CSC 160 LAB 8-1 DIGITAL PICTURE FRAME

PROFESSOR GODFREY MUGANDA
DEPARTMENT OF COMPUTER SCIENCE

1. Introduction

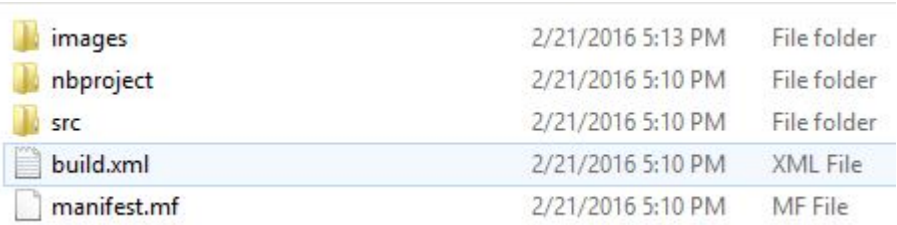
Download and unzip the images folder from the course website. The folder contains 28 images in files named `car0.png`, `car1.png`, ..., `car27.png`.

We are going to work with these images in a lab assignment that focuses on working with graphics. After you have successfully completed this lab, you, should be able to substitute images of your loved ones instead of the images of cars.

If you let the mouse hover over the icons that for the individual images, you will see that each of them is 128×128 pixels.

2. Displaying a Single Image

We will begin by displaying a single image, say `car15.png`. Start by creating a Netbeans project. Then, locate the Netbeans folder for the project on your computer and copy the unzipped `images` folder into it. When you look into the project folder, its contents should look like this:



Next, edit your Java source file so that it has two classes: a main class with the `main` method, and a second class `ImagePanel` that extends `JPanel`. You will use the second class to display an image, so equip it with a constructor that takes an `Image` as parameter:

```
public class Muganda160Lab8_1
{
    public static void main(String[] args) throws IOException
    {

    }
}


class ImagePanel extends JPanel
{
    private Image image;
```

```
    public ImagePanel(Image image)
    {

    }

    public void paintComponent(Graphics g)
    {

    }
}
```

3. Completing the `ImagePanel`

Complete the `ImagePanel` class so that

(1) the constructor stores the `image` parameter into the `image` field in `this` object.

(2) the `paintComponent()` method draws the image using the `Graphics` class method

```
void drawImage(Image image, int x, int y,
               ImageObserver observer)
```

Use $(0, 0)$ for the points (x, y) and use `null` for the `ImageObserver`.

4. The `main` method

Complete the `main()` method so that it creates a `JFrame` object, sets its size, title, default close operation method, and makes it visible.

Next, create an image in main using this statement:

```
Image image = ImageIO.read(new File("images/car15.png"));
```

This statement creates a `File` object that represents an image file, and then uses `ImageIO.read()` method to read and return the image contained in this file.

Finally, add code to `main()` that uses the created image to create a `MyImagePanel` object that displays the image. When you run your program, you should see something like this:

5. Centering the Image

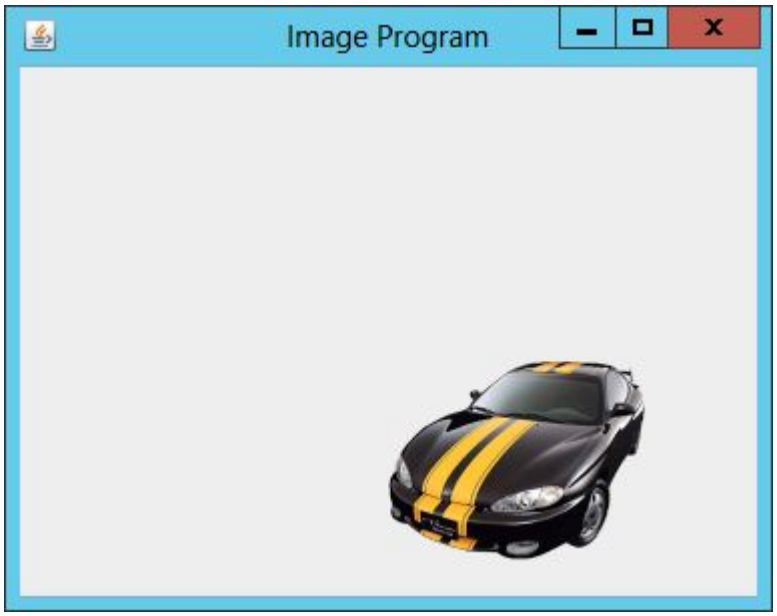
Next, we want to display the image in such a way that it is always centered in the `MyImagePanel`. To start, you can try to use

```
void drawImage(Image image, int x, int y, ImageObserver observer)
```

so that x is set to half the width of the panel, while y is set to half the height. When you are inside the paintComponent method, you can get the width and height of the panel by calling the `getWidth()` and `getHeight()` method of the `JPanel` class:

```
int getWidth()
int getHeight()
```

If you do this, this is what you get:



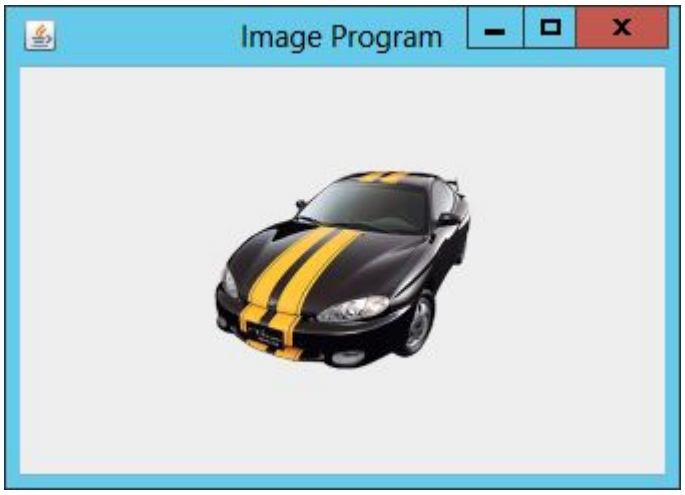
As you can see, this is not quite centered. The problem is that it puts the top left corner of the bounding box of the image at the center of the panel, so that the image is too far down and too far to the right. What you need is to put the center of the bounding box of the image at the center of the panel. To get the width and height of an image `image`, use this code

```
int imageWidth = image.getWidth(null);
int imageHeight = image.getHeight(null);
```

Using this knowledge, figure out how to set the (x, y)

```
void drawImage(Image image, int x, int y, ImageObserver observer)
```

so that the image is centered:



6. Digital Picture Frame

To create a digital picture frame, you need an array of images. Let us create an array of 10 images, using the images in the files `car0.png` through `car9.png`

For any integer k in the range $0 \ldots 9$, you can use the code

```
new File("images/car" + k + ".png")
```

to create a file that accesses that image. Next, create an array of `Image` objects

```
Image images = new Image[10];
```

and use a loop to set the images.

Next, modify the `ImagePanel` class so that it takes as parameter, an array of `Image` objects. In addition, the modified `ImagePanel` class will have a field

```
int currentImageIndex = 0;
```

that is initialized to zero, and indicates which image in the array of images should be drawn on the panel. Note that the fields in `ImagePanel` are *not* private. This is because they must be accessible to the timer listener. Here is a shell of the `ImagePanel` class.

```
class ImagePanel extends JPanel
{
    final Image[] images;
    int currentImageIndex = 0;

    ImagePanel(Image[] images)
    {
        this.images = images;
    }

    @Override
    public void paintComponent(Graphics g)
    {
```

```
    }
}
```

7. Timer

The timer will need access to the `ImagePanel`. Each time the timer's actionPerformed() method is called, it circularly increment the `currentImageIndex` in the panel, and then tell the image panel to repaint itself by calling its *repaint()* method:

```
repaint();
```

Here is shell of the `TimerListener` class.

```
class TimerListener implements ActionListener
{
    public TimerListener(ImagePanel imPanel)
    {

    }
    @Override
    public void actionPerformed(ActionEvent e)
    {

    }
}
```

Finally, add code to `main()` to create a timer with a delay of 3 seconds and an appropriate listener. Start the timer.

You do not need buttons to start and stop the timer: just start the timer as soon as the frame is shown.

8. Due Date

Monday of Week 9.