SPRAWOZDANIE

ZEGAR

Do stworzenie wigetu zegara trzeba było utworzyć projekt „Aplication window".

Po utworzeniu projektu w ten sposób wygenerowało nam się kilka klas takich jak:

ClockPanel, start, run oraz paintComponent, które są odpowiedzialne za:

- Klasa ClockPanel posiada konstruktor, który przyjmuje dwie wartości szerokości i wysokości, ustawiając odpowiednio rozmiar panelu. Właściwość opaque jest ustawiana na false. Wartość początkowa dla dateTime jest ustawiana na aktualny czas.
- Metoda start() ustawia wartość star" na true i tworzy nowy wątek, który jest uruchamiany.
- Metoda run() jest wywoływana w nowym wątku i w pętli sprawdza aktualny czas. Jeśli sekunda się zmieni, to metoda repaint() jest wywoływana. Wątek jest uśpiony na 200 milisekund.
- Metoda paintComponent(Graphics g) jest nadpisana i jest używana do rysowania zegara na panelu. Właściwość RenderingHint jest ustawiana, aby uzyskać antyaliasing. Metoda PaintClock(g2, dateTime) jest wywoływana, aby narysować zegar.

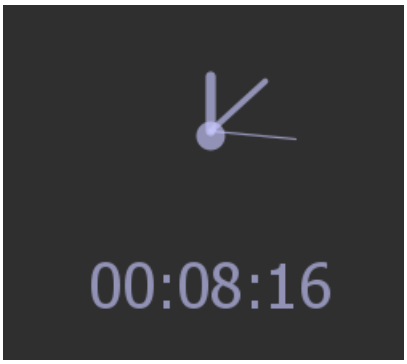
Główna funkcja, która jest odpowiedzialna za obliczenia:

```
public void PaintClock(Graphics2D g2, LocalDateTime dateTime) {
        float lineWidth = 3.0f;
        Stroke line = new BasicStroke(lineWidth, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND);
        Font font = new Font("Tahoma", Font.PLAIN, getHeight()/10);
        g2.setColor(new Color(200, 200, 255));
        g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC, 0.65f));
        g2.setStroke(line);
        g2.setFont(font);

        Point2D origin = new Point2D.Double(getWidth() / 2.0, getHeight() / 10);

        String time = dtFormatter.format(dateTime);
        Rectangle2D timeBounds = font.getStringBounds(time, g2.getFontRenderContext());
        g2.drawString(time, Math.round(origin.getX() - timeBounds.getWidth() / 2.0),
Math.round(origin.getY() + getHeight() / 3.0));

        double radius = getWidth() / 2.0  - 2.0 * lineWidth;

        //TO DO START
        double hour = 2*Math.PI*(dateTime.getHour())/12-0.5*Math.PI;
        double minute = 2*Math.PI*(dateTime.getMinute())/60-0.5*Math.PI;
        double second = 2*Math.PI*(dateTime.getSecond())/60-0.5*Math.PI;
        //godzina
        int godzina1=(int)Math.round(20*Math.cos(2*Math.PI*hour/12-0.5*Math.PI)); //20 to długość
        int godzina2=(int)Math.round(20*Math.sin(2*Math.PI*hour/12-0.5*Math.PI));
        line = new BasicStroke(4, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND); //4 to grubość
        g2.setStroke(line);
        g2.drawLine((int)origin.getX(),  (int)origin.getY()+12, godzina1+(int)origin.getX(),
godzina2+(int)origin.getY()+12);

        //minuty
        int minuta1=(int)Math.round(35*Math.cos(2*Math.PI*minute/60-0.5*Math.PI));
        int minuta2=(int)Math.round(35*Math.sin(2*Math.PI*minute/60-0.5*Math.PI));
        line = new BasicStroke(4, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND);
        g2.setStroke(line);
```

```
        g2.drawLine((int)origin.getX(),  (int)origin.getY()+12, minuta1+(int)origin.getX(),
minuta2+(int)origin.getY()+12);

        //sekundy
        int sekunda1=(int)Math.round(40*Math.cos(2*Math.PI*second/60-0.5*Math.PI));
        int sekunda2=(int)Math.round(40*Math.sin(2*Math.PI*second/60-0.5*Math.PI));
        line = new BasicStroke(2, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND);
        g2.setStroke(line);
        g2.drawLine((int)origin.getX(),(int)origin.getY()+12, sekunda1+(int)origin.getX(),
sekunda2+(int)origin.getY()+12);

        //TODO end

        //rysowanie kółka
        double innerRadius = radius / 10;
        g2.fill(new Ellipse2D.Double(origin.getX() - innerRadius/2.0, origin.getY() +
innerRadius/2.0, innerRadius, innerRadius));
          }
```

Metoda PaintClock jest odpowiedzialna za rysowanie zegara na podstawie czasu podanego jako argument. W metodzie ustawiany jest styl linii (grubość, styl zakończenia, styl łączenia), styl czcionki, kolor i przezroczystość. Następnie wyznaczana jest pozycja środka zegara, a aktualny czas jest wyświetlany na ekranie. Potem obliczane są kąty dla wskaźników godziny, minut i sekund, a następnie rysowane są wskaźniki na odpowiedniej pozycji. Na końcu rysowany jest mały koło wewnątrz zegara.

PREZENTACJA ZEGARA:



Na białym i czarnym tle.

KLASA MAIN:

```
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                try{
                    GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
                    GraphicsDevice gd = ge.getDefaultScreenDevice();
                    boolean isPerPixelTranslucencySupported =
gd.isWindowTranslucencySupported(GraphicsDevice.WindowTranslucency.PERPIXEL_TRANSLUCENT);
                    if(isPerPixelTranslucencySupported){
                        Main window = new Main();
                        window.frame.pack();
                        window.frame.setVisible(true);
```

```
                }
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    });
}
```

W tej funkcji interfejs Runnable jest uruchamiany za pomocą metody EventQueue.invokeLater. Następnie, w ramach tego interfejsu, pobierane jest lokalne środowisko graficzne GraphicsEnvironment ge i domyślne urządzenie ekranu GraphicsDevice gd. Następnie sprawdzane jest, czy urządzenie ekranu obsługuje translację na poziomie piksela boolean. Jeśli tak, tworzony jest obiekt klasy Main, który jest pakowany i wyświetlany. W przypadku wystąpienia wyjątku, jego szczegóły są wyświetlane.

FUNKCJA INITIALIZE:

```
public void initialize(){
        frame = new JFrame();
        frame.setType(Window.Type.UTILITY);
        frame.setUndecorated(true);
        frame.setBackground(new Color(0, 0, 0, 0));
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocation(0, 350); //położenie na ekranie
        frame.setAlwaysOnTop(true);
        ClockPanel clockPanel = new ClockPanel(300, 300);  //wielkość
        clockPanel.addMouseListener(new MouseAdapter() {
          @Override
            public void mouseReleased(MouseEvent me) {
                if(me.isPopupTrigger())System.exit(0);
                frame.setLocation(frame.getLocation().x + me.getX()-125, frame.getLocation().y +
me.getY()-35);

            }
        });
        frame.setContentPane(clockPanel);
        clockPanel.start();
    }
```

Metoda initialize tworzy i konfiguruje obiekt okna JFrame. Ustawia typ okna na UTILITY i wyłącza jego dekorację. Następnie ustawia kolor tła na przezroczysty. Operacją domyślną przy zamknięciu okna jest zakończenie działania programu. Położenie okna jest ustawione na współrzędne (0,350). Okno jest zawsze na wierzchu innych okien. Zawartość okna to panel zegara ClockPanel o wielkości 300x300. Panel zegara oddziaływa na zdarzenia myszy, które pozwala na przenoszenie okna w miejsce, w którym zostało kliknięte i zakończenie programu po kliknięciu prawym przyciskiem myszy. W końcowej części metoda uruchamia działanie panelu zegara.

WNIOSKI:

- W tym laboratorium jest tworzona aplikacja okna zegara, które jest transparentne i zawsze na wierzchu innych okien.
- Użyłam kilku klas i interfejsów Java, takich jak Runnable, JFrame, GraphicsDevice, itp. aby stworzyć i wyświetlić zegar w postaci okna na pulpicie.

- Tworzenie aplikacji jak tego zegara jest procesem wymagającym dobrej znajomości języka programowania i narzędzi takich jak Java, oraz zrozumienia wymagań i specyfikacji projektu.