Algoritmizace a programování 1

- Program v jazyce Java
- Jak napsat první program
- Co je v kódu
- Příkazy
- Deklarace
- Výrazy
- Výpis na konzoli
- Načtení dat z konzole

PRVNÍ KROKY

Struktura programu v jazyce Java

- Jazyk Java je objektový – základem programu je **třída** (**class**)
- **Program** sestává minimálně z jedné, zpravidla z více tříd
- Třída obsahuje deklaraci **členských proměnných** a **metod** (funkcí, podprogramů)
- Program zapisujeme v podobě prostého (strukturovaného) textu
- Každá třída (veřejná) musí být uložena v souboru stejného jména jako má třída, zdrojový soubor každé třídy má příponu `java` (název třídy a název souboru musí být totožné)
- Třídy jsou členěny do **balíků** (**package**)
- Struktura balíků (struktura pojmenovaní) musí odpovídat struktuře adresářů, tj. zařazení třídy do balíku znamená mimo jiné umístění zdrojového souboru třídy do příslušného adresáře

```
public class MojeTrida {
  // tělo třídy - deklarace proměnných, metod apod.
}
```

- Vstupním bodem programu může být pouze třída obsahující metodu `main` deklarovanou jako

```
public static void main (String[] args) {
  // tělo metody příkazy
}
```

Jak napsat první program

- Libovolný textový editor
- Zápis kódu:

```
public class MojeTrida {
  public static void main (String[] args) {
    // tělo metody příkazy
  }
}
```

- Soubor uložit pod jménem `MojeTrida.java`, třída je v implicitním balíku (bez jména)
- Umístění třídy do balíku – přidání klauzule `package` do záhlaví třídy

```
package mujbalik;
public class MojeTrida {
  public static void main (String[] args) {
    // tělo metody příkazy
  }
}
```

- Třídu umístíme do adresáře `mujbalik`
- Třídu můžeme pomocí prostředků Java SDK přeložit (javac.exe) a poté spustit (java.exe) – zadat z příkazové řádky systémové konzole, je nutné mít správně nastavené prostředí Javy
- Výše uvedený program je prázdný, po spuštění okamžitě skončí

Jak vytvořit první program/projekt v

NetBeans

- Provedení posloupnosti akcí
  - File
  - New Project
  - Java Application
    - Project Location, Project Name, Create main class
- NetBeans vytvoří nový adresář se všemi soubory projektu v zadaném umístění. V tomto adresáři, v jeho podadresáři `src` bude podadresář balíku aplikace (jméno odvozené od názvu projektu), v něm je umístěná třída se jménem odvozeným od názvu projektu. V prostředí lze program editovat, ladit, spouštět …

```
package prvni;
/**
 * @author Jirina
 */
public class Prvni {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

Eclipse

- Při spouštění Eclipse nutno zadat Workspace tj. umístění projektů
- Ve vývojovém prostředí Eclipse
  - File
  - New
  - Java Project
- V aplikaci, src
  - New
  - Package
- V package
  - New
  - Class

První programy

```
public class Prvni {
  public static void main (String[] args) {
    System.out.println("Toto je muj prvni program v jazyce Java");
  }
}
```

Program vypisuje text na konzoli

```
package alp.app;
public class Druhy {
  public static void main (String[] args) {
    System.out.println("Toto je muj druhy program v jazyce Java");
    System.out.println("Konec programu");
  }
}
```

V parametru `args` metody `main` jsou programu při jeho spuštění předány parametry příkazové řádky

```
package alp.app;
public class Ctvrty {
  public static void main (String[] args) {
    System.out.println("Vypisuji vsechny parametry zadane na prikazove radce");
    for (int i = 0; i < args.length; i++) {
      System.out.println(args[i]);
    }
    System.out.println("Konec programu");
  }
}
```

```java
package alp.app;

import alp.tools.Bod;

public class Treti {
  public static void main (String[] args) {
    System.out.println("Toto je muj dalsi program");
    Bod b = new Bod(3, 4);
    System.out.println("Bod se souradnicemi: " +
                       b.getX() + "  " +b.getY());
    System.out.println("Konec programu");
  }
}
```

Program sestávající ze dvou tříd
Třída s hlavním programem
Použití třídy z jiného balíku vyžaduje `import` této třídy
Třída `Bod` – obsahuje data a metody
Vytvoření jedné konkrétní instance třídy `Bod` operátorem `new`

```java
package alp.tools;

public class Bod {
  double x, y;

  public Bod (double x, double y) {
    this.x = x;
    this.y = y;
  }

  public double getX() {
    return x;
  }

  public double getY() {
    return y;
  }
}
```

CO JE V KÓDU – SYNTAKTICKÉ ELEMENTY

Lexikální struktura programu

- Program v jazyce Javě využívá znakovou sadu Unicode – abeceda jazyka obsahuje všechny znaky Unicode (včetně například písmen s diakritikou)
- Jazyk Java je „**case sensitive**“ – při zápisu kódu jsou rozlišována malá a velká písmena

- Po lexikální stránce obsahuje kód především
  - **Prázdná místa –** mezery, tabulátory, znaky konce řádku
  - **Komentáře**
    - Jednořádkový `//`
    - Víceřádkový `/* */`
    - Dokumentační `/** */` - dokumentační komentář může obsahovat anotace, anotace začínají znakem `@` například `@Override, @Deprecated` …
  - **Rezervovaná slova**
  - **Identifikátory**
    - Způsob zápisu identifikátorů
    - Case sensitive
  - **Literály** – konstantní hodnoty
  - **Specální symboly** – separátory (oddělovače) , operátory a další speciální symboly

Rezervovaná slova a speciální symboly

- **Rezervovaná slova** – nelze je používat jako identifikátory
  - **Klíčová slova** – slouží k zápisu konstrukcí programu (zápis příkazů a podobně)
    - `abstract assert boolean do break byte case catch char class const continue default double else enum extends final finally float for goto if implements import instanceof int interface long native new package private protected public return short static strictfp super switch synchronized this throw throws transient try void volatile while`
  - **Další rezervovaná slova**
    - Literály typu `boolean` – `true, false`
    - Literál `null`

- **Separátory**
  - `( ) { } [ ] ; , .`

- Operátory
  - `= > < ! ~ ? : == <= >= != && ||`
    `++ -- + - * /`
    `& | ^ % << >> >>>`
    `+= -= *= /= &= |= ^= %= <<= >>= >>>=`

Identifikátory

- Identifikátor = jméno, používáme k pojmenování vlastních
  - Tříd, rozhraní
  - Metod
  - Proměnných (v metodách, třídách)
  - Konstant
  - Balíků
- Identifikátor
  - Sekvence písmen, číslic, znak podtržítko, nesmí začínat číslicí
    - (JavaLetter, JavaDigit)
  - Pro identifikátory nelze používat vyhrazená slova
  - Délka identifikátoru není omezena
  - Java je „case sensitive“
- Znak tečka „.“ odděluje jednotlivé části ve složených identifikátorech
  - `System.out.println()`
- Doporučení
  - **Třídy a rozhraní** – identifikátor začínat velkým písmenem, další slova v identifikátoru začínat velkým písmenem

    `String, StringBuffer`
  - **Proměnné a metody** – identifikátor začínat malým písmenem

    `pocetCisel, getSize()`
  - **Konstanty** – používají se pouze velká písmena, ve víceslovných identifikátorech se používá znak podtržítko „_“ pro oddělení slov

    `MAX_HODNOTA`
  - **Balíčky** – identifikátory se skládají pouze z malých písmen, ve složených jménech je oddělovačem jednotlivých částí znak „tečka“

    `mypkg, java.lang`

CO JE V KÓDU – STRUKTURA

Třída a „spustitelná“ třída

- Základní schéma třídy
  - Deklarace atributů – proměnných (ukončeny středníkem)
  - Deklarace metod
  -
  - **Proměnná** – vyhrazená část paměti pro uchování dat, proměnná má deklarací určený typ a jméno (identifikátor) – proměnné lze přiřadit hodnotu a hodnotu číst (použít)
  - **Metoda** – část programového kódu – při deklaraci definujeme typ, jméno, parametry, tělo metody (výkonná část) – metodu lze volat v jiné části kódu – po zavolání metody se provede kód metody s konkrétními parametry předanými při volání, metoda vrací hodnotu určeného typu
- Libovolná třída může (nemusí) obsahovat metodu `main`
  - Třída deklarovaná jako `public`
  - Třída potom může (nemusí) být tzv. hlavní třídou – vstupním bodem celého programu
  - Po spuštění programu se začnou vykonávat jednotlivé příkazy umístěné v těle metody `main`

```
public class <jméno třídy> {

  // schéma deklarace proměnné
  <typ> <jméno>;

  // schéma metody
  <typ> <jméno>
   (<deklarace parametrů>) {
      <příkazy>
  }

  public static void main(String[] args) {
    // kód
    // příkazy
  }

}
```

Tělo metody

- Tělo metody (kód) obsahuje příkazy a bloky (bloky příkazů)
- Kód v těle metody se postupně provede po spuštění metody
- Příkazy jsou v kódu ukončeny středníkem
- Příkazy
  - Deklarace lokálních proměnných – zajistí přidělení požadované paměti proměnné
  - Prázdný příkaz
  - Výrazy použité jako příkazy
    - Přiřazení
    - Inkrementace a dekrementace
    - Volání metody – metodu voláme jménem, do okrouhlých závorek uvedeme skutečné parametry – provede se spuštění metody
  - Příkazy jazyka Java – `if, switch, do, while, for, break, continue, return, throw, try, synchronized, assert`
- Blok příkazů
  - `{<příkazy>}`

```
public class VypocetCtverce {

 /**
  * @param args
  */
 public static void main(String[] args) {

    float a = 3.6f;
    float obvod;
    float obsah;
    obvod = 4 * a;
    obsah = a * a;

    System.out.println("Strana " + a);
    System.out.println("Obvod " + obvod);
    System.out.println("Plocha " + obsah);
  }
}
```

Deklarace

- Deklarace proměnné – je konstrukce, která přidělí proměnné určitého typu **jméno**, **paměťový prostor** a **počáteční hodnotu**

```
<typ> <jmeno>;
<typ> <jmeno>, <jmeno>, <jmeno>;
<typ> <jmeno> = <hodnota>;
```

- Počáteční hodnota
  - Explicitně uvedena deklarací
  - Členské proměnné tříd – implicitně inicializovány
  - Lokální proměnné metod – počáteční hodnota není deklarací určená

- Deklarace v kódu mohou být uvnitř třídy, kdekoli v metodě
- Deklarace musí být uvedena před prvním použitím
- Hodnotu proměnné lze v kódu měnit – viz dále operátory přiřazení, načtení z proudu

- Deklarace konstanty – proměnné s konstantní (neměnnou) hodnotou
- Deklaraci předchází klíčové slovo **`final`**

```
final <typ> <jmeno> = <hodnota>;
final <typ> <jmeno>;
```

- Konstantě lze přiřadit hodnotu právě jednou, hodnotu konstanty nelze poté v kódu dále měnit
- Příklady použití

```
final int MAX = 200;
final int MAX;
MAX = 200;
```

- Základní typy
  - Celočíselné – `byte, short, int, long`
  - Reálné – `float, double`
  - Znakový – `char`
  - Logický – `boolean`

Výrazy

- Výraz – série operací
- Výrazy obsahují konstantní hodnoty, proměnné, operátory, volání metod

- **Operce přiřazení**
  - Operátor přiřazení je v jazyce Java reprezentován znakem **=**
    operaci přiřazení zapisujeme

    <promenna> = <vyraz>
    - <promenna> - identifikátor proměnné
    - <vyraz> - může být reprezentován konstantní hodnotou, proměnnou, konstantou nebo obecně výrazem, hodnota výrazu na pravé straně přiřazení musí být typově kompatibilní vzhledem k přiřazení s typem proměnné, která je na levé straně operátoru přiřazení, Java je jazyk se silnou typovou kontrolou
  - Operace přiřazení má výslednou hodnotu rovnou přiřazené hodnotě
  - Přiřazení se vyhodnocuje zprava doleva (na rozdíl od ostatních operací, které se vyhodnocují zleva doprava)
  - Příkaz přiřazení – ukončení operace středníkem

- Aritmetické operace a operátory
  - Binární
    - **+** součet
    - **-** rozdíl
    - ***** násobení
    - **/** dělení reálné, dělení celočíselné
    - **%** zbytek po celočíselném dělení, dělení modulo
  - Unární
    - **+** unární plus
    - **-** unární mínus
    - **++** inkrementace hodnoty operandu, prefixový i postfixový zápis
    - **--** dekrementace hodnoty operandu, prefixový i postfixový zápis

- Relační operátory
- Logické operátory
- Bitové operátory
- Operátor zřetězení
- Operátory přiřazení
- Přetypování
- Vyhodnocování výrazů, priorita, závorky

Konstantní hodnoty, textové řetězce

- Konstanty
  - **Pojmenovaná konstanty** – viz dříve část „Deklarace“
  - **Literály** – konkrétní konstantní hodnoty zapsané/použité přímo v kódu

- Literály
  - Celočíselné hodnoty – `1.865`, `8.965E-5`, `-14.3f`, `15.`, `.865`
  - Reálné hodnoty – `126`, `-365`, `013`, `0xFF`, `9876543210123456L`
  - Znakové – `'A'`, `'\u0041'`, `'\n'`, `'\101'`,
  - Logické – `true`, `false`
  - Textové řetězce – `"Strana "`

- Textové řetězce
  - Konstantní textové řetězce – odvozeny od jedné třídy jazyka Java – třídy `String`
  - Operace zřetězení
    - Operátor `+`
    - Pokud je jeden z operandů operace + textový řetězec provede se implicitní konverze hodnoty druhého operandu (jiného datového typu) na hodnotu typu `String`
    - `"Strana " + a`
    - `40 + 60 + „ mají společné dělitele " + 2 + 4 + 5 + 10 + 20`

☺ výsledek OK

☹ výsledek bude asi jiný, než bylo zamýšleno

INTERAKCE S UŽIVATELEM

Výpis hodnot na konzoli

- Konzolový vstup a výstup budeme používat jako základní způsob interakce programu s uživatelem

- Základní prostředky pro výpis hodnot na konzoli

```
System.out.print(<co>);
System.out.println(<co>);
```

  - `System` – třída jazyka Java – tato třída je umístěna v balíčku `java.lang` – tento balíček je implicitně importován, není nutné zapisovat import explicitně
  - `out` – členská proměnná třídy `System` typu `PrintStream`, umožňuje práci se standardním (terminálovým. konzolovým) výstupem
  - `print(), println(), format()` – metody, která má k dispozici proměnná `out` odvozená od třídy `PrintStream`, `println()` – provede výpis a navíc ukončí aktuální řádek výstupu, `println()` lze použít i bez parametrů
  - `<co>` – výraz libovolného typu – výraz se vyhodnotí, hodnota se převede na textový řetězec, řetězec se vypíše na příslušný výstup

Konzolový vstup

- Pro terminálový vstup budeme používat instanci (objekt odvozený od) třídy `Scanner`
- Tato třída je umístěna v baličku `java.util`
- Baliček a jednotlivé prostředky v něm obsažené nejsou implicitně importovány. Abychom mohli používat třídu a její prostředky, je třeba zajistit import této třídy zařazením import `java.util.Scanner` před deklaraci třídy, blíže bude import balíčků probírán později
- Instanci třídy `Scanner` musíme nejdříve vytvořit voláním metody konstruktoru, parametrem je vstupní proud, ze kterého budeme číst – tímto vstupním proudem je v případě konzolového/terminálového vstupu `System.in`
- Deklarace a inicializace proměnné pro načítání ze standardního vstupu bude potom vypadat následovně

```
Scanner sc = new Scanner(System.in);
```

- Poté je možné prostřednictvím `sc` volat metody třídy `Scanner` pro načítání hodnot různých základních typů z příslušného proudu.
  - Čtení celého čísla typu `int` – metoda `sc.nextInt()`
  - Čtení reálného čísla typu `double` – metoda `sc.nextDouble()`
  - Čtení znaku – prvního znak zadaného textového řetězce – metod `sc.nextLine().charAt(0)`
  - Čtení textového řetězce do prvního prázdného znaku – metoda `sc.next()`
  - Čtení celé řádky do textového řetězce – metoda `sc.nextLine()`
  - Metody vrací načtenou hodnotu svým jménem
  - Obdobné metody pro načtení hodnot typu `byte, short, long, float, boolean`
  - Lokalizace – `sc.useLocale(locale.US)`

Co už umíme, známe

- Vytvořit nový konzolovou aplikaci v Jave – projekt v NetBeans
- Základní strukturu třídy
- Co je třeba pro to, aby třída mohla být hlavní třídou programu
- Co je v kódu metody
- Jak deklarovat proměnné
- Některé základní typy
- Konstrukce aritmetických výrazů
- Jak vypsat zprávy a výsledky na konzoli (standardní výstup)
- Jak načíst celočíselné, reálné hodnoty, slova, textové zprávy a znaky ze standardního vstupu (z konzole)

- Vše zatím pouze v základním přehledu bez podrobností

```
public class VypocetObdelnika {

 /**
  * @param args
  */
 public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    float a, b;
    float obvod;
    float obsah;
    System.out.println("Zadej delku stran obdelnika ");
    a = sc.nextFloat();
    b = sc.nextFloat();
    obvod = 2 * (a + b);
    obsah = a * b;

    System.out.println("Obdelnik o stranach " +
       a + " a " + b + " ma ");
    System.out.println("obvod " + obvod);
    System.out.println("plochu " + obsah);
    System.out.println("Konec programu.");
  }
}
```