

# JavaFX GUI, hodně zjednodušený úvod

---

*Všechny údaje platí pro JavaFX verze 8.0.1 a SceneBuilder verze 2.0*

# Virtuální PC

Abych usnadnil práci i těm studentům, kteří z různých důvodů buď nemohou, nebo nechtějí instalovat programy, potřebné pro tento předmět, vytvořil jsem virtuální PC.

Protože se jedná o mocný a dosti spolehlivý nástroj, doporučuji jeho použití všem studentům. Nezanedbatelnou výhodou je, že jeho používáním vznikne jednotné prostředí, úplně stejné jak pro všechny studenty, tak pro vyučujícího, což zjednoduší ladění programů a případné hledání chyb.

Virtuální PC je vytvořeno v prostředí Oracle VM VirtualBox  
4.3.12.

# První zapnutí

## Stažení VirtualBox

Nejnovější verzi VirtualBoxu si stáhněte z webu <https://www.virtualbox.org/wiki/Downloads>. Instalace probíhá podle pokynů instalačního programu. Ještě existuje přenosná verze Portable VirtualBox, která se dá nainstalovat na USB paměť.

Jak spolehlivě funguje a jestli je plně kompatibilní, jsem netestoval. Jeho použití přichází v úvahu na počítači, u kterého nemáte dostatečná práva k tomu, abyste na něj mohli něco instalovat.

Všechny verze VirtualBox jsou pro osobní použití licencovány licencí [GNU](#).

Čerstvě nainstalovaný VirtualBox je úplně prázdný, bez jakýchkoliv programů a dokonce i bez operačního systému. Aby se dal používat, je potřeba do něj nejprve nainstalovat virtuální počítač. To se dá udělat více způsoby. Já vám doporučuji, stáhnout si a nainstalovat **appliance**.

## Appliance

Nainstalovaný a nastavený VirtualBox jsem pro vás připravil ve formě tzv. appliance. Appliance je binární obraz počítače přizpůsobený tak, aby ho šlo nainstalovat na jakýkoliv počítač.

Naše appliance je soubor **Tutorial\_2014-15.ova** a můžete si ho stáhnout na stránkách předmětu.

Jeho velikost je asi 7GB a obsahuje Windows 7 ve 32-bitové verzi.

V čerstvě nainstalovaném VirtualBoxu zadejte Import Appliance. Tím se spustí instalační program.

Na běžném počítači instalace trvá asi 10 minut.

Zatímco probíhá instalace, můžete se seznámit s licenčními podmínkami. Z nich nejdůležitější je, že Windows na virtuálním počítači nejsou licencované, proto je potřeba aktivovat licenci.

Škola má pro výuku multilicenci, postup aktivace najdete na školním webu.

Další programy, zejména NetBeans a Scene Builder, jsou pro uživatele **bezplatné**.

Programy MS Project a Enterprise Architect jsou nainstalované jako **trial** (pokusné) verze na omezený počet dní S tím je potřeba kalkulovat, až si budete plánovat čas na studium.

První věc, kterou musíte udělat po nainstalování appliance je, upravit velikost paměti. Defaultně je pro virtualbox přiděleno jen 512MB paměti. To sice postačuje ke spuštění, ale virtuální počítač pak je zoufale pomalý. Doporučuji, nastavit aspoň polovinu fyzické paměti.

Nastavení se provede přes Nastavení → Systém → Operační paměť, viz [obrázek](#).

## Spuštění a ukončení

Virtuální počítač spustíte tlačítkem [zobrazit](#).

Vypnutí je překvapivě trochu komplikovanější. Pokud virtuální počítač vypnete běžným způsobem, uloží se celý jeho stav a při příštím zapnutí se zase obnoví.

To nemusí být z různých důvodů výhodné. Například, pokud si omylem vymažeme nějaké soubory nebo pokud něco v systému špatně nastavíme, je výhodné, vrátit se před tuto změnu.

Virtuální počítač to řeší pomocí **snímků** ([snapshot](#)).

Pomocí tlačítka Snímky (červený ovál) se přepneme do zobrazení snímků a potom tlačítkem „fotoaparát“ (modrý ovál) můžeme zaznamenávat snímky. K zaznamenaným snímkům se lze kdykoliv vrátit.

Ve zvláštních situacích je vhodné vůbec neukládat změny stavu.

Pokud nechceme, aby se změny uložily, prostě násilně [ukončíme](#) [běh](#) virtuálního počítače. Při příštím spuštění se normálně nashoduje naposledy uložený stav. Není potřeba se obávat „nashodění“ windows.



# NetBeans

## Překladač online

Jsou případy, kdy si vystačíte s pouhým textovým rozhraním.

Například při výuce existuje řada příkladů, u kterých vůbec nepotřebujete pracovat s grafikou, kdy si úplně vystačíte s textovým vstupem a výstupem. V takových případech můžete pracovat s některým on-line kompilátorem. Na internetu jich najdete celou řadu.

Dobrý a vyzkoušený on-line kompilátor jazyka JAVA naleznete na <http://www.compileonline.com/>.

Okno kompilátoru ukazuje [obrázek](#).

K tomu, abyste ho mohli správně používat, potřebujete vědět, kam se zadávají hodnoty.

Pokud nějaký program spouštíte s argumenty, musíte parametry programu zadat do políčka „Command Line Arguments“, viz černá šipka. Většinou tuto možnost nevyužijeme.

Pokud chcete do programu předat nějaká data z klávesnice, musíte je předem zapsat do políčka pro standardní vstup, „STDIN Input“. Zvýrazňuje ho zelená šipka.

Na horním okraji formuláře jsou tři ouška. To první patří editačnímu oknu, do kterého se zapisuje hlavní třída (žlutá šipka).

Vedle je ouško „input.txt“ (červená šipka). Bohužel je špatně vidět, překrývá ho prvek pro volbu editoru. Na formuláři „input.txt“ zadáváme texty, které si program má za běhu načíst ze souboru „input.txt“. Jinými slovy, tento formulář emuluje diskový soubor „input.txt“.

Pokud bychom potřebovali pracovat s více soubory, musíme zaškrtnout volbu „Multiple Files“ (modrý ovál) a tím získáme možnost, pracovat s více soubory.

Program přeložíte a spustíte [tlačítkem](#) „Compile and Run“

# NetBeans v textovém režimu

Použití NetBeans v grafickém režimu si ukážeme později. Nyní si nejdřív ukážeme, jak s NetBeans pracovat v klasickém textovém režimu. Textovým režimem zde míním, že se kompletní programový kód zapíše z klávesnice, v textové podobě. To je způsob, jakým pracuje většina programátorů.

## První kroky

Nový projekt zahájíme přes File → New Project.

Prozatím budeme pracovat v textovém režimu, a tak zvolíme **Java** a **Java Application**. Otevře se [dialog](#).

**Důležité:** Dialog, který se otevře, není dobré jen tak bezmyšlenkovitě odklikat. Musíme zadat správný název [projektu](#), **Project Name**. Pokud ho v tomto formuláři zadáme špatně, i adresáře a názvy souborů se vytvoří špatně a potom dá dost práce všechno uvést do pořádku.

Nezapomeňte, že název třídy se musí psát s velkým počátečním písmenem. Proto i název souboru s touto třídou musí mít velké počáteční písmeno (pro případ, že bychom náš projekt portovali do operačního systému, který rozlišuje malá a velká písmena).

Z téhož důvodu i název projektu by měl začínat velkým písmenem.

Na levé straně [formuláře](#), v okně Projects, vidíme strukturu projektu (černá šipka). Zatím tam je jen soubor s naším novým projektem a v něm je soubor *JavaApplication1.java*.

Příponu **.java** mají soubory se zdrojovým kódem.

Do pravé poloviny formuláře se píše zdrojový kód. Připraveno je místo, do kterého se zapisuje dokumentace k programu (zelená šipka). Java je konstruována tak, že komentáře a dokumentaci k programu je možné (resp. nutné) zapisovat přímo do zdrojového kódu.

Programová dokumentace se potom vygeneruje automaticky pomocí programu **JavaDoc**.

Místo, kam budeme psát kód programu, obsahuje řádku `//TODO code application logic here` a je označeno modrou šipkou.

Jakmile máme kód napsán, zahájíme jeho kompilaci a spuštění.

Dělá se to pomocí tlačítka „šipka“, které je na [obrázku](#) zvýrazněno červeným oválem.

## Hello World

Stalo se zvykem, že první program v jakémkoliv jazyce se demonstruje na jednoduché aplikaci „Hello World!“, která nedělá nic jiného, než že vypíše pozdrav.

I my se s NetBeans seznámíme na jednoduchém příkladu „Hello World“, který vypíše pozdrav.

Přestože ještě neznáme syntaxi programovacího jazyka, přesto si krátký program napíšeme, abychom si ukázali, jak NetBeans funguje.

V podokně zdrojového textu vymažte řádku `//TODO code application logic here` a namísto ní napište `// Demo program`. Povšimněte si, že řádka musí začínat dvojitým lomítkem, protože to je komentář a komentáře začínají dvojitým lomítkem (... i jinak, jak uvidíme později).



Potom napíšeme první řádku programu. Budeme chtít napsat `System.out.println("Hello World");` což bychom dokázali napsat i rovnou, ale my si to předvedeme pomalu a podrobněji, aby vyniknul postup.

Začneme tedy psát `System.` a jakmile zapíšeme tečku, objeví se pomocná okna, která se vztahují k právě napsanému slovu (v tomto případě ke slovu `System`). Ukazuje to [obrázek](#).

Spodní okno s modře zvýrazněnou řádkou nám nabízí všechny možnosti, které v tomto místě smíme napsat. Můžeme buď napsat „out“ rovnou, nebo na „out“ můžeme najet zvýrazněním, nebo konečně začneme psát třeba „o“ a postupně, jak píšeme,

výběr se postupně zužuje. Nakonec máme celé „out“ napsané. Zapišeme tečku a celý postup opakujeme pro zbytek řádky.

Pokud jde o horní okno, to zobrazuje jednoduchou on-line nápovědu.

Pokud při psaní uděláme chybu, NetBeans ji ohlásí červeným kroužkem na začátku řádky.

Na příkladu jsme úmyslně vymazali zakončovací uvozovku a systém ohlásil chybu.

### *Důležité*

**Zapamatujte si, že NetBeans hlásí chybu až tam, kde ji objeví. To znamená, že chyba může ležet i o několik řádků dříve!**

Celý program máme k dispozici v [souboru](#), ze kterého jej můžeme načíst a do NetBeans nakopírovat. Program zkompilujeme a spustíme.

Program přeložíme a spustíme kliknutím na zelený symbol ► (na [obrázku](#) je označena červeným oválem) anebo klávesou F6.

## **Vstup/výstup do souboru**

Souborový vstup/výstup je mnohem komplikovanější, protože musíme udělat složitější přípravy než v případě klávesnice. Důležité části jsou vyznačeny tučně.

Čtení ze souboru „a.txt“:

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
...
public static void main(String[] args)
throws IOException {
    FileReader fr = new
FileReader("a.txt");
    BufferedReader orig = new
BufferedReader(fr);
    String s;
...
    s = orig.readLine();
```

Zápis do souboru „b.txt“:

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
...
public static void main(String[] args)
throws IOException {
    FileWriter fw = new
FileWriter("b.txt");
    BufferedWriter copy = new
BufferedWriter(fw);
```

```
String s;  
...  
    copy.write(s) ;  
    copy.newLine() ;  
...  
    copy.close() ;  
    fw.close() ;  
    fw.close() ;
```

## Příklad

Fungující příklad pro NetBeans najdete v souboru <..\files\03-20FileIO.zip>

## Konstruktory pro Scanner

Podobně jako při vstupu z klávesnice, také pro vstup z textového souboru se dá použít třída Scanner. Poslouží v případech, kdy potřebujeme načíst více údajů z jedné řádky. Třída Scanner má velké množství konstruktorů, z nich použijeme jen ty [nejdůležitější](#).

## Důležité metody třídy Scanner

```
public String findInLine(String string)
```

Najde následující výskyt stringu. Pozor, ignoruje oddělovače!

```
public boolean hasNext()
```

Vrací true, pokud ještě něco následuje.

```
public boolean hasNextInt()
```

Vrací true, pokud ještě následuje celé číslo.

```
public boolean hasNextLine()
```

Vrací true, pokud následuje řádka. Lze použít k čekání na vstup (třeba z klávesnice).

```
public String next()
```

Vrací následující element.

```
public String next(String string)
```

Vrací následující element, pokud odpovídá vzoru (popsaný regulárním výrazem).

```
public int nextInt()
```

```
public double nextDouble()
```

Vracejí následující číslo daného typu.



```
public String nextLine()
```

Vrací celý nepřečtený zbytek řádku a přejde na další řádek.

## Příklad vstupu int + String

```
int i;  
String s;  
Scanner scanner = new Scanner( new  
    File(fileName), "CP1250" );  
while (scanner.hasNextInt()) {  
    i = scanner.nextInt();  
    s = scanner.nextLine();  
    . . .  
}
```

# JavaFX grafické rozhraní

## JavaFX

JavaFX je moderní framework pro tvorbu bohatých okenních aplikací. Bohatých je zde myšleno vizuálně. JavaFX přináší podporu obrázků, videa, hudby, grafů, CSS stylů a dalších technologií, které zajistí, že výsledná aplikace je opravdu líbivá. Zároveň je kladen důraz na jednoduchost tvorby, všechny zmiňované věci jsou v JavaFX v základu. JavaFX se hodí jak pro desktopové aplikace, tak pro webové applety nebo mobilní aplikace.

# Tři cesty ke grafickému rozhraní

V JavaFX je možné vyvíjet podobně jako ve starším Swingu, tedy tak, že tvoříte instance jednotlivých formulářových prvků (tlačítko, textové pole). Ty poté vkládáte do tzv. *layoutů*, což jsou vlastně kontejnery na formulářové komponenty.

Druhým způsobem je FXML. FXML je jazyk pro návrh formulářů. Asi vás podle názvu nepřekvapí, že je to další jazyk odvozený z XML. Použití XML pro návrh prezentační části aplikace (to je ta část, se kterou komunikuje uživatel) není nic nového, naopak se jedná o osvědčený princip z webových aplikací. Java zde přenáší principy HTML a CSS do desktopových aplikací.

A konečně třetí možností je, tvořit grafický interface přímo v grafice. Tento „klikací“ způsob tvorby je bezkonkurenčně nejnázornější a také nejjednodušší. Tuto třetí možnost umožnil **Java Scene Builder**.

Základní „životní funkce“ aplikace jsou již integrovány do grafického integrovaného prostředí, a proto je programátor nepotřebuje programovat. Jediné, co musí programovat, je obsluha **událostí**, které vznikají za běhu aplikace. Takovou událostí, například, může být stisknutí tlačítka. Programátor tedy píše obsluhy (*handlers*) k obsluze jednotlivých událostí. Tomuto způsobu programování se říká **events-driven**, řízený událostmi.

# Java Scene Builder

Java Scene Builder není přímo v NetBeans, ale jedná se o samostatnou aplikaci. Pod názvem JavaFX Scene Builder stáhnete ze serveru [www.oracle.com](http://www.oracle.com).

**Java Scene Builder už je předinstalovaný ve výukovém virtuálním počítači.**

Přestože zatím nevíme nic o jazyku Java, můžeme si už teď ukázat základy práce se Scene Builderem, na které znalosti jazyka nebudeme potřebovat.

# Seznámení

## Nový projekt

Nejprve si v NetBeans vytvoříme projekt. Projekt nazýváme místo (obvykle adresář), ve kterém se shromažďují veškeré informace, soubory, nastavení i přeložené programy vztahující se k řešenému problému.

Otevřete NetBeans a zvolte **Soubor-> Nový projekt**, vyberte **JavaFX** a pak zvolte **JavaFX FXML aplikace (!)**, viz [obrázek](#). Tím jsme vytvořili velmi jednoduchou aplikaci v JavaFX, která obsahuje hlavní třídu aplikace, třídu controller poskytující aktuální podkladovou logiku pro okna definované v Scene

Builder a soubor FX Markup Language (FXML), který obsahuje XML kód pro naše definice okna. Žádnému z těchto souborů nepotřebujete rozumět a nechcete-li, nemusíte je ani číst.

Na následující stránce **je nutné** vyplnit [název projektu](#). Pokud zapomenete zadat jeho jméno, bude použito defaultní - ale dá dost velkou práci ho později přejmenovat. Takže je lepší použít správný název rovnou.

V levém horním rohu obrazovky NetBeans, v záložce [Projects](#), máme tři soubory:



- **EasyProject.java** obsahuje hlavní třídu aplikace. V našem příkladu s touto třídou nebudeme nic dělat, protože jejím hlavním úkolem je načíst definiční kód obsažený v souboru

FXML a zobrazit hlavní scénu (*stage*). JavaFX lépe pochopíte, když si ho přirovnáte k divadlu: platforma JavaFX je jeviště, na kterém se střídají scény.

- **FXMLDocumentController.java** je řídicí třída, která poskytuje řídicí logiku, "mozek" aplikace, a to s použitím grafického rozhraní. Pravým tlačítkem a Edit si otevřete SampleController a uvidíte, že obsahuje vlastnosti a metody, označené přívlastkem (**tagem**) @FXML. Tento tag umožňuje integraci vizuálních ovládacích prvků, které definujete pomocí Scene Builder s těmi které jsou uloženy v souboru FXML. Princip si ukážeme dále.
- **FXMLDocument.fxml** je soubor definující vzhled a obsah okna.



Hned můžeme projekt přeložit a spustit. Podle starého dobrého zvyku v něm totiž je předpřipraven program „[Hello World!](#)“ čili jednoduchý pozdrav.

Program přeložíme a spustíme kliknutím na zelený symbol  (na [obrázku](#) je označena červeným oválem) anebo klávesou F6.<EN>We can compile and run the program clicking the green symbol  (it is highlighted by red oval in [figure](#)).

Klikněme pravým tlačítkem na soubor [FXMLDocument.fxml](#) a vyberte **Open**. Tím se automaticky otevře Scene Builder – aplikace určená pro grafickou editaci.

Myší uchopte tlačítko a odtáhněte jej do jiného místa, aby bylo vidět, jak se pohybuje. Načež **scénu uložíme (Ctrl+S)**, přepneme se do NetBeans a zkusíme program znovu spustit.

### **Problém! Pozice tlačítka se vůbec nezměnila!**

Vysvětlení je prosté. NetBeans a SceneBuilder nejsou propojeny přímo, ale nepřímo, přes soubor FXML. Když jsme uložili scénu ve SceneBuilder, nová poloha se zapsala do souboru FXML, ale NetBeans o tom ještě nic neví.

**Změny v souboru FXML musíme načíst.** Uděláme to tak, že klikneme pravým tlačítkem na *FXMLDocument.fxml* a zvolíme [Edit](#). Mimochodem, tím se také otevře editační okno, ve kterém bychom mohli soubor editovat ručně...kdybychom to uměli.

Nyní znovu spustíme program a všechno už je OK.

## **Atributy čili vlastnosti**

Znovu se přepneme do SceneBuilder a vybereme Button.

V pravé části obrazovky vidíme všechny vlastnosti, které Button má. Tak například vlastnost Text obsahuje nadpis, který se má na tlačítku zobrazit- Zkuste ho změnit a ověřte si, že se nadpis změnil i v programu.

Úplně analogicky: scéna obsahuje nadpis Label. Na obrázku ho nevidíme, protože je prázdný, ale vidíme ho v navigačním panelu vpravo. Když ho tam vybereme, uvidíme ho i na scéně. Pokud změníme jeho vlastnost Rotate na 45, nadpis se pootočí o 45

stupňů. Vlastnost Rotate, stejně jako všechny vlastnosti, které se týkají vzhledu, najdete na záložce Layout.

## Editace

Poslední příklad se týká přidávání objektů do scény.

Na panelu „Library“ na levé straně vyberte záložku Controls. Najděte CheckBox a přetáhněte ho na scénu. Pak program obvyklým způsobem spusťte a uvidíte, že zaškrťovací políčko je funkční. Jen nám to je k ničemu, protože neumíme napsat kód, který by ho použil. Mimochodem, v souboru FXML přibyl jakýsi CheckBox 😊.

## Pro kaskadéry

Kdo chce, může ještě pokračovat dál.

Přepněte se na ouško *FXMLDocumentController.java*. Jsou tam řádky

```
private void handleButtonAction(ActionEvent  
event) {  
    System.out.println("You clicked me!");  
    label.setText("Hello World!");  
}
```

`handleButtonAction` říká, že tento kód se provede, když aktivujeme tlačítko Button (*handle* = obsloužit).

Vidíme, že tam je `System.out.println` a to už známe, po stisknutí tlačítka se na výstup vypíše text. Také tam je řádka

```
label.setText("Hello World!");
```

Můžeme zkusit jí porozumět. Říká, že v nadpisu `label` máme nastavit text na hodnotu „Hello world!“.

Zkusíme napsat svůj první příkaz. Přidáme řádku

```
label.setRotate(label.getRotate() + 45);
```

Její význam je ten, že vezmeme současné natočení nadpisu

```
label.getRotate()
```

, přičteme k němu 45 stupňů a

výsledek zase vložíme do nadpisu `label.setRotate(...)` .

Projekt naleznete v souboru [../files/03-43%20SimpleProgram.zip](#).

# Cvičení

---

Otevřete NetBeans a vytvořte jednoduchou aplikaci, která vypíše dva stejné řádky „Hello World!“.

Demonstrujte krokování.

Demonstrujte run-time chybu.

Tutéž aplikaci přeložte a spusťte v *on-line* kompilátoru.

Ve *SceneBuilder* vytvořte tuto [scénu](#) (semafor). Přeložte ji, aby bylo jasné, že je bez chyb.



Napište krátký program, který postupně načte řádky z klávesnice a vypíše je na obrazovku, každý řádek 2x.

Napište krátký program, který postupně načte řádky ze [souboru](#) a vypíše je na obrazovku.

Napište krátký program, který postupně načte řádky z klávesnice a zapíše je do souboru TEST.TXT.



# Obsah

---

Virtuální PC.....	2
První zapnutí.....	3
Stažení VirtualBox .....	3
Appliance.....	4
Spuštění a ukončení .....	7
NetBeans .....	9
Překladač online .....	9
NetBeans v textovém režimu .....	12
První kroky.....	12

Hello World ..... 15

Ladění ...**Chyba! Záložka není definována.**

Spuštění aplikace v JAVA**Chyba! Záložka není definována.**

NetBeans **Chyba! Záložka není definována.**

Nainstalované běhové prostředí**Chyba! Záložka není definována.**

Počítač bez Javy**Chyba! Záložka není definována.**

Základy vstupu a výstupu**Chyba! Záložka není definována.**

Důležité ..**Chyba! Záložka není definována.**

Vstup/výstup na konzoli**Chyba! Záložka není definována.**

Schéma pro výpis na obrazovku:**Chyba! Záložka není definována.**

Schéma pro vstup z klávesnice:**Chyba! Záložka není definována.**

Příklad...**Chyba! Záložka není definována.**

Vstup/výstup do souboru.....	19
Příklad.....	22
Konstruktory pro Scanner .....	23
Důležité metody třídy Scanner.....	23
Příklad vstupu int + String .....	25
JavaFX grafické rozhraní.....	26

JavaFX.....	26
Tři cesty ke grafickému rozhraní .....	27
Java Scene Builder .....	29
Seznámení .....	30
Nový projekt .....	30
Atributy čili vlastnosti.....	35
Editace .....	36
Pro kaskadéry .....	37