Grafični vmesniki v programskem jeziku Java

Splošnonamenski programski jezik **Java** je bil razvit kot **varen jezik za poljubno napravo**. Sintaksa zahteva daljše programe kot v programskem jeziku Python, ki pa so navadno veliko hitrejši. Jezik se prevaja, kar pomeni, da program `Demo.java` v ukazni vrstici najprej prevedemo kot `javac Demo.java`, kar ustvari vmesno datoteko `Demo.class`, katero nato izvedemo kot `java Demo`.

Programming is not science, it is a skill! If you want to run as fast as Usain Bolt, you have to do a lot of running. There is no other way! And it is the same with programming. Just try to run a lot =)

V programskem jeziku **Java grafični uporabniški vmesnik** sestavimo z uporabo ogrodja Abstract Window Toolkit, ki v paketu `java.awt` definira potrebne razrede za standardne grafične elemente. Razširitev slednjih predstavljajo razredi ogrodja Swing v paketu `javax.swing`, ki prevzamejo tudi izgled operacijskega sistema.

```java
public class Visuals {

    public static void main(String[] args) {
        // ...
    }

}
```

Programska okna, paneli in postavitve

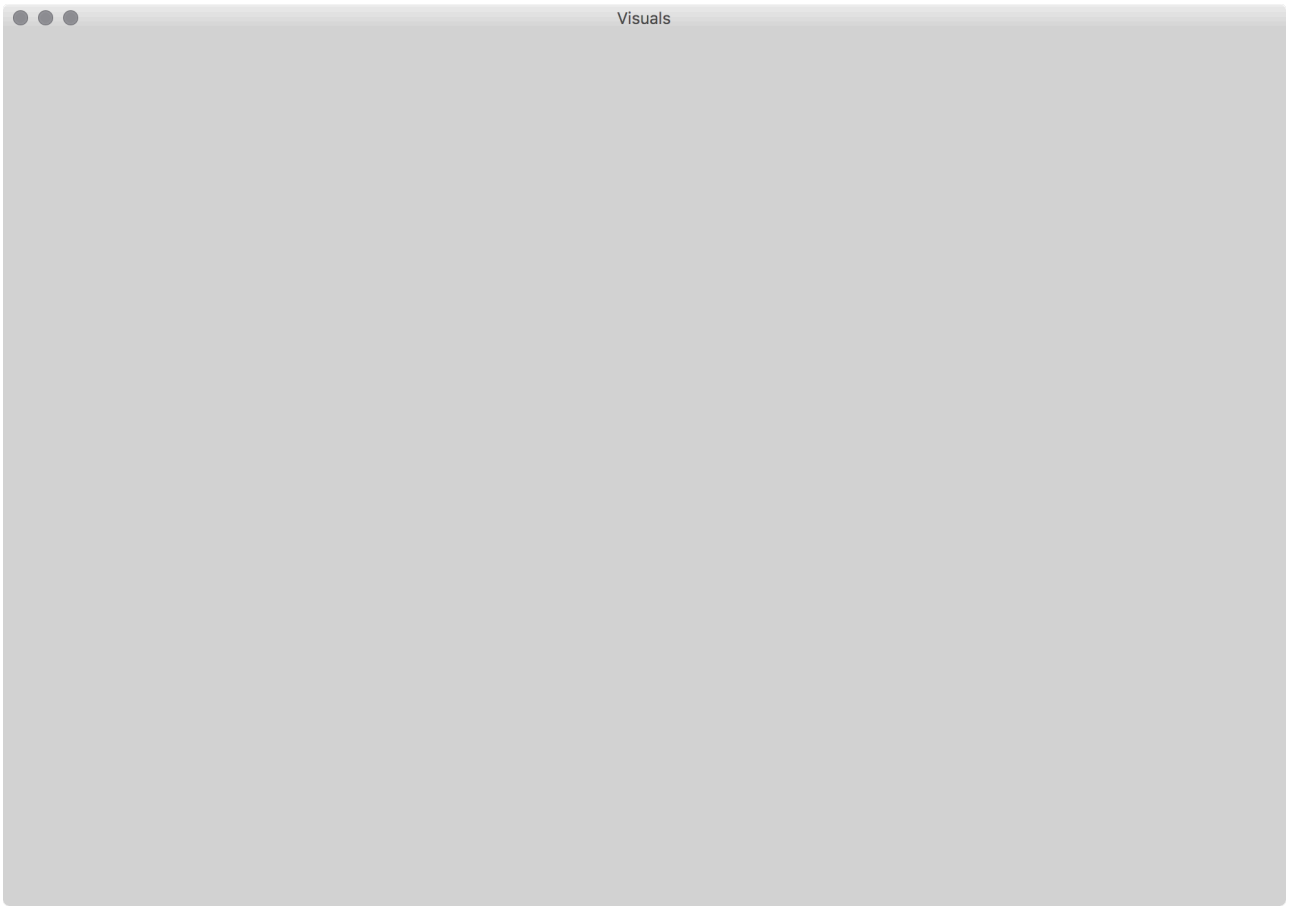
Programsko okno predstavlja **ogrodje grafičnega vmesnika** programa, ki prevzame izgled oken operacijskega sistema. V programskem jeziku **Java okno** ustvarimo kot objekt razreda `JFrame` v paketu `javax.swing`, ki mu lahko določimo naslov, izgled, velikost, obnašanje itd. Okno prikažemo z uporabo metode `setVisible(boolean visible)`, dočim lahko zahtevamo ponoven izris celotnega okna *kakor hitro se da* z uporabo metode `repaint()`.

```java
JFrame frame = new JFrame("Visuals");
frame.setSize(new Dimension(1024, 768));
frame.setMinimumSize(new Dimension(800, 600));
frame.setResizable(true);
// ...
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);

while (true) {
    // ...
    frame.repaint(); // ponoven izris okna
    try {
        Thread.sleep(50); // počakaj 50 ms
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

V praksi običajno definiramo nov podrazred razreda `JFrame`, pri čimer prvi del zgornjega programa vključimo v konstruktor (pod)razreda.

Primer izgleda grafičnega vmesnika je prikazan spodaj.

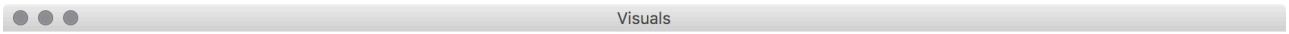


V programskem jeziku **Java** lahko z uporabo metode `add(JPanel panel)` oknu dodamo panele, ki so

objekti razreda `JPanel` v paketu `javax.swing`. Po **panelih** lahko **rišemo, dodajamo grafične elemente** (npr. gumbi, vnosna polja, drsniki, izbirniki) itd. Panel zopet prevzame izgled operacijskega sistema, dočim pa mu lahko tudi sami določimo izgled, velikost, ozadje, obnašanje itd. V praksi običajno definiramo nov podrazred razreda `JPanel`.

```java
JPanel panel = new JPanel();
panel.setBackground(Color.WHITE);
frame.add(panel);
```

Primer izgleda grafičnega vmesnika je prikazan spodaj.

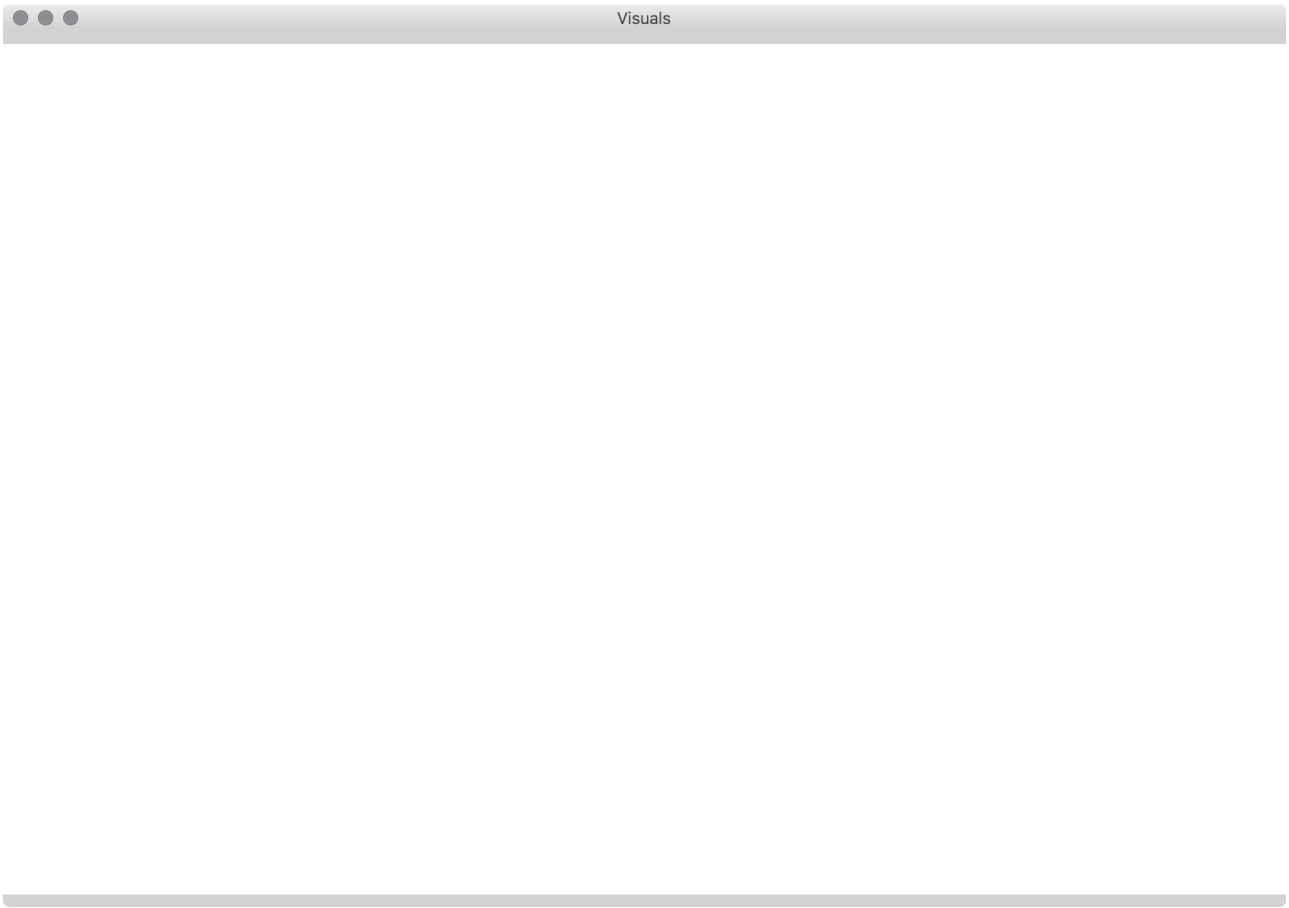


Tako kot zgoraj lahko **vsem grafičnim elementom**, ki so objekti razreda `Containter` v paketu `java.awt` ali njegovih podrazredov (npr. `frame`), z uporabo metode `add(Component component)` **dodamo** druge **grafične elemente**, ki so objekti razreda `Component` ali njegovih podrazredov (npr. `panel`). Pred tem lahko z uporabo metode `setLayout(LayoutManager layout)` **določimo postavitev elementov** kot objekt razreda, ki implementira vmesnik `LayoutManager`.

Programska okna `JFrame` privzeto uporabljajo postavitev `BorderLayout`, ki okno razdeli na osrednji del (tj. `BorderLayout.CENTER`) in štiri stranske dele (npr. `BorderLayout.WEST`). Pri tem je velikost slednjih najmanjša možna (tj. takšna kot zahtevajo grafični elementi), dočim se celoten preostali prostor okna dodeli osrednjemu delu grafičnega elementa.

```java
frame.setLayout(new BorderLayout());
frame.add(panel, BorderLayout.CENTER);
JPanel north = new JPanel();
frame.add(north, BorderLayout.NORTH);
JPanel south = new JPanel();
frame.add(south, BorderLayout.SOUTH);
```

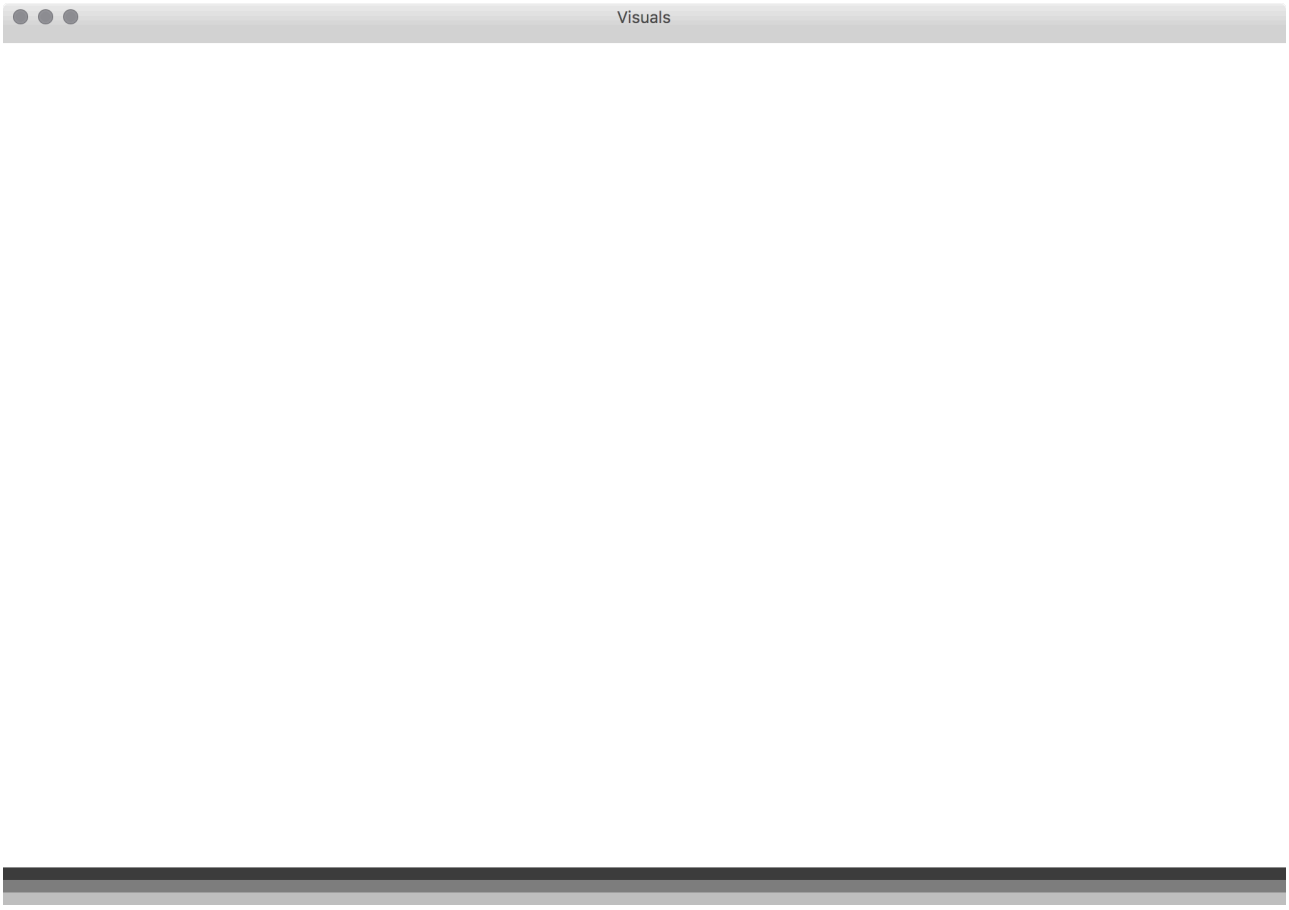
Primer izgleda grafičnega vmesnika je prikazan spodaj.



Postavitev `GridLayout` grafični element razdeli na regularno mrežo z izbranim številom vrstic in stolpcev. Pri tem je ves prostor enakomerno razdeljen med celicami mreže.

```java
south.setLayout(new GridLayout(3, 1));
for (int i = 0; i < 3; i++) {
    float rgb = (i + 1) / 4.0f;
    Color color = new Color(rgb, rgb, rgb);
    JPanel subsouth = new JPanel();
    subsouth.setBackground(color);
    south.add(subsouth);
}
```

Primer izgleda grafičnega vmesnika je prikazan spodaj.



Postavitev `GridBagLayout` omogoča tudi, da se grafični elementi raztezajo preko več celic mreže, dočim lahko poljubno določimo razporeditev prostora po celicah.

Risanje likov in grafičnih oblik

Po panelih razreda `JPanel` lahko **rišemo tako, da redefiniramo metodo** `paint(Graphics g)`, ki se izvede vsakič, ko je potrebno panel ponovno izrisati (npr. sprememba velikosti, klic metode `repaint()`). Pri tem je parameter metode objekt razreda `Graphics` ali `Graphics2D` v paketu `java.awt`, nad katerim lahko uporabimo različne **metode za risanje** geometrijskih **likov in** grafičnih **oblik**.

Primer razreda `Panel`, ki naj predstavlja osrednji panel programskega okna, je prikazan spodaj.

```java
class Panel extends JPanel {

    public Panel() {
        super();
        setBackground(Color.WHITE);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g); // klic metode nadrazreda
        Graphics2D graphics = (Graphics2D)g; // pretvarjanje tipov
        int indent = 32, size = 96; // pomožne spremenljivke
        // ...
    }

}
```

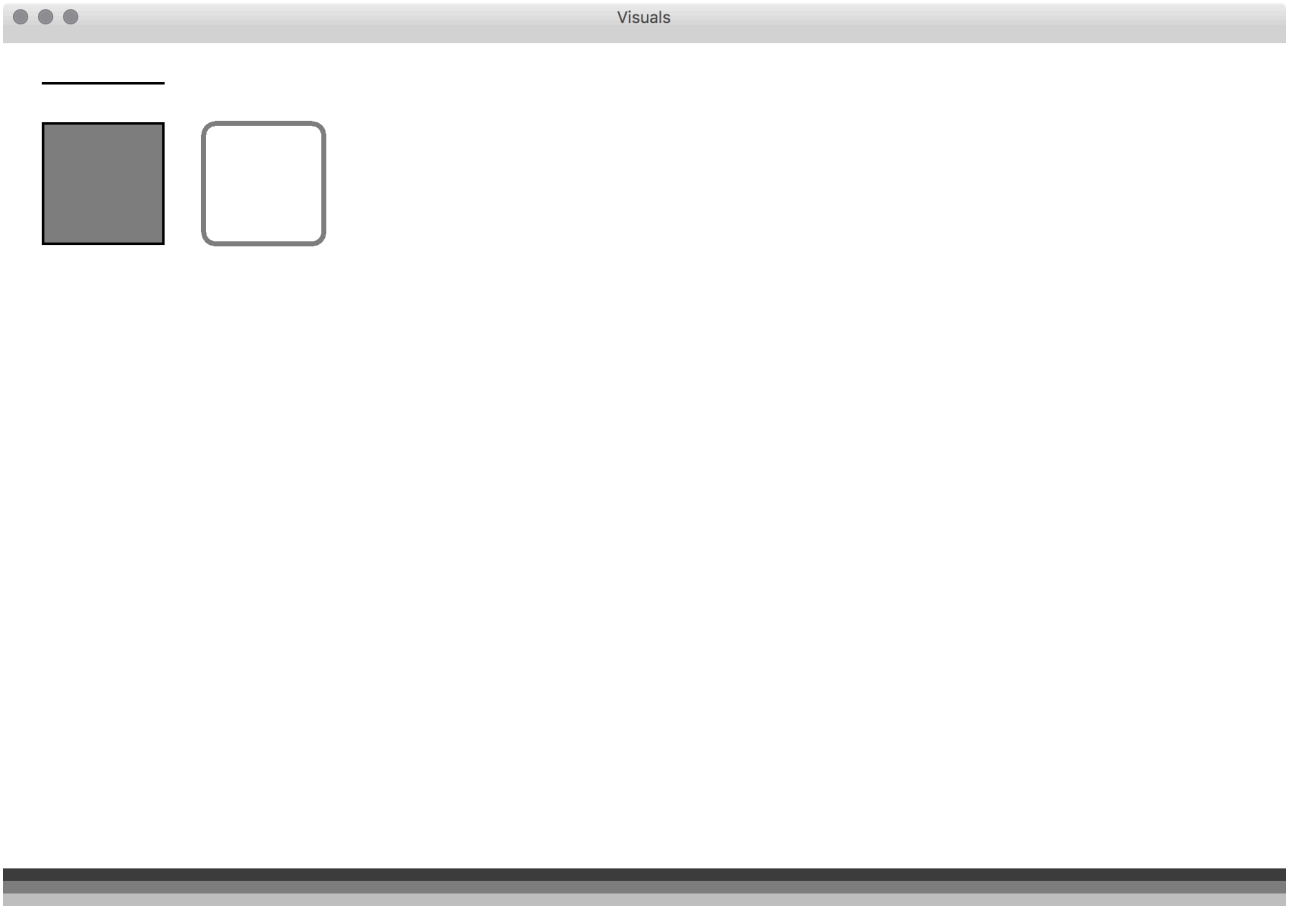
Ravno črto narišemo z uporabo metode `drawLine(int x1, int y1, int x2, int y2)`, pri čimer parametri predstavljajo koordinate začetka in konca črte. Pazite, da je **koordinatno izhodišče** postavljeno v **zgornje levo oglišče** panela! To pomeni, da koordinate `(32, 64)` predstavljajo točko 32 pikslov od levega roba panela in 64 pikslov od zgornjega roba panela. Pred samim risanjem lahko z uporabo metode `setColor(Color color)` določimo barvo čopiča in z uporabo metode `setStroke(Stroke stroke)` obliko čopiča (npr. debelino).

```java
int x = indent, y = indent;
graphics.setColor(Color.BLACK);
graphics.setStroke(new BasicStroke(2.0f));
graphics.drawLine(x, y, x + size, y);
```

Kvadrat ali pravokotnik narišemo z uporabo metode `drawRect(int x, int y, int width, int height)`, pri čimer parametri predstavljajo koordinati zgornjega levega oglišča pravokotnika ter njegovo širino in višino. Za večino `draw` metod v razredu `Graphics` obstaja tudi pripadajoča `fill` metoda, ki **zapolni površino** geometrijskega lika, in v nekaterih primerih tudi pripadajoča `drawRound` metoda, ki nariše geometrijski lik z **zaobljenimi oglišči**.

```java
y += indent;
graphics.setColor(Color.GRAY);
graphics.setStroke(new BasicStroke(1.0f));
graphics.fillRect(x, y, size, size);
graphics.setColor(Color.BLACK);
graphics.setStroke(new BasicStroke(2.0f));
graphics.drawRect(x, y, size, size);
x += indent + size;
graphics.setColor(Color.GRAY);
graphics.setStroke(new BasicStroke(4.0f));
graphics.drawRoundRect(x, y, size, size, size / 5, size / 5);
```

Primer izgleda grafičnega vmesnika je prikazan spodaj.



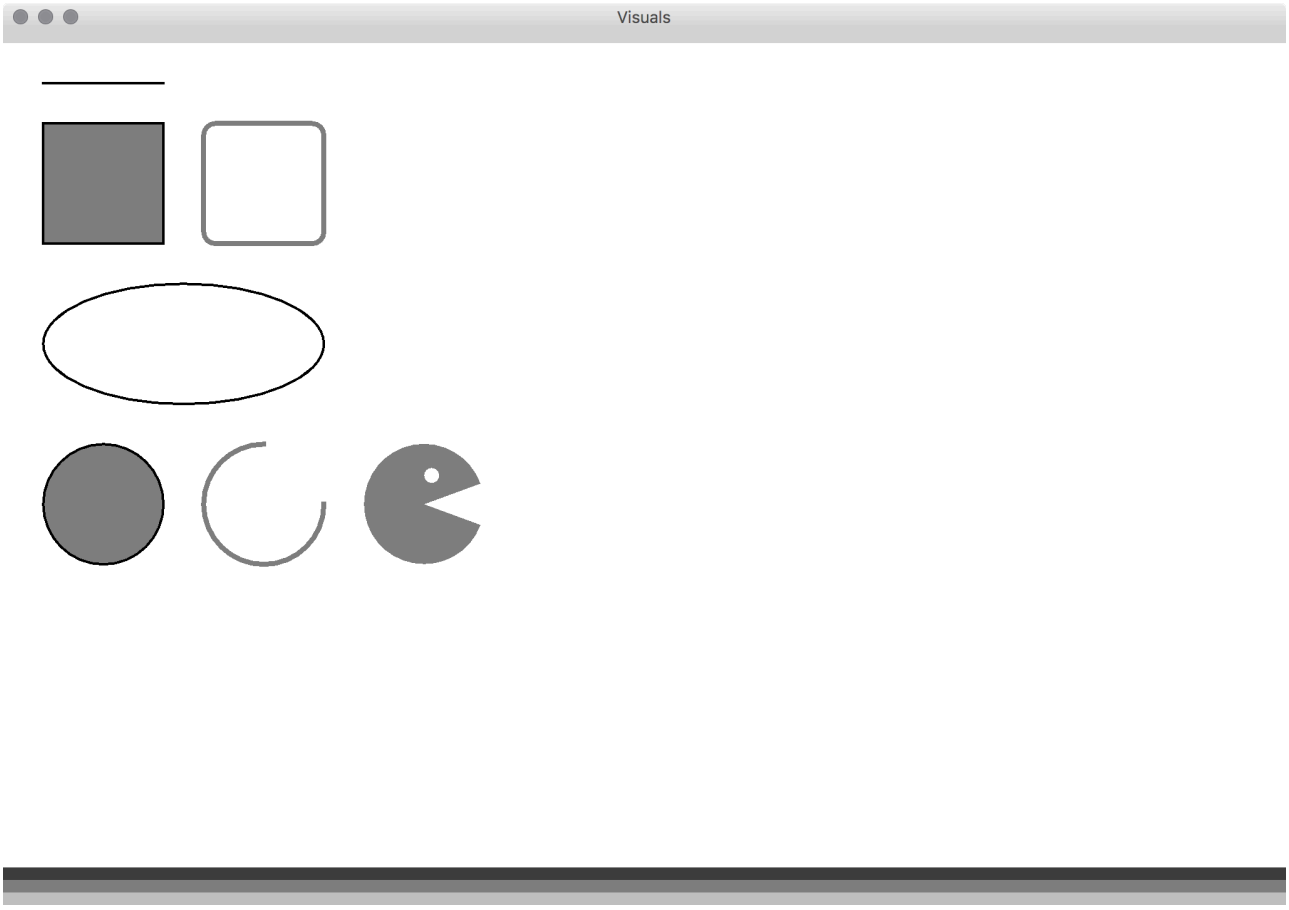
Krog ali elipso narišemo z uporabo metode `drawOval(int x, int y, int width, int height)`, pri čimer parametri predstavljajo koordinati zgornjega levega oglišča najmanjšega očrtanega pravokotnika ter širino in višino elipse.

```java
x = indent; y += indent + size;
graphics.setColor(Color.BLACK);
graphics.setStroke(new BasicStroke(2.0f));
graphics.drawOval(x, y, 2 * size + indent, size);
y += indent + size;
graphics.setColor(Color.GRAY);
graphics.setStroke(new BasicStroke(1.0f));
graphics.fillOval(x, y, size, size);
graphics.setColor(Color.BLACK);
graphics.setStroke(new BasicStroke(2.0f));
graphics.drawOval(x, y, size, size);
```

Krožni izsek narišemo z uporabo metode `drawArc(int x, int y, int width, int height, int start, int arc)`, pri čimer zadnja dva parametra predstavljata začetek in velikost krožnega izseka v stopinjah.

```java
x += indent + size;
graphics.setColor(Color.GRAY);
graphics.setStroke(new BasicStroke(4.0f));
graphics.drawArc(x, y, size, size, 90, 270);
x += indent + size;
graphics.setColor(Color.GRAY);
graphics.setStroke(new BasicStroke(1.0f));
graphics.fillArc(x, y, size, size, 20, 320);
graphics.setColor(Color.WHITE);
graphics.setStroke(new BasicStroke(2.0f));
graphics.fillOval(x + size / 2, y + size / 5, size / 8, size / 8);
```

Primer izgleda grafičnega vmesnika je prikazan spodaj.

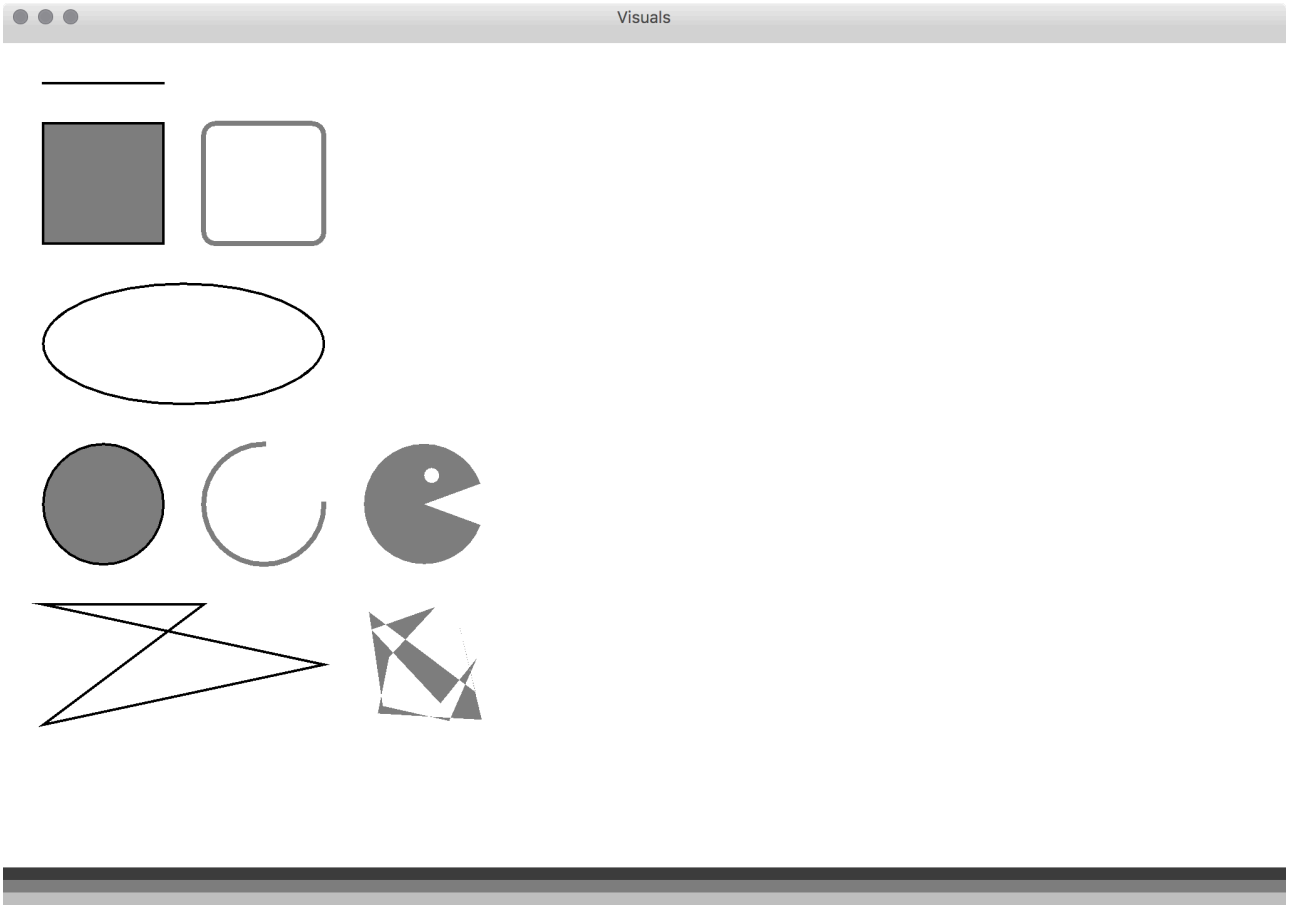


Zaprt poligon narišemo z uporabo metode `drawPolygon(Polygon polygon)`, pri čimer je edini parameter objekt razreda `Polygon` v paketu `java.awt`. Le-tega sestavimo kot urejeno zaporedje oglišč, ki jih dodamo poligonu z uporabo metode `addPoint(int x, int y)`.

Java

```
x = indent; y += indent + size;
graphics.setColor(Color.BLACK);
graphics.setStroke(new BasicStroke(2.0f));
Polygon quadrilateral = new Polygon();
quadrilateral.addPoint(x, y); quadrilateral.addPoint(x + 2 * size + indent, y + size / 2);
quadrilateral.addPoint(x, y + size); quadrilateral.addPoint(x + size + indent, y);
graphics.drawPolygon(quadrilateral);
x += 2 * (indent + size);
graphics.setColor(Color.GRAY);
graphics.setStroke(new BasicStroke(4.0f));
Polygon polygon = new Polygon();
for (int i = 0; i < 12; i++)
    polygon.addPoint((int)(x + Math.random() * size), (int)(y + Math.random() * size));
graphics.fillPolygon(polygon);
```

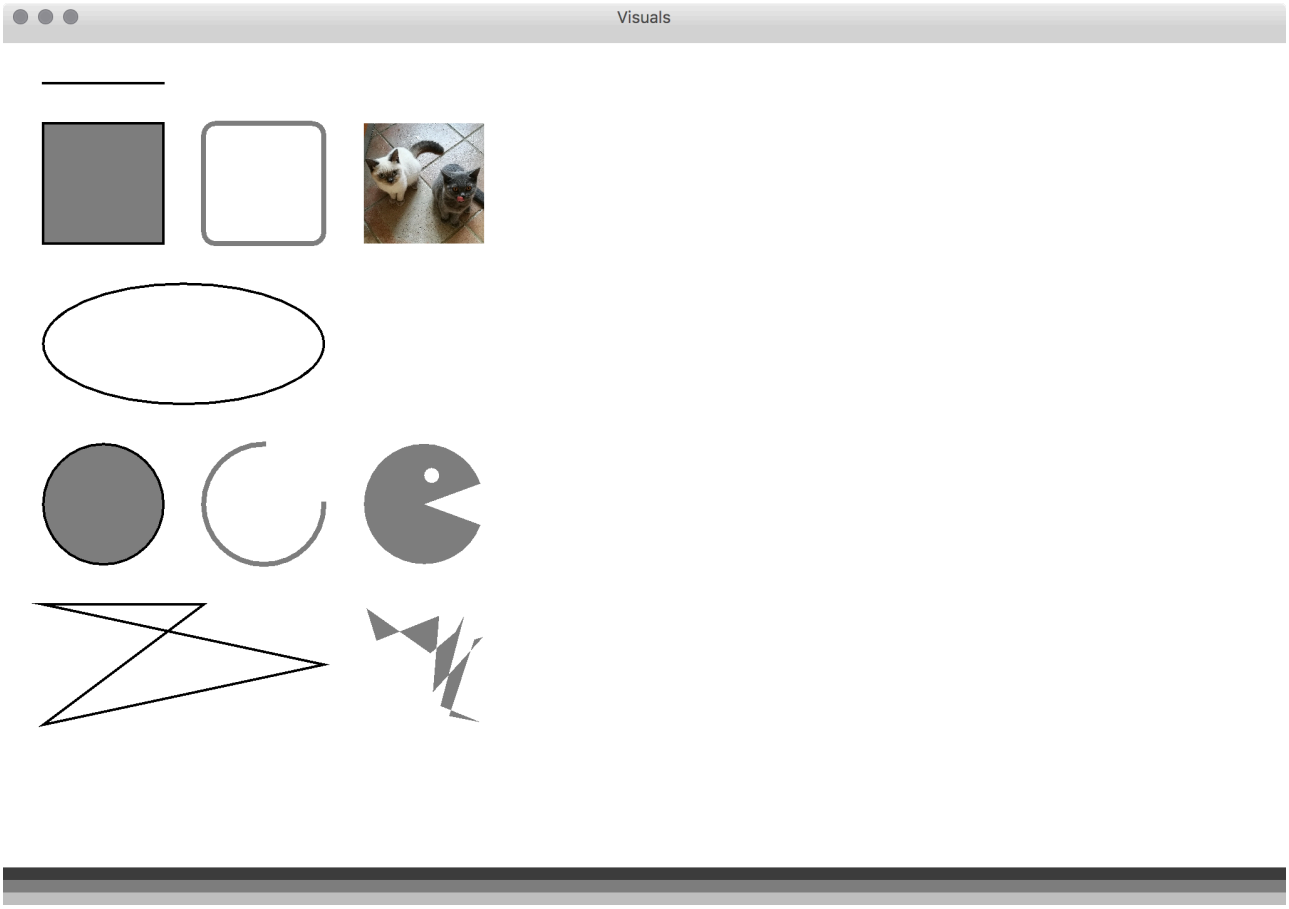
Primer izgleda grafičnega vmesnika je prikazan spodaj.



Poljubno sliko narišemo z uporabo metode `drawImage(Image image, int x, int y, int width, int height, ...)`, pri čimer parametri zaporedoma predstavljajo samo sliko, ki je objekt razreda `Image` v paketu `java.awt`, koordinati zgornjega levega oglišča slike ter širino in višino. Sliko lahko preberemo iz datoteke z uporabo statične metode `read(File file)` v razredu `javax.imageio.ImageIO`.

```java
y = 2 * indent;
try { // shrani sliko v spremenljivko ali konstanto!
    graphics.drawImage(ImageIO.read(new File("images", "cats.jpg")), x, y, size, size, null
} catch (IOException e) {
    e.printStackTrace();
}
```

Primer izgleda grafičnega vmesnika je prikazan spodaj.



Niz znakov narišemo z uporabo metode `drawString(String string, int x, int y)`, pri čimer parametri predstavljajo sam niz znakov in koordinati **spodnjega levega oglišča** očrtanega pravokotnika. Pred samim risanjem lahko z uporabo metode `setFont(Font font)` določimo obliko pisave, dočim geometrijske lastnosti trenutno izbrane pisave dobimo z uporabo funkcije `getFontMetrics()`, ki vrne objekt razreda `java.awt.FontMetrics`.

Z uporabo funkcij `getWidth()` in `getHeight()` dobimo **trenutno širino in višino** poljubnega grafičnega elementa.

Java

```
x = 2 * indent + size; y = indent;
String string = getWidth() + "x" + getHeight();
graphics.setColor(Color.BLACK);
graphics.setFont(new Font("Montserrat", Font.BOLD, 11));
graphics.drawString(string, x, y);
x += size / 2; y += indent + size / 2;
graphics.setColor(Color.GRAY);
graphics.setFont(new Font(Font.SANS_SERIF, Font.BOLD, 14));
FontMetrics metrics = graphics.getFontMetrics();
graphics.drawString(string, x - metrics.stringWidth(string) / 2, y + metrics.getAscent() / 
```

Primer izgleda grafičnega vmesnika je prikazan spodaj.

```
x = getWidth() / 2 + indent; y = indent;
graphics.setColor(Color.BLACK);
graphics.setStroke(new BasicStroke(2.0f));
graphics.drawLine(x, y, x + width, y);

y += indent;
graphics.setColor(Color.GRAY);
graphics.setStroke(new BasicStroke(1.0f));
graphics.fillRect(x, y, width, height);
graphics.setColor(Color.BLACK);
graphics.setStroke(new BasicStroke(2.0f));
graphics.drawRect(x, y, width, height);
x += indent + width;
graphics.setColor(Color.GRAY);
graphics.setStroke(new BasicStroke(4.0f));
graphics.drawRoundRect(x, y, width, height, width / 5, height / 5);

x = getWidth() / 2 + indent; y += indent + height;
graphics.setColor(Color.BLACK);
graphics.setStroke(new BasicStroke(2.0f));
graphics.drawOval(x, y, 2 * width + indent, height);
y += indent + height;
graphics.setColor(Color.GRAY);
graphics.setStroke(new BasicStroke(1.0f));
graphics.fillOval(x, y, width, height);
graphics.setColor(Color.BLACK);
graphics.setStroke(new BasicStroke(2.0f));
graphics.drawOval(x, y, width, height);

x += indent + width;
graphics.setColor(Color.GRAY);
graphics.setStroke(new BasicStroke(4.0f));
graphics.drawArc(x, y, width, height, 90, 270);
x += indent + width;
graphics.setColor(Color.GRAY);
graphics.setStroke(new BasicStroke(1.0f));
graphics.fillArc(x, y, width, height, 20, 320);
graphics.setColor(Color.WHITE);
graphics.setStroke(new BasicStroke(2.0f));
graphics.fillOval(x + width / 2, y + height / 5, width / 8, height / 8);

x = getWidth() / 2 + indent; y += indent + height;
graphics.setColor(Color.BLACK);
graphics.setStroke(new BasicStroke(2.0f));
quadrilateral = new Polygon();
quadrilateral.addPoint(x, y); quadrilateral.addPoint(x + 2 * width + indent, y + height / 2);
quadrilateral.addPoint(x, y + height); quadrilateral.addPoint(x + width + indent, y);
graphics.drawPolygon(quadrilateral);
```