

JavaFx



Dnes:

- základné komponenty knižnice JavaFx
- spracovanie udalostí
- spôsoby návrhu jednoduchej (pravouhlej) hry

Zdroj a literatúra:

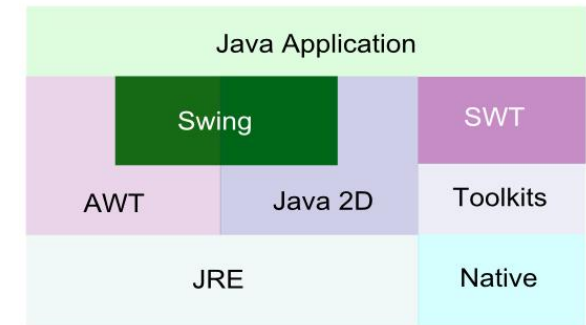
- ” [What Is JavaFX](#)
- ” [JavaFX 2.0: Introduction by Example](#)
- ” [Introduction to Java Programming, !!!!Tenth Edition](#)

Cvičenia:

jednoduché aplikácie s interakciou:

- maľovátka, euro-kalkulačka,
- logické hry: pexeso, piškvorky, ...
- dynamické hry: tenis

AWT-SWING SWT-JavaFx



Java Graphics - The Layer Cake

JAVA vznikla v r. 1995,

- “ Java 1.1, 1997, integrovaný grafický balík [AWT](#) (Abstract Windows Toolkit),
- “ Java 1.2, 1998, integrovaný grafický balík [SWING](#),
- “ od cca 2005, [SWT](#) (Standard Widget Toolkit), Eclipse Foundation, IBM
- “ Java 1.7, 2012, integrovaný grafický balík JavaFx, pôvodný vývoj od 2007.

SWING (AWT):

- “ je aktuálne m tvy, nevyvíja sa,
- “ nie je multi-platformový,
- “ je nepou0ite ný na mobilných zariadeniach,
- “ je aso pomerne a0kopádny pre programátora (ale zvykli sme si0).

medzi asom vznikli a odzneli: Adobe Flash, Microsoft Silverlight, ...

preto sme tie0 nahradili v prednázkach Swing a AWT platformou JavaFx

RIA

(Rich Internet application)

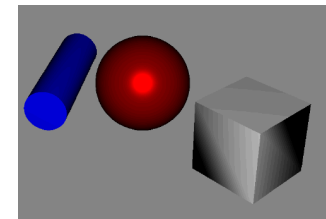


Cie : poskytnú rovnakú funkcionálnu a vlastnosti web-aplikáciám ako ich poznáme u desktopových aplikácií

Najrozzírenejšie platformy: Adobe Flash, Microsoft Silverlight, JavaFx, HTML5/JS

JavaFx :

- “ umožňuje spustiť lokálnu desktopovú aplikáciu v browseri, WebView,
- “ podporuje GUI ztvárňované pomocou CSS,
- “ podporuje multi-touch, multi-platform,
- “ podporuje SWING, komponent SwingNode ☺,
- “ 2D a 3D grafiku, animácie,
- “ multi-média (audio, video),
- “ nájdete ju inde, [ScalaFx](#), Ruby, ...

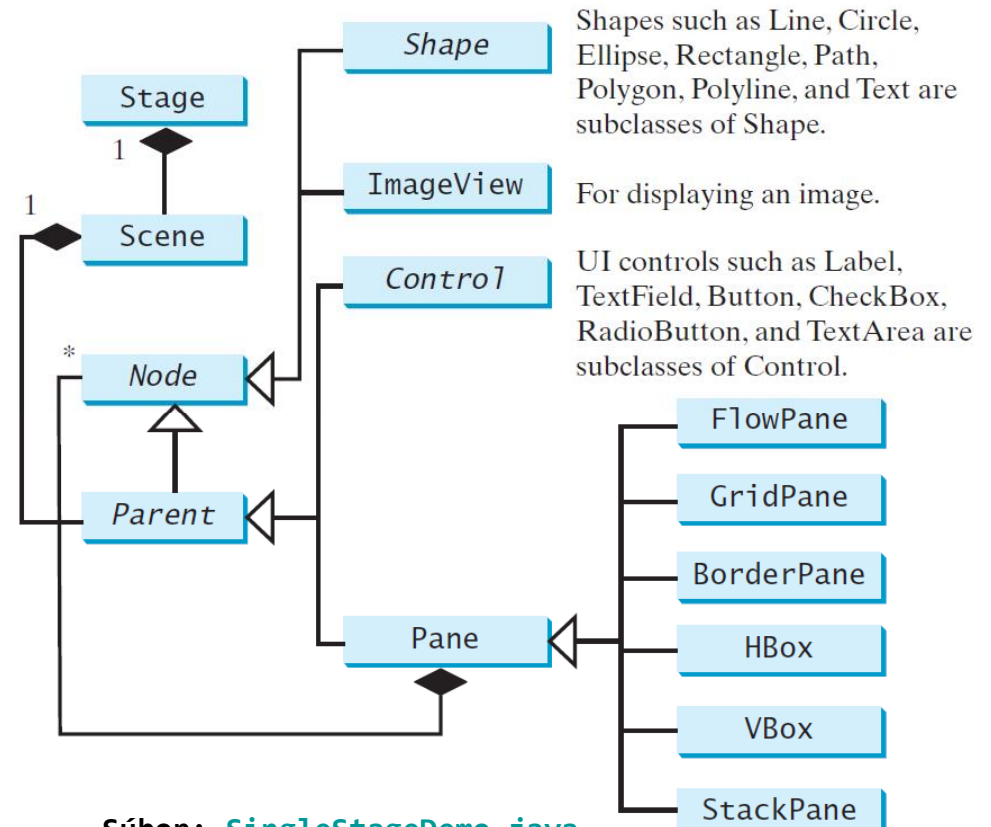
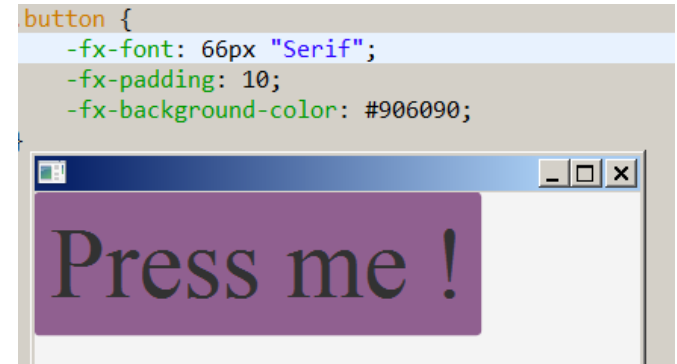
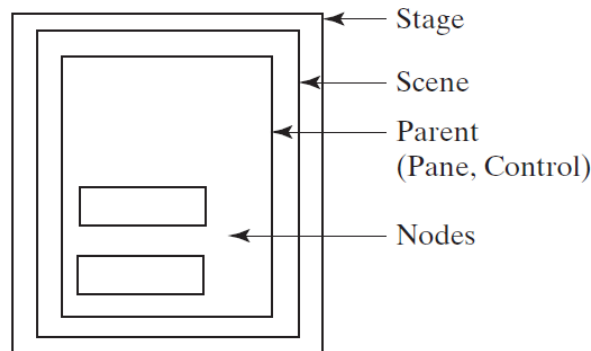


Ako začať :

- “ bezproblémová verzia eclipse (>= 4.4), java 1.8, a **e(fx)clipse** viac info tu:
<https://www.eclipse.org/efxclipse/install.html>
- “ **NetBeans**, <https://netbeans.org/features/java-on-client/javafx.html>

JavaFx aplikácia

```
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
            Button btn = new Button("Press me !");
            Pane root = new Pane(btn);
            Scene scene = new Scene(root,400,400);
            scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```



Source: [SingleStageDemo.java](#)

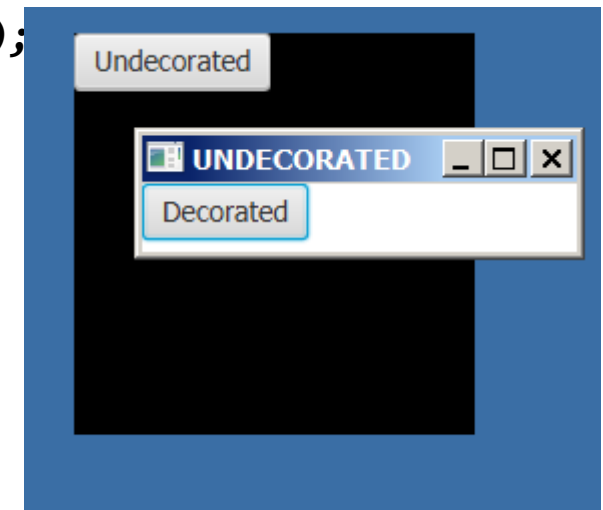
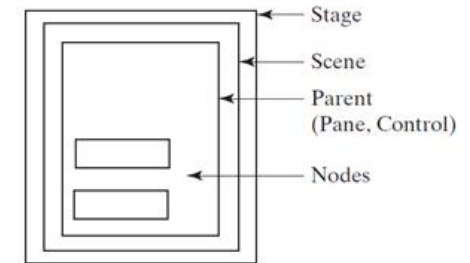


Stage

Stage je najvrchnejší kontajner, teda okno v rámci OS.

```
Group root = new Group(new Button("Undecorated"));
Scene scene = new Scene(root, 200, 200, Color.BLACK);
Stage newStage = new Stage(StageStyle.UNDECORATED);
newStage.setScene(scene);
newStage.initModality(Modality.WINDOW_MODAL);
newStage.setTitle("UNDECORATED");
newStage.show();
```

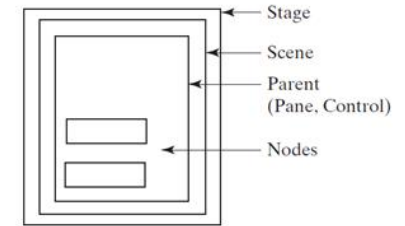
```
Group root = new Group(new Button("Decorated"));
Scene scene = new Scene(root);
Stage newStage = new Stage(StageStyle.DECORATED);
newStage.setScene(scene);
newStage.setTitle("UNDECORATED");
newStage.show();
```



```
.setTitle()
.setScene()
.sizeToScene()
.initStyle()
.initModality()
.show()
```

Súbor: [MultiStageDemo.java](#)

Scene



Scéna predstavuje vrchný element stromovej zruktúry elementov typu Node, resp. Parent

```
" Scene(Parent root)
  new Scene(root);
" Scene(Parent root, double width, double height)
  new Scene(root, 400, 400);
" Scene(Parent root, double w, double h, Paint fill)
  new Scene(root, 200, 200, Color.BLACK);
```

Parent má deti typu Node, presnejšie poskytuje metódu `ObservableList<Node> getChildren()`

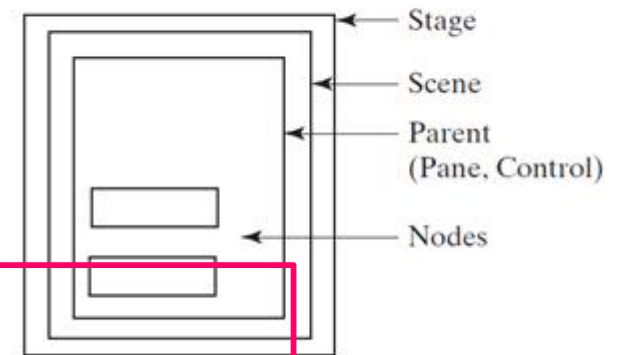
```
root.getChildren().addAll(node1, node2, ...)
```

Parent

```
" Control,
" Group,
" Region,
  " Axis,
  " Chart,
  " Pane
    " BorderPane,
    " FlowPane,
    " GridPane,
    " HBox,
    " StackPane,
    " Vbox,
  "
```

...

¥truktúra tried



Node

```
" Canvas,  
" ImageView,  
" Parent,  
" MediaView,  
" Shape  
  " Circle,  
  " Ellipse,  
  " Line,  
  " Polygon,  
  " Polyline,  
  " Rectangle,  
  " Text  
  ...
```

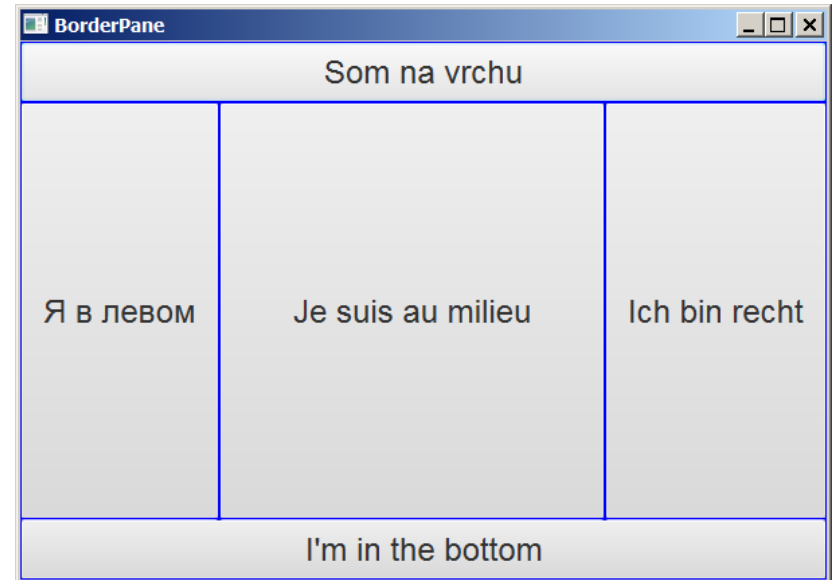
Control

```
" ChoiceBox,  
" ComboBoxBase,  
  " ComboBox  
" Labeled,  
  " ButtonBase,  
    " Button,  
    " CheckBox,  
    " ToggleButton  
  " Label,  
" ListView,  
" TextInputControl,  
  " TextArea,  
  " TextField
```

BorderPane

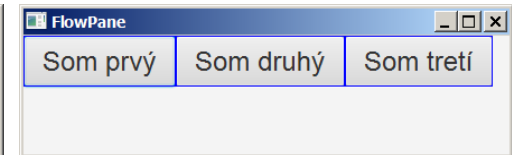
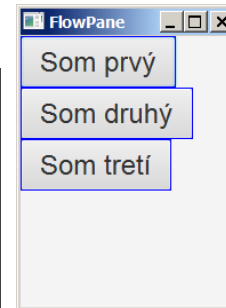
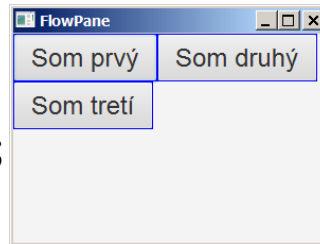
```
class MyButton extends Button {  
    public MyButton(String text) {  
        super(text);  
        setMaxWidth(Double.MAX_VALUE);  
        setMaxHeight(Double.MAX_VALUE);  
        setStyle("-fx-border-color: blue;  
                -fx-font: 24px 'Arial'");  
    }  
}
```

```
BorderPane root = new BorderPane();  
root.setTop(new MyButton("Som na vrchu"));  
root.setBottom(new MyButton("I'm in the bottom"));  
root.setRight(new MyButton("Ich bin recht"));  
root.setLeft(new MyButton("Я в левом"));  
root.setCenter(new MyButton("Je suis au milieu"));  
Scene scene = new Scene(root, 600, 400);  
primaryStage.setScene(scene);  
primaryStage.setTitle("BorderPane");  
primaryStage.show();
```



FlowPane, GridPane

```
FlowPane root = new FlowPane(  
    new MyButton("Som prvý"), new MyButton("Som druhý"), new MyButton("Som tretí"));  
Scene scene = new Scene(root, 300, 400);  
Stage newStage = new Stage();  
newStage.setScene(scene);  
newStage.setTitle("FlowPane");  
newStage.show();
```



```
GridPane root = new GridPane();  
for (int i = 0; i < 5; i++)  
    for (int j = 0; j < 5; j++)  
        root.add(new MyButton(i + "x" + j), j, i);  
root.setHgap(10);  
root.setVgap(10);  
Scene scene = new Scene(root, 400, 400);  
Stage newStage = new Stage();  
newStage.setScene(scene);  
newStage.setTitle("GridPane");  
newStage.show(); }
```



Súbor: [Layouts.java](#)

HBox, VBox, StackPane

```
HBox root = new HBox(  
    new MyButton("Som prvý"), new MyButton("Som druhý"), new MyButton("Som tretí"));
```

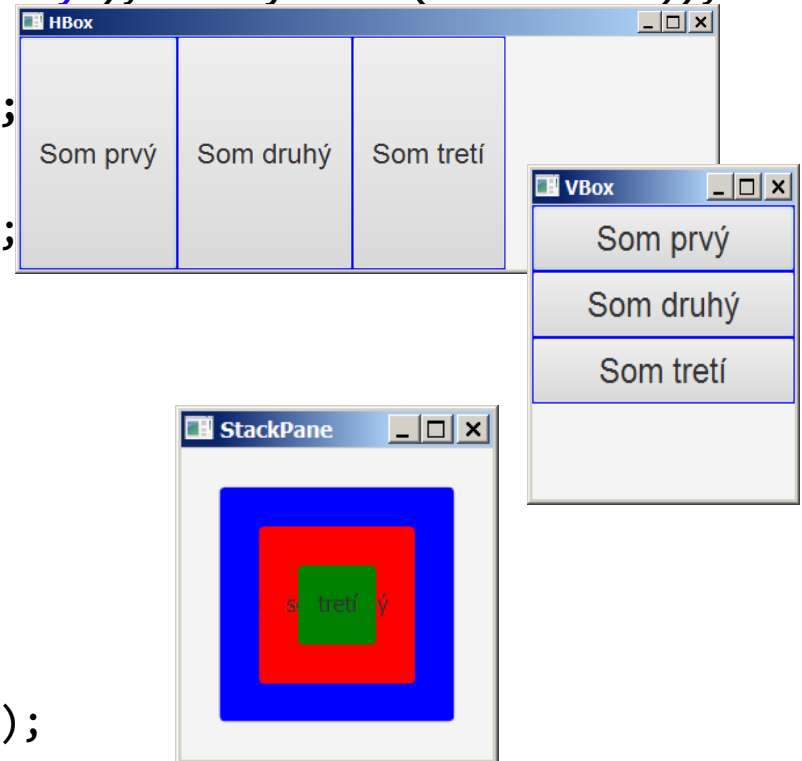
```
VBox root = new VBox(  
    new MyButton("Som prvý"), new MyButton("Som druhý"), new MyButton("Som tretí"));
```

```
Button btn1 = new Button("naozaj som prvý");  
btn1.setPrefSize(150,150);  
btn1.setStyle("-fx-background-color: blue");
```

```
Button btn2 = new Button("som druhý");  
btn2.setPrefSize(100,100);  
btn2.setStyle("-fx-background-color: red");
```

```
Button btn3 = new Button("tretí");  
btn3.setPrefSize(50,50);  
btn3.setStyle("-fx-background-color: green");
```

```
StackPane root = new StackPane(btn1, btn2, btn3);
```



EventHandler

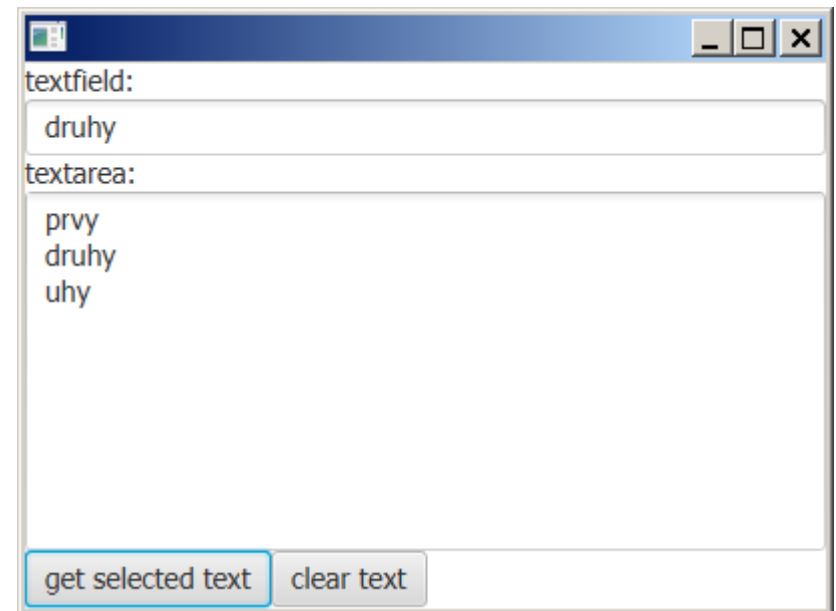
```
class MyButton extends Button {  
    setOnAction(new EventHandler<ActionEvent>() {  
        @Override  
        public void handle(ActionEvent event) {  
            System.out.println("stlačil si " + text);  
        }  
    });  
    setOnAction(event -> {  
        System.out.println("stlačil si " + text);  
    });  
    setOnMouseClicked(event -> {  
        System.out.println("klikol si " + text + ", " +  
            event.getX() + ", " + event.getY());  
    });  
    setOnKeyPressed(event -> {  
        System.out.println("stlačil si " + text + ", " +  
            event.getCode());  
    });  
});
```

TextField, TextArea

```
Button b1 = new Button("get selected text");
Button b2 = new Button("clear text");
TextField tf = new TextField(); tf.setPrefWidth(100);
TextArea ta = new TextArea(); ta.setPrefWidth(100);
ta.setEditable(false);

b1.setOnAction(event -> {
    ta.setText(ta.getSelectedText());});
b2.setOnAction(event -> {
    ta.clear(); });
tf.setOnAction(event -> {
    ta.appendText(tf.getText() + "\n");});

VBox fp = new VBox(
    new Label("textfield:"), tf,
    new Label("textarea:"), ta,
    new FlowPane(b1, b2));
```



Malá kalkulačka

The screenshot shows a Java Swing window titled "Hypotéka". It contains five text input fields with labels: "Úrok [%]:" (5.5), "Délka [roky]:" (20), "Suma:" (100000), "Mesačně:" (687,89), and "Spolu:" (165092,95). Below the fields is a button labeled "Vyhodnot".

```
public class Hypoteka extends Application {
    TextField tfUrokovMiera = new TextField(),
               tfPocetRokov = new TextField(),
               tfSuma = new TextField(),
               tfMesacneSplatky = new TextField(),
               tfSpolu = new TextField();
    Button btVypocet = new Button("Vyhodnot");
    GridPane gridPane = new GridPane();
    gridPane.setHgap(5);
    gridPane.setVgap(5);
    gridPane.add(new Label("Úrok [%]:"),0,0);
    gridPane.add(tfUrokovMiera, 1, 0);
    gridPane.add(new Label("Délka [roky]:"),0,1);
    gridPane.add(tfPocetRokov, 1, 1);
    gridPane.add(new Label("Suma:"),0,2);
    gridPane.add(tfSuma, 1, 2);
    gridPane.add(new Label("Mesačně:"),0,3);
    gridPane.add(tfMesacneSplatky, 1,3);
    gridPane.add(new Label("Spolu:"),0,4);
    gridPane.add(tfSpolu, 1, 4);
    gridPane.add(btVypocet, 1, 5);

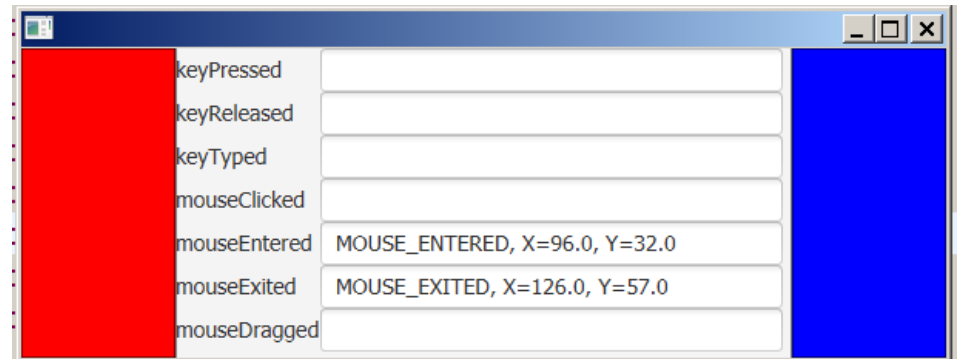
    tfUrokovMiera.setAlignment(Pos.BOTTOM_RIGHT);
    tfPocetRokov.setAlignment(Pos.BOTTOM_RIGHT);
    tfSuma.setAlignment(Pos.BOTTOM_RIGHT);
    tfMesacneSplatky.setAlignment(Pos.BOTTOM_RIGHT);
    tfSpolu.setAlignment(Pos.BOTTOM_RIGHT);
    tfMesacneSplatky.setEditable(false);
    tfSpolu.setEditable(false);

    btVypocet.setOnAction(e -> {
        rocnyUrok = Double.parseDouble(
            tfUrokovMiera.getText());
        pocetRokov = Integer.parseInt(
            tfPocetRokov.getText());
        suma = Double.parseDouble(
            tfSuma.getText());
        tfMesacneSplatky.setText(
            String.format("%.2f",
                mesacneSplatky()));
        tfSpolu.setText(
            String.format("%.2f",
                getTotalPayment()));
    });
}
```

Súbor: [Hypoteka.java](#)

MouseEvent, KeyEvent

```
Hashtable<String, Node> h = new Hashtable<String, Node>();
String[] event = { "keyPressed", "keyReleased", "keyTyped",
    "mouseClicked", "mouseEntered", "mouseExited", "mouseDragged"};
SmallPane bluePane = new SmallPane(this, Color.BLUE),
    redPane = new SmallPane(this, Color.RED);
GridPane gp = new GridPane();
for (int i = 0; i < event.length; i++) {
    TextField t = new TextField();
    t.setPrefWidth(300); t.setEditable(false);
    gp.add(new Label(event[i]), 0, i);
    gp.add(t, 1, i);
    h.put(event[i], t);
}
BorderPane bp = new BorderPane();
bp.setCenter(gp);
bp.setRight(bluePane);
bp.setLeft(redPane);
Scene scene = new Scene(bp, 600, 200);
```



Pokračovanie

```
class SmallPane extends Pane {
    SmallPane(AutoEvent parent, Color color) {
        this.parent = parent;
        this.color = color;
        setPrefWidth(100);
        setFocusTraversable(true);
        setOnKeyPressed(event -> {
            TextField t = (TextField) parent.h.get("keyPressed");
            t.setText(event.getEventType() + ", keyCode="+ event.getCode());
            paint();
            event.consume();
        });
        setOnMouseClicked(event -> {
            TextField t = (TextField) parent.h.get("mouseClicked");
            t.setText(event.getEventType() + ", X="+ event.getX() + ", Y="+ event.getY());
            paint();
            event.consume();
        });
    }
}
```

Event

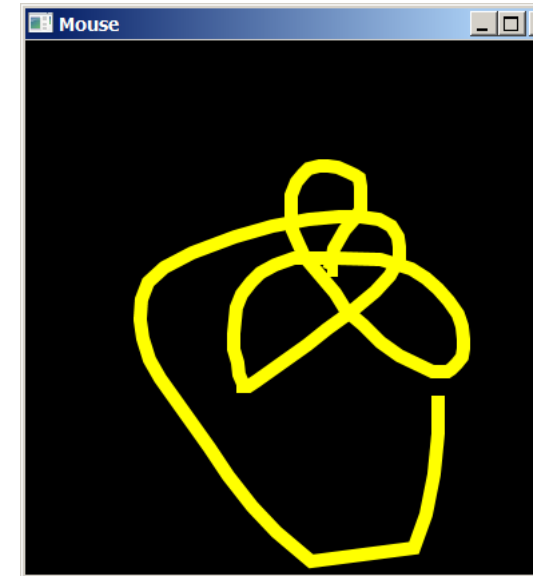
- " ActionEvent,
- " InputEvent,
 - . DragEvent
 - . KeyEvent,
 - . MouseEvent,
 - . TouchEvent
 -
- " WindowEvent,
- " ...

Súbor: MouseEvent.java

Polyline, Polygon

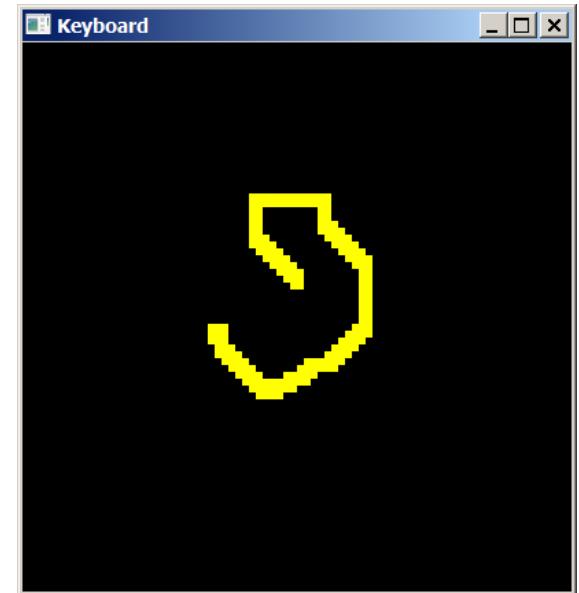
```
MousePane p = new MousePane();
Scene scene = new Scene(p, 400, 400, Color.BLACK);
scene.setOnMouseMoved(event -> {
    if (listOfPositions.size() >= 200) {
        listOfPositions.removeElementAt(0);
        listOfPositions.removeElementAt(1);
    }
    listOfPositions.addElement(
        event.getX());
    listOfPositions.addElement(
        event.getY());
    p.paint();
    event.consume();
} );
```

```
class MousePane extends Pane {
    public void paint() {
        getChildren().clear();
        Double[] d =
            listOfPositions.toArray(new Double[]{});
        Polyline p1 = new Polyline();
        p1.setStroke(Color.YELLOW);
        p1.setStrokeWidth(10);
        p1.getPoints().addAll(d);
        getChildren().add(p1);
    }
}
```



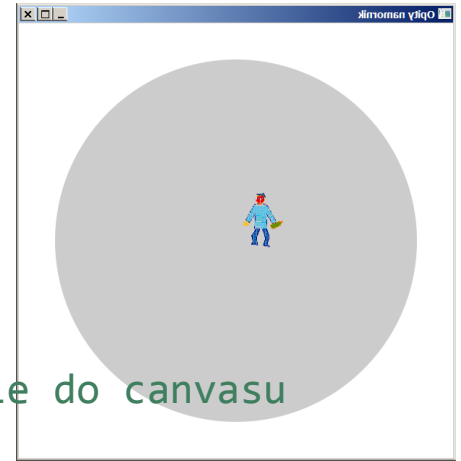
Pomocou zípiek

```
MousePane p = new MousePane();
Scene scene = new Scene(p, 400, 400, Color.BLACK);
scene.setOnKeyPressed(event -> {
    if (listOfPositions.size() >= 200) {
        listOfPositions.removeElementAt(0);
        listOfPositions.removeElementAt(0);
    }
    if (event.getCode() == KeyCode.UP) y -= 5;
    if (event.getCode() == KeyCode.DOWN) y += 5;
    if (event.getCode() == KeyCode.LEFT) x -= 5;
    if (event.getCode() == KeyCode.RIGHT) x += 5;
    listOfPositions.addElement(x);
    listOfPositions.addElement(y);
    p.paint();
    event.consume();
} );
```



Canvas

```
public void paintCanvas() {  
    GraphicsContext gc = getGraphicsContext2D();// kreslenie do canvasu  
    gc.clearRect(0, 0, sizeX, sizeY);  
    gc.setFill(Color.gray(0.2));  
    gc.fillOval(centerX - scale * moloSize, centerY - scale * moloSize,  
                scale * 2 * moloSize, scale * 2 * moloSize);  
    if (namornik.alive) { // ak sa este neutopil, nakresli obrazok namornika  
        gc.drawImage(new Image("namornik.gif"), // namornik.img,  
                     namornik.getXPixel(false),  
                     namornik.getYPixel(false));  
    } else { // ak je utopeny, nakresli vlny zobraz v strede vln pocet krokov  
        gc.setStroke(Color.RED);  
        gc.strokeText(Integer.toString(namornik.steps),  
                      namornik.getXPixel(true) - 8,  
                      namornik.getYPixel(true) + 7);  
    }  
}
```

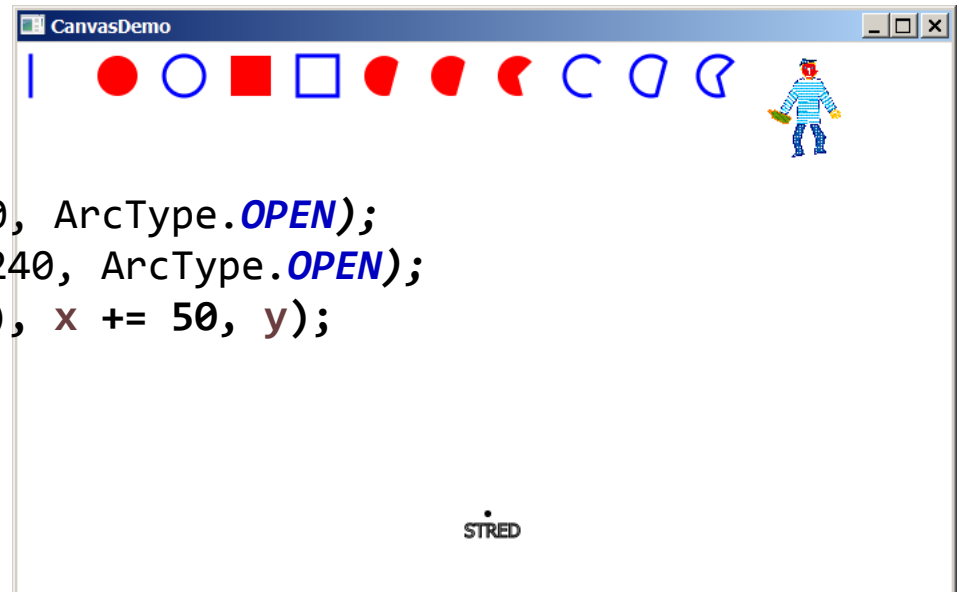


Kreslenie do Canvas

```
Canvas canvas = new Canvas(700, 700);
GraphicsContext gc = canvas.getGraphicsContext2D();
gc.fillOval(350, 350, 5, 5);
gc.strokeText("STRED", 335, 370);

.gc.setFill(Color.RED);
.gc.setStroke(Color.BLUE);
.gc.setLineWidth(3);
.gc.strokeLine(x, y, x, y + 30);
.gc.fillOval(x += 50, y, 30, 30);
.gc.strokeOval(x += 50, y, 30, 30);
.gc.fillRect(x += 50, y, 30, 30);
.gc.strokeRect(x += 50, y, 30, 30);

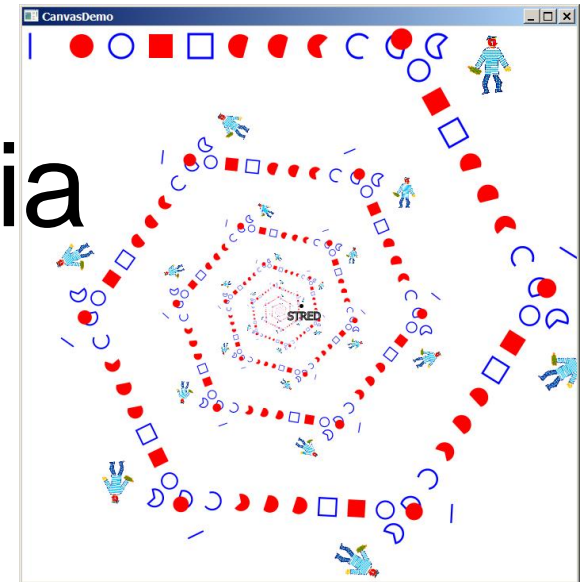
.gc.fillArc(x += 50, y, 30, 30, 45, 240, ArcType.OPEN);
.gc.strokeArc(x += 50, y, 30, 30, 45, 240, ArcType.OPEN);
.gc.drawImage(new Image("namornik.gif"), x += 50, y);
```



Afinné zobrazenia

Z lineárnej algebry:

- ” otočenie o uhol θ okolo stredu x, y ,
- ” posunutie dx, dy
- ” rovnoľahlosť /natiahnutie kx, ky pod a stredu x, y



```
Affine af = new Affine();
```

// afinné zobrazenie

```
for (int i = 0; i < 100; i++) {
```

```
    af.append(Affine.scale(0.9, 0.9, 350, 350)); // rovnoľahlosť
```

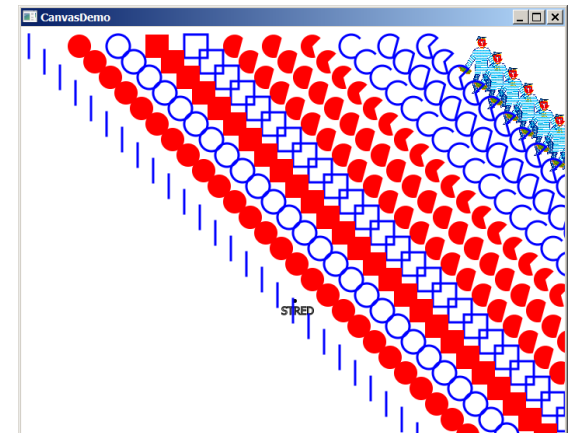
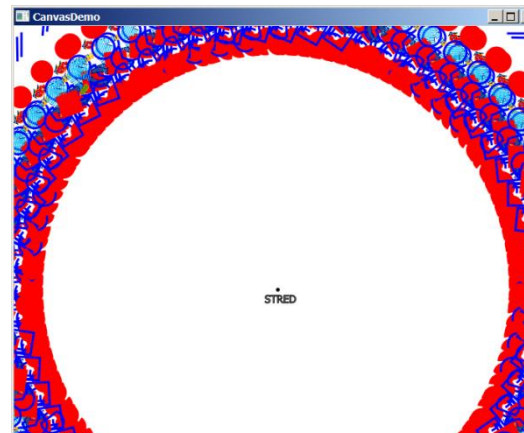
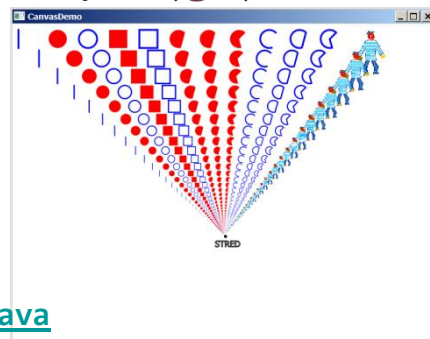
```
    af.append(Affine.rotate(60, 350, 350)); // otočenie
```

```
    af.append(Affine.translate(20, 20)); // posunutie
```

```
    gc.setTransform(af);
```

```
    paintShapes(gc);
```

```
}
```

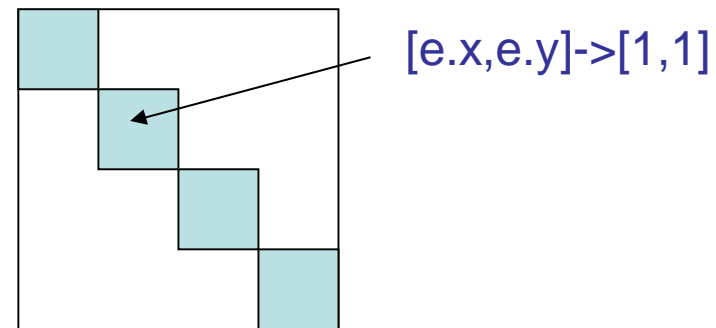


Hracia plocha

hracia plocha je často šachovnica rôznych rozmerov. Ako ju implementujeme:

1. jeden veľký canvas/Pane-l

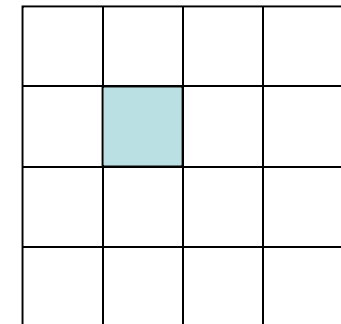
- musíme riešiť transformáciu pixelových súradníc do súradníc hracej plochy:



- a naopak, v metóde paint/paintComponent $[i, j] \rightarrow [x, y]$

2. grid canvasov/Pane-lov:

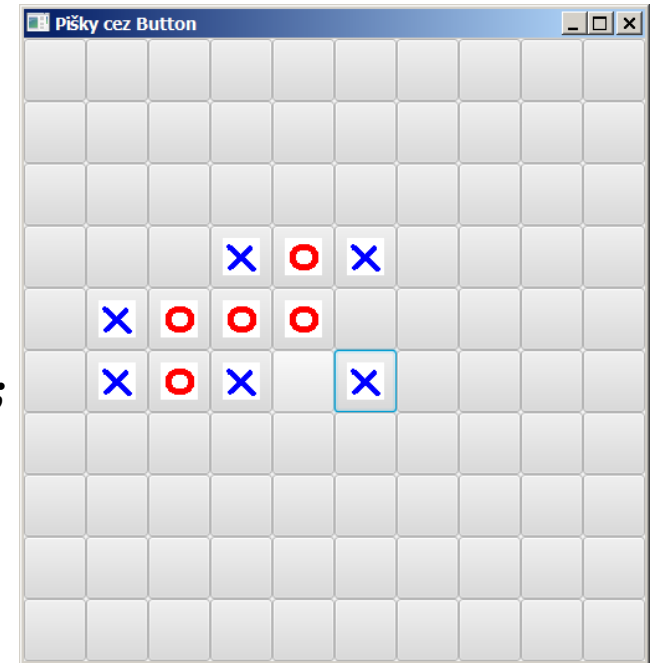
- každý canvas/panel má svoje súradnice od $[0, 0]$
- každý canvas/panel má svoj mouse lhandler
- každý canvas panel má svoju metódu paint/paintComponent
- veľkosť gridu upravíme podľa veľkosti obrázkov, resp. veľkosť obrázku upravíme podľa veľkosti panelu



3. grid buttonov/Button-ov:

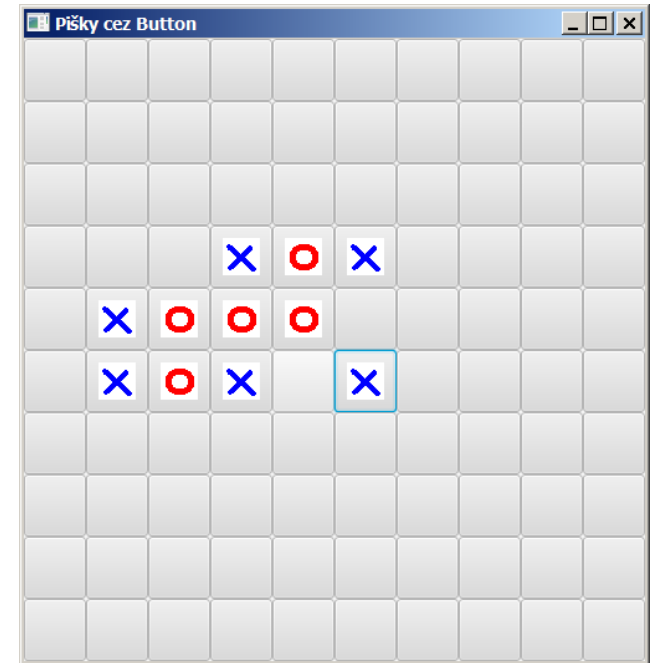
Riezenie Grid/Button

```
class PiskyState implements Serializable {  
    public int[][] playground = new int[SIZE][SIZE];  
    public boolean nextPlayerIsX = false;  
    public long elapsedTime = 0;  
}  
class Piskyground extends GridPane {  
    public Piskyground() {  
        for (int i = 0; i < PiskvorkyGridButton.SIZE; i++)  
            for (int j = 0; j < PiskvorkyGridButton.SIZE; j++)  
                add(new PiskyCell(i, j), j, i);  
    }  
}
```



Riezenie Grid/Button

```
class PiskyCell extends Button {  
    int i, j;  
    public PiskyCell(int i, int j) {  
        this.i = i;  
        this.j = j;  
        setPreferredSize(50, 50);  
        setOnAction(event -> {  
            if (ps.playground[i][j] != 0) return;  
            if (ps.nextPlayerIsX) {  
                ps.playground[i][j] = 1;  
                setGraphic(new ImageView(new Image("x.gif")));  
            } else {  
                ps.playground[i][j] = -1;  
                setGraphic(new ImageView(new Image("o.gif")));  
            }  
            ps.nextPlayerIsX = !ps.nextPlayerIsX;  
        } );  
    }  
}
```

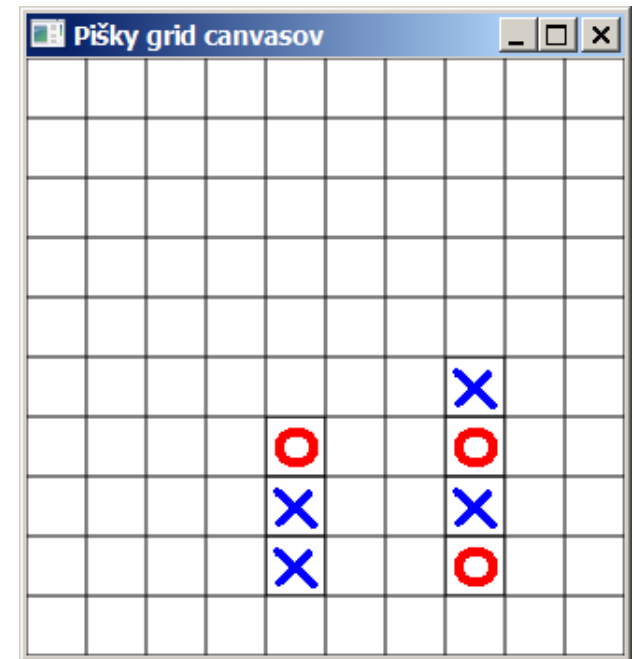


Riezenie Grid/Canvas

```
class PiskyCell extends Canvas {
    int i, j;

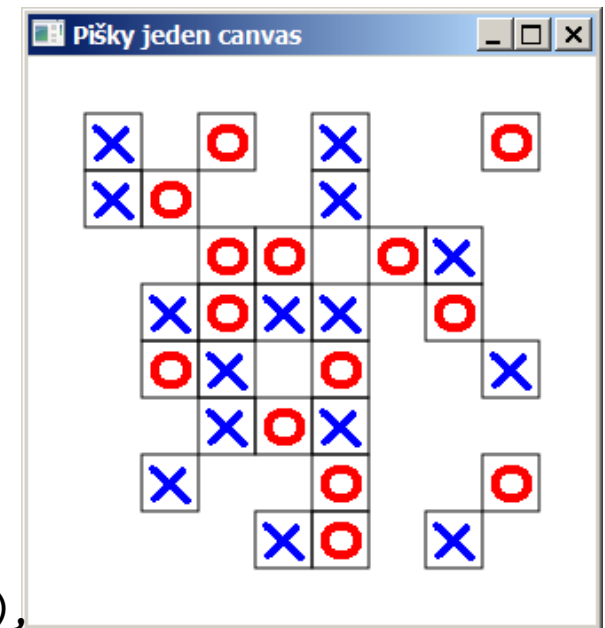
    public PiskyCell(int i, int j) {
        this.i = i; this.j = j;
        setWidth(imageX.getWidth()+2);
        setHeight(imageX.getHeight()+2);
        setOnMouseClicked(event -> {
            if (ps.playground[i][j] != 0) return;
            ps.playground[i][j] = (ps.nextPlayerIsX)?1:-1;
            ps.nextPlayerIsX = !ps.nextPlayerIsX;
            paintCell();
        });
    }

    public void paintCell() {
        GraphicsContext gc = getGraphicsContext2D();
        gc.strokeRect(0, 0, getWidth(), getHeight());
        if (ps.playground[i][j] == 1) gc.drawImage(new Image("x.gif"), 1, 1);
        else if (ps.playground[i][j] == -1) gc.drawImage(new Image("o.gif"), 1,1);
    }
}
```



Riezenie Canvas

```
class Piskyground extends Canvas {
    Image imageX = new Image("o.gif");
    Image imageO = new Image("x.gif");
    double cellSize = 2+Math.max(
        Math.max(imageX.getWidth(), imageO.getWidth()),
        Math.max(imageX.getHeight(), imageO.getHeight()));
    public Piskyground() {
        setWidth(SIZE * (imageX.getWidth() + 2));
        setHeight(SIZE * (imageX.getHeight() + 2));
        setOnMouseClicked(event -> {
            int i = getRow(event.getX());
            int j = getCol(event.getY());
            if (ps.playground[i][j] != 0) return;
            ps.playground[i][j] = (ps.nextPlayerIsX) ? 1 : -1;
            paintCell(i, j);
            ps.nextPlayerIsX = !ps.nextPlayerIsX;
        });
    }
}
```



Riezenie Canvas

```
class Piskyground extends Canvas {  
    public void paintCell(int i, int j) {  
        double px = getPixelX(i);  
        double py = getPixelY(j);  
        GraphicsContext gc = getGraphicsContext2D();  
        gc.strokeRect(px, py, cellSize, cellSize);  
        if (ps.playground[i][j] == 1) gc.drawImage(imageX, px+1, py+1);  
        else if (ps.playground[i][j] == -1) gc.drawImage(imageO, px+1, py+1);  
    }  
    private int getRow(double pixel) { == private int getCol(double pixel) {  
        return (int)(pixel/cellSize);  
    }  
    private double getPixelX(int i) { == private double getPixelY(int i) {  
        return i*cellSize;  
    }  
}
```

