Thursday

Began working on the program's first step. After getting the base code together I was looking throughout your website and didn't find much clear reference on how to properly make a border layout. Finished step one with help of Ben, since he already had architecture down.

Setbacks: Website is not clear at all, very hard to find code.

Development: Ben showed me his code and gave me the general idea of how to implement a Border Layout.

Friday

After checking your website I learned how to Implement a grid panel that is within the east section of the layout. After that I began working on an array of buttons, checked with Ben for syntax as that was his most recent development. I made an array of 10 buttons which in the future will be used to switch between 10 preset pictures. I managed to get interaction with center panel and east panel by printing a variable from within the east panel in the center panel.

Monday

I attempted to seperate the code into 5 classes, but upon separation certain lines became errors.

Setbacks: (Refrence to pandisp) gained an error after

Tuesday

No coding today, meeting with southridge.

Wednesday.

No coding, In class discussion and Apps for Learning took priority.

Thursday

Modification, created a grid panel in the south for future interfacing with center panel

Friday

Tried to get the drag image code to work well with the current code I have, completing step 4 and half of step 9, but the results were rather bad.

I tried to replace PanDisp with ComDragImage as the center panel but the following resulted only in errors. Later I tried moving the main of ComDragImage to the main of LayoutViewer, but then only less errors appeared.

I added:

String sImageFile = "Zappa1.jpg";

Image imgZappa = Toolkit.getDefaultToolkit().getImage(

PanDisp.class.getResource(sImageFile));

imgZappa = imgZappa.getScaledInstance(nW, nH, Image.SCALE\_DEFAULT);

frame.getContentPane().add(new PanDisp(imgZappa));

to the main code of the layoutviewer, and frame.getContentPane().add(new PanDisp(imgZappa)); refused to work no matter what.

After yelling at the screen for an hour I decided to give up.

Started to work backwards, by working on the drag image code on the website and adding the architecture onto that.

It was almost a success, but whenever you would click on or move the image, the buttons would disappear, I assume it has to do with the repaint. I will play around with it.

After working on the original program (not the working backwards version) I now have a pan disp that looks like this:

public class PanDisp extends JPanel implements MouseMotionListener {

int nX,nY;

public PanDisp(){

//panMovement.add(lblName);//Uppercase doesn't work

}

public void mouseDragged(MouseEvent e) {

nX = e.getX();

nY = e.getY();

System.out.println("yay interaction");

repaint();

}

public void mouseMoved(MouseEvent e) {

}

}

It does nothing as it has no pictures yet, but it has no errors which is a development for tonight.

Saturday

Tried to get repaint to accept a single variable, such as

repaint(imgZappa); it was very unsuccessful

After working on the PanDisp class that is supposed to have an image pop up in it, I have:

public class PanDisp extends JPanel implements MouseMotionListener {

int nW=60,nH=60;

int nX,nY;

image imgZappa;

String sImageFile = "Zappa1.jpg";

public PanDisp(Image \_imgZappa){

imgZappa = \_imgZappa;

Image imgZappa = Toolkit.getDefaultToolkit().getImage(

PanDisp.class.getResource(sImageFile));

addMouseMotionListener(this);

PanDisp.getContentPane().add(new PanDisp(imgZappa));

imgZappa = imgZappa.getScaledInstance(nW, nH, Image.SCALE\_DEFAULT);

//panMovement.add(lblName);//Uppercase doesn't work

}

public void paint(Graphics g) {

Graphics2D g2 = (Graphics2D) g;

g2.drawImage(imgZappa, nX, nY, this);

}

public void mouseDragged(MouseEvent e) {

nX = e.getX();

nY = e.getY();

repaint();

}

public void mouseMoved(MouseEvent e) {

}

}

I got an error when the constructor had \_imgZappa in it so I had to take it out, with this new advancement the code has no errors and is missing one line of your drag image code:

public PanDisp(){

addMouseMotionListener(this);

String sImageFile = "Zappa1.jpg";

Image imgZappa = Toolkit.getDefaultToolkit().getImage(

PanDisp.class.getResource(sImageFile));

imgZappa = imgZappa.getScaledInstance(nW, nH, Image.SCALE\_DEFAULT);

//frame.getContentPane().add(new PanDisp(imgZappa));

//frame.getContentPane().add(new PanDisp(imgZappa));

//panMovement.add(lblName);//Uppercase doesn't work

}

the frame.getContentPane().add(new PanDisp(imgZappa));, doesn't work I'll assume it's because I've only seen frame. Lines put in main, so I tried putting in main and it told me non-static variable cannot be referenced by static content.

Next I tried making main non-static by taking the static out of it, it proceeded to tell me I have no main so that's a dead end.

Finally I tried adding a frame within PanDisp itself since it couldn't access main's frame,

I ended up calling the project JavaGraphics4unstable and began to try to find the internet's way of adding an image.

Oracle's site was no help.

The internet was little help, I've decided to go back to the code where the image worked but the buttons reset every time it was moved, the code looked like:

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class ComDragImage extends JComponent

implements MouseMotionListener {

JLabel lblName,lblStuff,lblNew;

ComDragImage.PanDisp panDisp = new ComDragImage.PanDisp();

ComDragImage.PanMovement panMovement = new ComDragImage.PanMovement();//panDisp);

ComDragImage.PanMenu panMenu = new ComDragImage.PanMenu();//panDisp);

static int nW = 60, nH = 60;

int nGrid = 10;

int nX, nY;

Image imgZappa;

public ComDragImage(Image \_imgZappa) {

setLayout(new BorderLayout());

add(panMovement, BorderLayout.SOUTH);

add(panDisp, BorderLayout.CENTER);

add(panMenu, BorderLayout.EAST);

imgZappa = \_imgZappa;

addMouseMotionListener(this);

}

public void mouseDragged(MouseEvent e) {

nX = e.getX();

nY = e.getY();

repaint();

}

public void mouseMoved(MouseEvent e) {

}

public void paint(Graphics g) {

Graphics2D g2 = (Graphics2D) g;

g2.drawImage(imgZappa, nX, nY, this);

}

public class PanDisp extends JPanel {

public PanDisp(){

lblNew = new JLabel("Enter a Number");

add(lblNew); //add it to the Frame

}

}

public class PanStatus extends JPanel{

}

public class PanMovement extends JPanel {

public PanMovement(){

lblName = new JLabel("Enter a Number");

add(lblName); //add it to the Frame

lblStuff = new JLabel("More text to the south");

add(lblStuff);

panDisp.add(lblName);

}

}

public class PanMenu extends JPanel {

JButton[] btnButtons = new JButton[10];

public PanMenu(){

setLayout(new GridLayout(5,2));

for(int i=0;i<10;i++){

btnButtons[i]= new JButton("Button "+(i+1));

add(btnButtons[i]);

}

}

}

public static void main(String[] args) {

String sImageFile = "Zappa1.jpg";

Image imgZappa = Toolkit.getDefaultToolkit().getImage(

ComDragImage.class.getResource(sImageFile));

imgZappa = imgZappa.getScaledInstance(nW, nH, Image.SCALE\_DEFAULT);

JFrame frame = new JFrame("DragImage");

frame.getContentPane().add(new ComDragImage(imgZappa));

frame.setSize(800, 800);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

I added function to the buttons now, which will be used to change images.

Sunday

At first I had the idea that the button would change

String sImageFile = "Zappa1.jpg";

and change it to correspond to the button number according to which button was pressed but

My first attempt at changing images with buttons,

class Pressed implements ActionListener {

public void actionPerformed(ActionEvent event) {

sNum = event.getActionCommand();

JButton btnNew = new JButton(sNum);

panMenu.add(btnNew);

String sImageFile = "Zappa2.jpg";

System.out.println("Got here");

Image imgZappa = Toolkit.getDefaultToolkit().getImage(

ComDragImage.class.getResource(sImageFile));

JFrame frame = new JFrame("DragImage");

frame.getContentPane().add(new ComDragImage(imgZappa));

frame.setVisible(true);

revalidate();

}

}

It successfully changes the image, but it has to create a new frame to do so.

Upon fixing the code to make it reasonable, I've realized even If I close the frame upon opening a new one and making the new one look good, it will still be a pain to work with.

class Pressed implements ActionListener {

public void actionPerformed(ActionEvent event) {

sNum = event.getActionCommand();

JButton btnNew = new JButton(sNum);

panMenu.add(btnNew);

int nH = 60, nW = 60;

String sImageFile = "Zappa2.jpg";

System.out.println("Got here");

Image imgZappa = Toolkit.getDefaultToolkit().getImage(

ComDragImage.class.getResource(sImageFile));

imgZappa = imgZappa.getScaledInstance(nW, nH, Image.SCALE\_DEFAULT);

JFrame frame = new JFrame("DragImage");

frame.getContentPane().add(new ComDragImage(imgZappa));

frame.setSize(300, 300);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

revalidate();

}

}

My current ideas include making an array of the images that can change and are displayed by

Graphics2D g2 = (Graphics2D) g;

g2.drawImage(imgZappa, nX, nY, this);

In an attempt to get main to integrate with a class in the border layout, (which previously it complained that a static variable tried to talk to a non-static variable) now as an example and test I've tried to separate PanDisp:

import javax.swing.JPanel;

public class PanDisp extends JPanel{

public PanDisp(){

}

void Printer(){

System.out.println("Yay");

}

}

After moving panMenu over there were so many errors and fixing variables that were previously accessible that it was a dead end in terms of current and future endeavors.

Now I've fleshed out an idea, there will be an array of images, arimg Zappa[], and there will be img ZappaCurrent, which will be img Zappa = arimg Zappa[button pressed number]

The current LayoutViewer looks like:

static int nW = 60,nH = 60;

public static void main(String[] args) {

//String sImageFile = "Zappa1.jpg";

String sImageFile[]=new String[10];

Image imgZappa[]= new Image[10];

for(int i =0;i<10;i++){

sImageFile[i]="Zappa"+(i+1)+".jpg";

imgZappa[i] = Toolkit.getDefaultToolkit().getImage(

ComDragImage.class.getResource(sImageFile[i]));

imgZappa[i] = imgZappa[i].getScaledInstance(nW, nH, Image.SCALE\_DEFAULT);

}

JFrame frame = new JFrame("DragImage");

frame.getContentPane().add(new ComDragImage(imgZappa[0],imgZappa[1]));

frame.setSize(300, 300);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

Currently it only passes the first two images

as a test I tried to manually displayed the second image:

public void paint(Graphics g) {

Graphics2D g2 = (Graphics2D) g;

g2.drawImage(arimgZappa[1], nX, nY, this);

}

It was surprisingly successful.

I couldn't pass an array through, frame.getContentPane().add(new ComDragImage(imgZappa[])); so I decided to rectify it by passing the individual values and reforming the array.

After a while, I now have 10 images that change and I am on step 7.

I have gotten image rotating code from the internet:

private Image image;

AffineTransform identity = new AffineTransform();

Graphics2D g2d = (Graphics2D)g;

AffineTransform trans = new AffineTransform();

trans.setTransform(identity);

trans.rotate( Math.toRadians(45) );

g2d.drawImage(image, trans, this);

//

/\*\*

\* Rotates an image. Actually rotates a new copy of the image.

\*

\* @param img The image to be rotated

\* @param angle The angle in degrees

\* @return The rotated image

\*/

public static Image rotate(Image img, double angle)

{

double sin = Math.abs(Math.sin(Math.toRadians(angle))),

cos = Math.abs(Math.cos(Math.toRadians(angle)));

int w = img.getWidth(null), h = img.getHeight(null);

int neww = (int) Math.floor(w\*cos + h\*sin),

newh = (int) Math.floor(h\*cos + w\*sin);

BufferedImage bimg = toBufferedImage(getEmptyImage(neww, newh));

Graphics2D g = bimg.createGraphics();

g.translate((neww-w)/2, (newh-h)/2);

g.rotate(Math.toRadians(angle), w/2, h/2);

g.drawRenderedImage(toBufferedImage(img), null);

g.dispose();

return toImage(bimg);

}

and:

// The required drawing location

int drawLocationX = 300;

int drawLocationY = 300;

// Rotation information

double rotationRequired = Math.toRadian(45);

double locationX = image.getWidth() / 2;

double locationY = image.getHeight() / 2;

AffineTransform tx = AffineTransform.getRotateInstance(rotationRequired, locationX, locationY);

AffineTransformOp op = new AffineTransformOp(tx, AffineTransformOp.TYPE\_BILINEAR);

// Drawing the rotated image at the required drawing locations

g2d.drawImage(op.filter(image, null), drawLocationX, drawLocationY, null);

After trying to work with rotation I’ve got a paint method that looks like:

public void paint(Graphics g) {

double rotationRequired = Math.toRadians(45);

double locationX = imgZappaCurrent.getWidth() / 2;

double locationY = imgZappaCurrent.getHeight() / 2;

AffineTransform tx = AffineTransform.getRotateInstance(rotationRequired, locationX, locationY);

AffineTransformOp op = new AffineTransformOp(tx, AffineTransformOp.TYPE\_BILINEAR);

Graphics2D g2 = (Graphics2D) g;

g2.drawImage(op.filter(imgZappaCurrent, null), nX, nY, null);

//g2.drawImage(imgZappaCurrent, nX, nY, this);

} but it detects the majority of the code as errors so that code isn’t very transferable.

I also tried the code that uses a buffered image, but as implied in the name it is meant to work with buffered images so that was no help. In the end I’m abandoning how to rotate a picture because I’ve spent to long on it and it won’t even be integral to my apps for learning code.

**Modification to Steps;** Step 8 is unnecessary, due to step 9 being almost the same thing but more user friendly

Step 8 will be replaced by Step 9 entirely.

I made an actionlistener called

class Pressed implements ActionListener {

public void actionPerformed(ActionEvent event) {

nX+=10;

System.out.println(nX);

repaint();

} Which should change the x coordinate and move the image over 10 pixels.

I put paint inside panMovement which successfully made an image that could be moved by keyboard but the problem is now there are two images on the screen.

Finally I tried to work with keylisteners

int keyCode = e.getKeyCode();

if (keyCode == KeyEvent.VK\_UP){

nY ++;

}if(keyCode == KeyEvent.VK\_DOWN){

nY --;

}if (keyCode == KeyEvent.VK\_LEFT){

nX --;

}if(keyCode == KeyEvent.VK\_RIGHT){

nX --;

}

repaint();

System.out.println(nX + " "+ nY);

According to the internet it should work, but it doesn’t, I added addKeyListener(this);

And implemented a keylistener and even no errors are there but it gives no contact from pressing keys.