Graphics Using a Two-dimensional Data

Mitsu Ogihara

Department of Computer Science
University of Miami

Table of Contents

Problem

Write an application for drawing rectangles on a graphic window by specying on the terminal screen the location of the rectangle and then saving the picture in a file

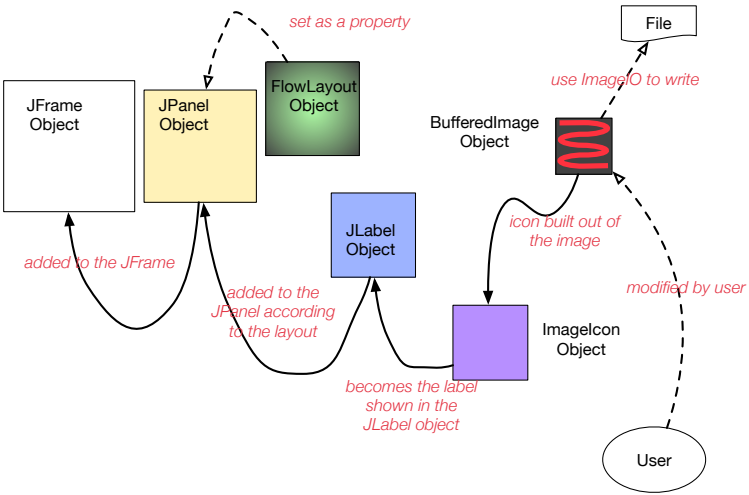
Pixels

- 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

Relations Among the Classes Used



More Description

- Create a `BufferedImage` object by specifying its dimension (the # of pixels horizontal, the # of pixels vertical, the color specification type) ... call this object `image`
- The color of any pixel of `image` can be changed
  - Set all of them to white initially
  - To draw rectangles, use this characteristic to change the pixels of the rectangle to green
- Using `ImageIO`'s `write` method, save `image` as a JPEG file

More Description (cont'd)

- Create a `ImageIcon` object `icon` from `image`
- Create a `JLabel` object `label` and set this icon to be its label
- Create a `JPanel` object `panel` and set its graphic object layout to a `FlowLayout` object
  - A `FlowLayout` object must be used since, the class `FlowLayout` allows changes in its property (e.g., the gaps between the aligned objects)
  - Here since there is only one project aligned, all these properties can be ignored and so we simply supply a new object to `panel` without naming it
- Add `panel` to `panel`

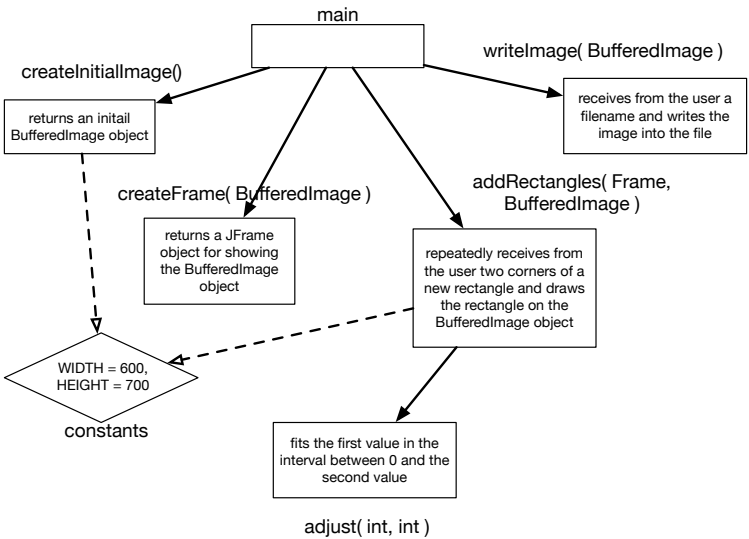
More Descriptions (cont'd)

- Create a `JFrame` object `frame`
- Set the property `EXIT_ON_CLOSE` of `frame` so that the program quits when the frame is closed
- Add `panel` to `frame`
- Execute `frame.pack()` to show `panel` inside `frame` with some wiggle room
- Execute `frame.setVisible( true )` to make the frame visible
  - By executing `frame.repaint()`, redraw the graphics
  - By executing `frame.dispose()`, the graphics can be removed from the screen

Major Actors in the Code

- `image`
  - Created as a fixed size white image
  - User interactions will make changes on the pixels of this object
  - Before quitting the contents of the image will be written to a file
- `frame`:
  - Built to show `image`
  - Each time a rectangle is drawn, `frame.repaint()` should be executed
  - At the end of the code, `frame.dispose()` should be executed

Relations Among Methods



Rectangle Drawing

- Occurs in the method `addRectangles`
- Use a while loop whose termination condition is the string `response` starts with `"y"`
- In the body first ask whether to add a rectangle
  - If “yes” receive the coordinates for the upper left corner and for the lower left corner and add one such rectangle
    - Each entered value is adjusted using the method `adjust` to fit between 0 and the maximum (`WIDTH` for a horizontal value, `HEIGHT` for a vertical value)
      A negative value is turned to 0
      A value greater than the maximum is turned to the maximum
    - Use a double for loop to set the pixels inside the rectangle to green.
      Use `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: PlayWithImageOld.java (header)

```
1  import java.awt.image.BufferedImage;
2  import java.awt.FlowLayout;
3  import java.awt.Color;
4  import javax.imageio.ImageIO;
5  import javax.swing.JFrame;
6  import javax.swing.JPanel;
7  import javax.swing.JLabel;
8  import javax.swing.ImageIcon;
9
10 import java.io.*;
11 import java.util.*;
12
13 public class PlayWithImageOld {
14   //--- size constants
15   public static final int WIDTH = 600;
16   public static final int HEIGHT = 700;
```

Three classes imported from `java.awt`

The Code: PlayWithImageOld.java (header)

```
1  import java.awt.image.BufferedImage;
2  import java.awt.FlowLayout;
3  import java.awt.Color;
4  import javax.imageio.ImageIO;
5  import javax.swing.JFrame;
6  import javax.swing.JPanel;
7  import javax.swing.JLabel;
8  import javax.swing.ImageIcon;
9
10 import java.io.*;
11 import java.util.*;
12
13 public class PlayWithImageOld {
14   //--- size constants
15   public static final int WIDTH = 600;
16   public static final int HEIGHT = 700;
```

Classe imported from `javax.imageio`

The Code: PlayWithImageOld.java (header)

```
1  import java.awt.image.BufferedImage;
2  import java.awt.FlowLayout;
3  import java.awt.Color;
4  import javax.imageio.ImageIO;
5  import javax.swing.JFrame;
6  import javax.swing.JPanel;
7  import javax.swing.JLabel;
8  import javax.swing.ImageIcon;
9
10 import java.io.*;
11 import java.util.*;
12
13 public class PlayWithImageOld {
14   //--- size constants
15   public static final int WIDTH = 600;
16   public static final int HEIGHT = 700;
```

Four classes imported from `javax.swing`

The Code: PlayWithImageOld.java (header)

```
1  import java.awt.image.BufferedImage;
2  import java.awt.FlowLayout;
3  import java.awt.Color;
4  import javax.imageio.ImageIO;
5  import javax.swing.JFrame;
6  import javax.swing.JPanel;
7  import javax.swing.JLabel;
8  import javax.swing.ImageIcon;
9
10 import java.io.*;
11 import java.util.*;
12
13 public class PlayWithImageOld {
14   //--- size constants
15   public static final int WIDTH = 600;
16   public static final int HEIGHT = 700;
```

The usual stuff

The Code: PlayWithImageOld.java (header)

```
1  import java.awt.image.BufferedImage;
2  import java.awt.FlowLayout;
3  import java.awt.Color;
4  import javax.imageio.ImageIO;
5  import javax.swing.JFrame;
6  import javax.swing.JPanel;
7  import javax.swing.JLabel;
8  import javax.swing.ImageIcon;
9
10 import java.io.*;
11 import java.util.*;
12
13 public class PlayWithImageOld {
14   //--- size constants
15   public static final int WIDTH = 600;
16   public static final int HEIGHT = 700;
```

The dimensional constants

The Code: PlayWithImageOld.java (main)

```
18    //--- main method
19    public static void main( String[] args ) throws IOException {
20      BufferedImage image = generateInitialImage();
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: PlayWithImageOld.java (main)

```
18    //--- main method
19    public static void main( String[] args ) throws IOException {
20      BufferedImage image = generateInitialImage();
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: PlayWithImageOld.java (main)

```
18    //--- main method
19    public static void main( String[] args ) throws IOException {
20      BufferedImage image = generateInitialImage();
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: PlayWithImageOld.java (main)

```
18    //--- main method
19    public static void main( String[] args ) throws IOException {
20      BufferedImage image = generateInitialImage();
21      JFrame frame = generateFrame( image );
22      addRectangles( frame, image );
23      writeImage( image );
24      frame.dispose();
25    }
```

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

The Code: PlayWithImageOld.java (main)

```
18    //--- main method
19    public static void main( String[] args ) throws IOException {
20      BufferedImage image = generateInitialImage();
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: PlayWithImageOld.java (generateInitialImage)

```
27    //--- create an all-white initial image and return
28    public static BufferedImage generateInitialImage() {
29      BufferedImage image = new BufferedImage(
30          WIDTH, HEIGHT, BufferedImage.TYPE_3BYTE_BGR );
31      for ( int i = 0; i < WIDTH; i ++ ) {
32        for ( int j = 0; j < HEIGHT; j ++ ) {
33          image.setRGB( i, j, Color.WHITE.getRGB() );
34        }
35      }
36      return image;
37    }
```

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

The Code: PlayWithImageOld.java (generateInitialImage)

```
27    //--- create an all-white initial image and return
28    public static BufferedImage generateInitialImage() {
29      BufferedImage image = new BufferedImage(
30          WIDTH, HEIGHT, BufferedImage.TYPE_3BYTE_BGR );
31      for ( int i = 0; i < WIDTH; i ++ ) {
32        for ( int j = 0; j < HEIGHT; j ++ ) {
33          image.setRGB( i, j, Color.WHITE.getRGB() );
34        }
35      }
36      return image;
37    }
```

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

The Code: PlayWithImageOld.java (generateInitialImage)

```
27    //--- create an all-white initial image and return
28    public static BufferedImage generateInitialImage() {
29      BufferedImage image = new BufferedImage(
30          WIDTH, HEIGHT, BufferedImage.TYPE_3BYTE_BGR );
31      for ( int i = 0; i < WIDTH; i ++ ) {
32        for ( int j = 0; j < HEIGHT; j ++ ) {
33          image.setRGB( i, j, Color.WHITE.getRGB() );
34        }
35      }
36      return image;
37    }
```

Return the image object

The Code: PlayWithImageOld.java (generateFrame)

```
39    //--- generate JFrame from a BufferedImage object
40    public static JFrame generateFrame( BufferedImage image ) {
41      //--- create an ImageIcon from the image
42      //--- set it to a JLabel
43      ImageIcon icon = new ImageIcon( image );
44      JLabel label = new JLabel();
45      label.setIcon( icon );
46      //--- crate a JPanel and add the JLabel object to it
47      JPanel panel = new JPanel();
48      panel.setLayout( new FlowLayout() );
49      panel.add( label );
50      //--- crate a JFrame and add the JLabel object to it
51      JFrame frame = new JFrame();
52      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
53      frame.add( panel );
54      frame.pack();
55      frame.setVisible( true );
56      //--- make the JFrame object visible and return it
57      return frame;
58    }
```

Method header; it returns a `JFrame` object

The Code: PlayWithImageOld.java (generateFrame)

```
39    //--- generate JFrame from a BufferedImage object
40    public static JFrame generateFrame( BufferedImage image ) {
41      //--- create an ImageIcon from the image
42      //--- set it to a JLabel
43      ImageIcon icon = new ImageIcon( image );
44      JLabel label = new JLabel();
45      label.setIcon( icon );
46      //--- crate a JPanel and add the JLabel object to it
47      JPanel panel = new JPanel();
48      panel.setLayout( new FlowLayout() );
49      panel.add( label );
50      //--- crate a JFrame and add the JLabel object to it
51      JFrame frame = new JFrame();
52      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
53      frame.add( panel );
54      frame.pack();
55      frame.setVisible( true );
56      //--- make the JFrame object visible and return it
57      return frame;
58    }
```

Create an `ImageIcon` object from the image and then create a `JLabel` object and set the icode of the `JLabel` object to the icon thus has been generated

The Code: PlayWithImageOld.java (generateFrame)

```
39    //--- generate JFrame from a BufferedImage object
40    public static JFrame generateFrame( BufferedImage image ) {
41      //--- create an ImageIcon from the image
42      //--- set it to a JLabel
43      ImageIcon icon = new ImageIcon( image );
44      JLabel label = new JLabel();
45      label.setIcon( icon );
46      //--- crate a JPanel and add the JLabel object to it
47      JPanel panel = new JPanel();
48      panel.setLayout( new FlowLayout() );
49      panel.add( label );
50      //--- crate a JFrame and add the JLabel object to it
51      JFrame frame = new JFrame();
52      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
53      frame.add( panel );
54      frame.pack();
55      frame.setVisible( true );
56      //--- make the JFrame object visible and return it
57      return frame;
58    }
```

Create a `JPanel` object, set its layout to a new `FlowLayout` object, and then add the label

The Code: PlayWithImageOld.java (generateFrame)

```
39    //--- generate JFrame from a BufferedImage object
40    public static JFrame generateFrame( BufferedImage image ) {
41      //--- create an ImageIcon from the image
42      //--- set it to a JLabel
43      ImageIcon icon = new ImageIcon( image );
44      JLabel label = new JLabel();
45      label.setIcon( icon );
46      //--- crate a JPanel and add the JLabel object to it
47      JPanel panel = new JPanel();
48      panel.setLayout( new FlowLayout() );
49      panel.add( label );
50      //--- crate a JFrame and add the JLabel object to it
51      JFrame frame = new JFrame();
52      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
53      frame.add( panel );
54      frame.pack();
55      frame.setVisible( true );
56      //--- make the JFrame object visible and return it
57      return frame;
58    }
```

Create a `JFrame` object `frame`, set the close option to force program termination upon close, add the panel, call `frame.pack()`, and then make the frame visible

The Code: PlayWithImageOld.java (generateFrame)

```
39    //--- generate JFrame from a BufferedImage object
40    public static JFrame generateFrame( BufferedImage image ) {
41      //--- create an ImageIcon from the image
42      //--- set it to a JLabel
43      ImageIcon icon = new ImageIcon( image );
44      JLabel label = new JLabel();
45      label.setIcon( icon );
46      //--- crate a JPanel and add the JLabel object to it
47      JPanel panel = new JPanel();
48      panel.setLayout( new FlowLayout() );
49      panel.add( label );
50      //--- crate a JFrame and add the JLabel object to it
51      JFrame frame = new JFrame();
52      frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
53      frame.add( panel );
54      frame.pack();
55      frame.setVisible( true );
56      //--- make the JFrame object visible and return it
57      return frame;
58    }
```

Return the frame

The Code: PlayWithImageOld.java (adjust)

```
60    //--- ensure that a number is between 0 and max
61    public static int adjust( int value, int max ) {
62      if ( value >= max ) {
63        return max;
64      }
65      if ( value < 0 ) {
66        return 0;
67      }
68      return value;
69    }
```

Method header; it takes the value and the maimum and returns an adjusted value

The Code: PlayWithImageOld.java (adjust)

```
60    //--- ensure that a number is between 0 and max
61    public static int adjust( int value, int max ) {
62      if ( value >= max ) {
63        return max;
64      }
65      if ( value < 0 ) {
66        return 0;
67      }
68      return value;
69    }
```

If the value is negative, return 0; if the value is greater than the maximum, return the maximum; otherwise, return the value without changing
Note that there is no `else` appearing in the method

The Code: PlayWithImageOld.java (addRectangle)

```
72    public static void addRectangles(
73        JFrame frame, BufferedImage image ) {
74      Scanner console = new Scanner( System.in );
75      String response = "y";
76      while ( response.startsWith( "y" ) ) {
77        System.out.print( "Add rectangles? " );
78        response = console.next().toLowerCase();
79        if ( response.startsWith( "y" ) ) {
80          System.out.print( "Enter ulX, ulY, lrX, lrY " );
81          int upperLeftX = adjust( console.nextInt(), WIDTH );
82          int upperLeftY = adjust( console.nextInt(), HEIGHT );
83          int lowerRightX = adjust( console.nextInt(), WIDTH );
84          int lowerRightY = adjust( console.nextInt(), HEIGHT );
85          for ( int i = upperLeftX; i < lowerRightX; i ++ ) {
86            for ( int j = upperLeftY; j < lowerRightY; j ++ ) {
87              image.setRGB( i, j, Color.GREEN.getRGB() );
88            }
89          }
90          frame.repaint();
91        }
92      }
93    }
```

Method header; it takes both the image and the frame

The Code: PlayWithImageOld.java (addRectangle)

```
72    public static void addRectangles(
73        JFrame frame, BufferedImage image ) {
74      Scanner console = new Scanner( System.in );
75      String response = "y";
76      while ( response.startsWith( "y" ) ) {
77        System.out.print( "Add rectangles? " );
78        response = console.next().toLowerCase();
79        if ( response.startsWith( "y" ) ) {
80          System.out.print( "Enter ulX, ulY, lrX, lrY " );
81          int upperLeftX = adjust( console.nextInt(), WIDTH );
82          int upperLeftY = adjust( console.nextInt(), HEIGHT );
83          int lowerRightX = adjust( console.nextInt(), WIDTH );
84          int lowerRightY = adjust( console.nextInt(), HEIGHT );
85          for ( int i = upperLeftX; i < lowerRightX; i ++ ) {
86            for ( int j = upperLeftY; j < lowerRightY; j ++ ) {
87              image.setRGB( i, j, Color.GREEN.getRGB() );
88            }
89          }
90          frame.repaint();
91        }
92      }
93    }
```

Creats a console scanner and creates a response string initialized to "y"

The Code: PlayWithImageOld.java (addRectangle)

```
72    public static void addRectangles(
73        JFrame frame, BufferedImage image ) {
74      Scanner console = new Scanner( System.in );
75      String response = "y";
76      while ( response.startsWith( "y" ) ) {
77        System.out.print( "Add rectangles? " );
78        response = console.next().toLowerCase();
79        if ( response.startsWith( "y" ) ) {
80          System.out.print( "Enter ulX, ulY, lrX, lrY " );
81          int upperLeftX = adjust( console.nextInt(), WIDTH );
82          int upperLeftY = adjust( console.nextInt(), HEIGHT );
83          int lowerRightX = adjust( console.nextInt(), WIDTH );
84          int lowerRightY = adjust( console.nextInt(), HEIGHT );
85          for ( int i = upperLeftX; i < lowerRightX; i ++ ) {
86            for ( int j = upperLeftY; j < lowerRightY; j ++ ) {
87              image.setRGB( i, j, Color.GREEN.getRGB() );
88            }
89          }
90          frame.repaint();
91        }
92      }
93    }
```

The while loop is executed because of the initial value

The Code: PlayWithImageOld.java (addRectangle)

```
72    public static void addRectangles(
73        JFrame frame, BufferedImage image ) {
74      Scanner console = new Scanner( System.in );
75      String response = "y";
76      while ( response.startsWith( "y" ) ) {
77        System.out.print( "Add rectangles? " );
78        response = console.next().toLowerCase();
79        if ( response.startsWith( "y" ) ) {
80          System.out.print( "Enter ulX, ulY, lrX, lrY " );
81          int upperLeftX = adjust( console.nextInt(), WIDTH );
82          int upperLeftY = adjust( console.nextInt(), HEIGHT );
83          int lowerRightX = adjust( console.nextInt(), WIDTH );
84          int lowerRightY = adjust( console.nextInt(), HEIGHT );
85          for ( int i = upperLeftX; i < lowerRightX; i ++ ) {
86            for ( int j = upperLeftY; j < lowerRightY; j ++ ) {
87              image.setRGB( i, j, Color.GREEN.getRGB() );
88            }
89          }
90          frame.repaint();
91        }
92      }
93    }
```

Prompt the user to answer whether to add a retangle, examine the answer (after turning it into lowercase), and check its first letter (since `next` is used, the user cannot enter an empty string)

The Code: PlayWithImageOld.java (addRectangle)

```
72    public static void addRectangles(
73        JFrame frame, BufferedImage image ) {
74      Scanner console = new Scanner( System.in );
75      String response = "y";
76      while ( response.startsWith( "y" ) ) {
77        System.out.print( "Add rectangles? " );
78        response = console.next().toLowerCase();
79        if ( response.startsWith( "y" ) ) {
80          System.out.print( "Enter ulX, ulY, lrX, lrY " );
81          int upperLeftX = adjust( console.nextInt(), WIDTH );
82          int upperLeftY = adjust( console.nextInt(), HEIGHT );
83          int lowerRightX = adjust( console.nextInt(), WIDTH );
84          int lowerRightY = adjust( console.nextInt(), HEIGHT );
85          for ( int i = upperLeftX; i < lowerRightX; i ++ ) {
86            for ( int j = upperLeftY; j < lowerRightY; j ++ ) {
87              image.setRGB( i, j, Color.GREEN.getRGB() );
88            }
89          }
90          frame.repaint();
91        }
92      }
93    }
```

Prompt for the coordinates

The Code: PlayWithImageOld.java (addRectangle)

```
72    public static void addRectangles(
73        JFrame frame, BufferedImage image ) {
74      Scanner console = new Scanner( System.in );
75      String response = "y";
76      while ( response.startsWith( "y" ) ) {
77        System.out.print( "Add rectangles? " );
78        response = console.next().toLowerCase();
79        if ( response.startsWith( "y" ) ) {
80          System.out.print( "Enter ulX, ulY, lrX, lrY " );
81          int upperLeftX = adjust( console.nextInt(), WIDTH );
82          int upperLeftY = adjust( console.nextInt(), HEIGHT );
83          int lowerRightX = adjust( console.nextInt(), WIDTH );
84          int lowerRightY = adjust( console.nextInt(), HEIGHT );
85          for ( int i = upperLeftX; i < lowerRightX; i ++ ) {
86            for ( int j = upperLeftY; j < lowerRightY; j ++ ) {
87              image.setRGB( i, j, Color.GREEN.getRGB() );
88            }
89          }
90          frame.repaint();
91        }
92      }
93    }
```

Receive the coordinates and make the adjustments on the sppot

The Code: PlayWithImageOld.java (addRectangle)

```
72    public static void addRectangles(
73        JFrame frame, BufferedImage image ) {
74      Scanner console = new Scanner( System.in );
75      String response = "y";
76      while ( response.startsWith( "y" ) ) {
77        System.out.print( "Add rectangles? " );
78        response = console.next().toLowerCase();
79        if ( response.startsWith( "y" ) ) {
80          System.out.print( "Enter ulX, ulY, lrX, lrY " );
81          int upperLeftX = adjust( console.nextInt(), WIDTH );
82          int upperLeftY = adjust( console.nextInt(), HEIGHT );
83          int lowerRightX = adjust( console.nextInt(), WIDTH );
84          int lowerRightY = adjust( console.nextInt(), HEIGHT );
85          for ( int i = upperLeftX; i < lowerRightX; i ++ ) {
86            for ( int j = upperLeftY; j < lowerRightY; j ++ ) {
87              image.setRGB( i, j, Color.GREEN.getRGB() );
88            }
89          }
90          frame.repaint();
91        }
92      }
93    }
```

Make the color changes using a double for-loop

The Code: PlayWithImageOld.java (addRectangle)

```
72    public static void addRectangles(
73        JFrame frame, BufferedImage image ) {
74      Scanner console = new Scanner( System.in );
75      String response = "y";
76      while ( response.startsWith( "y" ) ) {
77        System.out.print( "Add rectangles? " );
78        response = console.next().toLowerCase();
79        if ( response.startsWith( "y" ) ) {
80          System.out.print( "Enter ulX, ulY, lrX, lrY " );
81          int upperLeftX = adjust( console.nextInt(), WIDTH );
82          int upperLeftY = adjust( console.nextInt(), HEIGHT );
83          int lowerRightX = adjust( console.nextInt(), WIDTH );
84          int lowerRightY = adjust( console.nextInt(), HEIGHT );
85          for ( int i = upperLeftX; i < lowerRightX; i ++ ) {
86            for ( int j = upperLeftY; j < lowerRightY; j ++ ) {
87              image.setRGB( i, j, Color.GREEN.getRGB() );
88            }
89          }
90          frame.repaint();
91        }
92      }
93    }
```

Repaing the frame to update

The Code: PlayWithImageOld.java (writeImage)

```
95    //--- write an image to a file
96    public static void writeImage( BufferedImage image )
97        throws IOException {
98      Scanner console = new Scanner( System.in );
99      System.out.print( "Enter output file name with extension .jpg: " );
100     String outFileName = console.next();
101     if ( outFileName.endsWith( ".jpg" ) ) {
102       File outFile = new File( outFileName );
103       ImageIO.write( image, "jpg", outFile );
104     }
105   }
```

Method header; the method may throw `IOException`

The Code: PlayWithImageOld.java (writeImage)

```
95   //--- write an image to a file
96   public static void writeImage( BufferedImage image )
97       throws IOException {
98     Scanner console = new Scanner( System.in );
99     System.out.print( "Enter output file name with extension .jpg: " );
100    String outFileName = console.next();
101    if ( outFileName.endsWith( ".jpg" ) ) {
102      File outFile = new File( outFileName );
103      ImageIO.write( image, "jpg", outFile );
104    }
105  }
```

Create a scanner, prompt the user, and receive a file path

The Code: PlayWithImageOld.java (writeImage)

```
95    //--- write an image to a file
96    public static void writeImage( BufferedImage image )
97        throws IOException {
98      Scanner console = new Scanner( System.in );
99      System.out.print( "Enter output file name with extension .jpg: " );
100     String outFileName = console.next();
101     if ( outFileName.endsWith( ".jpg" ) ) {
102       File outFile = new File( outFileName );
103       ImageIO.write( image, "jpg", outFile );
104     }
105   }
```

If the filename ends with the required "`.jpg`" use the `ImageIO.write` method to write the image to the file

`ImageIO.write( BufferedImage, String, File )`

where the middle parameter species the file type (here it is "`jpg`")