**Painting**

**jFrame.add(new** MyPaint)

**static class** MyComponent **extends JComponent{** - add variable from this class to the **jFrame**  **@Override  
 protected void paintComponent(Graphics** g**){ …** - creating class with graphycs

**Graphics2D** g2 **= (Graphics2D)**g**;** - reconstructing graphics variable into graphics2D **Font** font **= new Font("…", Font.***BOLD***,** size**);** - creating personal font g2**.setFont(font);** - setting font

**Color** color **= new Color(r, g, b, opacity)** – creating color

g2**.drawString("…",** x,y**);** - painting strings **Ellipse2D** e2 **= new Ellipse2D.Double(**x**,**y**,**width**,**height**);** - painting circle  **Rectangle2D** r2 **= new Rectangle2D.Double(**x**,**y**,**width**,**height**);** - painting square

**Line2D** l2 **= new Line2D.Double(**x**,**y**,**x2**,**y2**);** - painting line

**QuadCurve2D** quadCurve2D **= new QuadCurve2D.Double(x1,y1,xc,yc, x2,y2);** - curve(кривая), 3 and 4 parameters are about where center of the line will be

**CubicCurve2D** quadCurve2D **= new CubicCurve2D.Double( x1, y1, cx1,cy1,cx2,cy2,x2,y2)** – curve(кривая) with 2 central points

**Arc2D** arc2D **= new Arc2D.Double(**x, y, width, height, start deg, end deg,Arc2D….**);** - circle but not all, you write its width and height and start\end degrees

Create break solid line

**GeneralPath** generalPath **= new GeneralPath();**generalPath**.moveTo(**x, y**);** - start pointgeneralPath**.lineTo(**x, y**);** - line to …generalPath**.quadTo(**x1, y1, x2, y2**);** - curve line and end point

generalPath**.curveTo(**x1, y1, x2, y2, x3, y3**); -** curve line with 2 points in the middle

create and manage area (concat, split atc) like in **дискретка**

**Area** area **= new Area();** - create areaarea**.add(new Area(new Ellipse2D.Double(**…**)));** - add area to this areaarea**.intersect(new Area(new Ellipse2D.Double(**…**)));** - take only common from both, which first and second have at once area**.subtract(new Area(new Ellipse2D.Double(**…**)));** - take first without second

Styles

**g2.setStroke(new BasicStroke(**width**f, BasicStroke. …, BasicStroke. …,** mitterLimit**f, new float[]{**width between line segments**},** dash space**));** - create custom line (required 1 or 1,2,3 parameters, else just for fun)

**g2.setPaint(new GradientPaint(new Point(x, y), Color…., new Point(x, y), Color….));** - set gradient paint

operations

g2**.scale(x,y);** - make figure biggerg2**.translate(…, …);** - move paintg2**.shear(…, …);** - transform paint (like in Photoshop)g2**.rotate(…);** - rotate paint (in radians)

g2**.clip(new …);** - cut figure (insert figure, which will be limit to the function)

choose faster of more beautiful

**RenderingHints** renderingHints **= new RenderingHints(null);** - create valuerenderingHints**.put(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_OFF);** - set to be faster but less beautiful

renderingHints**.put(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);** - set to be slover but more beautifulg2**.setRenderingHints(**renderingHints**);** - set object

g2**.draw(**l2**);** - line of figureg2**.setPaint(Color. …);** - set colorg2**.fill(**r2**);** - filled figure

**IMAGES**

Dangerous way

**File** file **= new File("1.jpg");** - getting directory **BufferedImage** image **= ImageIO.read(file);** - getting image ( can have errors)

formal way

**File** file **= new File("1.jpg");  
ImageReader** reader **= null;** - set reader null **Iterator<ImageReader>** iterator **= ImageIO.getImageReadersByFormatName("**…**"); - create iterator  
if(**iterator**.hasNext()){** reader **= iterator.next();** - set reader **}  
  
InputStream** inputStream **= new FileInputStream(new File("…"));  
ImageInputStream** imageInputStream **= ImageIO.createImageInputStream(inputStream);** - input for imagesreader**.setInput(imageInputStream, true);** - setting input **BufferedImage** image **= reader.read(reader.getNumImages(true));** - getting image **int** count **= reader.getNumThumbnails(0);** - number of thumbnails **BufferedImage** image2 **= reader.readThumbnail(0,count);** - get thumbnails

Change pixels

**BufferedImage** img **= ImageIO.read(new File("**where**"));** - getting file **WritableRaster** raster **=** img**.getRaster();** - getting raster (needed variable)

**Int[] …** = raster**.getPixel(**i**,** j**, new int[**4**])** – returns int array with 4 variables of RGBA format GETTING PIXELS

raster**.setPixel(**i**,** j**,** array**)** – setting pixels

**for (int i = 0; i <** raster**.getWidth(); i++) {** - cross the whole width **int[]** pixel **=** raster**.getPixel(**x**,** y**, new int[4]);** - get pixels by coordinates and save them in variablepixel**[0] = …;** - set red from rgb formatpixel**[1] = …;** - set green from rgb formatpixel**[2] = …;** - set blue from rgb format **raster.setPixel(i, i,** pixel**);** - set new pixel **}**img**.setData(**raster**);** - setting this raster to the img **ImageIO.write(**img**, "**format**", new File("**where and with name…**"));** - saving img

Transformation and filters

Classes with methods for rendering:

**AffineTransform**

**RescaleOp**

**LookupOp**

**ColorConvertOp**

**ConvolveOp**

Example:

**BufferedImage** bufferedImage **= new BufferedImage(**img**.getWidth(),**img**.getHeight(),**img**.getType());  
AffineTransform** affineTransform **= new AffineTransform();  
BufferedImageOp** op **= new AffineTransform(affineTransform, AffineTransform. ...);**op**.filter(**image**,** bufferedImage**);**

**ImageIO.write(**image**, "png", new File("2.png"));**