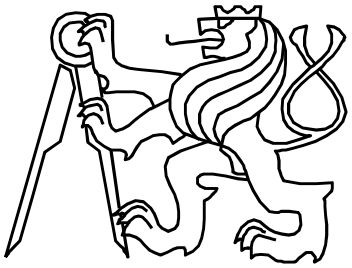
Enterprise Java (BI-EJA)
Technologie programování v jazyku Java (X36TJV)

Ing. Zdeněk Troníček, Ph.D.

Katedra softwarového inženýrství

Fakulta informačních technologií ČVUT v Praze

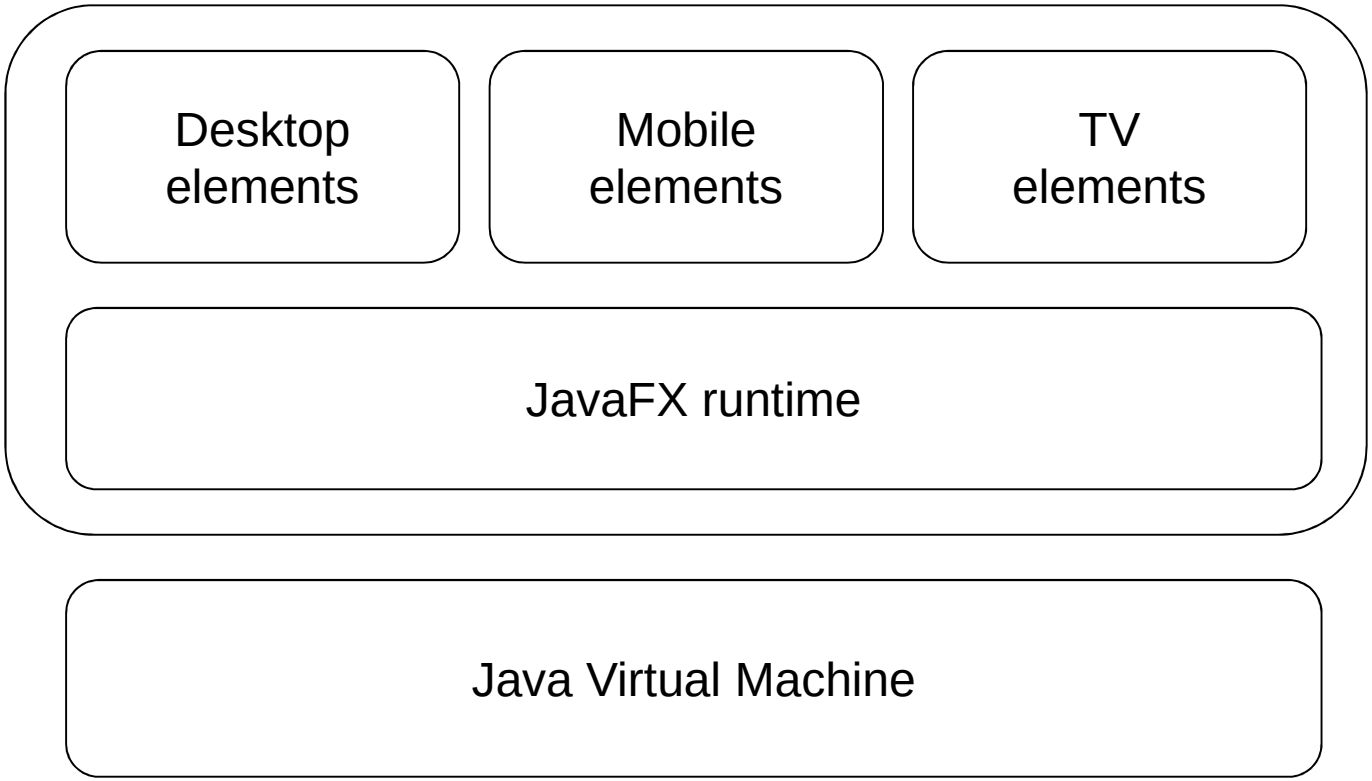


Letní semestr 2010/2011, přednáška č. 8
https://edux.fit.cvut.cz/courses/BI-EJA
https://edux.feld.cvut.cz/courses/X36TJV

JavaFX

- Úvod
- Datové typy
- Funkce
- Třídy, objekty
- GUI, graph model

JavaFX platform



JavaFX script

- Type Inference
- Data Types
- Sequences
- Expressions
- Binding
- Triggers
- Function Types
- Classes
- Mixins
- …

Type Inference

- JavaFX script je staticky typovaný jazyk
- překladač umí odvodit typ proměnné
- každá proměnná je inicializována na defaultní hodnotu

```
var i = 42;  // i je typu Integer
var s : String = "hi";
var j;
j = 3.14; // j je typu Number


def max = 10;
```

String

- lze používat uvozovky i apostrofy
- řetězce mohou obsahovat výrazy v {}

```
var s1 = "one";
var s2 = 'two';
var s3 = "rock 'n' roll";
var s4 = "{s1} and {s2}";      // one and two
var s5 = "three"
         " + "
         "four";               // three + four
```

Duration

```
var d1 : Duration = 100ms;    // 100 milliseconds
var d2 : Duration = 2s;       // 2 seconds
var d3 = 3.5m;                // 3.5 minutes
var d4 = 0.2h;                // 0.2 hours
var d5 = 2m + 11s;


var d6 = d2 – d1;
var d7 = 5 * d3;
var n = d4 / d5;     // n je typu Integer
```

Sequences (1)

```
var s1: Integer[];   // sekvence hodnot typu Integer
var s2: Integer[] = [ 1, 2, 3 ];

var s = [ 3, 5, 7 ];    // type inference
insert 11 into s;     // [ 3, 5, 7, 11 ]
delete 7 from s;    // [ 3, 5, 11 ]

insert 7 after s [ 1 ];            // [ 3, 5, 7, 11 ]
insert 2 before s [ 0 ];         // [ 2, 3, 5, 7, 11 ]
```

Sequences (2)

```
var s1 = [ 1..5 ];              // [ 1, 2, 3, 4, 5 ]
var s2 = [ 1..10 step 2 ];      // [ 1, 3, 5, 7, 9 ]
var s3 = s1 [ 1..3 ];           // [ 2, 3, 4 ]
var s4 = [ s3, 11 ];            // [ 2, 3, 4, 11 ]
var s5 = [ 1 ];
insert s3 into s5;              // [ 1, 2, 3, 4 ]

var s6 = s5 [ n | n >= 3 ];     // [ 3, 4 ]
var s7 = reverse s6;            // [ 4, 3 ]
var n = sizeof s7;              // 2
println( s6 == s7 );            // false

println( s7 [ 0 ] );            // 4
println( s7 [ 2 ] );            // 0 (defaultní hodnota)
```

Expressions (1)

JavaFX is expression language

Block expression

```
var primes = [ 2, 3, 5, 7 ];
var sum = {
    var s = 0;
    for ( p in primes ) {
        s += p
    }
    s
}
```

If expression

```
var y = if ( x > 0 ) {
    ++x;
} else {
    --x;
}
```

Expressions (2)

For expression

```
var s = for ( n in [ 1..5 ] ) {
   n * n
}
println( s );  // [ 1, 4, 9, 16, 25 ]
```

While expression

```
var c = 0;
var d = 1234;
while ( d > 0 ) {
    c++;
    d /= 10;
}
```

While expression
is of type Void

Functions (1)

```
function answer(): Integer {
    return 42;
}
```

```
function answer(): Integer {
    return 42
}
```

```
function answer() {
    return 42;
}
```

```
function answer() {
    return 42
}
```

```
function answer() {
    42;
}
```

```
function answer() {
    42
}
```

Functions (2)

```
function add ( x: Integer, y: Integer ) {
    return x + y
}
```

```
function add ( x, y ) {  // x a y jsou typu Number
    return x + y
}
```

Binding (1)

```
var x = 10;
var y = bind x mod 2;
println ( y );
x--;
println ( y );
```

```
var p = 42;
var q = bind p with inverse;
println ( q );
q = 29;
println ( p );
```

```
function max ( x : Number, y : Number ) : Number {
    if ( x > y ) x else y
}
var i = 3.141593;
var j = 22.0 / 7;
var m = bind max ( i, j );
```

Binding (2)

```
function f ( ) : Double {
    println ( "random" );
    return Math.random ( );
}

var x = 1.23;
var y = bind f ( ) + x;          // funkce f se zavolá pouze 1x
println ( y );
x = 2.34;
println ( y );
```

Bound Function

```
var c = 1.5;
bound function multiply ( u : Number ) {
    return c * u;
}

var w = 2.0;
var v = bind multiply ( w );
c = 1.4;
println ( v );
```

Replace Trigger

```
var t = 1 on replace {
   println ( "změna hodnoty t" );
}
```

```
var t = 1 on replace tt {
   println ( "změna {tt} -> {t}" );
}


t = 2;
println( t );
```

Function Types

```
var f : function ( : Integer ) : Void; // proměnná typu funkce
f = function ( x : Integer ) : Void {
    println( "x = {x}" );
}
f ( 123 );
```

funkce může být

- parametrem funkce
- návratovou hodnotou funkce

Classes

```
class Point {

    var x : Integer;  // instance variables
    var y : Integer;

    // instance function
    function move ( dx : Integer, dy : Integer ) : Void {
        x += dx;
        y += dy;
    }
}
```

Visibility

- default
  (script level)
- package
- protected
- public
- public-read
- public-init
  (public-read + init)

```
class Visibility {
    var p1 : Integer;
    package var p2 : Integer;
    protected var p3 : Integer;
    public var p4 : Integer;
    public-read var p5 : Integer;
    public-init var p6 : Integer;
    protected public-read var p7 : Integer;
    package public-init var p8 : Integer;
}
```

Object Literals

```
var p: Point;
p = Point {          // žádné new ani konstruktor
    x: 1
    y: 2
}
```

```
var p: Point = Point {
    x: 3
    y: 4
}
```

```
var p = Point {
    x: 5
    y: 6
}
```

Inheritance

```
class Parent {
    var x : Integer = 10;
    function printX ( ) : Void {
        println ( "Parent: {x}" );
    }
}
```

```
class Child extends Parent {
    override var x = 20;  // přepíše pouze inicializaci
    override function printX ( ) : Void {
        super.printX ( );
        println ( "Child: {x} {super.x}" );
    }
}
```

Covariant return & contravariant arguments

```
class Parent {
    function makeShape ( ) : Shape { ... }
    function processShape ( r : Rectangle ) : Void { ... }
}

class Child extends Parent {
    override function makeShape ( ) : Rectangle { ... }
    override function processShape ( s : Shape ) : Void { ... }
}
```

Rectangle is a subclass of Shape

Mixins (1)

```
mixin class Sailer {
    function sail() : Void {
        println( "sailing..." );
    }
    abstract function dock(): Void;
}
```

```
class Vehicle {
    var speed = 0;
    function changeSpeed ( speed : Integer ) {
        println( "changing speed..." );
        this.speed = speed;
    }
}
```

Mixins (2)

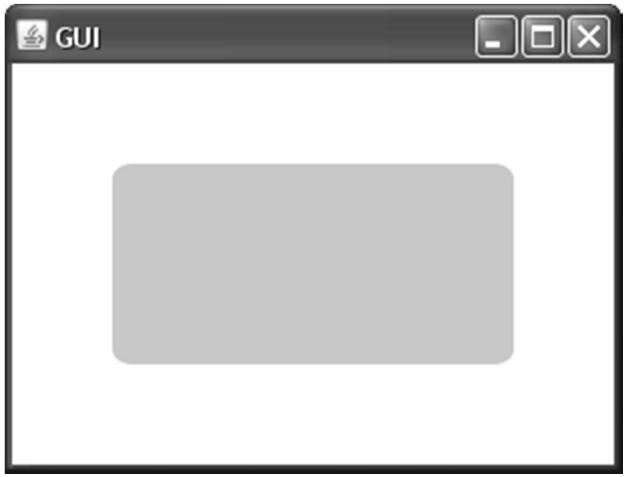
```
class Yacht extends Vehicle, Sailer {


    override function dock() : Void {
        println("docking...");
    }
}
```

Class can inherit from one class and any number of mixins

GUI

```
Stage {
    title: "GUI"
    scene: Scene {
        width: 300
        height: 200
        content: Rectangle {
            x: 50
            y: 50
            width: 200
            height: 100
            …
        }
    }
}
```



Graph Model

- Stage – top level container
- Scene – a drawing surface
- Node – an element in a scene graph
- Group – a sequence of nodes
- CustomNode – designed to be subclassed

Effects

Lighting

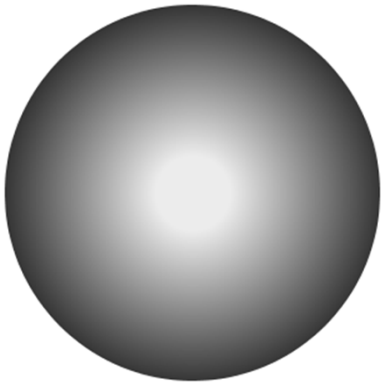
- DistantLight
- PointLight
- SpotLight

Transforms

- Affine
- Rotate
- Scale
- Shear
- Translate

Gradients

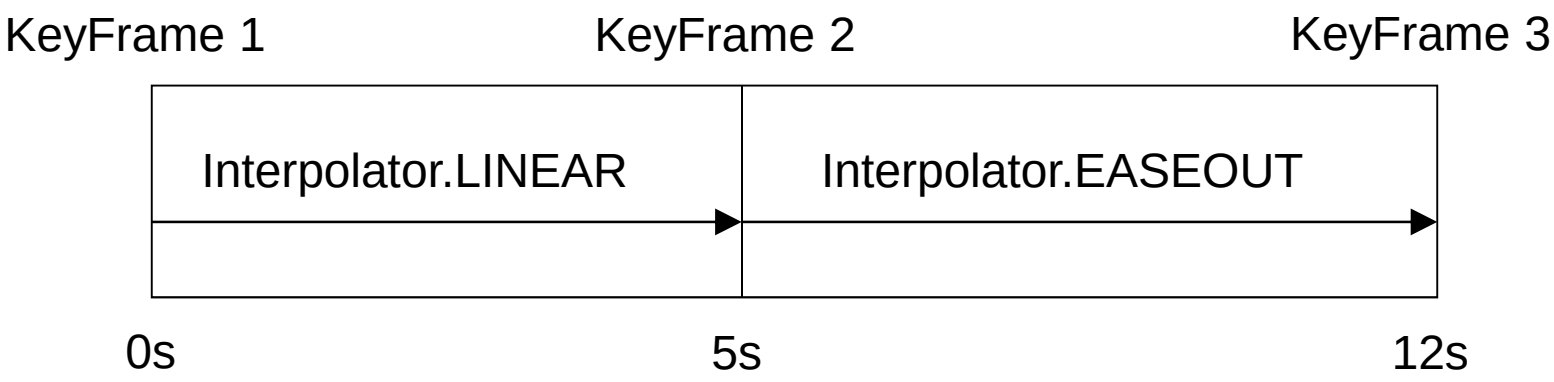
- LinearGradient
- RadialGradient



Demo: effects

- LightingUse.fx
- TransformUse.fx
- LinearGradientUse.fx
- RadialGradientUse.fx

Animation




```
Timeline {
    ...
    keyFrames: [
        at (0s) { x => 0 },
        at (5s) { x => 300 tween Interpolator.LINEAR },
        ...
    ]
}
```

Demo: animation

- AnimationUse.fx

Demo: transitions

- FadeTransition (FadeTransitionUse.fx)
- ParallelTransition
- PathTransition (PathTransition.fx)
- PauseTransition
- RotateTransition (RotateTransitionUse.fx)
- ScaleTransition (ScaleTransitionUse.fx)
- SequentialTransition
- TranslateTransition (TranslateTransitionUse.fx)

Video

```
scene: Scene {
    content: MediaView {
        mediaPlayer: MediaPlayer {
            media: Media {
                source: "file:/…"
            }
        }
    }
}
```

- Media
- MediaPlayer
- MediaView

Otázky & odpovědi

tronicek@fit.cvut.cz