# Java és web programozás

Kovács Kristóf

Budapesti Műszaki Egyetem

2015. 03. 18.

# 6. Előadás

# Grafika Java-ban

Emlékezzünk kicsit vissza a **tikz**-re:

```
\begin{tikzpicture}
    \draw (0,0) node[draw,circle] (S) {s};
    \draw (3,2) node[draw,circle] (A) {a};
    \draw (3,-2) node[draw,circle] (B) {b};
    ...
```

Ehhez kicsit hasonló lesz a rajzolás Java-ban.

# Keretek

Kell valami amibe tudunk rajzolni. Három ilyennel fogunk megismerkedni:

- Frame
- JFrame
- JPanel

# Keretek

Kell valami amibe tudunk rajzolni. Három ilyennel fogunk megismerkedni:

- Frame
- JFrame
- JPanel

Kezdjük a **Frame**-el:

```
public class Jatek extends Frame {
    ...
    public void paint(Graphics g) {
        // Rajzolas
    }
    ...
}
```

Csak akkor fogjuk használni, ha csak nagyon kicsi progit írunk, pl gyakorlati feladatokat.

# JFrame

Tekintsünk erre úgy, mint a programjaink fõablakára. Ez lesz a kiinduló pontja (mondhatjuk úgy, hogy a main-je) a grafikus programjainknak.
A konstruktorában érdemes használni a következõ metódusait:

- **setSize(int, int)**: beállítja az ablak méretét
- **setTitle(String)**: beállítja az ablak nevét
- **setVisible(true)**: láthatóvá teszi az ablakban rajzolt dolgokat

# JFrame példa

```
public class AsteroidsGame extends JFrame {
    public AsteroidsGame() {
        initUI();
    }
    private void initUI() {
        setSize(screenX, screenY);
        setTitle("Asteroids");
        setVisible(true);

        // Ezek mindig itt lesznek:
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }
    ...
}
```

# Main a JFrame-hez

Ezt nem kell teljesen megérteni, elég ha copy-paste-eljük mindig.
Ezzel indul el biztonságosan a megírt JFrame osztályunk.

```
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            AsteroidsGame game = new AsteroidsGame();
        }
    });
}
```

# JPanel

A **JPanel**-re gondolhatunk úgy, mint ablak az ablakban, ennek a segítségével feloszthatjuk az ablakunkat több részre. Viszont ha ezt nem is tervezzük, akkor is érdemes inkább **JPanel**-be rajzolni és nem közvetlen a **JFrame**-be.
Hogy összekössünk egy **JPanel**-t egy **JFrame**-el, a **JFrame**-hez hozzá kell adnunk a **JPanel**-t. Ezt egyszerûen megtehetjük az **add** metódussal a **JFrame**-ben:

```
...
JPanel myPanel = new JPanel();
add(myPanel);
...
```

Errõl részletesebben a következõ elõadáson tanulunk, most nézzük inkább hogyan lehet rajzolni.

# Rajzolás

A **Frame**-be rajzoláshoz, a **paint(Graphics)** metódust használjuk:

```
public class MyFrame extends Frame {
    @Override
    public void paint(Graphics g) {
        // Rajzolas
    }
    ...
}
```

Míg a **JPanel**-ben a **paintComponent(Graphics)** metódust:

```
    @Override
    public void paintComponent(Graphics g) {
        Graphics2D g2d = (Graphics2D) g.create();
        ...
    }
```

Érdemes **Graphics2D**-vé alakítani a **Graphics**-unkat, hogy több lehetõségünk legyen.

# Rajzolás

Az elõzõ két metódusban használhatjuk a paraméterként kapott **Graphics2D** objektumot a rajzoláshoz. A következõ legalapvetõbb metódusai vannak:

- **g2d.setColor(Color)**: beállítja a színt amivel rajzolni fogunk
- **g2d.drawRect(int, int, int, int)**: kirajzol egy téglalapot, az elsõ kettõ paraméter a téglalap koordinátái, a második kettõ a szélessége és magassága
- **g2d.fillRect(int, int, int, int)**: kitöltött téglalapot rajzol, a paraméterek mint korábban
- **g2d.drawOval(int, int, int, int)**: egy kör / ellipszist rajzol, a paraméterek mint korábban
- **g2d.drawArc(int, int, int, int, int, int)**: egy körívet rajzol, elsõ 4 paraméter mint korábban, utolsó elõtti, a körív kezdete, a második a körív hossza

Az utolsó kettõbõl is van fill-es változat, **fillOval** és **fillArc**.

# Rajzolás példa

```
g2d.setColor(Color.black); // Szín beállítása
g2d.drawRect(50, 50, 50, 70); // Keret rajzolása
g2d.setColor(Color.blue);
g2d.fillRect(50, 50, 50, 70); // Kitöltés

g2d.fillArc(120, 130, 110, 100, 5, 150);
g2d.fillOval(270, 130, 50, 50);
```

A kód elsõ része rajzol egy téglalapot fekete kerettel kékkel kitöltve.
A második része rajzol egy körívet és egy kört.

# Bonyolultabb rajzolások

Rajzolhatunk tetszõleges poligont a **GeneralPath** osztály használatával. Három fontos metódusa van:

- **moveTo(float, float)**: a ceruzát az adott pontra helyezi
- **lineTo(float, float)**: az elõzõ pontból húz egy vonalat az adottba
- **closePath()**: bezárja a poligont (a legelsõ pontot az utolsóval összeköti)

Ha így létrehoztunk egy poligont, akkor még ki kell rajzolnunk a **Graphics2D draw** vagy **fill** metódusával. Paraméterként kell átadni nekik a **GeneralPath** objektumot.

# Bonyolultabb rajzolások példa

```
GeneralPath pent = new GeneralPath();

pent.moveTo(0, 0);

pent.lineTo(0, 75);
pent.lineTo(75, 75);
pent.lineTo(75, 0);

pent.closePath();

g2d.setColor(Color.blue);
g2d.fill(pent);
```

Ez a példa rajzol egy téglalapot. Figyeljük meg, hogy elõször le kell tenni valahova a ceruzát a **moveTo** metódussal és csak utána tudunk vonalakat húzni a **lineTo** metódussal.

# Stringek kirajzolása

Ahhoz, hogy **String**-et ki tudjunk rajzolni létre kell hozni egy **Font** objektumot, ami a betûk megjelenését szabályozza:

- **Font(String, int, int)**:
  - az elsõ paraméter a betûkészletet adja meg (pl **Ariel**)
  - a második a betû stílusát (pl **Font.PLAIN** vagy **Font.BOLD**)
  - a harmadik a betûméretet.

A **Graphics2D** osztály **setFont** metódusával állíthatjuk be a fontot, majd a **drawString** metódusával írhatunk ki **String**et.

- **g2d.drawString(String, int, int)**: az elsõ paraméter a kiírandó **String** a második kettõ a koordinátái a szövegnek

# Stringek kirajzolása példa

```
g2d.setColor(Color.black);
g2d.setFont(new Font("Ariel", Font.PLAIN, 20));
g2d.drawString("Volt egyszer egy kiskutya", 50, 40);
```

Itt közvetlenül a **setFont** metódusban hoztam létre a **Font**-ot, de külön változóba is menthetjük, de ez felesleges.

# Koordináta transzformáció

Elég gyakran esik meg, hogy el kell forgatnunk valamilyen objektumot, vagy az egész képernyõt el kell tolnunk egy vektorral. Ezeket is viszonylag egyszerû megoldani három metódus segítségével:

- **g2d.translate(double, double)**: az adott vektorral eltolja a koordinátatengely középpontját
- **g2d.rotate(double, double, double)**: az elsõ paraméterû radiánnal elforgatja a második két paraméter körüli pont körül a koordinátatengelyt
- **g2d.scale(double, double)**: az adott értékekkel felskálázza a koordinátatengelyeket

# Koordináta transzformáció

Ha vissza szeretnénk térni az eredeti koordinátázáshoz miután egy transzformációt végeztünk, akkor érdemes elõször egy másolatot készíteni a **Graphics**-ról, még a transzformáció elõtt:

```
Graphics2D g2d = (Graphics2D) g.create();
```

Majd, ha végeztünk a transzformálással és a rajzolással (g2d-t kell használnunk), akkor a következõ paranccsal visszaállunk az eredeti koordinátázáshoz:

```
g2d.dispose();
```

# Koordináta transzformáció példa

```
Graphics2D g2d = (Graphics2D) g.create();
g2d.setColor(Color.black);

g2d.rotate(Math.toRadians(25), 100, 30);
g2d.translate(10, 10);
g2d.scale(2, 2);

g2d.drawRect(100, 30, 50, 70);

g2d.dispose();
```

Itt egy téglalapot forgatok el 25 fokkal és rajzolom ki a (10, 10) vektorral eltolva és kétszeres skálázással. Ha megtörtént a kirajzolás akkor visszatérek az eredeti koordinátázásra, hogy a többi rajzolásnál ne legyen probléma a transzformálásból.