**UIManager.setLookAndFeel("…");** - change style of the frame

**Creating frame**

**JFrame** jFrame **= new JFrame();** - creating frame variablejFrame**.setVisible(true);** - set it to be bisiblejFrame**.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);** - function on closingjFrame**.setSize(…,…);** - set sizejFrame**.setLocation(…,…);** - set locationjFrame**.setTitle("…");** - set titlejFrame**.setBounds(**x**,**y**,**width**,**height**);** - set location and then size

**Toolkit** toolkit **= Toolkit.*getDefaultToolkit*();** - get variable with parameters  **Dimension** dimension **= toolkit.getScreenSize();** - getting screen formatdimension**.height** - heightdimension**.width** – width

**Point2D** p **= new Point2D.Double(…, …);** - create coordinate point… **.setFrameFromDiagonal(**p**,**p**);** - set coordinate point (from the top)

Variable from this class has method **repaint**, which repaints picture

**Buttons and actions**

**JPanel** jPanel **= new JPanel();** - create panel with instrumentsjFrame**.add(jPanel);** - add panel to the frame

jPanel**.setLayout(new GridLayout(…, …));** - create grid box with … rows and .. columns

**JButton** jButton **= new JButton(new** MyAction(), **BorderLayout…**.**);** - creating button and add class, which will take care of actionsjPanel**.add(**jButton**);** - add button

jButton**.setToolTipText(“…”)** - setting tips

jButton**.setText("…");**

**... .requestFocusInWindow()** – set default focus in that menu

**… .setFocusable(false)** – set unable to focus in

jButton**.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) { …** - overriding method, which happens after pushing the button

jButton**.addActionListener(EventHandler.create(ActionListener.class,** where**, "**what**", "**what to set**"));** - short type to make one function, without creating class

**putValue(AbstractAction. …, "**…**");** - setting value ( in action of button )

**static class** MyAction **extends AbstractAction{  
@Override  
public void actionPerformed(ActionEvent e){ … }** – to set actions

**KeyStroke** keyStroke **= KeyStroke.getKeyStroke("…");** - get key or combination of keys to create event with it **InputMap** inputMap **=** jPanel**.getInputMap(JComponent.**WHEN\_IN\_FOCUSED\_WINDOW**);** - create map massive, where you can save buttons with their ids inputMap**.put(keyStroke, "a");** - add key with id **ActionMap** actionMap **= jPanel.getActionMap();** - connect key with action in this variableactionMap**.put("a",** action**);** - connecting (method has to be from class, which extends AbstractAction)

jFrame**.addMouseMotionListener(new MouseAdapter() {** - set mouse events by overriding methods

**JCheckBox** jCheckBox **= new JCheckBox("…"));** - create checkbox

**ButtonGroup** buttonGroup **= new ButtonGroup();** - create a group for radiobuttons  **JRadioButton** jRadioButton **= new JRadioButton("…");** - create rediobuttonbuttonGroup**.add(**jRadioButton**);** - add radiobutton to the groop

**JComboBox<…>** comboBox **= new JComboBox<>();** - create list with opportunity of values (selection) **comboBox.addItem("one");** - add item to the combox

**Border** border **= BorderFactory.createEtchedBorder();** - create border **Border** border1 **= BorderFactory.createTitledBorder("…");** - create border with titlejPanel**.setBorder(**border**);** - set border

**JSlider jSlider = new JSlider();** - creating slider (line with values) **jSlider.setMinorTickSpacing(…);** - set min **jSlider.setMajorTickSpacing(…);** - set max **jSlider.setPaintTicks(true);** - paint parts of the line by making line under them **jSlider.setSnapToTicks(true);** - cursor automatically get to the closest value  **jSlider.setPaintLabels(true);**  - paint numbers

**Events with frame**

jFrame**.addWindowListener(new WindowListener() { … }** – add method, where you override ALL events connected with window for example when it closing

jFrame**.addWindowListener(new WindowAdapter() { … }** – add method, where you override ONE event connected with window for example when it closing

**Filds**

jPanel**.add(…);** - to add something in the bottom

jPanel**.revalidate();** - start filds

**JLabel** jLabel **= new JLabel("…");** - create label **JTextField** txt **= new JTextField("**common text**",** number of chars**);** - create text fildJTextArea **jTextArea = new JTextArea(…, …);** - text areajTextArea**.setLineWrap(true);** - enable to push lines

**JScrollPane** jScrollPane **= new JScrollPane(**jTextArea**);** - scrollable text area

jTextField**.setDragEnabled(true);** - thanks to it you can swap texts between the lines using mouse (hold and move)

**Menu bar**

**JMenuBar** jMenuBar **= new JMenuBar();** - creating menu bar

**JMenu** … **= new JMenu("…");** - creating menu part in the bar (don’t have any functions, just wrapper)

**JMenuItem** jMenuItem **= new JMenuItem("…", ‘…’);** - creating element of menu (it has to be child of JMenu variable) second value is symbol, which is going to be underlined

jMenuBar**.add(…);** - adding **JMenuItem**  to the **JMenu**

**… .addSeparator();** - separate by line

jFrame**.setJMenuBar(jMenuBar);** - adding menu barjFrame**.revalidate();** - don’t know for what it is, just add in the end

**jMenuItem.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 System.exit(0);  
 }  
});** - create function on this button **jMenuItem.setAccelerator(KeyStroke.getKeyStroke("E"));** - bind button on this function

**Toolbar**

**JToolBar** jToolBar **= new JToolBar();** - creating toolbar

**new JButton("…");** - this variable for toolbars, in this kind variable you can add tip **setToolTipText(“…”)**

jToolBar**.add( … )** – add menus

jPanel**.add(**jToolBar**);** - add tollbar

**Context menu**

**JPopupMenu … = new JPopupMenu();** - creating context menu **… .add(new JMenuItem("…"));** - add menus jPanel**.setComponentPopupMenu(…);** - setting context menu

**Grid positioning**

jPanel**.setLayout(new GridBagLayout());** - set panel to have grid background  **GridBagConstraints** constraints1 **= new GridBagConstraints();**constraints1**.weightx = …;** - width of the elementconstraints1**.weighty = …;** - height of the elementconstraints1**.gridx = …;** - position X of the element (like in exel)constraints1**.gridy = …;** - position Y of the element (like in exel)constraints1**.gridheight = …;** - fills blocks (like make one block from two)constraints1**.gridwidth = …;** - fills blocks (like make one block from two)jPanel**.add(new …),** constraints1**);** - add element with those parametrs

**Modal window**

**JOptionPane.showMessageDialog(**jPanel**, "**message**","**title**",JOptionPane.**INFORMATION\_MESSAGE**);** - common message with “ok” button **JOptionPane.showConfirmDialog(**jPanel**, "**message**", "**title**", JOptionPane.**NO\_OPTION**);** - confirm message with options **JOptionPane.showInputDialog(**jPanel**, "**message**");** - message with input line

**JOptionPane.showOptionDialog(**jPanel**, "**message**", "**title**",JOptionPane.**CANCEL\_OPTION**,JOptionPane.**INFORMATION\_MESSAGE**, null, new Object[]{**…**},** default item **);** - modal window with your own properties

**Color** color **= JColorChooser.showDialog(**jPanel**, "**title**", Color.** …**);** - get modal with choosing colors, third parameter if default color

**Create custom dialog window**

**static class** CunstoDialog **extends JDialog{** - extend this class **public** CunstoDialog**(){** - overrode this method **super(**jFrame**,"**title**",**true**);** - set default parameters  **add(new …, BorderLayout.**NORTH**);** - add what you want **setBounds(…, …, …,…);**  - locate  **setVisible(true);** - set visible **}  
}**

**file chooser**

**JFileChooser** jFileChooser **= new JFileChooser();**jFileChooser**.setCurrentDirectory(new File("…"));** - set default directoryjFileChooser**.setSelectedFile(new File(""));** - set default filejFileChooser**.setMultiSelectionEnabled(true);** - enable multi selecting jFileChooser**.setFileSelectionMode(JFileChooser.**FILES\_AND\_DIRECTORIES**);** - filter

jFileChooser**.setFileFilter(new javax.swing.filechooser.FileFilter() {  
 @Override  
 public boolean accept(File f) {**  if(f.getName().endsWith("gif")){  
 return true;  
 }else {  
 return false;  
 } **}  
 @Override  
 public String getDescription() {  
 return "…";  
 }  
 });** - custom filterjFileChooser**.showDialog(jPanel, "…");** - add to the dialog with acceptable button “…”

**File** file **= jFileChooser.getSelectedFile()** – getting selected file

**jFrame.revalidate()** – very important!!!!!!!!!

**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

**Robot**

to use program as you want (move mouse for example)

GraphicsEnvironment graphicsEnvironment = GraphicsEnvironment.getLocalGraphicsEnvironment(); - getting graphics  
GraphicsDevice device = graphicsEnvironment.getDefaultScreenDevice(); - getting screen  
try{  
 Robot robot = new Robot(device); - getting robot  
 robot.mouseMove(x, y); - move mouse  
  
 robot.mousePress(InputEvent….); - press mouse button  
 robot.mouseRelease(InputEvent….); - free mouse button  
  
 Rectangle rectangle = new Rectangle(0,0,jFrame.getWidth(), jFrame.getHeight()); - getting rectangle  
 BufferedImage image = robot.createScreenCapture(rectangle); - getting screenshot by with parametrs  
}catch (AWTException ee){}