19 import javafx.stage.Stage;  
 20   
 21 public class Assignment6 extends Application {  
 22   
 23 public static void main(String[] args) {  
 24 launch(args);  
 25 }  
 26   
 27 @Override  
 28 public void start(Stage primaryStage) throws Exception {  
 29 Pane root = buildRootPane();  
 30 Scene scene = new Scene(root, 1000, 350);  
 31   
 32 primaryStage.setTitle("Assignment 6"); // Set the stage title  
 33 primaryStage.setScene(scene); // Place the scene in the stage  
 34 primaryStage.show(); // Display the stage  
 35 }  
 36   
 37 private Pane buildRootPane() {  
 38 GridPane rootPane = new GridPane();  
 39   
 40 Pane clock = buildClockPane();  
 41 clock.setPrefHeight(5000);  
 42 clock.setPrefWidth(5000);  
 43   
 44 Pane fan = buildFanPane();  
 45 fan.setPrefHeight(5000);  
 46 fan.setPrefWidth(5000);  
 47   
 48 Pane hangman = buildHangmanPane();  
 49 hangman.setPrefHeight(5000);  
 50 hangman.setPrefWidth(5000);  
 51   
 52 rootPane.setAlignment(Pos.CENTER);  
 53 rootPane.add(clock, 1, 0);  
 54 rootPane.add(fan, 2, 0);  
 55 rootPane.add(hangman, 3, 0);  
 56   
 57 return rootPane;  
 58 }  
 59   
 60 private Pane buildClockPane() {  
 61 // Create a clock and a label  
 62 Random rand = new Random();  
 63 int hour = rand.nextInt(23) + 0;  
 64 int minute = rand.nextInt(60) + 0;  
 65 int second = rand.nextInt(60) + 0;  
 66   
 67 Text name = new Text("Shyam Rajendren");  
 68   
 69 ClockPane clock = new ClockPane(hour, minute, second);  
 70 String timeString = clock.getHour() + ":" + clock.getMinute() + ":" + clock.getSecond();  
 71 Label lblCurrentTime = new Label(timeString);  
 72   
 73 StackPane stack = new StackPane();  
 74 stack.getChildren().addAll(clock, name);  
 75   
 76 // Place clock and label in border pane  
 77 BorderPane pane = new BorderPane();  
 78 pane.setCenter(stack);  
 79 pane.setBottom(lblCurrentTime);  
 80 BorderPane.setAlignment(lblCurrentTime, Pos.TOP\_CENTER);  
 81   
 82 return pane;  
 83 }  
 84   
 85   
 86 private Pane buildFanPane() {  
 87 GridPane grid = new GridPane();  
 88 FanPane fan00 = new FanPane();  
 89 fan00.setPrefHeight(5000);  
 90 fan00.setPrefWidth(5000);  
 91   
 92 FanPane fan01 = new FanPane();  
 93 fan01.setPrefHeight(5000);  
 94 fan01.setPrefWidth(5000);  
 95   
 96 FanPane fan10 = new FanPane();  
 97 fan10.setPrefHeight(5000);  
 98 fan10.setPrefWidth(5000);  
 99   
100 FanPane fan11 = new FanPane();  
101 fan11.setPrefHeight(5000);  
102 fan11.setPrefWidth(5000);   
103   
104 grid.setAlignment(Pos.CENTER);  
105 grid.add(fan00, 0, 0);  
106 grid.add(fan01, 0, 1);  
107 grid.add(fan10, 1, 0);  
108 grid.add(fan11, 1, 1);  
109   
110 return grid;  
111 }  
112   
113 private Pane buildHangmanPane() {  
114 Pane hangman = new HangmanPane();  
115 BorderPane border = new BorderPane();  
116 border.setCenter(hangman);  
117 return border;  
118 }  
119   
120   
121 class FanPane extends Pane {  
122 FanPane() {  
123 paintFan();  
124 }  
125   
126 public void paintFan() {  
127 double xCenter = getWidth() / 2;  
128 double yCenter = getHeight() / 2;  
129 double radius = (Math.min(getWidth(), getHeight()) \* 0.4);  
130 Circle circle = new Circle(xCenter, yCenter, radius);  
131 circle.setFill(Color.WHITE);  
132 circle.setStroke(Color.BLACK);  
133   
134 getChildren().clear();  
135 getChildren().add(circle);  
136   
137 for (int i = 60; i < 360; i = i + 90) {  
138 Arc arc = new Arc(getWidth() / 2, getHeight() / 2, radius \* 0.90, radius \* 0.90, i, -30);  
139 arc.setFill(Color.BLACK);  
140 arc.setType(ArcType.ROUND);  
141 getChildren().add(arc);  
142 }  
143   
144 }  
145   
146 @Override  
147 public void setWidth(double width) {  
148 super.setWidth(width);  
149 paintFan();  
150 }  
151   
152 @Override  
153 public void setHeight(double height) {  
154 super.setHeight(height);  
155 paintFan();  
156 }  
157 }  
158   
159 class ClockPane extends Pane {  
160 private int hour;  
161 private int minute;  
162 private int second;  
163   
164 /\*\* Construct a default clock with the current time\*/  
165 public ClockPane() {  
166 setCurrentTime();  
167 paintClock();  
168 }  
169   
170 /\*\* Construct a clock with specified hour, minute, and second \*/  
171 public ClockPane(int hour, int minute, int second) {  
172 setHour(hour);  
173 setMinute(minute);  
174 setSecond(second);  
175 paintClock();  
176 }  
177   
178 /\*\* Return hour \*/  
179 public int getHour() {  
180 return hour;  
181 }  
182   
183 /\*\* Set a new hour \*/  
184 public void setHour(int hour) {  
185 this.hour = hour;  
186 paintClock();  
187 }  
188   
189 /\*\* Return minute \*/  
190 public int getMinute() {  
191 return minute;  
192 }  
193   
194 /\*\* Set a new minute \*/  
195 public void setMinute(int minute) {  
196 this.minute = minute;  
197 }  
198   
199 /\*\* Return second \*/  
200 public int getSecond() {  
201 return second;  
202 }  
203   
204 /\*\* Set a new second \*/  
205 public void setSecond(int second) {  
206 this.second = second;  
207 paintClock();  
208 }  
209   
210 /\* Set the current time for the clock \*/  
211 public void setCurrentTime() {  
212 // Construct a calendar for the current date and time  
213 Calendar calendar = new GregorianCalendar();  
214   
215 // Set current hour, minute and second  
216 this.hour = calendar.get(Calendar.HOUR\_OF\_DAY);  
217 this.minute = calendar.get(Calendar.MINUTE);  
218 this.second = calendar.get(Calendar.SECOND);  
219 }  
220   
221 /\*\* Paint the clock \*/  
222 private void paintClock() {  
223 // Initialize clock parameters  
224 double clockRadius = Math.min(getWidth(), getHeight()) \* 0.8 \* 0.5;  
225 double centerX = getWidth() / 2;  
226 double centerY = getHeight() / 2;  
227 // Draw circle  
228 Circle circle = new Circle(centerX, centerY, clockRadius);  
229 circle.setFill(Color.WHITE);  
230 circle.setStroke(Color.BLACK);  
231 Text t1 = new Text(centerX - 5, centerY - clockRadius + 12, "12");  
232 Text t2 = new Text(centerX - clockRadius + 3, centerY + 5, "9");  
233 Text t3 = new Text(centerX + clockRadius - 10, centerY + 3, "3");  
234 Text t4 = new Text(centerX - 3, centerY + clockRadius - 3, "6");  
235   
236 // Draw second hand  
237 double sLength = clockRadius \* 0.8;  
238 double secondX = centerX + sLength \* Math.sin(second \* (2 \* Math.PI / 60));  
239 double secondY = centerY - sLength \* Math.cos(second \* (2 \* Math.PI / 60));  
240 Line sLine = new Line(centerX, centerY, secondX, secondY);  
241 sLine.setStroke(Color.RED);  
242   
243 // Draw minute hand  
244 double mLength = clockRadius \* 0.65;  
245 double xMinute = centerX + mLength \* Math.sin(minute \* (2 \* Math.PI / 60));  
246 double minuteY = centerY - mLength \* Math.cos(minute \* (2 \* Math.PI / 60));  
247 Line mLine = new Line(centerX, centerY, xMinute, minuteY);  
248 mLine.setStroke(Color.BLUE);  
249   
250 // Draw hour hand  
251 double hLength = clockRadius \* 0.5;  
252 double hourX = centerX + hLength \* Math.sin((hour % 12 + minute / 60.0) \* (2 \* Math.PI / 12));  
253 double hourY = centerY - hLength \* Math.cos((hour % 12 + minute / 60.0) \* (2 \* Math.PI / 12));  
254 Line hLine = new Line(centerX, centerY, hourX, hourY);  
255 hLine.setStroke(Color.GREEN);  
256   
257 getChildren().clear();   
258 getChildren().addAll(circle, t1, t2, t3, t4, sLine, mLine, hLine);  
259 }  
260   
261 @Override  
262 public void setWidth(double width) {  
263 super.setWidth(width);  
264 paintClock();  
265 }  
266   
267 @Override  
268 public void setHeight(double height) {  
269 super.setHeight(height);  
270 paintClock();  
271 }  
272 }  
273   
274 class HangmanPane extends Pane {  
275 HangmanPane() {  
276 paint();  
277 }  
278   
279 private void paint() {  
280 Line line1 = new Line();  
281 line1.startXProperty().bind(layoutXProperty().add(10));  
282 line1.startYProperty().bind(layoutYProperty().add(10));  
283 line1.endXProperty().bind(layoutXProperty().add(getWidth()\*0.5));  
284 line1.endYProperty().bind(layoutYProperty().add(10));  
285   
286 Line line2 = new Line();  
287 line2.startXProperty().bind(layoutXProperty().add(10));  
288 line2.startYProperty().bind(layoutYProperty().add(10));  
289 line2.endXProperty().bind(layoutXProperty().add(10));  
290 line2.endYProperty().bind(layoutYProperty().add(getHeight()\*0.9));  
291   
292 Circle circle = new Circle();  
293 circle.centerXProperty().bind(layoutXProperty().add(getWidth()\*0.5));  
294 circle.centerYProperty().bind(layoutYProperty().add(getHeight()\*0.2));  
295 circle.radiusProperty().bind(layoutXProperty().add(getWidth()\*0.1));  
296 circle.setFill(Color.WHITE);  
297 circle.setStroke(Color.BLACK);  
298   
299 Line line3 = new Line();  
300 line3.startXProperty().bind(circle.centerXProperty());  
301 line3.startYProperty().bind(layoutYProperty().add(10));  
302 line3.endXProperty().bind(circle.centerXProperty());  
303 line3.endYProperty().bind(circle.centerYProperty().subtract(circle.getRadius()));  
304   
305 Line line4 = new Line(); //body  
306 line4.startXProperty().bind(circle.centerXProperty());  
307 line4.startYProperty().bind(circle.centerYProperty().add(circle.getRadius()));  
308 line4.endXProperty().bind(circle.centerXProperty());  
309 line4.endYProperty().bind(line4.startYProperty().add(getHeight()\*0.3));  
310   
311 Line line5 = new Line(); //left arm  
312 line5.startXProperty().bind(circle.centerXProperty().subtract(getWidth()\*0.05));  
313 line5.startYProperty().bind(line4.startYProperty().subtract(getHeight()\*0.009));  
314 line5.endXProperty().bind(line5.startXProperty().subtract(getWidth()\*0.1));  
315 line5.endYProperty().bind(line5.startYProperty().add(getHeight()\*0.05));  
316   
317 Line line6 = new Line(); //right arm  
318 line6.startXProperty().bind(circle.centerXProperty().add(getWidth()\*0.05));  
319 line6.startYProperty().bind(line4.startYProperty().subtract(getHeight()\*0.009));  
320 line6.endXProperty().bind(line5.startXProperty().add(getWidth()\*0.2));  
321 line6.endYProperty().bind(line5.startYProperty().add(getHeight()\*0.05));  
322   
323 Line line7 = new Line(); //left leg  
324 line7.startXProperty().bind(line4.endXProperty());  
325 line7.startYProperty().bind(line4.endYProperty());  
326 line7.endXProperty().bind(line7.startXProperty().subtract(getWidth()\*0.15));  
327 line7.endYProperty().bind(line7.startYProperty().add(getHeight()\*0.09));  
328   
329 Line line8 = new Line(); //right leg  
330 line8.startXProperty().bind(line4.endXProperty());  
331 line8.startYProperty().bind(line4.endYProperty());  
332 line8.endXProperty().bind(line7.startXProperty().add(getWidth()\*0.15));  
333 line8.endYProperty().bind(line7.startYProperty().add(getHeight()\*0.09));  
334   
335 Arc arc = new Arc();  
336 arc.centerXProperty().bind(line2.endXProperty().add(getWidth()\*0.005));  
337 arc.centerYProperty().bind(line2.endYProperty().add(getHeight()\*0.10));  
338 arc.radiusXProperty().bind(layoutXProperty().add(getWidth()\*0.15));  
339 arc.radiusYProperty().bind(layoutYProperty().add(getHeight()\*0.15));  
340 arc.setStartAngle(30);  
341 arc.setLength(120);  
342 arc.setFill(Color.WHITE);  
343 arc.setStroke(Color.BLACK);  
344   
345 getChildren().clear();  
346 getChildren().addAll(line1, line2, circle, line3, line4, line5, line6, line7, line8, arc);  
347 }  
348   
349 @Override  
350 public void setWidth(double width) {  
351 super.setWidth(width);  
352 paint();  
353 }  
354   
355 @Override  
356 public void setHeight(double height) {  
357 super.setHeight(height);  
358 paint();  
359 }  
360 }  
361 }

