|  |
| --- |
| **Exercise 17.2**  The placement of the text is always centered horizontally and vertically. |
| **Exercise 17.3**  There are six constructors in the Scene class. |
| **Exercise 17.5**  **They are stacked on top of each other and remain so, regardless of the size of the frame.** |
| **Exercise 17.8**  **The components are obscured rather than being made tiny.** |
| **Exercise 17.9**  **If there is no Centre component, the frame starts with its space filled by the other four but if the frame is expanded, the central area remains empty - the other components do not take up the spare space.** |
| **Exercise 17.10**  **A BorderPane and GridPane.** |
| **Exercise 17.19**  **Nothing happens because no actions have been set on the menu items.** |
| **Exercise 17.22-17.24**  **See** imageviewer0-3**.** |
| **Exercise 17.26**  **The image remains the same size and the additional space is allocated to the left, right and bottom positions.** |
| **Exercise 17.27**  **The method is** setAccelerator**.** |
| **Exercise 17.28**  private void openFile(File file)  {  OFImage currentImage = ImageFileManager.loadImage(file);    if (currentImage != null) {  imageLabel.setGraphic(new ImageView(currentImage));  filenameLabel.setText("File: " + file.getPath());  stage.sizeToScene();  }  } |
| **Exercise 17.29**  private void closeAction(ActionEvent event)  {  currentImage = null;  imageLabel.setGraphic(  new ImageView(ImageFileManager.loadImage("emptyImage.png")));  filenameLabel.setText("No file displayed.");  statusLabel.setText(VERSION);  stage.sizeToScene();  } |
| **Exercise 17.30-17.35**  **See** imageviewer1-0**.** |
| **Exercise 17.36**  **In** makeMenuBar**:**  Menu helpMenu = new Menu("Help");  MenuItem aboutItem = new MenuItem("About ImageViewer...");  aboutItem.setOnAction(this::showAbout);  helpMenu.getItems().addAll(aboutItem);  menubar.getMenus().addAll(fileMenu, filterMenu, helpMenu); |
| **Exercise 17.37**  **AlertType predefines the following, with the API descriptions:**   * **CONFIRMATION:** suggests the content of the dialog is seeking confirmation from the user. * **ERROR:** suggests that something has gone wrong. * **INFORMATION:** suggests the content of the dialog is informing the user of a piece of information. * **NONE: does** not set any default properties in the Alert. * **WARNING:** suggests the content of the dialog is warning the user about some fact or action. |
| **Exercise 17.39**  **CONFIRMATION has OK and Cancel buttons.** |
| **Exercise 17.41**  private void showInvalidFileFormatError()  {  Alert alert = new Alert(AlertType.ERROR);  alert.setTitle("Image Load Error");  alert.setHeaderText(null);  alert.setContentText("The file was not in a recognized image file format.");  alert.showAndWait();  } |
| **Exercise 17.43**  **Once the new filter has been implemented as a subclass of** Filter**, little needs to be added to the** ImageViewer **class. An instance of the new class needs to be added to the list of filters in the** createFilters() **method. The rest is handled via polymorphism.** |
| **Exercise 17.44**  **See** imageviewer2-1**.** |
| **Exercise 17.45-17.50**  **See the** imageviewer-final **project.** |
| **Exercise 17.53-17.57**  **See the** imageviewer3-0 **project.** |
| **Exercise 17.58**  **The padding property affects the amount of** padding around a region's content. If padding is increased then the label/button/etc. will have additional space around its content and appear larger, where permitted. |
| **Exercise 17.59**  **The file is empty.** |
| **Exercise 17.61**  **The property is** -fx-underline **and it has a Boolean type.** |
| **Exercise 17.62-17.66**  **See the** imageviewer3-0 **project.** |
| **Exercise 17.67**  **The API page for the javafx.event.Event class lists some of its subtypes, which include DialogEvent, InputEvent and WindowEvent.** |
| **Exercise 17.68**  **From the API, '**AnchorPane allows the edges of child nodes to be anchored to an offset from the anchor pane's edges.' |
| **Exercise 17.69**  **From the API, '**The Slider Control is used to display a continuous or discrete range of valid numeric choices and allows the user to interact with the control. It is typically represented visually as having a "track" and a "knob" or "thumb" which is dragged within the track. The Slider can optionally show tick marks and labels indicating the different slider position values.' |
| **Exercise 17.70**  **From the API, '**A control that allows switching between a group of [Tabs](https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Tab.html). Only one tab is visible at a time.' |
| **Exercise 17.71**  **From the API, '**A single line text field that lets the user select a number or an object value from an ordered sequence. Spinners typically provide a pair of tiny arrow buttons for stepping through the elements of the sequence.' |