Jazyk Java 1

Seminář 1

Daniel Bazala

Katedra informatiky
Univerzita Palackeho v Olomouci

Organizační informace

Organizační informace

- email: daniel.bazala01@upol.cz
- konzultace
  - na hodinách
  - po individuální domluvě emailem
- www: http://apollo.inf.upol.cz/~urbanec/teaching/2021/jj1/java-1.html

Zápočet

- Pro získání zápočtu potřebujete splnit alespoň 7 z 10 pravidelných úkolů a odevzdat semestrální projekt splňující požadavky.

Pravidelné úkoly

- Pravidelné úkoly budou zadávány na seminářích.
- Pvní úkol na semináři 2, poslední na semináři 12.
- Na každý úkol budete mít čas do dalšího semináře (tj. týden)
- K odevzdávání úkolu budeme používat GitHub classrooms (automatické testy).
- U každého úkolu můžete odevzdat libovolný počet verzí, v potaz bude brána jen ta poslední.
- Po vypršení času budou všechny odevzdané úkoly zkontrolovány ručně a bude poskytnuta zpětná vazba. Případné nedodělky či chyby budete moci opravit do dalšího semináře. Po tomto termínu (dva týdny po zadání) už bude úkol uzavřen a nebude jej možné dále plnit.
- O splnění úkolu či výhradách budete informováni přes GitHub classrooms.
- Více k použití GitHub classrooms na druhém semináři.

Semestrální projekt

- Semestrální projekt by měla být netriviální (posuzuje vyučující) aplikace nebo knihovna (s ukázkou použití) pokrývající učivo probrané během semestru.
- V ideálním případě by výsledkem měla být skutečně použitelná aplikace nebo knihovna.
- Projekt musí být odevzdán nejpozději v čase semináře v zápočtovém týdnu.

Plagiátorství

- Z webu katedry:
  "Pokud se student dopustí plagiátorství, opisování při písemném testu, opisování při práci na domácím úkolu nebo se jiným způsobem pokusí o podvod, zahájí s ním vedoucí katedry kárné řízení. Pokud se takové jednání studenta opakuje, vedoucí katedry navrhne děkanovi fakulty vyloučit studenta ze studia."
- Všechny úkoly budou mimo automatických testů kontrolovány i vyučujícím a MOSSem. Pokud bude odhalena příliš velká shoda, budou všichni studenti, kterých se to týká, nahlášeni vedení katedry. Za daný úkol navíc nebudou moci získat žádné body.

Platforma Java

Java jako programovací jazyk

- objektově orientovaný jazyk (dědičnost s jedním rodičem)
- syntaxí podobný C/C++
- portabilita (Write-Once-Run-Anywhere)
- snaha o omezení chyb
- automatická správa paměti (GC)
- silně a staticky typový jazyk
- omezení dostupných prvků jazyka na ty bezpečné
- systém vyjímek
- konvence vynucené standardem

Další vlastnosti

- několik nezávislých implementací (OpenJDK, Oracle, IBM, IcedTea, GCJ, Harmony)
- rozsáhlá standardní knihovna, řada API
- množství dostupných knihoven a nástrojů třetích stran (Apache Foundation)

Java jako platforma

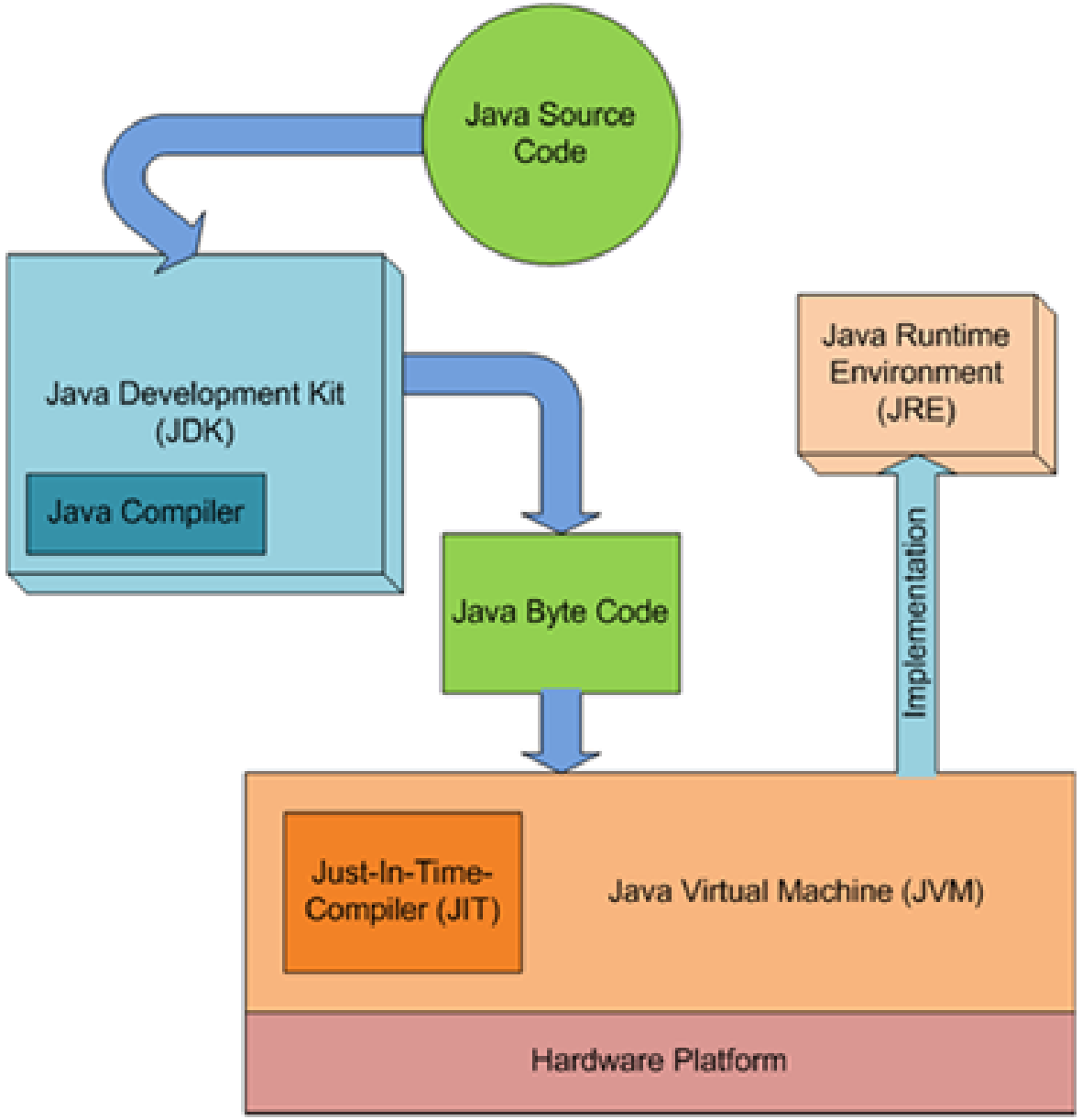
Infrastruktura

- Java Development Kit (JDK): překladač, knihovny + související nástroje
- Java Virtual Machine (JVM): vykonává kód vygenerovaný překladačem (Java Byte Code)
- Java Runtime Environment (JRE): běhové prostředí (JVM, knihovny, atd.)
- Java Applety – propojení JRE s prohlížečem
- s JDK/JRE: lze provozovat další jazyky, např. Scala, Groovy, Clojure, Jython, JRuby

Edice

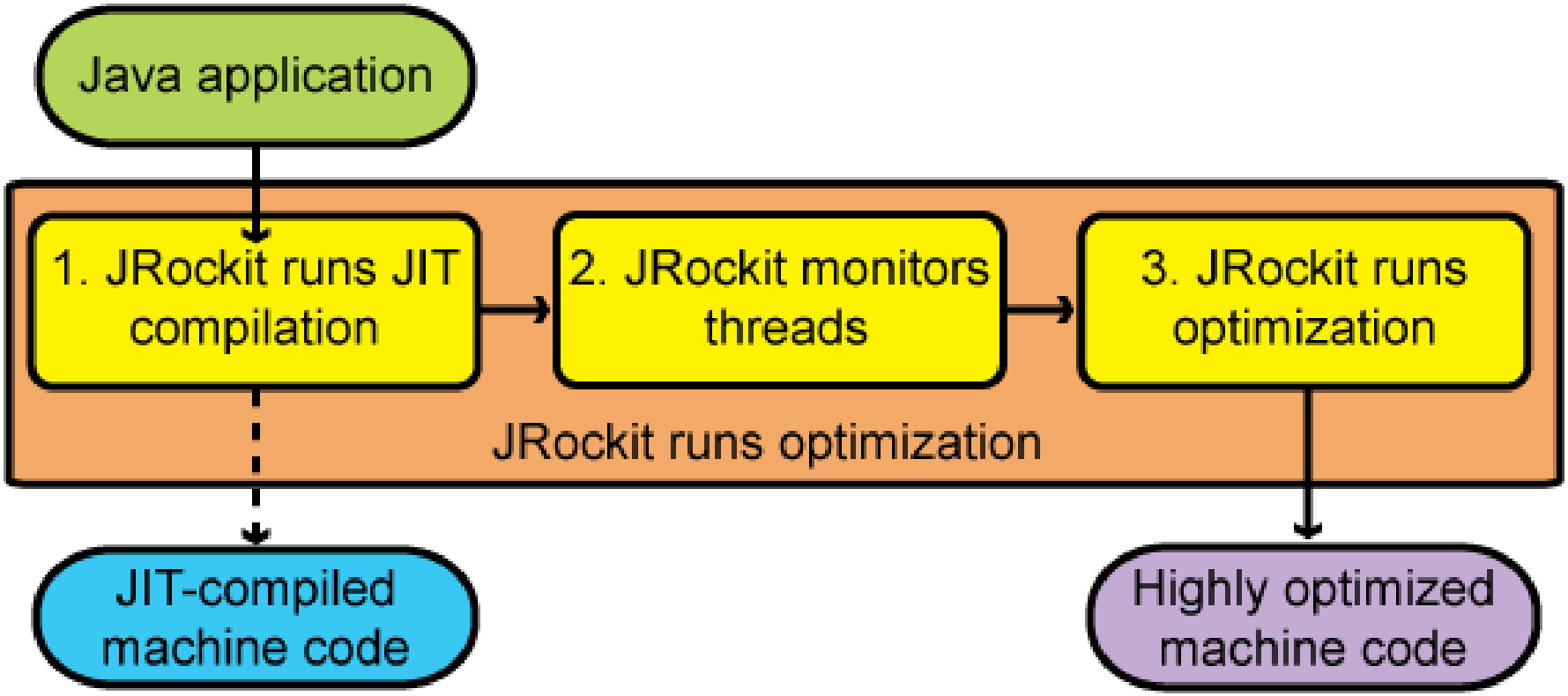
- Java ME – mobilní zařízení (definuje řadu omezení)
- Java SE – nejčastěji používaná
- Java EE – jako SE + knihovny a rozhraní

Java jako platforma



Java jako platforma

HotSpot compiler, JIT



Historický přehled

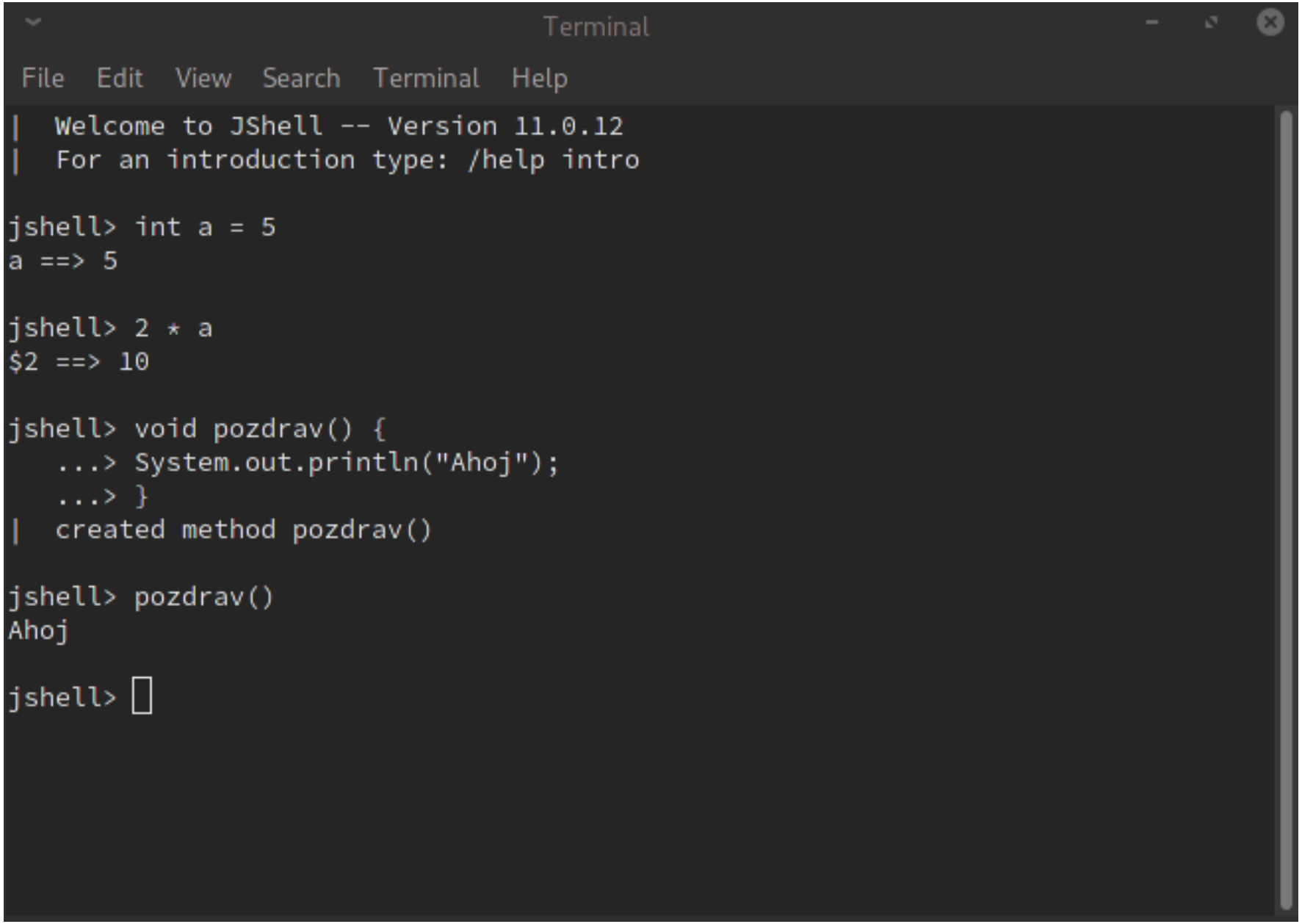
- 1990 – původně vyvíjeno SUN Microsystems jako projekt OAK
- 1996 – JDK 1.0 (první verze)
- 1997 – JDK 1.1
- 1998 – J2SE 1.2 (JIT překlad, kolekce) – Java2 (J2SE)
- 2000 – J2SE 1.3 (HotSpot JIT překladač)
- 2002 – J2SE 1.4 (řada nových API)
- 2005 – J2SE 5.0 (řada rozšíření – generické typy, metadata, for-each)
- 2006 – Java SE 6 (nové API, vylepšení JVM)
- 2010 – SUN je koupen Oraclem
- 2011 – Java SE 7 (mírné úpravy jazyka, rozšířené API)
- 2014 – Java SE 8 (lambda, Stream API)
- 2017 – Java SE 9
- 2018 – Java SE 10, 11
- 2019 – Java SE 12, 13
- 2020 – Java SE 14, 15
- 2021 – Java SE 16, 17

JShell

JShell

- Java REPL konzole
- Poprvé součástí JDK 9
- Nástroj pro rychlé testování kódu
- Dokumentace: https://docs.oracle.com/javase/9/jshell/JSHEL.pdf

JShell


Terminal
File Edit View Search Terminal Help
| Welcome to JShell -- Version 11.0.12
| For an introduction type: /help intro
jshell> int a = 5
a ==> 5
jshell> 2 * a
$2 ==> 10
jshell> void pozdrav() {
...> System.out.println("Ahoj");
...> }
| created method pozdrav()
jshell> pozdrav()
Ahoj
jshell>

Základy JShell

- Spouštění JShel (verbose)

```
% jshell -v
| Welcome to JShell -- Version 9
| For an introduction type: /help intro
```

- Deklarace/definice proměnných

```
jshell> int x = 45
x ==> 45
| created variable x : int
```

- Výrazy

```
jshell> 2 + 2
$3 ==> 4
| created scratch variable $3 : int
```

Základy JShell

- Funkce

```
jshell> int add1(int n) {
 ...> return n + 1;
 ...> }
| created method add1(int)

jshell> add1(2)
$5 ==> 3
```

- Předefinovaání

```
jshell> int add1(int n) {
 ...> return n - 1;
 ...> }
|  modified method add1(int)

jshell> add1(2)
$7 ==> 1
```

Základy JShell

- Budoucí reference

```
jshell> double volume(double radius) {
 ...> return 4.0 / 3.0 * PI * cube(radius);
 ...> }
| created method volume(double), however, it cannot
be invoked until variable PI,and method cube(double)
are declared

jshell> double PI = 3.1415926535
PI ==> 3.1415926535
| created variable PI : double

jshell> volume(2)
| attempted to call method volume(double)
which cannot be invoked until method
cube(double) is declared
```

Základy JShell

- Budoucí reference

```
jshell> double cube(double x) { return x * x * x; }
|  created method cube(double)
|  update modified method volume(double)

jshell> volume(2)
$5 ==> 33.51032163733334
|  created scratch variable $5 : double
```

Základy JShell

- Příkazy

```
jshell> /vars
|     int a = 5
|     int $8 = 3

jshell> /methods
|     int test()
|     int add1(int)
|       which cannot be invoked until variable l is declared

jshell> /list
1 : int add1(int n) {
        return n + 1;
        }
2 : add1(1)
3 : ...

jshell> /help
...
```

Základní konstrukce jazyka

Větvení programu

- If
  - Blok:

```
{
  prikaz1
  prikaz2
  ...
}
```

  - Syntaxe:

```
if (podminka) prikaz

if (podminka) prikaz1
else prikaz2

if (podminka1) prikaz1
else if (podminka2) prikaz2
else prikaz3
```

Větvení programu

- Příklad

```
int a = 42;
if (a < 42) {
  System.out.println("Promenna 'a' je mensi nez 42.");
} else if (a == 42) {
  System.out.println("Promenna 'a' obsahuje tajemstvi vesmiru!");
} else {
  System.out.println("Promenna 'a' je vetsi nez 42.");
}
```

Větvení programu

- Switch

```
switch (vyraz) {
  case konstanta1: prikaz_11 ... prikaz_1m break;
  ...
  case konstantan: prikaz_n1 ... prikaz_no break;
  default: default_prikazy
}
```

- Ternární operátor

```
(vyraz) ? hodnota1 : hodnota2
```

Větvení programu

- Příklady

```
int month = 8;
String monthString;
switch (month) {
  case 1: monthString = "January"; break;
  case 2: monthString = "February"; break;
  case 3: monthString = "March"; break;
  case 4: monthString = "April"; break;
  case 5: monthString = "May"; break;
  case 6: monthString = "June"; break;
  case 7: monthString = "July"; break;
  case 8: monthString = "August"; break;
  case 9: monthString = "September"; break;
  //...
  case 12: monthString = "December"; break;
  default: monthString = "Invalid month"; break;
}
System.out.println(monthString);

b = (a == 42) ? 10 : a + 1;
```

Cykly

- For
  - Syntaxe:

```
for (inicializace; podminka; inkrementace) {
  prikaz1
  ...
  prikazn
}
```

  - Příklad 1:

```
int a = 5;
for (int i = 0; i <= 5; i++) {
  System.out.println("Iterace: " + i);
}
```

  - Příklad 2:

```
int i = 0;
int a = 5;
for (;i <= 5;) {
  System.out.println("Iterace: " + i);
  i += 2;
}
```

Cykly

- While
  - Syntaxe:

```
while (podminka) {
  prikaz1
  ...
  prikazn
}
```

  - Příklad:

```
int i = 0;
while (i < 5) {
  System.out.println("Iterace: " + i);
  i++;
}
```

Pozor na:

```
while (true)
```

Cykly

- Do while
  - Syntaxe:

```
do {
  prikaz1
  ...
  prikazn
} while (podminka);
```

  - Příklad:

```
int i = 0;
do {
  System.out.println("Iterace: " + i);
  i++;
} while (i < 5);
```

- Možnost přerušení cyklu příkazem break, popř. přeskočit na začátek smyčky příkazem continue.

Domácí úkol

- Nainstalovat IDE: Netbeans, Eclipse nebo **IntelliJ IDEA**

Úkoly

- Úkol 1: Napište v JShell funkci, která zjistí dělitelnost čísla 'n' číslem 'd'.

```
jshell> checkDiv(8, 4);
Cislo 8 je cislem 4 delitelne.

jshell> checkDiv(8, 5);
Cislo 8 neni cislem 4 delitelne.

jshell> checkDiv(8, 0);
Nulou nelze delit.
```

- Úkol 2: Napište v JShell funkci, která vykreslí pomocí znaku "*" na obrazovku trojúhelník o zadané velikosti (velikost v proměnné 'a'). Příklad výstupu pro hodnotu 5 (jshell trojuhelnik(5);):

```
    *
   **
  * *
 *  *
*****
```

- Úkol 3: Napište v JShell funkci, která vypíše všechna prvočísla menší než 100.

```
jshell> prvocisla();
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79
83 89 97
```

Reference

- Slidy vychází z předlohy dr. Petra Krajči
- HotSpot Compiler Understanding JIT Compilation and Optimizations
- Obrázky: https://i.stack.imgur.com/eUqSJ.png