Images in JavaFX

When using images in a JavaFX application, there are two important classes to familiarise oneself with. These are:

* **`Image`** - An `Image` is essentially a raw image object used only to store an image in memory.
* **`ImageView`** - An `ImageView` is a graphical UI element used to display, manipulate and render an image in a JavaFX application.

## Instantiating an `Image` Object (Loading an Image into Memory)

Loading an image into memory is done by instantiating a new `Image` object, passing its constructor a file path (as a string), web URL (as a string), or InputStream object.

**Passing a Relative File Path**

Image image = new Image("file:pic.png");

The current working directory of a Java application is the same location as the program's executable `.class` file. This should be considered when using relative file paths.

**Passing an Absolute File Path**

Image image = new Image("file:/Users/username/Pictures/pic.png");

*\*When passing a file path to the `Image` constructor, it should have the prefix `file:`.*

**Passing a Web URL**

Image image = new Image("https://www.image.com/resources/pic.png:);

*\*When loading an image from the web, you can use the http or https protocols.*

**Passing an `InputStream`**

InputStream inputStream = new FileInputStream("pic.png");

Image image = new Image(InputStream);

## Instantiating an `ImageView` Object

An `ImageView` object takes an `Image` as an argument to its constructor. The `ImageView` provides mechanisms to render the image on the screen, apply transformations and control how the image is displayed. An `ImageView` will typically be a node in the scene graph.

ImageView imageView = new ImageView(image);

## Working with `ImageView` Objects

An ImageView can be added to a container such as an HBox, VBox or StackPane.

Some important methods of the ImageView class include:

.setPreserveRatio(boolean b)

The `preserveRatio` field of an `ImageView` class is `false` by default. This means that when the `ImageView` is resized, it is stretched to meet the new dimensions without maintaining the original aspect ratio (unless this is manually calculated). When `preserveRatio` is set to `true` using the above method, the original aspect ratio is preserved.

.isPreserveRatio()

This returns the value in `preserveRatio`.

.setFitWidth(double v)

This sets the width of the `ImageView`. If `preserveRatio` is `true`, the height of the `ImageView` will automatically adjust to maintain the original aspect ratio.

.setFitHeight(double v)

This sets the height of the `ImageView`. If preserveRatio is `true`, the width of the `ImageView` will automatically adjust to maintain the original aspect ratio.

.setImage(Image image)

This sets / updates the `Image` in the `ImageView`.

## Separation of Concerns

The reason for the separation of the `Image` and `ImageView` classes in JavaFX is adherence to the principle of separation of concerns.

The `Image` class encapsulates raw image information, such as pixel data, dimensions, and loading properties. By separating the image data from the view, the framework allows developers to manage and manipulate image resources without being tied to how they are displayed.

The `ImageView` class is responsible for displaying an image on the screen. It provides methods for positioning, scaling, rotating, and applying effects to the image. By focusing solely on rendering, the `ImageView` class can be optimised for performance and flexibility in the user interface.

Some other functionality that benefits from the separation of concerns is lazy loading - the `Image` class supports lazy loading and caching, which can improve application performance by loading images only when needed and reusing them efficiently. The `ImageView` class can leverage these optimisations without being burdened by the complexities of image loading and management.

Most notably, however, is that JavaFX follows a design pattern where the model (data) is separated from the view (presentation). This is consistent with other components in JavaFX, where data handling and rendering are typically managed by different classes. This approach promotes clean, maintainable code and adheres to established software engineering principles.