Graphics Using a Two-dimensional Data

Mitsu Ogihara

Department of Computer Science
University of Miami

Problem

Write an application for drawing overlapping rectangles on a graphic window where

- for each rectangle it receives from the user
  - the location and the size by way of the upper left corner and the lower right corner,
  - the color chosen from the preset colors;
- at the conclusion of the program receives from the user the file name and stores the image as a JPEG file

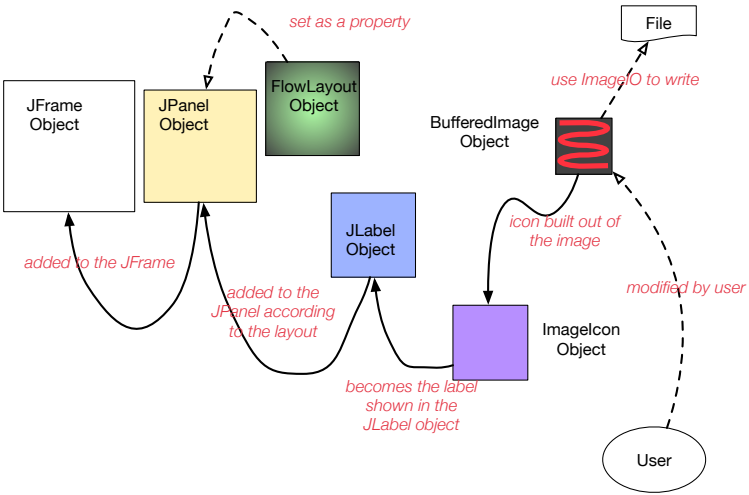
Pixel Graphics

- A graphic display is pixel-based, i.e., it is a large two-dimensional array of pixel, which consists of color and depth
- In Java it is possible to geterate an image using this feature
- Unlike the table-like view we have been using in data arrays, the pixel array is horizontal-by-vertical; i.e., the first dimension is for the horizontal coordinate and the second is for the vertical

New Imported Classes for the Application

- `JFrame`: The main graphic window object for Java
- `JPanel`: A virtual panel-like structure on which to place Java graphic objects
- `FlowLayout`: The class for specifying an inside-panel object layout, where a new object added are appended in a single line from left to right
- `JLabel`:A graphic object class for showing either text or icon
- `ImageIcon`: A graphic object class for generating an icon from an image type
- `BufferedImage`: A graphic image class in which the image is specified by its bitmap
- `ImageIO`: A class for writing an image object to a file
- `Color`: A class for creating and interpreting colors
- `JOptionPane`: A class for receiving input from user by way of pop-ups

Relations Among the Classes Used



More About How the Program Works

- Create a `BufferedImage` object `image` by specifying its dimension
  - the # of pixels horizontal
  - the # of pixels vertical
  - the color format to be used
- The color of any pixel of `image` can be changed
  - Set all of them to the default color
  - Draw rectangles by changing the color information of the pixels
- Using `ImageIO`'s `write` method, save `image` as a JPEG file

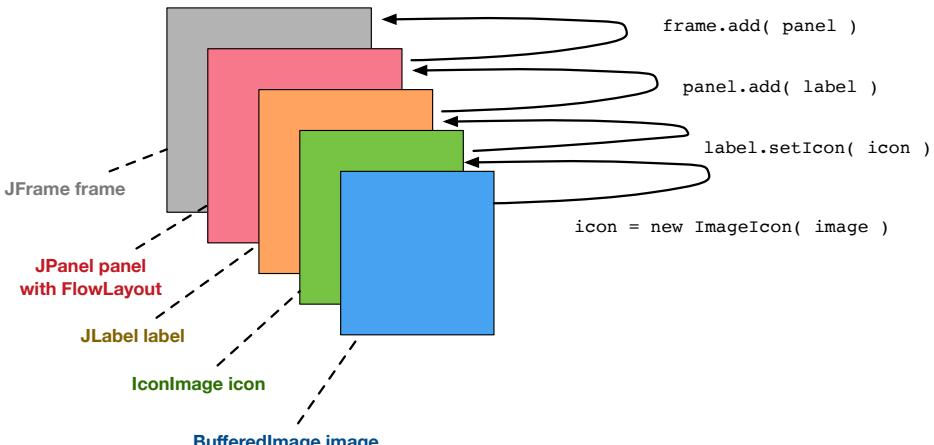
How the Program Works (cont'd)

- Create a `ImageIcon` object `icon` from `image`
- Create a `JLabel` object `label` and assign `icon` to `label`
- Create a `JPanel` object `panel`
- Set the layout of `panel` to a `FlowLayout` object
  - A `FlowLayout` object must be used since, `FlowLayout` allows changes in its property (e.g., the gaps between the aligned objects)
  - Since there is only one project aligned, these properties can be ignored
- Add `label` to `panel`
- Add `panel` to `frame`

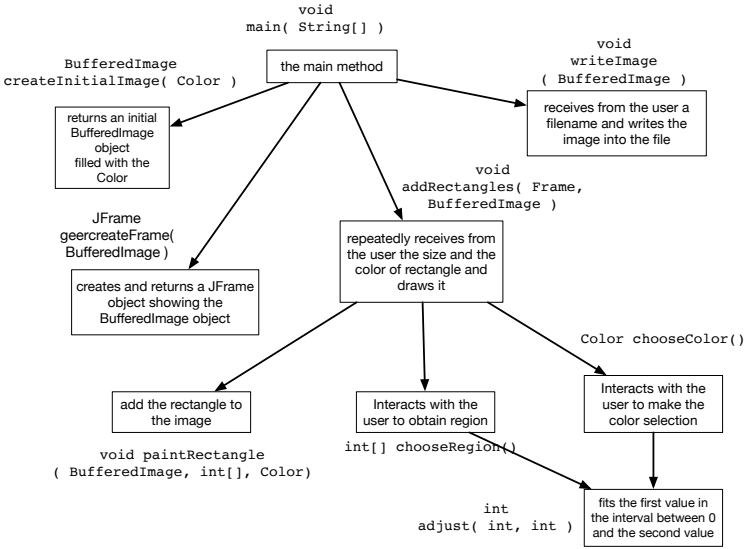
Additional Details

- Set the property `EXIT_ON_CLOSE` of `frame` so that the program quits when the frame is closed
- After adding `panel` to `frame`, execute `frame.pack()` to show `panel` inside `frame` with some wiggle room
- Execute `frame.setVisible( true )` to make the frame visible
- After that:
  - by executing `frame.repaint()`, redraw the graphics, and
  - by executing `frame.dispose()`, the graphics can be removed from the screen

Relations Among the Graphics Objects



Relations Among Methods



Rectangle Drawing

- Occurs in the method `addRectangles`
- Use a do-while loop whose termination condition is the string `response` starts with `"y"`
- Use `JOptionPane.showInputDialog( ARGUMENT )` to receive input:
  - The method generates a text box pop up and returns the text entered by the user
  - ARGUMENT is the string to be printed in the window above the text box
- If the answer starts with `"y"`, add a rectangle; otherwise, the loop quits and the method returns

Rectangle Drawing (If the response starts with "y")

- Using `JOptionPane.showInputDialog` to receive from the user the coordinates for the upper left corner and for the lower left corner
  - The values appear in one line
  - Trim the String using `trim` method
  - Replace any double white space to a single white space
  - Split the String using the white space as the delimiter
  - Convert the array elements to four numbers
  - Using method `adjust` ensure that the numbers are in the valid range between 0 and `WIDTH` for a horizontal value and between 0 and `HEIGHT` for a vertical value
- Ask the user to select a color
- Use a double for-loop to set the pixels inside the rectangle to the color: `image.setRGB( i, j, Color.GREEN.getRGB() )`, where the first two parameters are the coordinates and the last is the integer value of the color green, as define din class `Color`

Writing the Image

- Receive from the user a filename path
- If the filename path ends with `".jpg"` (the desingated extension for a JPEG file), use `ImageIO.write` to the write the image as a jpg file to the specified file

The Code: PlayWithImage.java (header)

```
1  // lots of imports for this class
2  import java.awt.image.BufferedImage;
3  import java.awt.FlowLayout;
4  import java.awt.Color;
5  import javax.imageio.ImageIO;
6  import javax.swing.JFrame;
7  import javax.swing.JPanel;
8  import javax.swing.JLabel;
9  import javax.swing.ImageIcon;
10 import javax.swing.JOptionPane;
11 import java.io.File;
12 import java.io.IOException;
13
14 // play with drawing colored rectangles
15 public class PlayWithImage {
```

Three classes imported from `java.awt`

The Code: PlayWithImage.java (header)

```
1  // lots of imports for this class
2  import java.awt.image.BufferedImage;
3  import java.awt.FlowLayout;
4  import java.awt.Color;
5  import javax.imageio.ImageIO;
6  import javax.swing.JFrame;
7  import javax.swing.JPanel;
8  import javax.swing.JLabel;
9  import javax.swing.ImageIcon;
10 import javax.swing.JOptionPane;
11 import java.io.File;
12 import java.io.IOException;
13
14 // play with drawing colored rectangles
15 public class PlayWithImage {
```

Classe imported from `javax.imageio`

The Code: PlayWithImage.java (header)

```
1  // lots of imports for this class
2  import java.awt.image.BufferedImage;
3  import java.awt.FlowLayout;
4  import java.awt.Color;
5  import javax.imageio.ImageIO;
6  import javax.swing.JFrame;
7  import javax.swing.JPanel;
8  import javax.swing.JLabel;
9  import javax.swing.ImageIcon;
10 import javax.swing.JOptionPane;
11 import java.io.File;
12 import java.io.IOException;
13
14 // play with drawing colored rectangles
15 public class PlayWithImage {
```

Five classes imported from `javax.swing`

The Code: PlayWithImage.java (header)

```
1  // lots of imports for this class
2  import java.awt.image.BufferedImage;
3  import java.awt.FlowLayout;
4  import java.awt.Color;
5  import javax.imageio.ImageIO;
6  import javax.swing.JFrame;
7  import javax.swing.JPanel;
8  import javax.swing.JLabel;
9  import javax.swing.ImageIcon;
10 import javax.swing.JOptionPane;
11 import java.io.File;
12 import java.io.IOException;
13
14 // play with drawing colored rectangles
15 public class PlayWithImage {
```

Two classes from `java.io`

The Code: PlayWithImage.java (header)

```
1  // lots of imports for this class
2  import java.awt.image.BufferedImage;
3  import java.awt.FlowLayout;
4  import java.awt.Color;
5  import javax.imageio.ImageIO;
6  import javax.swing.JFrame;
7  import javax.swing.JPanel;
8  import javax.swing.JLabel;
9  import javax.swing.ImageIcon;
10 import javax.swing.JOptionPane;
11 import java.io.File;
12 import java.io.IOException;
13
14 // play with drawing colored rectangles
15 public class PlayWithImage {
```

The class declaration

The Code: PlayWithImage.java (main)

```
16    ////////////////////////////////////////////////////////
17    // MAIN METHOD
18    ////////////////////////////////////////////////////////
19    public static void main( String[] args ) {
20      BufferedImage image = generateInitialImage( Color.BLACK );
21      JFrame frame = generateFrame( image );
22      addRectangles( frame, image );
23      writeImage( image );
24      frame.dispose();
25    }
```

Call `generateInitialImage` to create an initial image

The Code: PlayWithImage.java (main)

```
16    ////////////////////////////////////////////////////////
17    // MAIN METHOD
18    ////////////////////////////////////////////////////////
19    public static void main( String[] args ) {
20      BufferedImage image = generateInitialImage( Color.BLACK );
21      JFrame frame = generateFrame( image );
22      addRectangles( frame, image );
23      writeImage( image );
24      frame.dispose();
25    }
```

Call `generateFrame` with the image as a parameter to create a frame

The Code: PlayWithImage.java (main)

```
16    ////////////////////////////////////////////////////////
17    // MAIN METHOD
18    ////////////////////////////////////////////////////////
19    public static void main( String[] args ) {
20      BufferedImage image = generateInitialImage( Color.BLACK );
21      JFrame frame = generateFrame( image );
22      addRectangles( frame, image );
23      writeImage( image );
24      frame.dispose();
25    }
```

Call `addRectangles` with the image and the frame as parameters to perform interactions with the user to add rectangles

The Code: PlayWithImage.java (main)

```
16    ////////////////////////////////////////////////////////
17    // MAIN METHOD
18    ////////////////////////////////////////////////////////
19    public static void main( String[] args ) {
20      BufferedImage image = generateInitialImage( Color.BLACK );
21      JFrame frame = generateFrame( image );
22      addRectangles( frame, image );
23      writeImage( image );
24      frame.dispose();
25    }
```

Call `writeImage` with the image as parameter to write the image to a file

The Code: PlayWithImage.java (main)

```
16    ////////////////////////////////////////////////////////
17    // MAIN METHOD
18    ////////////////////////////////////////////////////////
19    public static void main( String[] args ) {
20      BufferedImage image = generateInitialImage( Color.BLACK );
21      JFrame frame = generateFrame( image );
22      addRectangles( frame, image );
23      writeImage( image );
24      frame.dispose();
25    }
```

Dispose the frame to close the program

The Code: PlayWithImage.java (generateInitialImage)

```
27    ////////////////////////////////////////////////////////
28    // INITIAL IMAGE CREATION
29    ////////////////////////////////////////////////////////
30
31    //--- size constants
32    public static final int WIDTH = 600;
33    public static final int HEIGHT = 700;
34
35    //--- method for creating an initial amage
36    public static BufferedImage generateInitialImage(
37         Color initialColor ) {
38      BufferedImage image = new BufferedImage(
39           WIDTH, HEIGHT, BufferedImage.TYPE_3BYTE_BGR );
40      for ( int i = 0; i < WIDTH; i ++ ) {
41        for ( int j = 0; j < HEIGHT; j ++ ) {
42          image.setRGB( i, j, initialColor.getRGB() );
43        }
44      }
45      return image;
46    }
```

Dimensional constants for image width and height

The Code: PlayWithImage.java (generateInitialImage)

```
27    ////////////////////////////////////////////////////////
28    // INITIAL IMAGE CREATION
29    ////////////////////////////////////////////////////////
30
31    //--- size constants
32    public static final int WIDTH = 600;
33    public static final int HEIGHT = 700;
34
35    //--- method for creating an initial amage
36    public static BufferedImage generateInitialImage(
37         Color initialColor ) {
38      BufferedImage image = new BufferedImage(
39           WIDTH, HEIGHT, BufferedImage.TYPE_3BYTE_BGR );
40      for ( int i = 0; i < WIDTH; i ++ ) {
41        for ( int j = 0; j < HEIGHT; j ++ ) {
42          image.setRGB( i, j, initialColor.getRGB() );
43        }
44      }
45      return image;
46    }
```

The method receives the color to fill the initial image color, `Color initialColor` and returns a new image

The Code: PlayWithImage.java (generateInitialImage)

```
27    ////////////////////////////////////////////////////////
28    // INITIAL IMAGE CREATION
29    ////////////////////////////////////////////////////////
30
31    //--- size constants
32    public static final int WIDTH = 600;
33    public static final int HEIGHT = 700;
34
35    //--- method for creating an initial amage
36    public static BufferedImage generateInitialImage(
37         Color initialColor ) {
38      BufferedImage image = new BufferedImage(
39          WIDTH, HEIGHT, BufferedImage.TYPE_3BYTE_BGR );
40      for ( int i = 0; i < WIDTH; i ++ ) {
41        for ( int j = 0; j < HEIGHT; j ++ ) {
42          image.setRGB( i, j, initialColor.getRGB() );
43        }
44      }
45      return image;
46    }
```

Create a new `BufferedImage` object `image` with the predetermined dimensions and with `BufferedImage.TYPE_3BYTE_BGR` as the color specification type

The Code: PlayWithImage.java (generateInitialImage)

```
27    ////////////////////////////////////////////////////////
28    // INITIAL IMAGE CREATION
29    ////////////////////////////////////////////////////////
30
31    //--- size constants
32    public static final int WIDTH = 600;
33    public static final int HEIGHT = 700;
34
35    //--- method for creating an initial amage
36    public static BufferedImage generateInitialImage(
37         Color initialColor ) {
38      BufferedImage image = new BufferedImage(
39          WIDTH, HEIGHT, BufferedImage.TYPE_3BYTE_BGR );
40      for ( int i = 0; i < WIDTH; i ++ ) {
41        for ( int j = 0; j < HEIGHT; j ++ ) {
42          image.setRGB( i, j, initialColor.getRGB() );
43        }
44      }
45      return image;
46    }
```

Use a double for loop to set the color of the pixels to white

The Code: PlayWithImage.java (generateInitialImage)

```
27    ////////////////////////////////////////////////////////
28    // INITIAL IMAGE CREATION
29    ////////////////////////////////////////////////////////
30
31    //--- size constants
32    public static final int WIDTH = 600;
33    public static final int HEIGHT = 700;
34
35    //--- method for creating an initial amage
36    public static BufferedImage generateInitialImage(
37         Color initialColor ) {
38      BufferedImage image = new BufferedImage(
39           WIDTH, HEIGHT, BufferedImage.TYPE_3BYTE_BGR );
40      for ( int i = 0; i < WIDTH; i ++ ) {
41        for ( int j = 0; j < HEIGHT; j ++ ) {
42          image.setRGB( i, j, initialColor.getRGB() );
43        }
44      }
45      return image;
46    }
```

Return the image object

The Code: PlayWithImage.java (generateFrame)

```
48    ////////////////////////////////////////////////////////
49    // GENERATE JFRAME
50    ////////////////////////////////////////////////////////
51    public static JFrame generateFrame( BufferedImage image ) {
52      //--- create an ImageIcon from the image and
53      //--- set it to a JLabel
54      ImageIcon icon = new ImageIcon( image );
55      JLabel label = new JLabel();
56      label.setIcon( icon );
57
58      //--- crate a JPanel and add the JLabel object to it
59      JPanel panel = new JPanel();
60      panel.setLayout( new FlowLayout() );
61      panel.add( label );
```

Method header. The method takes as a parameter a `BufferedImage` object `image` and returns a `JFrame` object

The Code: PlayWithImage.java (generateFrame)

```
48    ////////////////////////////////////////////////////////
49    // GENERATE JFRAME
50    ////////////////////////////////////////////////////////
51    public static JFrame generateFrame( BufferedImage image ) {
52      //--- create an ImageIcon from the image and
53      //--- set it to a JLabel
54      ImageIcon icon = new ImageIcon( image );
55      JLabel label = new JLabel();
56      label.setIcon( icon );
57
58      //--- crate a JPanel and add the JLabel object to it
59      JPanel panel = new JPanel();
60      panel.setLayout( new FlowLayout() );
61      panel.add( label );
```

Create an `ImageIcon` object `icon` from `image`

The Code: PlayWithImage.java (generateFrame)

```
48    ////////////////////////////////////////////////////////
49    // GENERATE JFRAME
50    ////////////////////////////////////////////////////////
51    public static JFrame generateFrame( BufferedImage image ) {
52      //--- create an ImageIcon from the image and
53      //--- set it to a JLabel
54      ImageIcon icon = new ImageIcon( image );
55      JLabel label = new JLabel();
56      label.setIcon( icon );
57
58      //--- crate a JPanel and add the JLabel object to it
59      JPanel panel = new JPanel();
60      panel.setLayout( new FlowLayout() );
61      panel.add( label );
```

Create a `JLabel` object

The Code: PlayWithImage.java (generateFrame)

```
48    ////////////////////////////////////////////////////////
49    // GENERATE JFRAME
50    ////////////////////////////////////////////////////////
51    public static JFrame generateFrame( BufferedImage image ) {
52      //--- create an ImageIcon from the image and
53      //--- set it to a JLabel
54      ImageIcon icon = new ImageIcon( image );
55      JLabel label = new JLabel();
56      label.setIcon( icon );
57
58      //--- crate a JPanel and add the JLabel object to it
59      JPanel panel = new JPanel();
60      panel.setLayout( new FlowLayout() );
61      panel.add( label );
```

Assign `icon` to the icon of the label using `setIcon`

The Code: PlayWithImage.java (generateFrame)

```
48    ////////////////////////////////////////////////////////
49    // GENERATE JFRAME
50    ////////////////////////////////////////////////////////
51    public static JFrame generateFrame( BufferedImage image ) {
52      //--- create an ImageIcon from the image and
53      //--- set it to a JLabel
54      ImageIcon icon = new ImageIcon( image );
55      JLabel label = new JLabel();
56      label.setIcon( icon );
57
58      //--- crate a JPanel and add the JLabel object to it
59      JPanel panel = new JPanel();
60      panel.setLayout( new FlowLayout() );
61      panel.add( label );
```

Create a new `JPanel` object `panel`

The Code: PlayWithImage.java (generateFrame)

```
48    ////////////////////////////////////////////////////////
49    // GENERATE JFRAME
50    ////////////////////////////////////////////////////////
51    public static JFrame generateFrame( BufferedImage image ) {
52      //--- create an ImageIcon from the image and
53      //--- set it to a JLabel
54      ImageIcon icon = new ImageIcon( image );
55      JLabel label = new JLabel();
56      label.setIcon( icon );
57
58      //--- crate a JPanel and add the JLabel object to it
59      JPanel panel = new JPanel();
60      panel.setLayout( new FlowLayout() );
61      panel.add( label );
```

Assign a new `FlowLayout` object to the panel using the `setLayout` method

The Code: PlayWithImage.java (generateFrame)

```
48    ////////////////////////////////////////////////////////
49    // GENERATE JFRAME
50    ////////////////////////////////////////////////////////
51    public static JFrame generateFrame( BufferedImage image ) {
52      //--- create an ImageIcon from the image and
53      //--- set it to a JLabel
54      ImageIcon icon = new ImageIcon( image );
55      JLabel label = new JLabel();
56      label.setIcon( icon );
57
58      //--- crate a JPanel and add the JLabel object to it
59      JPanel panel = new JPanel();
60      panel.setLayout( new FlowLayout() );
61      panel.add( label );
```

Add `label` to `panel`

generateFrame (cont'd)

```
63      //--- crate a JFrame and add the JLabel object to it
64      JFrame frame = new JFrame();
65      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
66      frame.add( panel );
67      frame.pack();
68
69      //--- make the JFrame object visible and return it
70      frame.setVisible( true );
71      return frame;
72    }
```

Create a `JFrame` object `frame`

generateFrame (cont'd)

```
63      //--- crate a JFrame and add the JLabel object to it
64      JFrame frame = new JFrame();
65      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
66      frame.add( panel );
67      frame.pack();
68
69      //--- make the JFrame object visible and return it
70      frame.setVisible( true );
71      return frame;
72    }
```

Set the close option to the one that forces program termination upon closing the frame

generateFrame (cont'd)

```
63      //--- crate a JFrame and add the JLabel object to it
64      JFrame frame = new JFrame();
65      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
66      frame.add( panel );
67      frame.pack();
68
69      //--- make the JFrame object visible and return it
70      frame.setVisible( true );
71      return frame;
72    }
```

Add `panel`

generateFrame (cont'd)

```
63      //--- crate a JFrame and add the JLabel object to it
64      JFrame frame = new JFrame();
65      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
66      frame.add( panel );
67      frame.pack();
68
69      //--- make the JFrame object visible and return it
70      frame.setVisible( true );
71      return frame;
72    }
```

Call `frame.pack()` to add some wiggle room surrounding the panel

generateFrame (cont'd)

```
63      //--- crate a JFrame and add the JLabel object to it
64      JFrame frame = new JFrame();
65      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
66      frame.add( panel );
67      frame.pack();
68
69      //--- make the JFrame object visible and return it
70      frame.setVisible( true );
71      return frame;
72    }
```

Make `frame` visible

generateFrame (cont'd)

```
63      //--- crate a JFrame and add the JLabel object to it
64      JFrame frame = new JFrame();
65      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
66      frame.add( panel );
67      frame.pack();
68
69      //--- make the JFrame object visible and return it
70      frame.setVisible( true );
71      return frame;
72    }
```

Return `frame`

The Code: PlayWithImage.java (adjust)

```
74    ////////////////////////////////////////////////////////
75    // TOOL FOR ADJUSTING A NUMBER BETWEEN 0 AND MAX (EXCLUSIVE)
76    ////////////////////////////////////////////////////////////
77    public static int adjust( int value, int max ) {
78      if ( value < 0 || max < 0 ) {
79        return 0;
80      }
81      return Math.min( value, max - 1 );
82    }
```

Method header; it takes the value and the maximum and returns an integer

The Code: PlayWithImage.java (adjust)

```
74    ////////////////////////////////////////////////////////
75    // TOOL FOR ADJUSTING A NUMBER BETWEEN 0 AND MAX (EXCLUSIVE)
76    ////////////////////////////////////////////////////////////
77    public static int adjust( int value, int max ) {
78      if ( value < 0 || max < 0 ) {
79        return 0;
80      }
81      return Math.min( value, max - 1 );
82    }
```

If either the value or the maximum is negative, return 0

The Code: PlayWithImage.java (adjust)

```
74    ////////////////////////////////////////////////////////
75    // TOOL FOR ADJUSTING A NUMBER BETWEEN 0 AND MAX (EXCLUSIVE)
76    ////////////////////////////////////////////////////////
77    public static int adjust( int value, int max ) {
78      if ( value < 0 || max < 0 ) {
79        return 0;
80      }
81      return Math.min( value, max - 1 );
82    }
```

Otherwise, return the minimum of the value received and maximum -1

The Code: PlayWithImage.java (chooseColor)

```
84    ////////////////////////////////////////////////////////
85    // COLOR OPERATION
86    ////////////////////////////////////////////////////////
87    //--- color choices
88    public static Color[] COLORS = {
89      Color.RED, Color.YELLOW, Color.GREEN, Color.BLUE,
90      Color.MAGENTA, Color.BLACK, Color.WHITE };
91    public static String COLOR_CHOICE_MESSAGE =
92      "Choose color\n" +
93      "0. Red, 1. Yellow, 2. Green, 3. Blue\n" +
94      "4. Magenta, 5. Black, 6. White";
95    //--- choose color
96    public static Color chooseColor() {
97      String response = JOptionPane.showInputDialog( COLOR_CHOICE_MESSAGE );
98      int choice = 0;  // default value
99      try {
100       choice = Integer.parseInt( response );
101     } catch( NumberFormatException e ) { // use default
102     }
103     choice = adjust( choice, COLORS.length );
104     return COLORS[ choice ];
105   }
```

The Color array

The Code: PlayWithImage.java (chooseColor)

```
84    ////////////////////////////////////////////////////////
85    // COLOR OPERATION
86    ////////////////////////////////////////////////////////
87    //--- color choices
88    public static Color[] COLORS = {
89      Color.RED, Color.YELLOW, Color.GREEN, Color.BLUE,
90      Color.MAGENTA, Color.BLACK, Color.WHITE };
91    public static String COLOR_CHOICE_MESSAGE =
92      "Choose color\n" +
93      "0. Red, 1. Yellow, 2. Green, 3. Blue\n" +
94      "4. Magenta, 5. Black, 6. White";
95    //--- choose color
96    public static Color chooseColor() {
97      String response = JOptionPane.showInputDialog( COLOR_CHOICE_MESSAGE );
98      int choice = 0;  // default value
99      try {
100       choice = Integer.parseInt( response );
101     } catch( NumberFormatException e ) { // use default
102     }
103     choice = adjust( choice, COLORS.length );
104     return COLORS[ choice ];
105   }
```

The message to present when receiving a color selection

The Code: PlayWithImage.java (chooseColor)

```
84    ////////////////////////////////////////////////////////
85    // COLOR OPERATION
86    ////////////////////////////////////////////////////////
87    //--- color choices
88    public static Color[] COLORS = {
89      Color.RED, Color.YELLOW, Color.GREEN, Color.BLUE,
90      Color.MAGENTA, Color.BLACK, Color.WHITE };
91    public static String COLOR_CHOICE_MESSAGE =
92      "Choose color\n" +
93      "0. Red, 1. Yellow, 2. Green, 3. Blue\n" +
94      "4. Magenta, 5. Black, 6. White";
95    //--- choose color
96    public static Color chooseColor() {
97      String response = JOptionPane.showInputDialog( COLOR_CHOICE_MESSAGE );
98      int choice = 0;  // default value
99      try {
100       choice = Integer.parseInt( response );
101     } catch( NumberFormatException e ) { // use default
102     }
103     choice = adjust( choice, COLORS.length );
104     return COLORS[ choice ];
105   }
```

Method header

The Code: PlayWithImage.java (chooseColor)

```
84    ////////////////////////////////////////////////////////
85    // COLOR OPERATION
86    ////////////////////////////////////////////////////////
87    //--- color choices
88    public static Color[] COLORS = {
89      Color.RED, Color.YELLOW, Color.GREEN, Color.BLUE,
90      Color.MAGENTA, Color.BLACK, Color.WHITE };
91    public static String COLOR_CHOICE_MESSAGE =
92      "Choose color\n" +
93      "0. Red, 1. Yellow, 2. Green, 3. Blue\n" +
94      "4. Magenta, 5. Black, 6. White";
95    //--- choose color
96    public static Color chooseColor() {
97      String response = JOptionPane.showInputDialog( COLOR_CHOICE_MESSAGE );
98      int choice = 0;  // default value
99      try {
100       choice = Integer.parseInt( response );
101     } catch( NumberFormatException e ) { // use default
102     }
103     choice = adjust( choice, COLORS.length );
104     return COLORS[ choice ];
105   }
```

Receive response from the user using `JOptionPane.showInputDialog` with the message

The Code: PlayWithImage.java (chooseColor)

```
84    ////////////////////////////////////////////////////////
85    // COLOR OPERATION
86    ////////////////////////////////////////////////////////
87    //--- color choices
88    public static Color[] COLORS = {
89      Color.RED, Color.YELLOW, Color.GREEN, Color.BLUE,
90      Color.MAGENTA, Color.BLACK, Color.WHITE };
91    public static String COLOR_CHOICE_MESSAGE =
92      "Choose color\n" +
93      "0. Red, 1. Yellow, 2. Green, 3. Blue\n" +
94      "4. Magenta, 5. Black, 6. White";
95    //--- choose color
96    public static Color chooseColor() {
97      String response = JOptionPane.showInputDialog( COLOR_CHOICE_MESSAGE );
98      int choice = 0;  // default value
99      try {
100       choice = Integer.parseInt( response );
101     } catch( NumberFormatException e ) { // use default
102     }
103     choice = adjust( choice, COLORS.length );
104     return COLORS[ choice ];
105   }
```

`choice` is the color choice in integer; initialize to 0

The Code: PlayWithImage.java (chooseColor)

```
84    ////////////////////////////////////////////////////////
85    // COLOR OPERATION
86    ////////////////////////////////////////////////////////
87    //--- color choices
88    public static Color[] COLORS = {
89      Color.RED, Color.YELLOW, Color.GREEN, Color.BLUE,
90      Color.MAGENTA, Color.BLACK, Color.WHITE };
91    public static String COLOR_CHOICE_MESSAGE =
92      "Choose color\n" +
93      "0. Red, 1. Yellow, 2. Green, 3. Blue\n" +
94      "4. Magenta, 5. Black, 6. White";
95    //--- choose color
96    public static Color chooseColor() {
97      String response = JOptionPane.showInputDialog( COLOR_CHOICE_MESSAGE );
98      int choice = 0;  // default value
99      try {
100       choice = Integer.parseInt( response );
101     } catch( NumberFormatException e ) { // use default
102     }
103     choice = adjust( choice, COLORS.length );
104     return COLORS[ choice ];
105   }
```

Interpret the response as integer; if error, the initial value of 0 will be used

The Code: PlayWithImage.java (chooseColor)

```
84    ////////////////////////////////////////////////////////
85    // COLOR OPERATION
86    ////////////////////////////////////////////////////////
87    //--- color choices
88    public static Color[] COLORS = {
89      Color.RED, Color.YELLOW, Color.GREEN, Color.BLUE,
90      Color.MAGENTA, Color.BLACK, Color.WHITE };
91    public static String COLOR_CHOICE_MESSAGE =
92      "Choose color\n" +
93      "0. Red, 1. Yellow, 2. Green, 3. Blue\n" +
94      "4. Magenta, 5. Black, 6. White";
95    //--- choose color
96    public static Color chooseColor() {
97      String response = JOptionPane.showInputDialog( COLOR_CHOICE_MESSAGE );
98      int choice = 0;  // default value
99      try {
100       choice = Integer.parseInt( response );
101     } catch( NumberFormatException e ) { // use default
102     }
103     choice = adjust( choice, COLORS.length );
104     return COLORS[ choice ];
105   }
```

Adjust the choice using the array `COLORS`'s length as the max; and returns the color chosen

The Code: PlayWithImage.java (chooseRegion)

```
108   ////////////////////////////////////////////////////////
109   // RECEIVE RECTANGLE REGION
110   ////////////////////////////////////////////////////////
111   public static final String DIM_MESSAGE =
112       "Enter upperLeftCorner's X Y and lowerRighthCorner's X Y\n" +
113        "with a white space in between.\n" +
114        "The range for X is 0 .. " + WIDTH + ".\n" +
115        "The range for Y is 0 .. " + HEIGHT + ".";
116
117   public static int[] chooseRegion() throws Exception {
118     //--- receive dimensional input in one line
119     String input = JOptionPane.showInputDialog( DIM_MESSAGE );
120     String[] parts = input.trim().replace( "  ", " " ).split( " " );
121     int[] region = new int[ 4 ];
122     for ( int i = 0; i < 4; i ++ ) {
123       region[ i ]  = Integer.parseInt( parts[ i ] );
124     }
125     region[ 0 ] = adjust( region[ 0 ], WIDTH );
126     region[ 1 ] = adjust( region[ 1 ], HEIGHT );
127     region[ 2 ] = adjust( region[ 2 ], WIDTH );
128     region[ 3 ] = adjust( region[ 3 ], HEIGHT );
129     return region;
130   }
```

The message to present when receiving region

The Code: PlayWithImage.java (chooseRegion)

```
108   ////////////////////////////////////////////////////////
109   // RECEIVE RECTANGLE REGION
110   ////////////////////////////////////////////////////////
111   public static final String DIM_MESSAGE =
112       "Enter upperLeftCorner's X Y and lowerRighthCorner's X Y\n" +
113        "with a white space in between.\n" +
114        "The range for X is 0 .. " + WIDTH + ".\n" +
115        "The range for Y is 0 .. " + HEIGHT + ".";
116
117   public static int[] chooseRegion() throws Exception {
118     //--- receive dimensional input in one line
119     String input = JOptionPane.showInputDialog( DIM_MESSAGE );
120     String[] parts = input.trim().replace( "  ", " " ).split( " " );
121     int[] region = new int[ 4 ];
122     for ( int i = 0; i < 4; i ++ ) {
123       region[ i ]  = Integer.parseInt( parts[ i ] );
124     }
125     region[ 0 ] = adjust( region[ 0 ], WIDTH );
126     region[ 1 ] = adjust( region[ 1 ], HEIGHT );
127     region[ 2 ] = adjust( region[ 2 ], WIDTH );
128     region[ 3 ] = adjust( region[ 3 ], HEIGHT );
129     return region;
130   }
```

Method header; return value is an int array containing the positions of the two corners

The Code: PlayWithImage.java (chooseRegion)

```
108   ////////////////////////////////////////////////////////
109   // RECEIVE RECTANGLE REGION
110   ////////////////////////////////////////////////////////
111   public static final String DIM_MESSAGE =
112       "Enter upperLeftCorner's X Y and lowerRighthCorner's X Y\n" +
113        "with a white space in between.\n" +
114        "The range for X is 0 .. " + WIDTH + ".\n" +
115        "The range for Y is 0 .. " + HEIGHT + ".";
116
117   public static int[] chooseRegion() throws Exception {
118     //--- receive dimensional input in one line
119     String input = JOptionPane.showInputDialog( DIM_MESSAGE );
120     String[] parts = input.trim().replace( "  ", " " ).split( " " );
121     int[] region = new int[ 4 ];
122     for ( int i = 0; i < 4; i ++ ) {
123       region[ i ]  = Integer.parseInt( parts[ i ] );
124     }
125     region[ 0 ] = adjust( region[ 0 ], WIDTH );
126     region[ 1 ] = adjust( region[ 1 ], HEIGHT );
127     region[ 2 ] = adjust( region[ 2 ], WIDTH );
128     region[ 3 ] = adjust( region[ 3 ], HEIGHT );
129     return region;
130   }
```

Receive response from the user using `JOptionPane.showInputDialog` with the message

The Code: PlayWithImage.java (chooseRegion)

```
108   ////////////////////////////////////////////////////////
109   // RECEIVE RECTANGLE REGION
110   ////////////////////////////////////////////////////////
111   public static final String DIM_MESSAGE =
112       "Enter upperLeftCorner's X Y and lowerRightCorner's X Y\n" +
113        "with a white space in between.\n" +
114        "The range for X is 0 .. " + WIDTH + ".\n" +
115        "The range for Y is 0 .. " + HEIGHT + ".";
116
117   public static int[] chooseRegion() throws Exception {
118     //--- receive dimensional input in one line
119     String input = JOptionPane.showInputDialog( DIM_MESSAGE );
120     String[] parts = input.trim().replace( "  ", " " ).split( " " );
121     int[] region = new int[ 4 ];
122     for ( int i = 0; i < 4; i ++ ) {
123       region[ i ]  = Integer.parseInt( parts[ i ] );
124     }
125     region[ 0 ] = adjust( region[ 0 ], WIDTH );
126     region[ 1 ] = adjust( region[ 1 ], HEIGHT );
127     region[ 2 ] = adjust( region[ 2 ], WIDTH );
128     region[ 3 ] = adjust( region[ 3 ], HEIGHT );
129     return region;
130   }
```

Trim the response, replace all double white space with a single white space, and then split it using the whitespace as the delimiter; saves it to an array `parts`

The Code: PlayWithImage.java (chooseRegion)

```
108   ////////////////////////////////////////////////////////
109   // RECEIVE RECTANGLE REGION
110   ////////////////////////////////////////////////////////
111   public static final String DIM_MESSAGE =
112       "Enter upperLeftCorner's X Y and lowerRighthCorner's X Y\n" +
113        "with a white space in between.\n" +
114        "The range for X is 0 .. " + WIDTH + ".\n" +
115        "The range for Y is 0 .. " + HEIGHT + ".";
116
117   public static int[] chooseRegion() throws Exception {
118     //--- receive dimensional input in one line
119     String input = JOptionPane.showInputDialog( DIM_MESSAGE );
120     String[] parts = input.trim().replace( "  ", " " ).split( " " );
121     int[] region = new int[ 4 ];
122     for ( int i = 0; i < 4; i ++ ) {
123       region[ i ]  = Integer.parseInt( parts[ i ] );
124     }
125     region[ 0 ] = adjust( region[ 0 ], WIDTH );
126     region[ 1 ] = adjust( region[ 1 ], HEIGHT );
127     region[ 2 ] = adjust( region[ 2 ], WIDTH );
128     region[ 3 ] = adjust( region[ 3 ], HEIGHT );
129     return region;
130   }
```

Create an int array of four elements

The Code: PlayWithImage.java (chooseRegion)

```
108   ////////////////////////////////////////////////////////
109   // RECEIVE RECTANGLE REGION
110   ////////////////////////////////////////////////////////
111   public static final String DIM_MESSAGE =
112       "Enter upperLeftCorner's X Y and lowerRighthCorner's X Y\n" +
113        "with a white space in between.\n" +
114        "The range for X is 0 .. " + WIDTH + ".\n" +
115        "The range for Y is 0 .. " + HEIGHT + ".";
116
117   public static int[] chooseRegion() throws Exception {
118     //--- receive dimensional input in one line
119     String input = JOptionPane.showInputDialog( DIM_MESSAGE );
120     String[] parts = input.trim().replace( "  ", " " ).split( " " );
121     int[] region = new int[ 4 ];
122     for ( int i = 0; i < 4; i ++ ) {
123       region[ i ]  = Integer.parseInt( parts[ i ] );
124     }
125     region[ 0 ] = adjust( region[ 0 ], WIDTH );
126     region[ 1 ] = adjust( region[ 1 ], HEIGHT );
127     region[ 2 ] = adjust( region[ 2 ], WIDTH );
128     region[ 3 ] = adjust( region[ 3 ], HEIGHT );
129     return region;
130   }
```

Convert the four elements of the string array to integers and say them in the int array

The Code: PlayWithImage.java (chooseRegion)

```
108   ////////////////////////////////////////////////////////
109   // RECEIVE RECTANGLE REGION
110   ////////////////////////////////////////////////////////
111   public static final String DIM_MESSAGE =
112       "Enter upperLeftCorner's X Y and lowerRighthCorner's X Y\n" +
113        "with a white space in between.\n" +
114        "The range for X is 0 .. " + WIDTH + ".\n" +
115        "The range for Y is 0 .. " + HEIGHT + ".";
116
117   public static int[] chooseRegion() throws Exception {
118     //--- receive dimensional input in one line
119     String input = JOptionPane.showInputDialog( DIM_MESSAGE );
120     String[] parts = input.trim().replace( "  ", " " ).split( " " );
121     int[] region = new int[ 4 ];
122     for ( int i = 0; i < 4; i ++ ) {
123       region[ i ]  = Integer.parseInt( parts[ i ] );
124     }
125     region[ 0 ] = adjust( region[ 0 ], WIDTH );
126     region[ 1 ] = adjust( region[ 1 ], HEIGHT );
127     region[ 2 ] = adjust( region[ 2 ], WIDTH );
128     region[ 3 ] = adjust( region[ 3 ], HEIGHT );
129     return region;
130   }
```

Apply adjustments to the values

The Code: PlayWithImage.java (chooseRegion)

```
108   ////////////////////////////////////////////////////////
109   // RECEIVE RECTANGLE REGION
110   ////////////////////////////////////////////////////////
111   public static final String DIM_MESSAGE =
112       "Enter upperLeftCorner's X Y and lowerRighthCorner's X Y\n" +
113        "with a white space in between.\n" +
114        "The range for X is 0 .. " + WIDTH + ".\n" +
115        "The range for Y is 0 .. " + HEIGHT + ".";
116
117   public static int[] chooseRegion() throws Exception {
118     //--- receive dimensional input in one line
119     String input = JOptionPane.showInputDialog( DIM_MESSAGE );
120     String[] parts = input.trim().replace( "  ", " " ).split( " " );
121     int[] region = new int[ 4 ];
122     for ( int i = 0; i < 4; i ++ ) {
123       region[ i ]  = Integer.parseInt( parts[ i ] );
124     }
125     region[ 0 ] = adjust( region[ 0 ], WIDTH );
126     region[ 1 ] = adjust( region[ 1 ], HEIGHT );
127     region[ 2 ] = adjust( region[ 2 ], WIDTH );
128     region[ 3 ] = adjust( region[ 3 ], HEIGHT );
129     return region;
130   }
```

Return the array

The Code: PlayWithImage.java (chooseRegion)

```
108   ////////////////////////////////////////////////////////
109   // RECEIVE RECTANGLE REGION
110   ////////////////////////////////////////////////////////
111   public static final String DIM_MESSAGE =
112       "Enter upperLeftCorner's X Y and lowerRighthCorner's X Y\n" +
113        "with a white space in between.\n" +
114        "The range for X is 0 .. " + WIDTH + ".\n" +
115        "The range for Y is 0 .. " + HEIGHT + ".";
116
117   public static int[] chooseRegion() throws Exception {
118     //--- receive dimensional input in one line
119     String input = JOptionPane.showInputDialog( DIM_MESSAGE );
120     String[] parts = input.trim().replace( "  ", " " ).split( " " );
121     int[] region = new int[ 4 ];
122     for ( int i = 0; i < 4; i ++ ) {
123       region[ i ]  = Integer.parseInt( parts[ i ] );
124     }
125     region[ 0 ] = adjust( region[ 0 ], WIDTH );
126     region[ 1 ] = adjust( region[ 1 ], HEIGHT );
127     region[ 2 ] = adjust( region[ 2 ], WIDTH );
128     region[ 3 ] = adjust( region[ 3 ], HEIGHT );
129     return region;
130   }
```

An error, a non-integer response, or a fewer elements in the response, will result in a run-time error

The Code: PlayWithImage.java (paintRectangle)

```
132   ////////////////////////////////////////////////////////
133   // ADD RECTANGLES
134   ////////////////////////////////////////////////////////
135   public static void paintRectangle( BufferedImage image,
136       int[] region, Color color ) {
137     for ( int i = region[ 0 ]; i < region[ 2 ]; i ++ ) {
138       for ( int j = region[ 1 ]; j < region[ 3 ]; j ++ ) {
139         image.setRGB( i, j, color.getRGB() );
140       }
141     }
142   }
```

Method header; it takes a `BufferedImage` object, an int array that holds size information, and a `Color` object that is the color to be used

The Code: PlayWithImage.java (paintRectangle)

```
132   ////////////////////////////////////////////////////////
133   // ADD RECTANGLES
134   ////////////////////////////////////////////////////////
135   public static void paintRectangle( BufferedImage image,
136       int[] region, Color color ) {
137     for ( int i = region[ 0 ]; i < region[ 2 ]; i ++ ) {
138       for ( int j = region[ 1 ]; j < region[ 3 ]; j ++ ) {
139         image.setRGB( i, j, color.getRGB() );
140       }
141     }
142   }
```

Use a double for-loop to change the color of the pixel in the region

The Code: PlayWithImage.java (addRectangles)

```
144   ////////////////////////////////////////////////////////
145   // ADD RECTANGLES
146   ////////////////////////////////////////////////////////
147   public static void addRectangles(
148       JFrame frame, BufferedImage image ) {
149     while ( JOptionPane.showInputDialog( "Add rectangles? y/n" )
150         .toLowerCase().startsWith( "y" ) ) {
151       try {
152         int[] region = chooseRegion();
153         Color color = chooseColor();
154         paintRectangle( image, region, color );
155         // repaint the frame
156         frame.repaint();
157       } catch ( Exception e ) {
158         JOptionPane.showMessageDialog( null, "failed!" );
159       }
160     }
161   }
```

Method header

The Code: PlayWithImage.java (addRectangles)

```
144   ////////////////////////////////////////////////////////
145   // ADD RECTANGLES
146   ////////////////////////////////////////////////////////
147   public static void addRectangles(
148       JFrame frame, BufferedImage image ) {
149     while ( JOptionPane.showInputDialog( "Add rectangles? y/n" )
150         .toLowerCase().startsWith( "y" ) ) {
151       try {
152         int[] region = chooseRegion();
153         Color color = chooseColor();
154         paintRectangle( image, region, color );
155         // repaint the frame
156         frame.repaint();
157       } catch ( Exception e ) {
158         JOptionPane.showMessageDialog( null, "failed!" );
159       }
160     }
161   }
```

Prompts the user and receives response; if the lower-case version of the String does not start with 'y', quit the loop

The Code: PlayWithImage.java (addRectangles)

```
144   ////////////////////////////////////////////////////////
145   // ADD RECTANGLES
146   ////////////////////////////////////////////////////////
147   public static void addRectangles(
148       JFrame frame, BufferedImage image ) {
149     while ( JOptionPane.showInputDialog( "Add rectangles? y/n" )
150         .toLowerCase().startsWith( "y" ) ) {
151       try {
152         int[] region = chooseRegion();
153         Color color = chooseColor();
154         paintRectangle( image, region, color );
155         // repaint the frame
156         frame.repaint();
157       } catch ( Exception e ) {
158         JOptionPane.showMessageDialog( null, "failed!" );
159       }
160     }
161   }
```

Obtain the region, obtain the color, and paint the rectangle

The Code: PlayWithImage.java (addRectangles)

```
144   ////////////////////////////////////////////////////////
145   // ADD RECTANGLES
146   ////////////////////////////////////////////////////////
147   public static void addRectangles(
148       JFrame frame, BufferedImage image ) {
149     while ( JOptionPane.showInputDialog( "Add rectangles? y/n" )
150         .toLowerCase().startsWith( "y" ) ) {
151       try {
152         int[] region = chooseRegion();
153         Color color = chooseColor();
154         paintRectangle( image, region, color );
155         // repaint the frame
156         frame.repaint();
157       } catch ( Exception e ) {
158         JOptionPane.showMessageDialog( null, "failed!" );
159       }
160     }
161   }
```

Call `repaint` on `frame` to reflect the change

The Code: PlayWithImage.java (addRectangles)

```
144   ////////////////////////////////////////////////////////
145   // ADD RECTANGLES
146   ////////////////////////////////////////////////////////
147   public static void addRectangles(
148       JFrame frame, BufferedImage image ) {
149     while ( JOptionPane.showInputDialog( "Add rectangles? y/n" )
150         .toLowerCase().startsWith( "y" ) ) {
151       try {
152         int[] region = chooseRegion();
153         Color color = chooseColor();
154         paintRectangle( image, region, color );
155         // repaint the frame
156         frame.repaint();
157       } catch ( Exception e ) {
158         JOptionPane.showMessageDialog( null, "failed!" );
159       }
160     }
161   }
```

If some error occurs, during the process, generate an error message using `JOptionPane.showMessageDialog`; the method takes two parameters, the first should be set to null in this case and the second is the message

The Code: PlayWithImage.java (writeImage)

```
164   ////////////////////////////////////////////////////////
165   // WRITE IMAGE TO A FILE
166   ////////////////////////////////////////////////////////
167   public static String DEFAULT_FILE_NAME = "default.jpg";
168
169   public static void writeImage( BufferedImage image ) {
170     String outFileName = JOptionPane.showInputDialog(
171         "Enter file name with extension .jpg" );
172     if ( !outFileName.endsWith( ".jpg" ) ) {
173       outFileName = DEFAULT_FILE_NAME;
174     }
175     File outFile = new File( outFileName );
176     try {
177       ImageIO.write( image, "jpg", outFile );
178     } catch ( IOException e ) {
179       JOptionPane.showMessageDialog( null,
180           "An error\n" + e.getMessage() );
181     }
182   }
```

Default file name

The Code: PlayWithImage.java (writeImage)

```
164   ////////////////////////////////////////////////////////
165   // WRITE IMAGE TO A FILE
166   ////////////////////////////////////////////////////////
167   public static String DEFAULT_FILE_NAME = "default.jpg";
168
169   public static void writeImage( BufferedImage image ) {
170     String outFileName = JOptionPane.showInputDialog(
171         "Enter file name with extension .jpg" );
172     if ( !outFileName.endsWith( ".jpg" ) ) {
173       outFileName = DEFAULT_FILE_NAME;
174     }
175     File outFile = new File( outFileName );
176     try {
177       ImageIO.write( image, "jpg", outFile );
178     } catch ( IOException e ) {
179       JOptionPane.showMessageDialog( null,
180           "An error\n" + e.getMessage() );
181     }
182   }
```

Method header

The Code: PlayWithImage.java (writeImage)

```
164   ////////////////////////////////////////////////////////
165   // WRITE IMAGE TO A FILE
166   ////////////////////////////////////////////////////////
167   public static String DEFAULT_FILE_NAME = "default.jpg";
168
169   public static void writeImage( BufferedImage image ) {
170     String outFileName = JOptionPane.showInputDialog(
171         "Enter file name with extension .jpg" );
172     if ( !outFileName.endsWith( ".jpg" ) ) {
173       outFileName = DEFAULT_FILE_NAME;
174     }
175     File outFile = new File( outFileName );
176     try {
177       ImageIO.write( image, "jpg", outFile );
178     } catch ( IOException e ) {
179       JOptionPane.showMessageDialog( null,
180           "An error\n" + e.getMessage() );
181     }
182   }
```

Receive from the user a file path

The Code: PlayWithImage.java (writeImage)

```
164   ////////////////////////////////////////////////////////
165   // WRITE IMAGE TO A FILE
166   ////////////////////////////////////////////////////////
167   public static String DEFAULT_FILE_NAME = "default.jpg";
168
169   public static void writeImage( BufferedImage image ) {
170     String outFileName = JOptionPane.showInputDialog(
171         "Enter file name with extension .jpg" );
172     if ( !outFileName.endsWith( ".jpg" ) ) {
173       outFileName = DEFAULT_FILE_NAME;
174     }
175     File outFile = new File( outFileName );
176     try {
177       ImageIO.write( image, "jpg", outFile );
178     } catch ( IOException e ) {
179       JOptionPane.showMessageDialog( null,
180           "An error\n" + e.getMessage() );
181     }
182   }
```

If the filename does not end with "`.jpg`" use the default file name

The Code: PlayWithImage.java (writeImage)

```
164   ////////////////////////////////////////////////////////
165   // WRITE IMAGE TO A FILE
166   ////////////////////////////////////////////////////////
167   public static String DEFAULT_FILE_NAME = "default.jpg";
168
169   public static void writeImage( BufferedImage image ) {
170     String outFileName = JOptionPane.showInputDialog(
171         "Enter file name with extension .jpg" );
172     if ( !outFileName.endsWith( ".jpg" ) ) {
173       outFileName = DEFAULT_FILE_NAME;
174     }
175     File outFile = new File( outFileName );
176     try {
177       ImageIO.write( image, "jpg", outFile );
178     } catch ( IOException e ) {
179       JOptionPane.showMessageDialog( null,
180           "An error\n" + e.getMessage() );
181     }
182   }
```

Create the file

The Code: PlayWithImage.java (writeImage)

```
164   ////////////////////////////////////////////////////////
165   // WRITE IMAGE TO A FILE
166   ////////////////////////////////////////////////////////
167   public static String DEFAULT_FILE_NAME = "default.jpg";
168
169   public static void writeImage( BufferedImage image ) {
170     String outFileName = JOptionPane.showInputDialog(
171         "Enter file name with extension .jpg" );
172     if ( !outFileName.endsWith( ".jpg" ) ) {
173       outFileName = DEFAULT_FILE_NAME;
174     }
175     File outFile = new File( outFileName );
176     try {
177       ImageIO.write( image, "jpg", outFile );
178     } catch ( IOException e ) {
179       JOptionPane.showMessageDialog( null,
180           "An error\n" + e.getMessage() );
181     }
182   }
```

Call the write method of the `ImageIO` to write the image to the file as a jpg file

The Code: PlayWithImage.java (writeImage)

```
164   ////////////////////////////////////////////////////////
165   // WRITE IMAGE TO A FILE
166   ////////////////////////////////////////////////////////
167   public static String DEFAULT_FILE_NAME = "default.jpg";
168
169   public static void writeImage( BufferedImage image ) {
170     String outFileName = JOptionPane.showInputDialog(
171         "Enter file name with extension .jpg" );
172     if ( !outFileName.endsWith( ".jpg" ) ) {
173       outFileName = DEFAULT_FILE_NAME;
174     }
175     File outFile = new File( outFileName );
176     try {
177       ImageIO.write( image, "jpg", outFile );
178     } catch ( IOException e ) {
179       JOptionPane.showMessageDialog( null,
180           "An error\n" + e.getMessage() );
181     }
182   }
```

Report error on screen