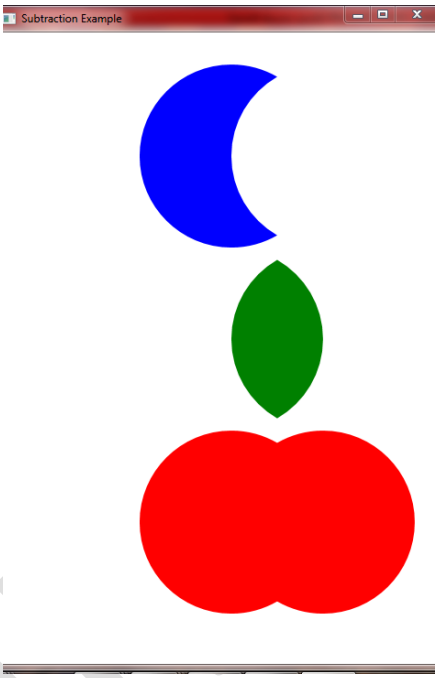
Operations on 2D Objects

If we add more than one shape to a group, the first shape is overlapped by the second one as shown below.

In addition to the transformations (rotate, scale, translate, etc.), transitions (animations), you can also perform three operations on 2D objects namely – **Union, Subtraction** and **Intersection**.

S.No	Operation & Description
1	**Union Operation** This operation takes two or more shapes as inputs and returns the area occupied by them.
2	**Intersection Operation** This operation takes two or more shapes as inputs and returns the intersection area between them.
3	**Subtraction Operation** This operation takes two or more shapes as an input. Then, it returns the area of the first shape excluding the area overlapped by the second one.

```
package javafxapplication32;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.Group;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Shape;

public class JavaFXApplication32 extends Application {

    @Override
    public void start(Stage stage) {
      //Drawing Circle1
      Circle circle1 = new Circle();

      //Setting the position of the circle
      circle1.setCenterX(250.0);
      circle1.setCenterY(135.0);

      //Setting the radius of the circle
     circle1.setRadius(100.0);
```




```
        //Setting the color of the circle
        circle1.setFill(Color.DARKSLATEBLUE);

        //Drawing Circle2
        Circle circle2 = new Circle();

        //Setting the position of the circle
        circle2.setCenterX(350.0);
        circle2.setCenterY(135.0);

        //Setting the radius of the circle
        circle2.setRadius(100.0);

        //Setting the color of the circle
        circle2.setFill(Color.BLUE);

        //Performing subtraction operation on the circle
        Shape shape1 = Shape.subtract(circle1, circle2);
        Shape shape2 = Shape.intersect(circle1, circle2);
        Shape shape3 = Shape.union(circle1, circle2);

        //Setting the fill color to the result
        shape1.setFill(Color.BLUE);
        shape2.setFill(Color.GREEN);
        shape3.setFill(Color.RED);

        shape1.setLayoutY(0);
        shape2.setLayoutY(200);
        shape3.setLayoutY(400);
        //Creating a Group object
        Group root = new Group();

        root.getChildren().addAll(shape1, shape2, shape3);

        //Creating a scene object
        Scene scene = new Scene(root, 600, 300);

        //Setting title to the Stage
        stage.setTitle("Subtraction Example");

        //Adding scene to the stage
        stage.setScene(scene);

        //Displaying the contents of the stage
        stage.show();
    }
}
```

Drawing More Shapes Through Path Class

The Path Class

To draw such complex structures JavaFX provides a class named **Path**. This class represents the geometrical outline of a shape.

It is attached to an observable list which holds various **Path Elements** such as moveTo, LineTo, HlineTo, VlineTo, ArcTo, QuadCurveTo, CubicCurveTo.

On instantiating, this class constructs a path based on the given path elements.

You can pass the path elements to this class while instantiating it as follows–

```
Path myshape = new Path(pathElement1, pathElement2, pathElement3);
```

Or, you can get the observable list and add all the path elements using **addAll()** method as follows –

```
Path myshape = new Path();
myshape.getElements().addAll(pathElement1, pathElement2, pathElement3);
```

You can also add elements individually using the add() method as –

```
Path myshape = new Path();
myshape.getElements().add(pathElement1);
```

The Move to Path Element

The Path Element **MoveTo** is used to move the current position of the path to a specified point. It is generally used to set the starting point of a shape drawn using the path elements.

It is represented by a class named **LineTo** of the package **javafx.scene.shape**. It has 2 properties of the double datatype namely –

- **X** – The x coordinate of the point to which a line is to be drawn from the current position.
- **Y** – The y coordinate of the point to which a line is to be drawn from the current position.

You can create a move to path element by instantiating the MoveTo class and passing the x, y coordinates of the new point as follows –

```
MoveTo moveTo = new MoveTo(x, y);
```

If you don't pass any values to the constructor, then the new point will be set to (0,0).

You can also set values to the x, y coordinate, using their respective setter methods as follows –

```
setX(value);
setY(value);
```

```
package javafxapplication33;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.Group;
import javafx.scene.shape.*;
import javafx.scene.shape.MoveTo;
import javafx.scene.shape.Path;


public class JavaFXApplication33 extends Application {

    @Override
    public void start(Stage stage) {
      //Creating a Path
      Path path = new Path();

      //Moving to the starting point
      MoveTo moveTo = new MoveTo(108, 71);

      //Creating 1st line
      LineTo line1 = new LineTo(321, 160);

      //Creating 2nd line
      VLineTo line2 = new VLineTo(126);

      //Creating 3rd line
      HLineTo line3 = new HLineTo(232);

      //Creating 4th line
      LineTo line4 = new LineTo(269, 250);

      //Creating 4th line
      LineTo line5 = new LineTo(108, 71);

       //Instantiating the class QuadCurve
      QuadCurveTo quadCurveTo = new QuadCurveTo();

      //Setting properties of the class QuadCurve
      quadCurveTo.setX(500.0);
      quadCurveTo.setY(220.0);
      quadCurveTo.setControlX(250.0);
      quadCurveTo.setControlY(0.0);

      //Instantiating the class CubicCurve
      CubicCurveTo cubicCurveTo = new CubicCurveTo();
```

```
        //Setting properties of the class CubicCurve
        cubicCurveTo.setControlX1(400.0);
        cubicCurveTo.setControlY1(40.0);
        cubicCurveTo.setControlX2(175.0);
        cubicCurveTo.setControlY2(250.0);
        cubicCurveTo.setX(500.0);
        cubicCurveTo.setY(150.0);

        //Instantiating the arcTo class
        ArcTo arcTo = new ArcTo();

        //setting properties of the path element arc
        arcTo.setX(300.0);
        arcTo.setY(50.0);

        arcTo.setRadiusX(50.0);
        arcTo.setRadiusY(50.0);

        //Adding all the elements to the path
        path.getElements().add(moveTo);
        path.getElements().addAll(cubicCurveTo, line1, line2,arcTo, line3, line4, line5, quadCurveTo);

        //Creating a Group object
        Group root = new Group(path);

        //Creating a scene object
        Scene scene = new Scene(root, 600, 300);

        //Setting title to the Stage
        stage.setTitle("Drawing an arc through a path");

        //Adding scene to the stage
        stage.setScene(scene);

        //Displaying the contents of the stage
        stage.show();
    }
    public static void main(String args[]){
        launch(args);
    }

}
```

- Text

Just like various shapes, you can also create a text node in JavaFX. The text node is represented by the class named **Text**, which belongs to the package **javafx.scene.text**. This class contains several properties to create text in JavaFX and modify its appearance. This class also inherits the Shape class which belongs to the package **javafx.scene.shape**.

Therefore, in addition to the properties of the text like font, alignment, line spacing, text, etc. It also inherits the basic shape node properties such as **strokeFill, stroke, strokeWidth, strokeType,** etc.

You can also apply decorations such as strike through; in which case a line is passed through the text. You can underline a text using the methods of the **Text** class.

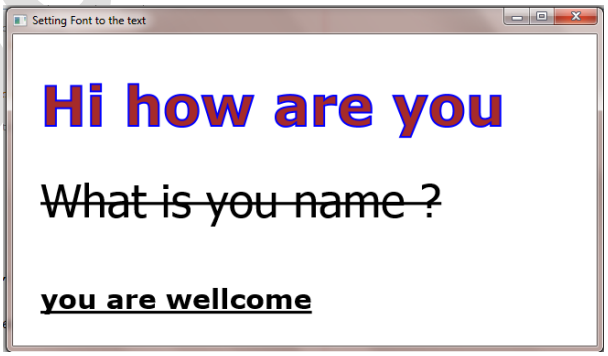
You can strike through the text using the method **setStrikethrough()**. This accepts a Boolean value, pass the value **true** to this method to strike through the text as shown in the following code box –

```
//Striking through the text
text1.setStrikethrough(true);
```

In the same way, you can underline a text by passing the value **true** to the method **setUnderLine()** as follows –

```
//underlining the text
text2.setUnderline(true);
```

```
package javafxapplication34;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.Group;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;
// you can use import javafx.scene.text.*; instead of the above 4 statemants
import javafx.scene.paint.Color;
public class JavaFXApplication34 extends Application {

    @Override
    public void start(Stage stage) {
      //Creating a Text object
      Text text = new Text();

      //Setting font to the text
      text.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 60));

      //setting the position of the text
      text.setX(30);
      text.setY(100);

      //Setting the text to be added.
      text.setText("Hi how are you");
```



```
        //Setting the color
        text.setFill(Color.BROWN);

        //Setting the Stroke
        text.setStrokeWidth(2);

        //Setting the stroke color
        text.setStroke(Color.BLUE);

        //Creating a Text_Example object
        Text text1 = new Text("What is you name ?");

        //Setting font to the text
        text1.setFont(Font.font("Tahoma", FontWeight.SEMI_BOLD, FontPosture.ITALIC, 50));

        //setting the position of the text
        text1.setX(30);
        text1.setY(200);


        //Striking through the text
        text1.setStrikethrough(true);

        //Creating a Text_Example object
        Text text2 = new Text("you are wellcome");

        //Setting font to the text
        text2.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 30));

        //setting the position of the text
        text2.setX(30);
        text2.setY(300);

        //underlining the text
        text2.setUnderline(true);

        //Creating a Group object
        Group root = new Group();
        root.getChildren().addAll(text, text1, text2);

        //Creating a scene object
        Scene scene = new Scene(root, 600, 300);

        //Setting title to the Stage
        stage.setTitle("Setting Font to the text");

        //Adding scene to the stage
        stage.setScene(scene);

        //Displaying the contents of the stage
        stage.show();
    }
}
```