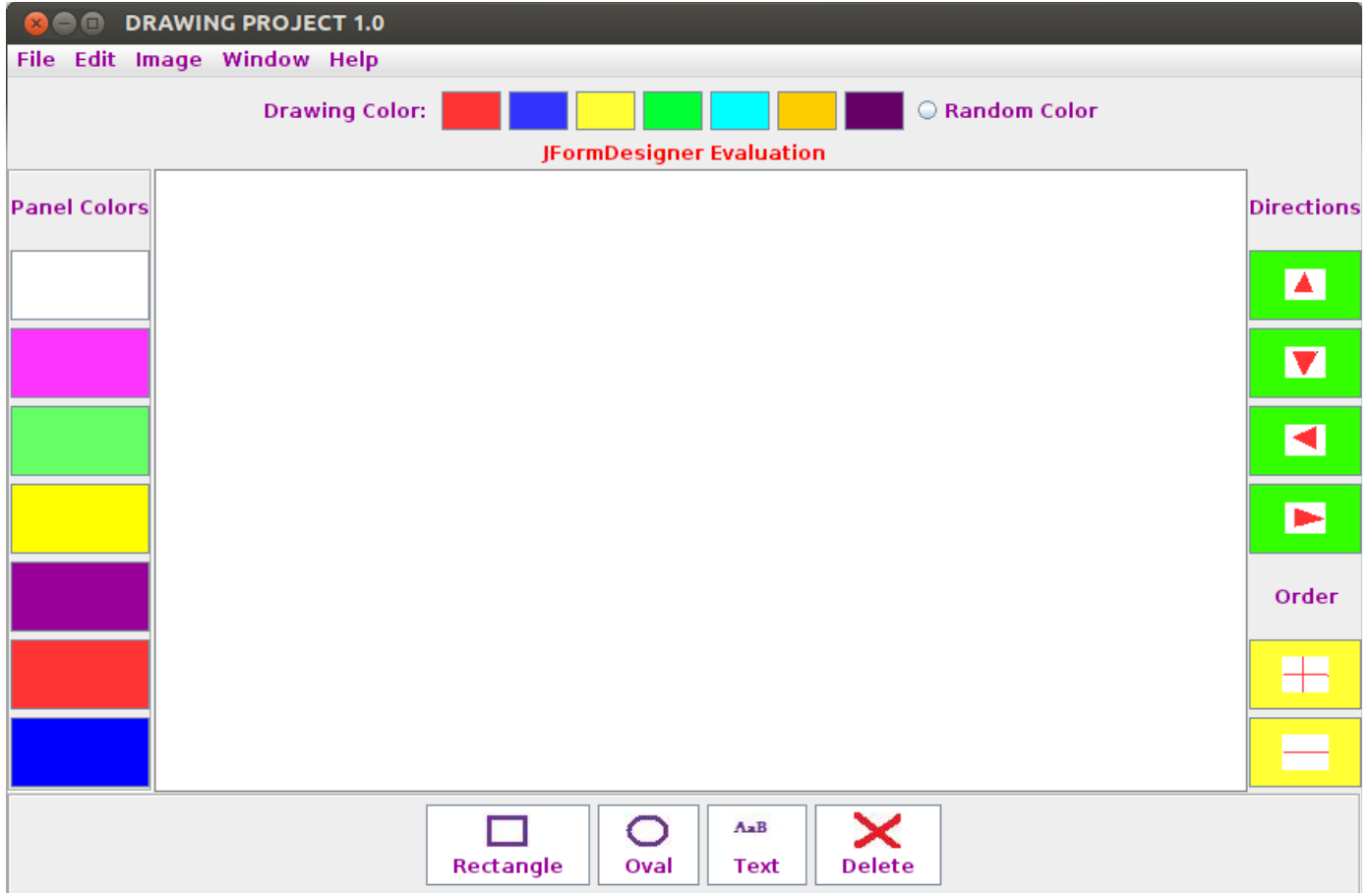


KISIM		YAPRAK NO : 09
YAPILAN İŞ	<i>Drawing Project'e Başlangıç</i>	TARİH : 03/07/2014

Chapter 9 Object-Oriented Programming: Inheritance ünitesine çalışıldı. Mühendisten bu konu hakkında bilgi alındı. Ayrıca bugün itibariyle **Drawing** projesine başlanılacağı söylendi. **Java**'da **GUT**ye henüz gelinmediği için **Drawing** projesinde arayüzün hazır verileceği, ancak **Toplu SMS Desktop** projesinde her şeyi sıfırdan oluşturmamız isteneceği belirtildi. Proje ile ilgili geniş çaplı bilgi alındı. Verilen arayüzde oluşturmamız gereken class yapısı anlatıldı. **Abstract classlarla** ilgili bilgi verildi. Class hiyerarşilerinin mantığı anlatıldı. Projede bulunması gereken ana değişkenler ve fonksiyonlar gösterildi. Bunların içeriğini bizim oluşturacağımız bilgisi verildi.



verilen arayüz

İlk olarak Shape abstract sınıfı oluşturuldu. Ardından Shapes sınıfından miras alan Rectangle sınıfı oluşturuldu. Shapes sınıfından miras alan her şekil için ortak olarak bulunması ve kendi içerlerinde tanımlanması gereken fonksiyon isimleri eklendi. Oluşturulan abstract fonksiyonların bir kısmı aşağıda yer almaktadır.

```
public abstract void animation();
```

```
public abstract void dP( DrawingPanel drwingPanel);
```

```
public abstract void draw( Graphics g);
```

```
public abstract void move( int px, int direction);
```

```
public abstract void move( Point ref, Point target);
```

```
public abstract void resize( Point ref, Point target, int direction);
```

```
public abstract void paste( Point p);

public abstract int getArea();

public abstract Shape copy();

public abstract boolean containsPoint( Point p);

public abstract Cursor getResizingCursor( Point p);
```

Her şekil sınıfında ortak olarak bulunan özelliklerde Shapes sınıfı içerisinde tanımlandı. Bu değişkenlerin get ve set metodları oluşturuldu. Tanımlanan değişkenlerin bir kısmı aşağıda örnek olarak verilmiştir.

```
public static final int SOUTH_WEST = 4;
public static final int SOUTH_EAST = 5;
public static final int NORTH_WEST = 6;
public static final int NORTH_EAST = 7;
public static final int NORTH = 8;
public static final int SOUTH = 9;
public static final int WEST = 10;
public static final int EAST = 11;
private boolean filled = false;
private boolean selected = false;
private Color color;
```

Oluşturulan get ve set metodlarının bir kısmı ise aşağıdaki gibi yer almaktadır.

```
public boolean isSelected()
{
    return selected;
}

public void setSelected( boolean selected)
{
    this.selected = selected;
}

public Color getColor()
{
    return color;
}

public void setColor( Color color)
{
    this.color = color;
}

public boolean isFilled()
{
    return filled;
}
```

```
public void setFilled( boolean filled)
{
    this.filled = filled;
}
```

Shape sınıfının ana yapısı oluşturulduktan sonra **Chapter 15: Graphics and Java 2D** 'ye çalışılmaya başlandı.

KISIM		YAPRAK NO : 10
YAPILAN İŞ	<i>Drawing Project'in Temel Sınıf ve Metodlarının Oluşturulması</i>	TARİH : 04/072014

Graphics and Java 2D konusu araştırılmaya devam edildi. *Drawing* projesi için *Shape sınıfından* miras alan *Rectangle sınıfı* oluşturuldu. *Dikdörtgen* çizimi için *x, y koordinatları* ile yükseklik ve genişlik bilgileri gerektiği için bunları ifade edecek değişkenler tanımlandı. Bunlar için aşağıdaki yapıcı oluşturuldu.

```
public Rectangle( int x, int y, int w, int h )
{
    this.x = x;
    this.y = y;
    this.height = h;
    this.width = w;
}
```

Ardından kontrol sınıfı olarak *DrawingPanel* oluşturuldu. Burda oluşturacağımız her şekli bir vektörde tutmak için aşağıdaki *shapes vektörü* oluşturuldu.

```
private Vector<Shape> shapes = new Vector<Shape>();
```

Varsayılan çizim rengi aşağıdaki kodla belirlendi.

```
private Color drawingColor = Color.RED;
```

Ayrıca renk panellerinde kullanılmak üzere bir Color[] dizisi oluşturuldu.

```
private static Color[] colors = { Color.MAGENTA, Color.BLUE, Color.CYAN, Color.LIGHT_GRAY,
    Color.GREEN, Color.PINK,
    Color.ORANGE, Color.YELLOW };
```

Hangi şekil seçilecekse o çizileceği için bunları ayırt edilebilmesi için şu değişkenler oluşturuldu.

```
public static final int DM_RECT = 0;
public static final int DM_OVAL = 1;
public static final int DM_TEXT = 2;
```

Sonra hazır olarak verilen arayüz sınıfı MainFrame'de DrawMode butonlarına tıklanınca modu seçmeye yarayan aşağıdaki ActionListener eklendi,

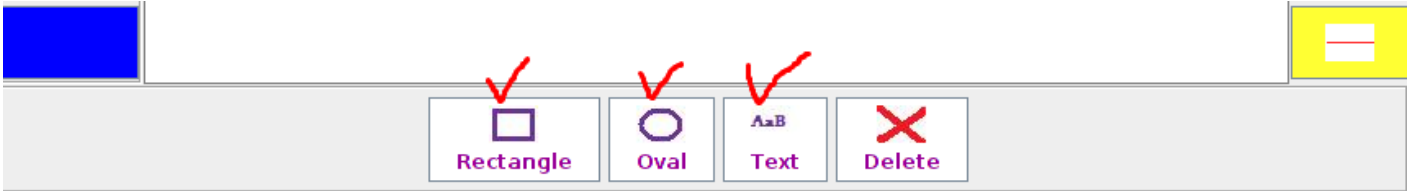
```
private class DrawModeButtonActionListener implements ActionListener
{
    @Override
    public void actionPerformed( ActionEvent e)
    {
        JButton button = (JButton) e.getSource();
        if ( btSelectedDM != null ) {
            btSelectedDM.setBorder( defaultButtonBorder );
        }

        button.setBorder( selectedButtonBorder );
        btSelectedDM = button;
        int dm = -1;
        if ( button == btRect ) {
            dm = DrawingPanel.DM_RECT;
        } else if ( button == btOval ) {
```

```

        dm = DrawingPanel.DM_OVAL;
    } else if ( button == btText ) {
        dm = DrawingPanel.DM_TEXT;
    }
    drawingPanel.setDrawingMode( dm );
}
}

```



Fareyle yeni şekil mi çizilecek(*draw*), var olan şekil mi taşıyacak(*move*) ya da şekil büyütülüp küçültülecek mi(*resize*) ayrımı için aşağıdaki değişkenler tanımlandı.

```

public static final int Drag_Draw = 0;
public static final int Drag_Move = 1;
public static final int Drag_Resize = 2;

```

Farenin tıklandığı nokta için *Point* türünden bir değişken oluşturuldu.

```

private Point pointMouseDown;

```

Çizilmiş şeklin üzerine tıklandığında bu şeklin seçilmiş şekil olarak *Shape* türünde tutulması için bir değişken tanımlandı.

```

public Shape selectedShape;

```

Şu anda çizilen şeklin ne olduğunun tutulduğu yine *Shape* türünde bir değişken oluşturuldu.

```

private Shape currentDrawingShape = null;

```

Shapes vektöründeki şekillerin her birinin çizilebilmesi için bir *paint* metodu *override* edildi.

```

public void paint( Graphics g)
{
    super.paint( g );
    for ( Shape shape : shapes ) {

        shape.draw( g );
    }
}

```

Tıklanılan noktadan başlayıp mouse'un release edildiği noktaya kadar bir dikdörtgen çizilebilmesi için *Rectangle sınıfında* aşağıdaki *constructor* oluşturuldu. Burada *pointMouseDown* başlangıç noktası, *mr-*(*mouse release*) ise bitiş noktasıdır. Yükseklik ve genişlik eksi değer çıkamayacağı için *Math.abs()* metodu ile mutlak değeri alınmıştır.

```
public Rectangle( Point pointMouseDown, Point mr )
{
    this.x = Math.min( pointMouseDown.x, mr.x );
    this.y = Math.min( pointMouseDown.y, mr.y );
    this.width = Math.abs( mr.x - pointMouseDown.x );

    this.height = Math.abs( mr.y - pointMouseDown.y );
}
```

Ardından Rectangle'in draw() metodu oluşturuldu.

```
public void draw( Graphics g)
{
    g.setColor( Color.BLACK );
    g.drawRect( x, y, width, height );
    g.fillRect( x , y, width, height );
}
```

KISIM		YAPRAK NO : 11
YAPILAN İŞ	Drawing Project'in Dikdörtgen Çizebilir Hale Getirilmesi	TARİH : 07/07/2014

Farenin sol click tuşuna basılıyken olması gereken olaylar aşağıdaki gibi oluşturuldu. Daha sonra kullanacağımız *şekli taşıma(move)*, *büyütme- küçültme(resize)* olayları için de gereken **switch case** yapısı oluşturuldu. Ardından *repaint()* metodu çağrıldı ki olaylar sonucunda en güncel durum ekrana yansiyabilsin.

```

public void mouseDragged( MouseEvent me)
{
    shapes.remove( currentDrawingShape );
    Point mr = me.getPoint();

    switch ( draggingMode )
    {
        case Drag_Draw :
            switch ( drawingMode )
            {
                case DM_RECT :
                    Rectangle rectangle = new Rectangle( pointMouseDown, mr );
                    rectangle.setColor( drawingColor );
                    currentDrawingShape = rectangle;
                    shapes.add( rectangle );

                    break;

                case DM_OVAL :
                    break;

                default :
                    break;
            }
            break;
        case Drag_Move :
            break;

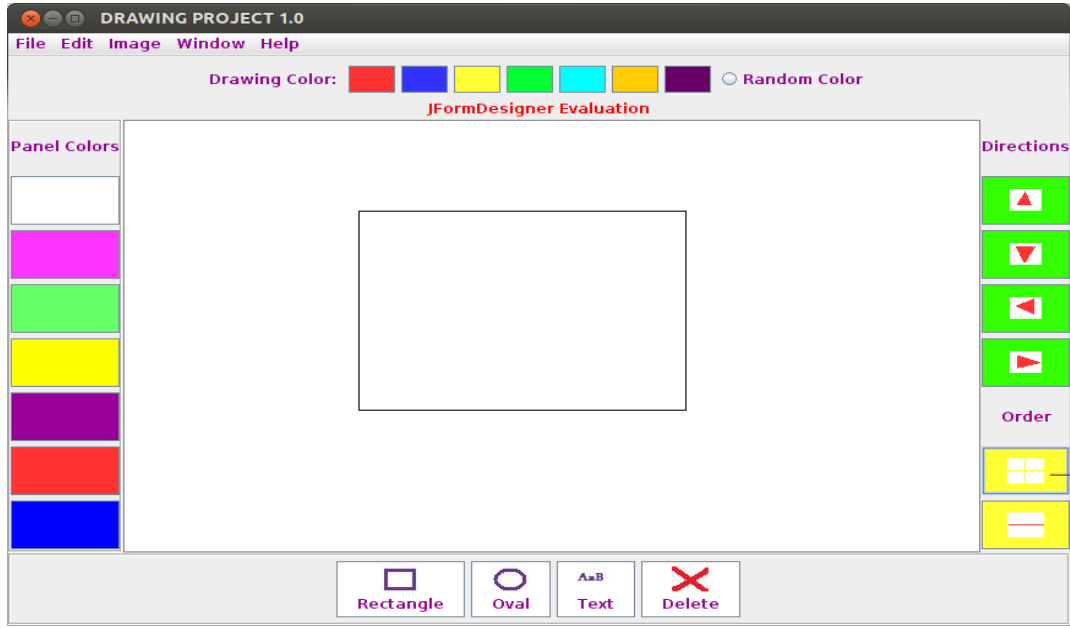
        case Drag_Resize :
            break;

        default :
            break;
    }

    repaint();
}

```

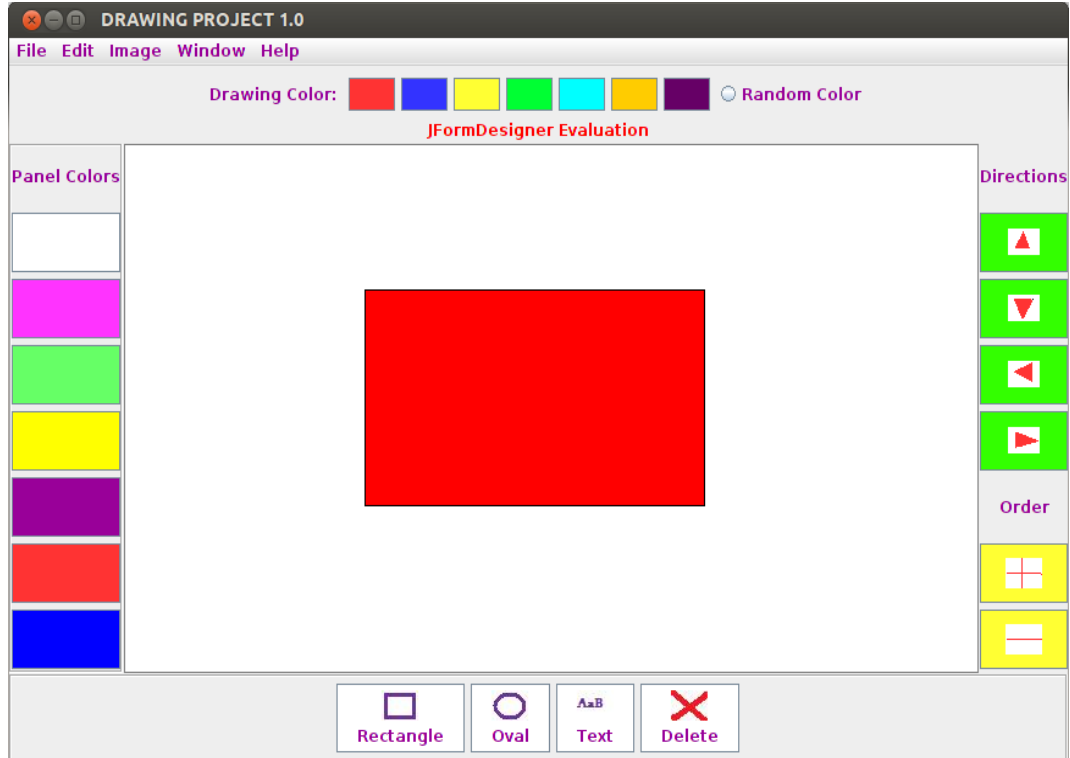
Yapılan deęiřikliklerden sonra program ařaęıdaki dikdörtgeni çizebilir hale hale geldi.



Dikdörtgen iini renkle doldurabilmek iin ařaęıdaki kod **Rectangle sınıfının Draw()** metoduna eklendi.

```
g.setColor( getColor() );  
g.fillRect( x + 1, y + 1, width - 1, height - 1 );
```

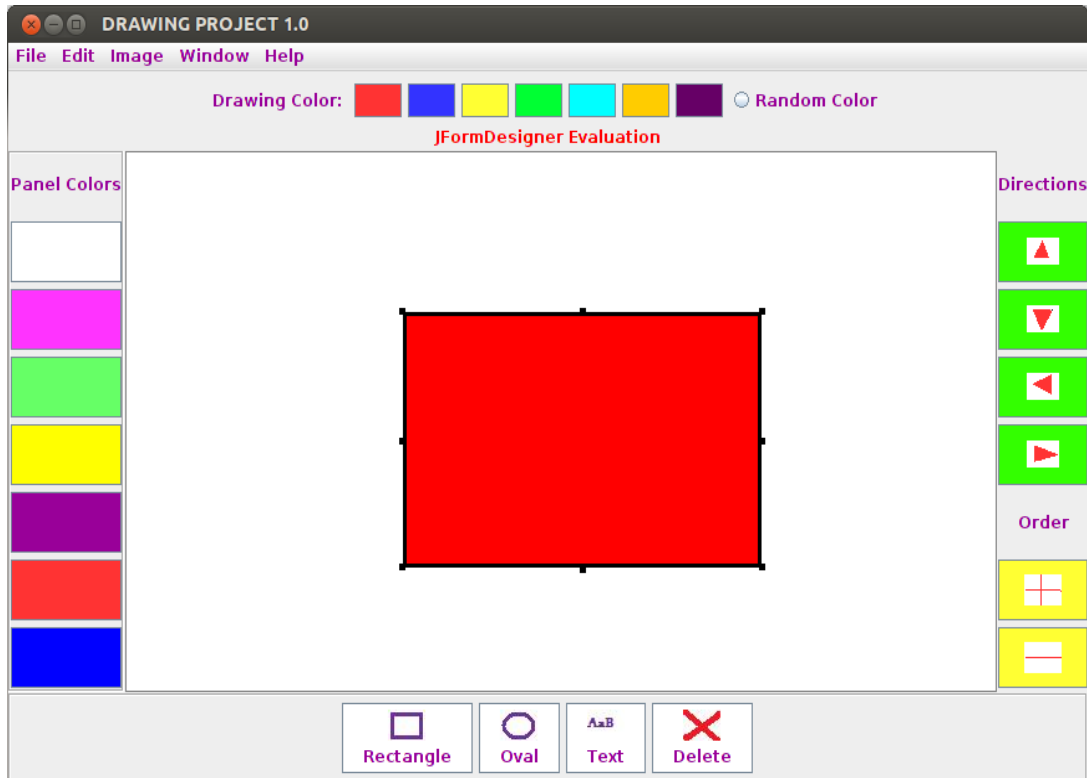
Bu kod eklendikten sonraki durum ařaęıdaki gibidir.



Ardından bir **dikdörtgen** şekli seçiliyken onun seçili olduğunu vurgulayan kenarlık için aşağıdaki kod **draw()** metodu içine eklendi.

```
if ( isSelected() ) {  
    g.setColor( Color.BLACK );  
    g.drawRect( x - 1, y - 1, width + 2, height + 2 );  
    g.drawRect( x + 1, y + 1, width - 2, height - 2 );  
  
    g.fillRect( x - 4, y - 4, 5, 5 );  
    g.fillRect( x - 4, y + height - 1, 5, 5 );  
    g.fillRect( x - 4, y + height / 2 - 1, 5, 5 );  
  
    g.fillRect( x + width, y - 4, 5, 5 );  
    g.fillRect( x + width, y + height / 2 + -1, 5, 5 );  
    g.fillRect( x + width, y + height - 1, 5, 5 );  
    g.fillRect( x + width / 2 - 1, y - 4, 5, 5 );  
    g.fillRect( x + width / 2 - 1, y + height + 1, 5, 5 );  
}
```

Ayrıca büyütme-küçütme işlemi için kullanılacak **resize noktaları** eklendi. **Kenarlıklar** ve **resize noktaları** eklenmiş şekil aşağıdaki gibidir.



Ayrıca **Chapter 16: Strings, Characters and Regular Expressions** ünitesine çalışıldı.

KISIM		YAPRAK NO : 12
YAPILAN İŞ	Directions Butonlarının Aktif Hale Getirilmesi	TARİH : 08/07/2014
<p>İlgili mühendis tarafından arayüzün sağ kısmında bulunan Directions butonlarıyla, <i>seçilen bir şeklin taşınması(move)</i> için gerekli fonksiyonların yazılması istendi. Öncelikle Rectangle sınıfı içerine move() metodu tanımlandı.</p> <pre> public void move(int px, int direction) { switch (direction) { case NORTH : this.y -= px; break; case SOUTH : this.y += px; break; case WEST : this.x -= px; break; case EAST : this.x += px; break; default : break; } } </pre> <p>Ardından DrawingPanel sınıfı içerisinde moveSelectedShape adında bir metod oluşturuldu.</p> <pre> public void moveSelectedShape(int direction) { if (selectedShape != null && selectedShape.isSelected() == true) { selectedShape.move(moveStep, direction); repaint(); } } </pre> <p>Burada moveStep 10 değerinde bir sabittir. Sonrasında MainFrame sınıfındaki Directions butonları, bu fonksiyonların ilgili yerlerde çağırılmasıyla aktive edildi.</p> <pre> private void btUpActionPerformed(ActionEvent e) { drawingPanel.moveSelectedShape(Shape.NORTH); } private void btDownActionPerformed(ActionEvent e) { drawingPanel.moveSelectedShape(Shape.SOUTH); } private void btLeftActionPerformed(ActionEvent e) { drawingPanel.moveSelectedShape(Shape.WEST); } </pre>		

```
private void btrightActionPerformed( ActionEvent e)
{
    drawingPanel.moveSelectedShape( Shape.EAST );
}
```

Bu son değişiklikten sonra **Directions** butonlarına tıklandığında; seçilen şekil *sağ, sol, aşağı ve yukarı* hareket edebilir hale geldi.

Fareyle her tıklandığında nokta büyüklüğünde şekiller oluşmaması için **DrawingPanel** sınıfında **ignorableShapeArea** değişkeni oluşturuldu.

```
private static int ignorableShapeArea = 20;
```

Rectangle sınıfında ise çizilen dikdörtgenin alanını hesaplayan bir metod tanımlandı.

```
public int getArea()
{
    return width * height;
}
```

Son olarak çizilen dikdörtgeni. vektöre ekleyen kod aşağıdaki gibi güncellenerek *alanı 20'den küçük* dikdörtgenlerin çizimi engellendi.

```
...
...
case DM_RECT :
    Rectangle rectangle = new Rectangle( pointMouseDown, mr );
    rectangle.setColor( drawingColor );
    currentDrawingShape = rectangle;
    if ( rectangle.getArea() > ignorableShapeArea )
        shapes.add( rectangle );
    break;
...
...
```

KISIM		YAPRAK NO : 13
YAPILAN İŞ	Fareyle Şekillerin Taşınması	TARİH : 09/07/2014
<p>Fareyle bir şeklin taşınması işlemi için (<i>sürükle-bırak</i>), <i>Rectangle</i> sınıfında aşağıdaki <i>move()</i> metodu override edildi.</p> <pre> public void move(Point ref, Point target) { int xd = target.x - ref.x; int yd = target.y - ref.y; this.x += xd; this.y += yd; } </pre> <p>Ardından <i>DrawingPanel</i> sınıfının <i>mouseDragged()</i> kısmında <i>move()</i> metodu çağrıldı.</p> <pre> public void mouseDragged(MouseEvent me) { case Drag_Move : if (selectedShape != null) { selectedShape.move(pointMouseDown, me.getPoint();); pointMouseDown = me.getPoint(); } break; } </pre> <p>Verilen noktanın şekil içerisinde mi olup olmadığını kontrol eden aşağıdaki metod <i>Rectangle</i> sınıfı içerisinde aşağıdaki gibi tanımlandı.</p> <pre> public boolean containsPoint(Point p) { return p.x > x && p.x < (x + width) && p.y > y && p.y < (y + height); } </pre> <p>İç içe geçmiş şekillere tıklandığında en üstteki şekil hangisiyse onu döndüren aşağıdaki <i>findTopMostShape()</i> metodu, <i>DrawingPanel</i> sınıfı içerisinde tanımlandı.</p> <pre> private Shape findTopMostShape(Point p) { for (int i = shapes.size() - 1; i >= 0; i--) { if (shapes.get(i).containsPoint(p)) return shapes.get(i); } return null; } </pre> <p>Ardışık birkaç şekle tıklayınca sadece en son tıklananın şeklin <i>selectedShape</i> olması için diğer şekillerin <i>selected</i> değişkenleri <i>false</i>'a setlendi.</p>		

```
public void mousePressed( MouseEvent me)
{
    pointMouseDown = me.getPoint();
    Shape shape = findTopMostShape( pointMouseDown );
    if ( shape != null ) {
        shape.setSelected( true );
        selectedShape = shape;
        repaint();

        for ( int i = 0; i < shapes.size(); i++ ) {
            if ( shape == shapes.get( i ) ) {}

            else
                shapes.get( i ).setSelected( false );
        }
    } else {
        for ( int i = 0; i < shapes.size(); i++ ) {
            shapes.get( i ).setSelected( false );
        }
    }

    repaint();
}
```

Ayrıca **Chapter 14 GUI Components: Part 1** ünitesine çalışılmaya başlandı.

KISIM		YAPRAK NO : 14
YAPILAN İŞ	Up, Down ve Color Butonlarının Aktifleştirilmesi	TARİH : 10/07/2014

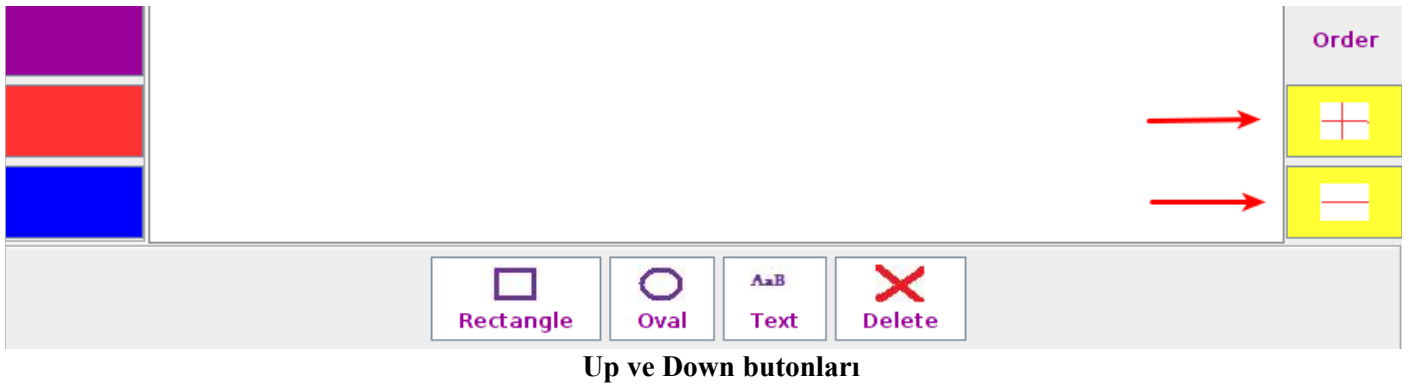
Farenin sol tuşu **release** edildiğinde **currentDrawingShape** değişkeninin **null**'a atanması için gereken kod **DrawingPanel** sınıfı içerisine aşağıdaki gibi eklendi.

```

public void mouseReleased( MouseEvent me)
{
    if ( currentDrawingShape != null ) {
        currentDrawingShape.setFilled( true );
        currentDrawingShape = null;
        repaint();
    }
}

```

Daha sonra program arayüzünün **sağ alt kısmında yer alan Up ve Down butonlarını** aktif hale getirme için aşağıdaki metodlar **DrawingPanel** sınıfına eklendi. Bu butonlar iç içe geçmiş şekillerden seçili olanı, daha altta veya daha üstte konumlanmasını sağlamaktadır.



Up ve Down butonları

```

public void increaseIndexofSelectedShape()
{
    int index = shapes.indexOf( selectedShape );
    if ( index < shapes.size() - 1 && selectedShape.isSelected() ) {
        index++;
        shapes.remove( selectedShape );
        shapes.add( index, selectedShape );
    }
    repaint();
}

public void decreaseIndexofSelectedShape()
{
    int index = shapes.indexOf( selectedShape );

    if ( index > 0 && selectedShape.isSelected() ) {
        index--;
        shapes.remove( selectedShape );
        shapes.add( index, selectedShape );
    }

    repaint();
}

```

Ardından bu fonksiyonlar *MainFrame* arayüz sınıfında ilgili *btUp* ve *btDown* adlı butonlara bağlandı.

```
private void btORUpActionPerformed( ActionEvent e)
{
    drawingPanel.increaseIndexOfSelectedShape();
}
```

```
private void btORDownActionPerformed( ActionEvent e)
{
    drawingPanel.decreaseIndexOfSelectedShape();
}
```

Bağlama işleminden sonra *Up* ve *Down* butonları da aktif hale geldi.

Arayüzün üst kısmındaki çizim rengini belirleyen butonların aktif hale gelmesi için yine *MainFrame* arayüz sınıfına aşağıdaki kodlar eklendi.

```
private class DrawingColorButtonActionListener implements ActionListener
{
    @Override
    public void actionPerformed( ActionEvent e)
    {
        JButton button = (JButton) e.getSource();

        // mark previos selected button as deselected
        if ( btSelectedDC != null ) {
            btSelectedDC.setBorder( defaultButtonBorder );
        }

        // make clicked button as selected,
        button.setBorder( selectedButtonBorder );
        btSelectedDC = button;

        // in case drawing panel is in random color mode, fix it.
        drawingPanel.setRandomColor( false );
        btDCRandomColor.setSelected( false );

        drawingPanel.setDrawingColor( button.getBackground() );
    }
}
```

Random Color butonunu aktif etmek için de **MainFrame** sınıfına eklenen kod aşağıdaki gibidir.

```
private void btDCRandomColorActionPerformed( ActionEvent e)
{
    if ( btSelectedDC != null ) {
        btSelectedDC.setBorder( defaultButtonBorder );
    }
    if ( btDCRandomColor.isSelected() ) {
        drawingPanel.selectRandomColor();
        drawingPanel.setRandomColor( true );

    } else {
        drawingPanel.setDrawingColor( Color.RED );
        drawingPanel.setRandomColor( false );
    }
}
```

Rastgele **RGB aralığında** renk seçimi yapan metod, **DrawingPanel** sınıfında aşağıdaki gibi tanımlanmıştır.

```
public void selectRandomColor()
{
    int r = new Random().nextInt( 256 );
    int g = new Random().nextInt( 256 );
    int b = new Random().nextInt( 256 );
    Color color = new Color( r, g, b );
    setDrawingColor( color );
}
```

Bu metod yine **DrawinPanel sınıfı** içerisinde **mouseReleased()** kısmında çağırılmıştır.

```
if ( isRandomColor() ) {
    selectRandomColor();
}
```

Ayrıca **GUI Components: Part 1** ünitesine çalışılmaya devam edildi.

KISIM		YAPRAK NO : 15
YAPILAN İŞ	<i>Mouse İşaretçilerinin Eklenmesi</i>	TARİH : 11/07/2014
<p>İlgili mühendis tarafından <i>fare şekil üzerine gelince fare işaretçisi move görünümüne, büyütüp küçültme modunda ise resize görünümüne geçecek şekilde</i> projeye eklenmesi ve <i>delete butonunun</i> aktifleştirilmesi istendi. Bunun üzerine <i>mouse cursors</i> konusu internet üzerinden araştırıldı. Öncelikle <i>Rectangle sınıfı</i> içerisine dikdörtgenlerde bu işlemi sağlayacak <i>getResizingCursor()</i> metodu tanımlandı ve aşağıdaki gibi override edildi.</p> <pre> public Cursor getResizingCursor(Point p) { if ((p.x > x - 5 && p.x < x + 5 && p.y > y - 5 && p.y < y + 5)) { return Cursor.getPredefinedCursor(Cursor.NW_RESIZE_CURSOR);} else if ((p.x > x + width - 5 && p.x < x + width + 5 && p.y > y - 5 && p.y < y + 5)) { return Cursor.getPredefinedCursor(Cursor.NE_RESIZE_CURSOR);} else if ((p.x > x - 5 && p.x < x + 5 && p.y > y + height - 5 && p.y < y + height + 5)) { return Cursor.getPredefinedCursor(Cursor.SW_RESIZE_CURSOR);} else if ((p.x > x + width - 5 && p.x < x + width + 5 && p.y > y + height - 5 && p.y < y + height + 5)) { return Cursor.getPredefinedCursor(Cursor.SE_RESIZE_CURSOR);} else if ((p.x > x + width - 5 && p.x < x + width + 5 && p.y > y && p.y < y + height)) { return Cursor.getPredefinedCursor(Cursor.E_RESIZE_CURSOR);} else if ((p.x > x - 5 && p.x < x + 5 && p.y > y && p.y < y + height)) { return Cursor.getPredefinedCursor(Cursor.W_RESIZE_CURSOR);} else if ((p.y > y + height - 5 && p.y < y + height + 5 && p.x > x && p.x < x + width)) { return Cursor.getPredefinedCursor(Cursor.S_RESIZE_CURSOR);} else if ((p.y > y - 5 && p.y < y + 5 && p.x > x && p.x < x + width)) { return Cursor.getPredefinedCursor(Cursor.N_RESIZE_CURSOR);} else if (containsPoint(p)) { return Cursor.getPredefinedCursor(Cursor.MOVE_CURSOR);} return null; } </pre> <p>Ardından bu metod <i>DrawingPanel</i> sınıfının <i>mouseMoved()</i> olayında aşağıdaki gibi kullanılarak işlem tamamlandı.</p> <pre> public void mouseMoved(MouseEvent me) { Shape shape = findTopMostShape(me.getPoint()); if (shape != null) { Cursor c = shape.getResizingCursor(me.getPoint()); if (c.equals(Cursor.getPredefinedCursor(Cursor.MOVE_CURSOR))) { draggingMode = Drag_Move; } else { draggingMode = Drag_Resize; } } } </pre>		

```
        setCursor( c );  
    } else {  
        setCursor( Cursor.getDefaultCursor() );  
        draggingMode = Drag_Draw;    }}
```

Sonrasında *delete* işlemi için *DrawingPanel* sınıfında aşağıdaki metod tanımlandı.

```
public void deleteSelectedShape()  
{  
    shapes.remove( selectedShape );  
  
    repaint();  
}
```

Son olarak *MainFrame* arayüz sınıfının ilgili buton olayında *deleteSelectedShape()* metodu çağrılarak buton aktifleştirildi.

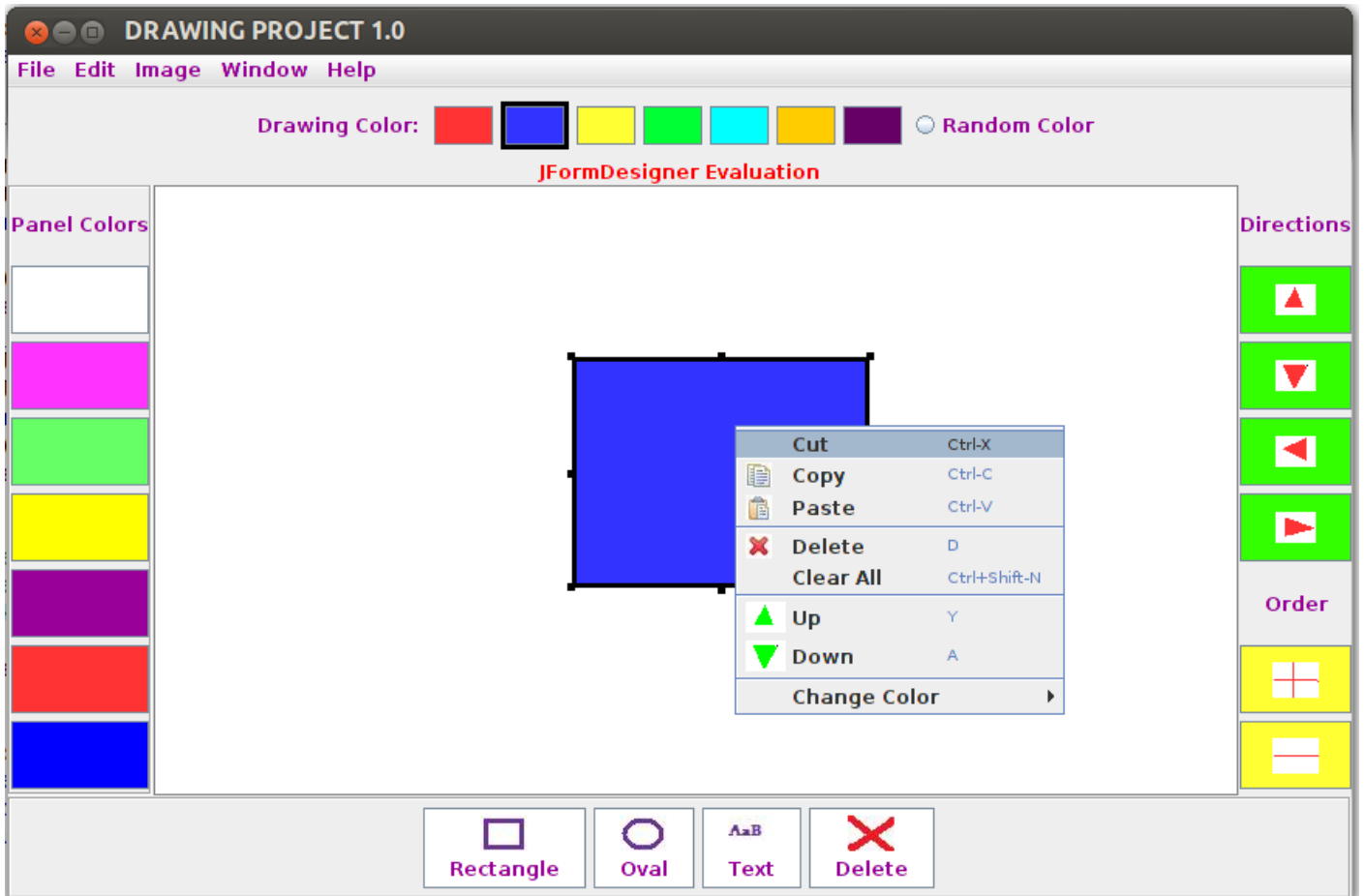
```
private void btDeleteActionPerformed( ActionEvent e)  
{  
    drawingPanel.deleteSelectedShape();  
}
```

KISIM		YAPRAK NO : 16
YAPILAN İŞ	<i>Sağ Tık ile Açılan Popup Menü Oluşturulması</i>	TARİH : 14/07/2014
<p>Mühendis tarafından, arayüzü hazırlanmış bir popup menünün, sadece fare herhangi bir şekil üzerindeyken ve sadece sağ click olayında açılması için gerekli kod ve düzenlemelerin gerçekleştirilmesi istendi. Bunun üzerine <i>Java</i>'da popup menü ve right-left click olayları internet üzerinden ayrıntılı şekilde araştırıldı. Öncelikle popup menü elemanlarının bulunduğu MyPopupMenu sınıfı DrawingPanel sınıfının yapıcısında aşağıdaki gibi çağrıldı.</p> <pre> popupMenu = new MyPopupMenu(this); add(popupMenu); </pre> <p>Ardından mouseReleased() olayına aşağıdaki kodlar eklenerek sadece şekil üzerinde ve sağ tık olayıyla popup menünün açılması sağlandı.</p> <pre> public void mouseReleased(MouseEvent me) { if (SwingUtilities.isRightMouseButton(me)) { Shape shape = findTopMostShape(me.getPoint()); if (shape != null cuttedOrCopiedShape != null) { popupMenu.setVisible(true); popupMenu.show(this, me.getX(), me.getY()); } } } </pre> <p>Sonrasında sağ tıklamayla çizim yapılmasını önlemek ve sadece farenin sol tuşu ile çizim yapılabilmesi için mouseDragged() olayı içerisindeki tüm kodlar aşağıdaki if yapısı içerisine alındı.</p> <pre> public void mouseDragged(MouseEvent me) { if (SwingUtilities.isLeftMouseButton(me)) { } } </pre> <p>Böylece yan sayfadaki popup menü sağ tıklamayla açılabilir hale geldi. Sonrasında açılır menüde görülen Delete işlemini aktifleştirmek için MyPopupMenu sınıfının ilgili kod kısmına aşağıdaki kod eklendi.</p> <pre> public void actionPerformed(ActionEvent e) { pmiDeleteActionPerformed(e); } </pre>		

```

private void pmiDeleteActionPerformed( ActionEvent e)
{
    drawingPanel.deleteSelectedShape();
}

```



KISIM		YAPRAK NO : 17
YAPILAN İŞ	Kes, Kopyala, Yapıştır Metodlarının Oluşturulması	TARİH : 15/07/2014
<p>Mühendis tarafından seçili şekilleri <i>kesme</i> ve <i>kopyalama</i>, ardından <i>boş bir alan üzerine yapıştırma</i> için gerekli <i>copySelectedShape()</i>, <i>cutSelectedShape()</i>, <i>pasteSelectedShape()</i> metodlarının oluşturulması istendi. Öncelikle DrawingPanel sınıfında <i>global cuttedOrCopiedShape</i> değişkeni oluşturuldu. Ardından ilgili metodlar aşağıdaki gibi tanımlandı.</p> <pre> public void cutSelectedShape() { if (selectedShape != null) { tempShape = selectedShape; cuttedOrCopiedShape = tempShape; cuttedOrCopiedShape.setSelected(false); shapes.remove(selectedShape); repaint(); } } public void copySelectedShape() { if (selectedShape.isSelected()) { tempShape = selectedShape; cuttedOrCopiedShape = tempShape.copy(); cuttedOrCopiedShape.setColor(tempShape.getColor()); } } public void pasteSelectedShape() { cuttedOrCopiedShape = cuttedOrCopiedShape.copy(); cuttedOrCopiedShape.setColor(tempShape.getColor()); shapes.add(cuttedOrCopiedShape); cuttedOrCopiedShape.paste(pointMouseDown); repaint(); } </pre> <p>Algoritma mantığımız <i>her çizimde yeni bir şekil nesnesi oluşturulup vektöre eklenmesi</i> olduğu için Rectangle sınıfı içerisinde <i>yeni nesne oluşturan bir metod</i> yazıldı.</p> <pre> public Shape copy() { Rectangle copyRect = new Rectangle(x, y, width, height); copyRect.setFilled(true); return copyRect; } </pre> <p>Bu metodların ilgili <i>popup menü</i> elemanlarına bağlanması ise MyPopupMenu sınıfında yan sayfadaki gibi çağrılmasıyla sağlanmıştır.</p>		

```

public void actionPerformed((ActionEvent e)
    {
        pmiCutActionPerformed( e );
    }

    private void pmiCutActionPerformed((ActionEvent e)
    {
        drawingPanel.cutSelectedShape();
    }







public void actionPerformed((ActionEvent e)
    {
        pmiCopyActionPerformed( e );
    }

    private void pmiCopyActionPerformed((ActionEvent e)
    {
        drawingPanel.copySelectedShape();
    }

public void actionPerformed((ActionEvent e)
    {
        pmiPasteActionPerformed( e );
    }







    private void pmiPasteActionPerformed((ActionEvent e)
    {
        drawingPanel.pasteSelectedShape();
    }

```

	Cut	Ctrl-X
	Copy	Ctrl-C
	Paste	Ctrl-V
	Delete	D
	Clear All	Ctrl+Shift-N
	Up	Y
	Down	A
	Change Color	►

Bu değişiklikler ardından *yukarıdaki popup menüdeki kes, kopyala, yapıştır* işlevleri aktifleştirilmiş oldu.

KISIM		YAPRAK NO : 18
YAPILAN İŞ	Klavye Kısayollarının Eklenmesi	TARİH : 16/07/2014
<p>Bugün Drawing projesini klavye kısayol tuşlarıyla kontrol edilebilecek hale getirmemiz istendi. Bu konuyla ilgili “KeyListener in Java” başlığı altında internetten araştırma yapıldıktan sonra aşağıdaki şekilde gerçekleştirildi.</p> <pre> public void keyPressed(KeyEvent ke) { if (selectedShape != null) { if (ke.getKeyCode() == KeyEvent.VK_UP) { //yukarı yön tuşu ile yukarı hareket selectedShape.move(moveStep, 8); repaint(); } if (ke.getKeyCode() == KeyEvent.VK_DOWN) { //aşağı yön tuşu ile aşağı hareket selectedShape.move(moveStep, 9); repaint(); } if (ke.getKeyCode() == KeyEvent.VK_LEFT) { //sol yön tuşu ile sola hareket selectedShape.move(moveStep, 10); repaint(); } if (ke.getKeyCode() == KeyEvent.VK_RIGHT) { //sağ yön tuşu ile sağa hareket selectedShape.move(moveStep, 11); repaint(); } if (ke.getKeyChar() == 'y') { increaseIndexofSelectedShape(); //y harfine basılınca şekli bir üste taşı } if (ke.getKeyChar() == 'a') { decreaseIndexofSelectedShape(); // a harfine basılınca şekli bir alta taşı } } } </pre> <p>DrawingPanel sınıfına eklenen yukarıdaki kodlarla birlikte seçilen şeklin klavye tuşları ile 4 yönde hareketine ve iç içe geçmiş şekilleri üste veya alta taşıma işlemine olanak sağlanmış oldu. Ayrıca MyPopupMenu sınıfına kopyalama işlemi için ctrl+c, kesme işlemi için ctrl+x, yapıştırma işlemi için ctrl+v, silme işlemi için d tuşu ve son olarak tüm şekilleri silme işlemi için ctrl+n kısayol tuşları yan sayfadaki kodlarla tanımlandı.</p>		

	Cut	Ctrl-X
	Copy	Ctrl-C
	Paste	Ctrl-V
	Delete	D
	Clear All	Ctrl+Shift-N
	Up	Y
	Down	A
	Change Color	▶

popup menüde belirtilen kısayollar

```
// ---- pmiCut ----
```

```
pmiCut.setAccelerator( KeyStroke.getKeyStroke( KeyEvent.VK_X, KeyEvent.CTRL_MASK ) );
```

```
// ---- pmiCopy ----
```

```
pmiCopy.setAccelerator( KeyStroke.getKeyStroke( KeyEvent.VK_C, KeyEvent.CTRL_MASK ) );
```

```
// ---- pmiPaste ----
```

```
pmiPaste.setAccelerator( KeyStroke.getKeyStroke( KeyEvent.VK_V, KeyEvent.CTRL_MASK ) );
```

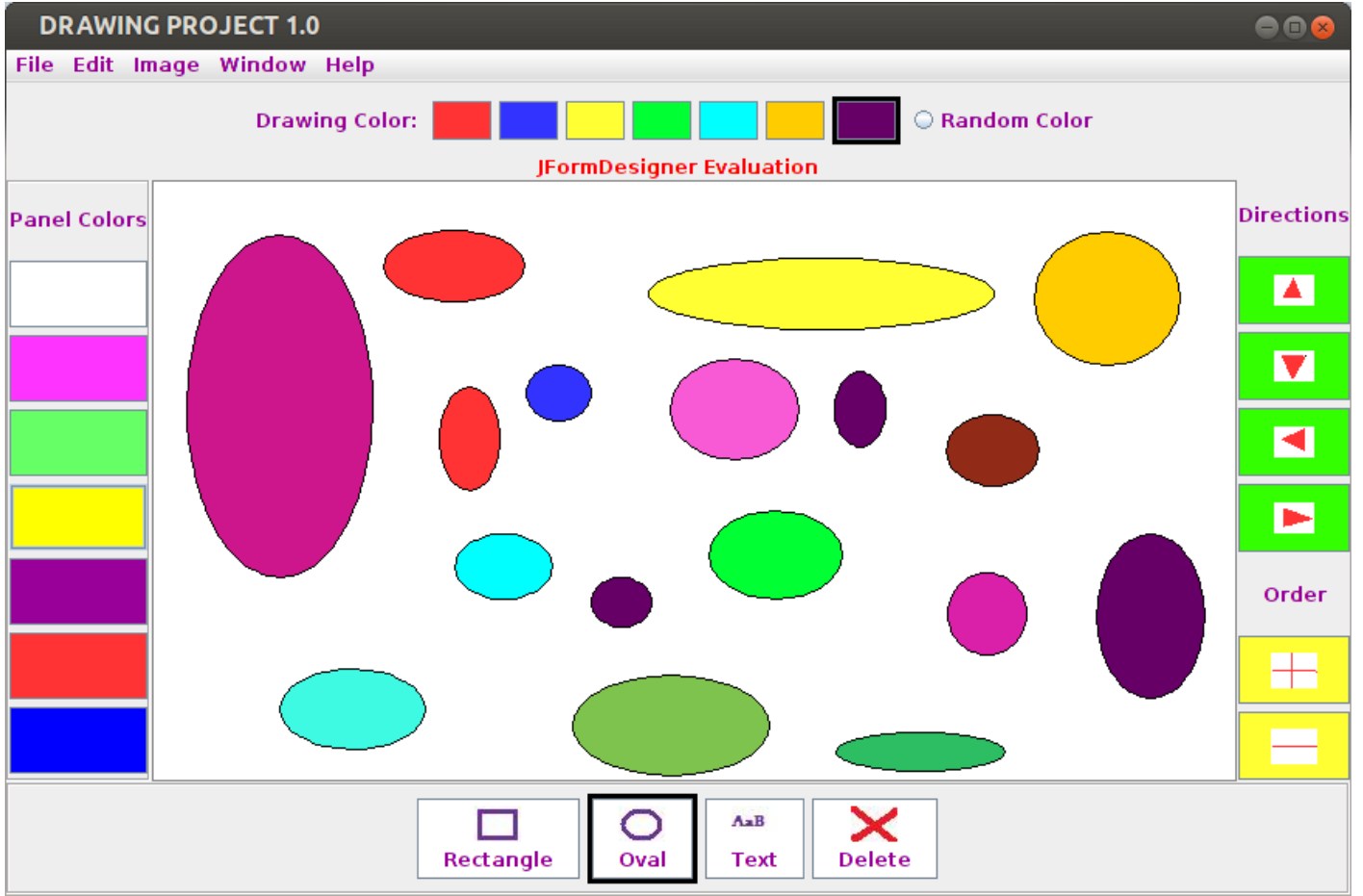
```
// ---- pmiDelete ----
```

```
pmiDelete.setAccelerator( KeyStroke.getKeyStroke( KeyEvent.VK_D, 0 ) );
```

```
// ---- pmiClearAll ----
```

```
pmiClearAll.setAccelerator( KeyStroke.getKeyStroke( KeyEvent.VK_N, KeyEvent.CTRL_MASK |
KeyEvent.SHIFT_MASK ) );
```


KISIM		YAPRAK NO : 19
YAPILAN İŞ	Oval Sınıfı Oluşturulması	TARİH : 17/07/2014
<p>Şu ana kadar <i>sadece dikdörtgen</i> çizebilen projeye bugün <i>oval</i> de çizmesini sağlayacak şekilde bir <i>Oval sınıfı</i> eklememiz istendi. Bunun üzerine <i>Java'da oval çizme</i> konusu çalışıldı. Oval ile dikdörtgen şekillerinin pek çok özelliği aynı olduğu için <i>Rectangle</i> sınıfından aşağıdaki gibi miras alan bir <i>Oval</i> sınıfı oluşturuldu.</p> <pre> public Oval(Point pointMouseDown, Point mr) { super(pointMouseDown, mr); } public Oval(int x, int y, int w, int h) { super(x, y, w, h); } </pre> <p>Ardından <i>Oval</i> sınıfının <i>draw()</i> metodu aşağıdaki gibi oluşturuldu. Seçiliyken etrafını saran <i>siyah bir kenarlık</i> ile şekli <i>alt-üst,sağ-sol</i> uç kısımlarına boyutlandırma noktaları eklendi.</p> <pre> public void draw(Graphics g) { g.setColor(getColor()); g.drawOval(x, y, width, height); g.fillOval(x, y, width, height); g.setColor(Color.BLACK); g.drawOval(x, y, width, height); if (isSelected()) { g.setColor(Color.BLACK); g.drawOval(x - 1, y - 1, width - 1, height - 1); g.drawOval(x, y, width, height); g.drawOval(x + 1, y + 1, width - 1, height - 1); g.fillRect(x - 4, y + height / 2 - 1, 5, 5); g.fillRect(x + width, y + height / 2 + -1, 5, 5); g.fillRect(x + width / 2 - 1, y - 4, 5, 5); g.fillRect(x + width / 2 - 1, y + height + 1, 5, 5); } } </pre> <p>Son olarak aşağıdaki <i>kopyalama</i> ve <i>kesme</i> işlemi için gerekli <i>copy() metodu</i> eklendikten sonra proje bir sonraki sayfadaki oval şekilleri çizebilir hale geldi.</p> <pre> public Shape copy() { Oval copyOval = new Oval(x, y, width, height); copyOval.setFilled(true); return copyOval; } </pre>		



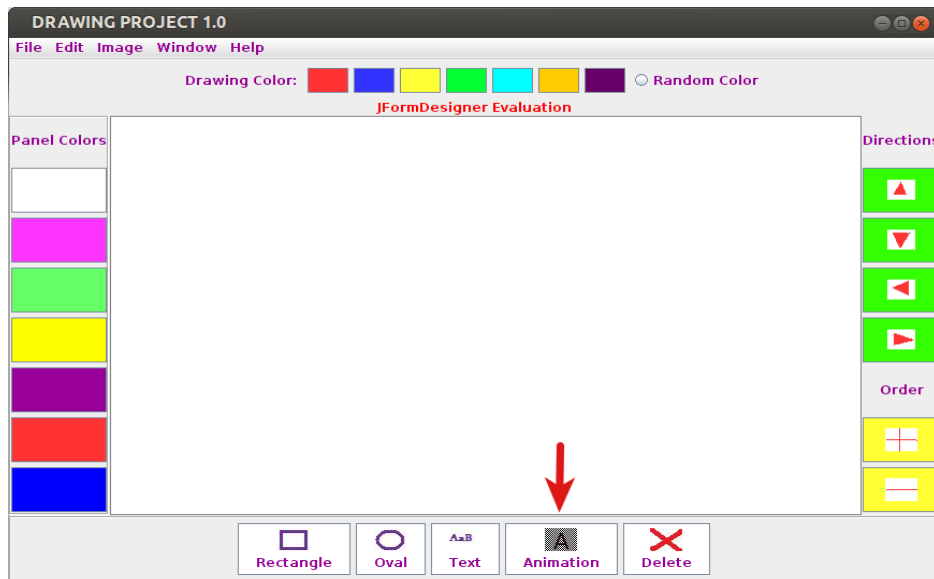
KISIM		YAPRAK NO : 20
YAPILAN İŞ	Animation Butonu ve Metodu Ekleme	TARİH : 18/07/2014

Bugün **Drawing Project**'e, çizilen şekillerin çizim ekranında rastgele renk değişirerek hareket edip duvarlara çarparak harekete devam etmesi şeklinde animasyonunu sağlayacak bir metod oluşturmamız istendi. Hem hareket hem de çizimin aynı anda olabilmesi için **thread yapısı** kullanılması gerektiği belirtildi. Ayrıca üzerine tıklandığında animasyonu başlatan bir buton eklenilmesi istendi.

Öncelikle **Java'da JButton** araştırılarak çalışıldı. Ardından buton eklemek için **MainFrame** sınıfına aşağıdaki kod eklendi ve ilgili bağlamalar yapıldı. Butona basıldığı zaman çağrılacak metod olarak da **startAnimate()** metodu verildi. Bu sayede aşağıdaki görüntüdeki bir animasyon butonu eklenmiş oldu.

```
// ---- btAnimation ----
btAnimation.setBackground( Color.white );
btAnimation.setFocusable( false );
btAnimation.setIcon( new ImageIcon( getClass().getResource( "/drawingprj/images/animation.jpg" ) ) );
btAnimation.setVerticalAlignment( SwingConstants.TOP );
btAnimation.setToolTipText( "Start Animation" );
btAnimation.setText( "Animation" );
btAnimation.setVerticalTextPosition( SwingConstants.BOTTOM );
btAnimation.setHorizontalTextPosition( SwingConstants.CENTER );
btAnimation.setForeground( new Color( 153, 0, 153 ) );
btAnimation.addActionListener( new ActionListener() {
    public void actionPerformed( ActionEvent e )
    {
        btAnimationActionPerformed( e );
    }
} );
southPanel.add( btAnimation );

private void btAnimationActionPerformed( ActionEvent e )
{
    drawingPanel.startAnimate();
}
```



Eklenen Animation Butonu

Sonrasında *Java'da threadler* konusu geniş çaplı araştırıldı ve aşağıdaki gibi bir *thread yapısı* kullanan metod oluşturuldu. Burada *isAnimActive* değişkeni *global olarak true* olarak tanımlandı. Metodun sadece bu değişken *true* iken çalışacak şekilde *while döngüsüne* alındı ki istenildiğinde *animasyon* durdurulabilsin.

```
public void startAnimate()
{
    isAnimActive = true;
    animThread = new Thread() {
        public void run()
        {
            while ( isAnimActive ) {
                for ( int i = 0; i < shapes.size(); i++ ) {
                    Shape shape = shapes.get( i );
                    if ( !shape.isFilled() ) continue;
                    selectRandomColor();
                    shape.setColor( getDrawingColor() );
                    shape.dP( drwingPanel );
                    shape.animation();
                }
                repaint();
                try {
                    sleep( 100 );
                } catch ( InterruptedException e ) {
                    e.printStackTrace();
                }
            }
        }
    };
    animThread.start();
}
```

Animasyon aktifken butona basılınca durdurulması için butona tıklanma olayı aşağıdaki gibi güncellendi.

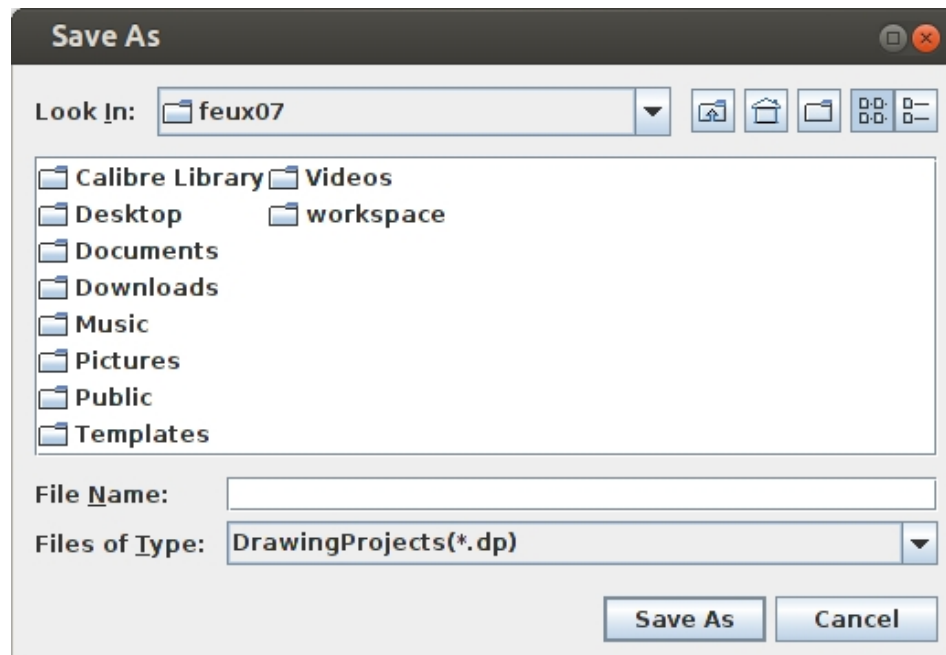
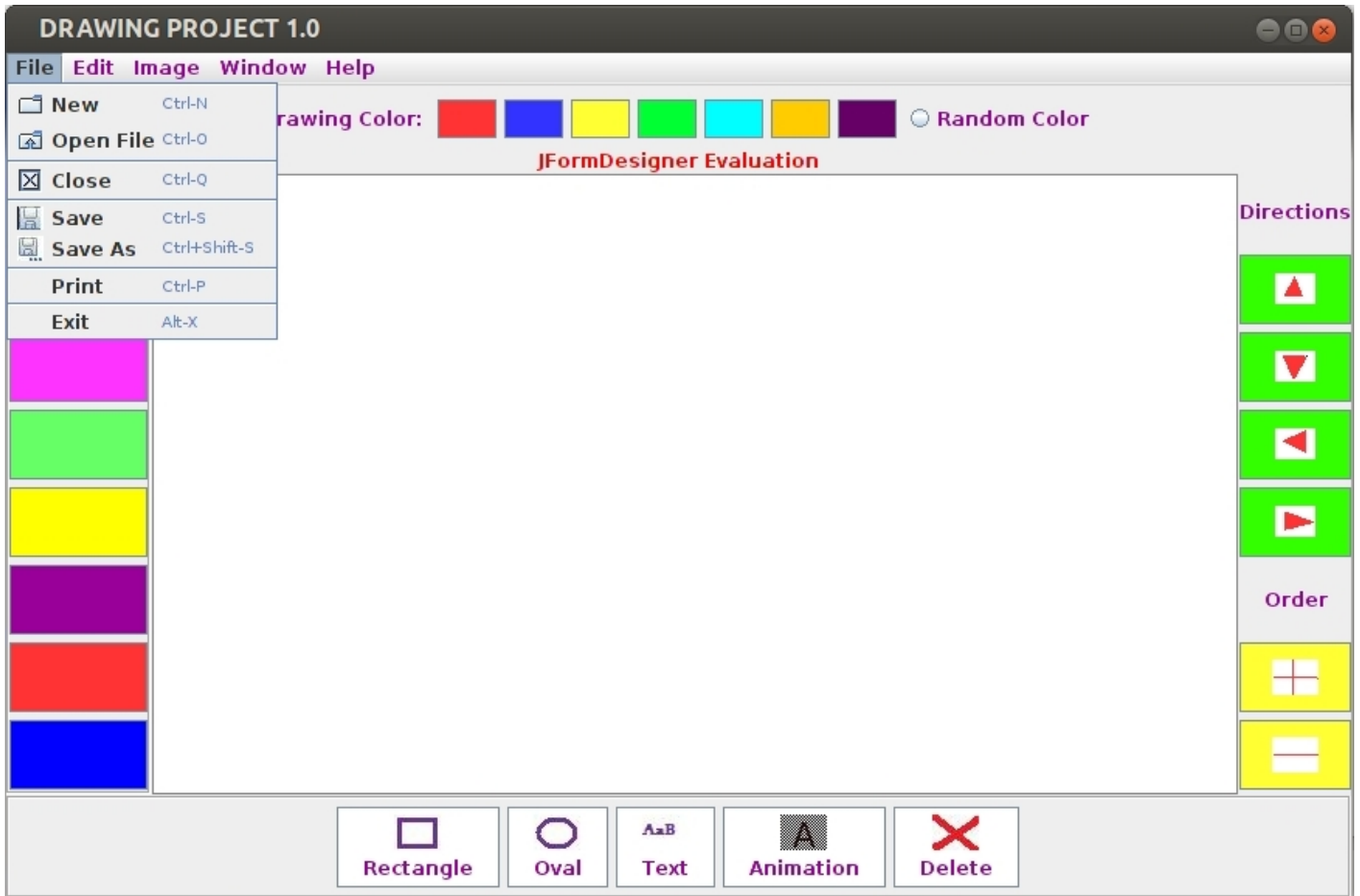
```
private void btAnimationActionPerformed( ActionEvent e)
{
    if(DrawingPanel.isAnimActive==false){
        drawingPanel.startAnimate();}
    else
        drawingPanel.stopAnimate();
}
```

Buradaki stopAnimate metodu da aşağıdaki gibi oluşturuldu.

```
public void stopAnimate()
{
    isAnimActive = false;
}
```

Bu değişikliklerden sonra animasyon işlevi eklenmiş oldu.

KISIM		YAPRAK NO : 21
YAPILAN İŞ	Save ve Save as Butonlarının Aktifleştirilmesi	TARİH : 21/07/2014
<p>Arayüzdeki save ve save as butonlarının aktif edilmesi istendi. Bunun üzerine <i>Java'da dosya işlemleri</i> araştırılıp, çalışıldıktan sonra MainFrame sınıfına aşağıdaki gibi kodlandı.</p> <pre> private void miSaveActionPerformed(ActionEvent e) { BufferedImage image2 = new BufferedImage(drawingPanel.getWidth(), drawingPanel.getHeight(), BufferedImage.TYPE_INT_RGB); Graphics2D graphics2D = image2.createGraphics(); drawingPanel.paint(graphics2D); JFileChooser jFile = new JFileChooser(); jFile.showSaveDialog(null); String pth = jFile.getSelectedFile().getPath(); JOptionPane.showMessageDialog(null, pth.toString()); try { ImageIO.write(image2, "png", new File(pth.toString() + ".png")); } catch (IOException ox) { ox.printStackTrace(); } } private void miSaveAsActionPerformed(ActionEvent e) { if (drawingPanel.isVisible() != true) return; JFileChooser fc = new JFileChooser(); FileFilter filter = new FileNameExtensionFilter("DrawingProjects(*.dp)", "dp"); fc.setAcceptAllFileFilterUsed(false); fc.addChoosableFileFilter(filter); fc.setFileFilter(filter); fc.getCurrentDirectory(); fc.getFileFilter(); int res = fc.showDialog(MainFrame.this, "Save As"); if (res == JFileChooser.APPROVE_OPTION) { try { ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(fc.getSelectedFile())); oos.writeObject(drawingPanel.getShapes()); oos.flush(); oos.close(); } catch (IOException e1) { e1.printStackTrace(); } } } </pre>		



KISIM		YAPRAK NO : 22
YAPILAN İŞ	<i>Text Butonunun Aktifleştirilmesi</i>	TARİH : 22/07/2014
<p>Öncelikle Rectangle sınıfından miras alan aşağıdaki Text sınıfı oluşturuldu.</p> <pre> public class Text extends Rectangle { public Text(int x, int y, int w, int h) { super(x, y, w, h); } } </pre> <p>Shape sınıfı içerisinde string türünde text değişkeni ile bu değişkenin set ve get metodları oluşturuldu.</p> <pre> public String text = " "; public String getText() { return text; } public void setText(char charText) { this.text += charText; } </pre> <p>Ardından girilen harfleri ekrana çizecek olan draw() metodu oluşturuldu. Varsayılan renk siyah seçildi. Girilen karakterleri büyük harfe çevirmek için toUpperCase() metodu kullanıldı.</p> <pre> public void draw(Graphics g) { g.drawString(getText().toUpperCase(), x, y); g.setColor(Color.<i>black</i>); } </pre> <p>DrawingPanel sınıfının mouseClicked() olayında texts nesnesi oluşturulup shapes vektörüne eklendi (çizim modu text olması durumunda)</p> <pre> public void mouseClicked(MouseEvent me) { if (drawingMode == <i>DM_TEXT</i>) { setDrawingMode(<i>DM_TEXT</i>); Text texts = new Text(pointMouseDown.x, pointMouseDown.y, 0, 0); currentDrawingShape = texts; shapes.add(texts); setSelectedShape(texts); repaint(); } repaint(); } </pre> <p>Klavyeden girilen karakterleri text değişkenine atayacak kod parçası DrawingPanel sınıfı içerisindeki keyTyped() olayı içerisinde oluşturuldu.</p>		

```

public void keyTyped( KeyEvent ke)
{
    if ( drawingMode == DM_TEXT ) {
        currentDrawingShape.setText( ke.getKeyChar() );
    }
    repaint();
}

```

Yazı girerken *fare işaretçisinin metin girme moduna geçmesi* için *setDrawingMode()* metodu aşağıdaki gibi güncellendi.

```

public void setDrawingMode( int drawingMode)
{
    this.drawingMode = drawingMode;
    if ( drawingMode == DM_TEXT ) {
        setCursor( Cursor.getPredefinedCursor( Cursor.TEXT_CURSOR ) );
    } else {
        setCursor( Cursor.getDefaultCursor() );
    }
}

```

Böylece *Drawing Project* aşağıdaki gibi ekrana *yazı yazabilir hale* geldi.

