Analýza a návrh informačných systémov II 1

objektovo orientovaný návrh

Peter Bednár

Štruktúra predmetu

Cvičenia

- Programovanie v jazyku Java
- Hodnotenie: 2x10 bodov test: teória + programovanie zadaných úloh, 20 bodov zadanie: programovanie + UML návrh

Skúška

- Hodnotenie: 30 bodov programovanie zadaných úloh, 30 bodov teoretické otázky

Programovacie paradigmy – Štruktúrované programovanie

Štruktúrované programovanie

- Dátové štruktúry združujú viacero súvisiacich atomických hodnôt – umožňujú programátorovi definovať vlastný zložený dátový typ
- Kód je rozdelený na podprogramy – funkcie, ktoré sa dajú opakovane volať na rôznych dátach

Programovacie paradigmy – Modulárne programovanie

Modulárne programovanie

- Rozdelenie programu na samostatné znovupoužiteľné moduly – knižnice, ktoré poskytujú vlastné funkcie a dátové typy
- Oddelené rozhranie od implementácie – tzn. je možné meniť vnútorný kód knižnice bez toho aby sa zmenil spôsob jej použitia/volania (napr. v jazyku C oddelená deklarácia typov a funkcií v hlavičkovom .h súbore a ich definícia v zdrojovom .c súbore)
- Zlepšenie znovupoužiteľnosti a udržateľnosti kódu

Programovacie paradigmy – Objektové programovanie (1)

Objektové programovanie

- Objekt = dátová štruktúra + funkcie určené na jej spracovanie
- Objekt má:
  - svoj stav, tvorený hodnotami dátových atribútov
  - správanie: kód naprogramovaný v metódach - funkciách určených na spracovanie stavu
  - svoju identitu, tzn. môžeme vytvoriť viacero objektov toho istého typu
- Program je potom tvorený objektmi a interakciou medzi nimi pomocou operácií, ktoré sú nad nimi vykonané

Programovacie paradigmy – Objektové programovanie (2)

- Typ objektov je definovaný ako Trieda
- Trieda predpisuje, ktoré dátové atribúty musí objekt daného typu mať a definuje kód pre všetky metódy
- Z jednej triedy môžeme vytvoriť viacero konkrétnych objektov daného typu – tzv. inštancií

UML – Diagram tried a Diagram objektov

- Pre objektový návrh budeme používať diagram tried jazyka UML
- Diagram tried umožňuje popísať triedy a vzťahy medzi nimi
- Podobne, Diagram objektov popisuje konkrétne objekty

Názov triedy
Atribúty triedy
Metódy

ID objektu: Názov triedy
Hodnoty atribútov

Diagram tried - príklad

- Triedy modelujúce osobný automobil a jeho motor

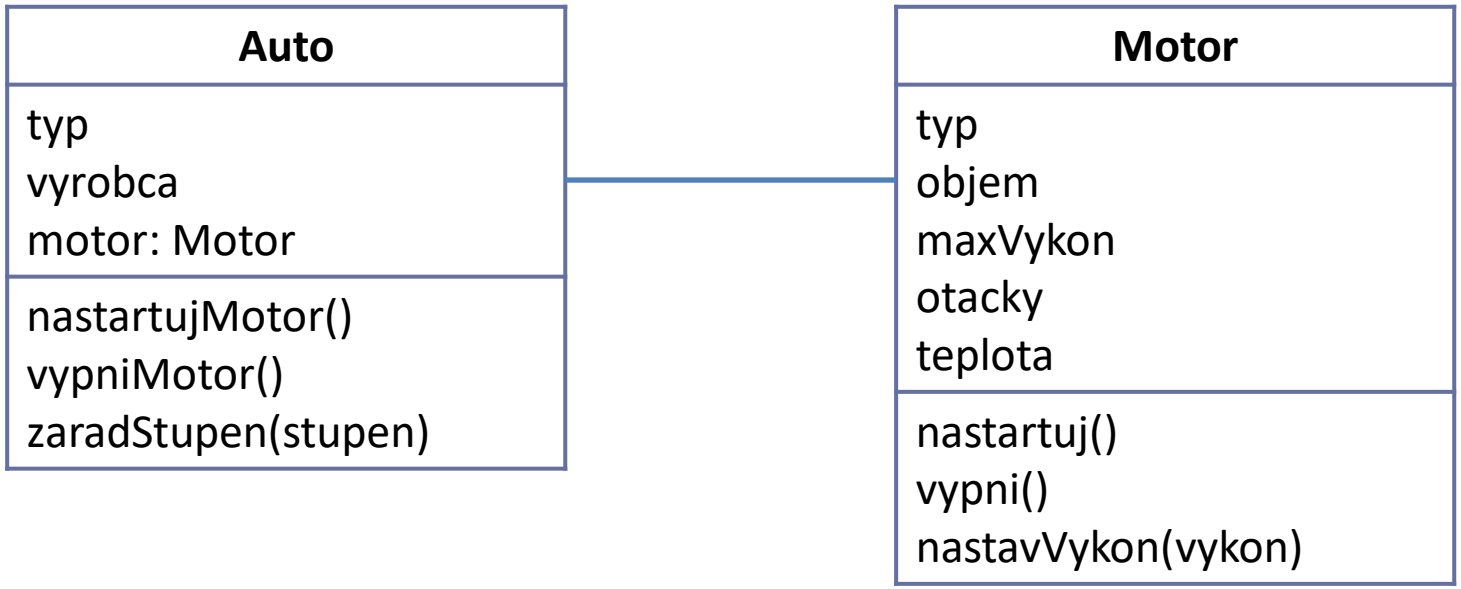


Diagram tried - príklad

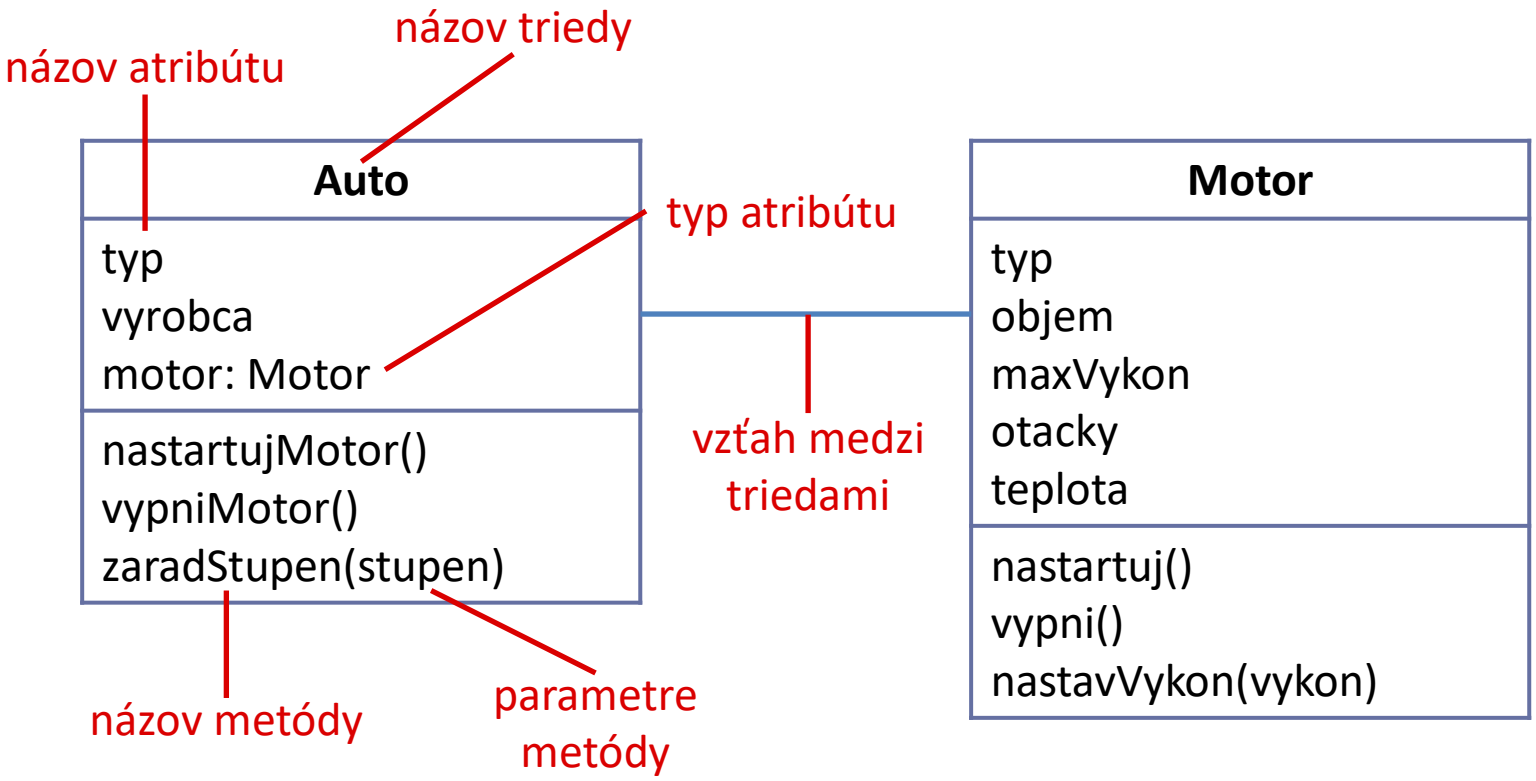


Diagram objektov - príklad

- Konkrétne inštancie modelujúce dve autá (jedno bez motora)

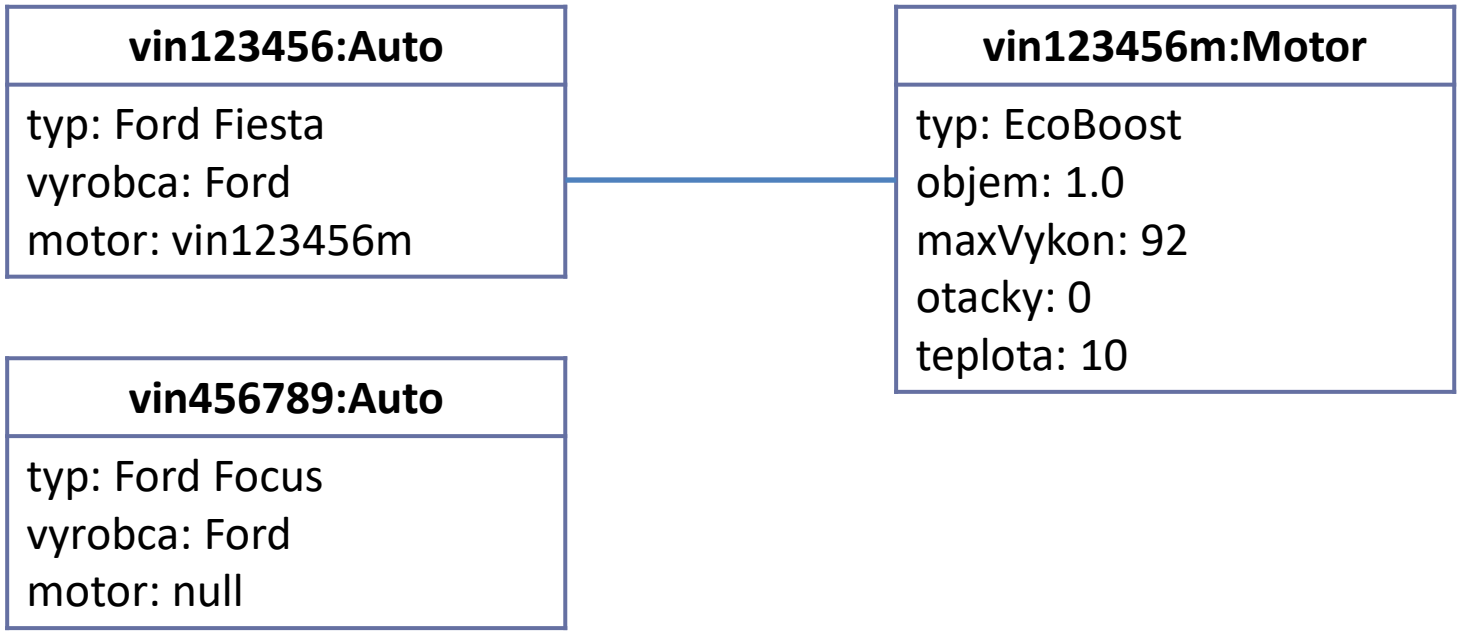
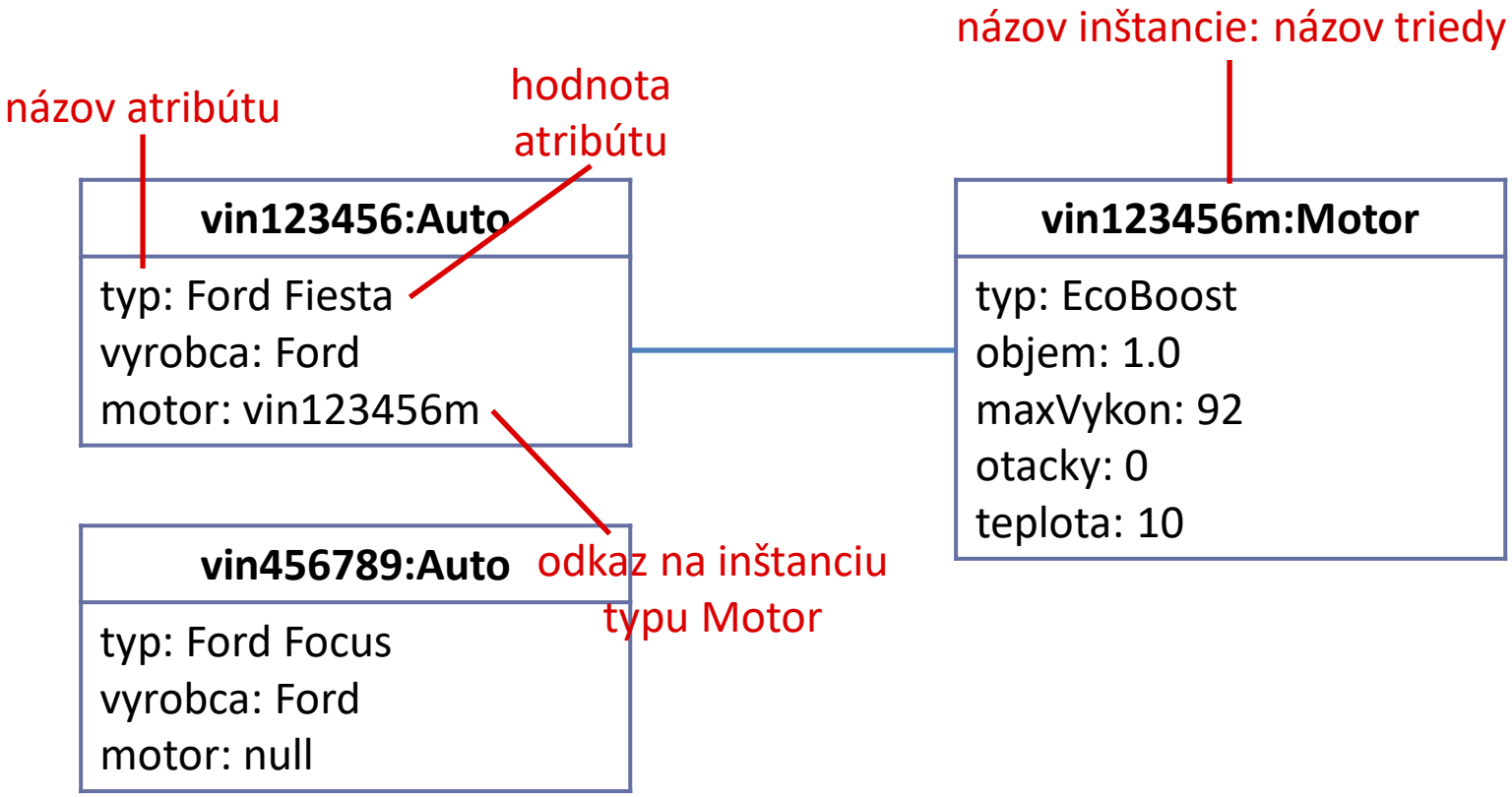


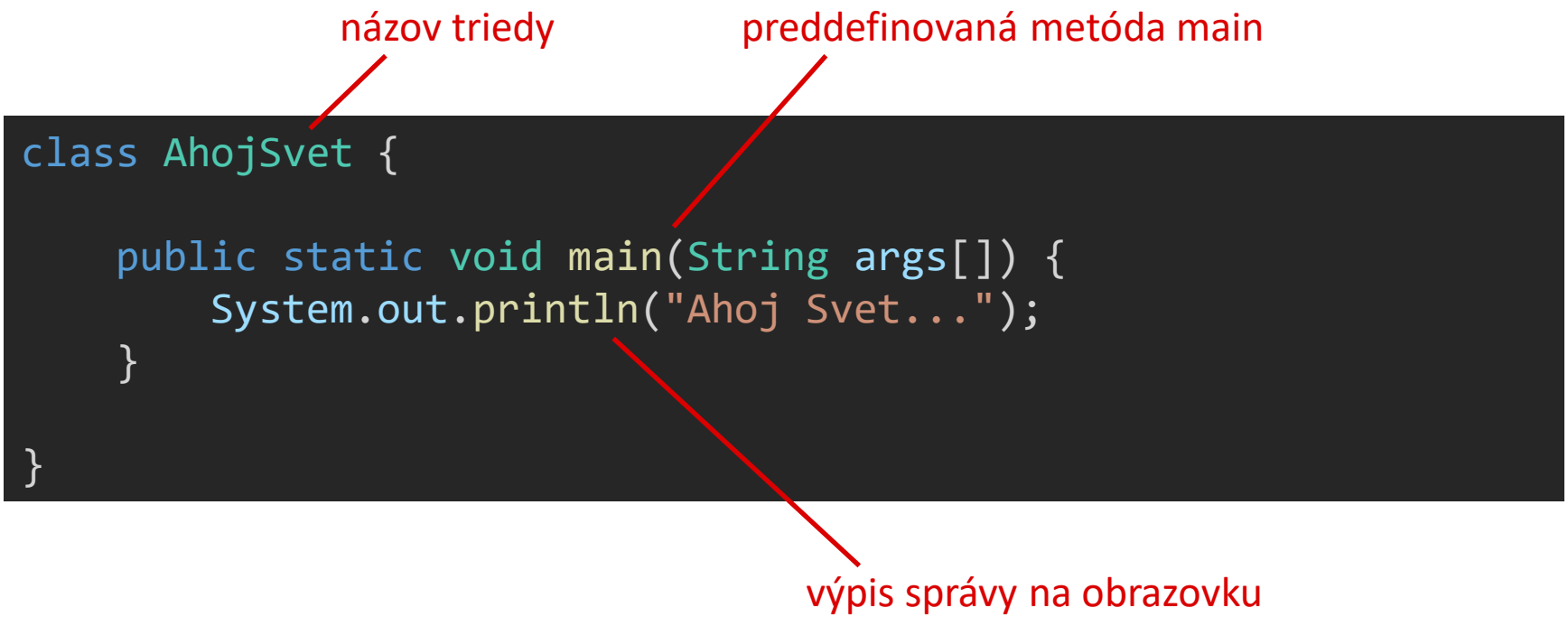
Diagram objektov - príklad



Programovací jazyk Java

- Objektovo-orientovaný jazyk
- Syntax založená na jazyku C resp. C++
- Navrhnutý v 1995 firmou Sun Microsystems (James Gosling)
- Zdrojový kód je preložený do univerzálneho byte-kódu, ktorý je interpretovaný vo virtuálnom stroji (JVM – Java Virtual Machine)
  - Preložené aplikácie sú prenositeľné na rôzne zariadenia a operačné systémy

Java Ahoj Svet



zdrojový kód musí byť uložený v súbore AhojSvet.java

Premenné, dátové typy a príkazy

Premenné

- Príkaz na definovanie premennej je podobný ako v C: `typ názov;`
- Napr.:

```
int maxVykon;
```

- Voliteľne je im možné priamo priradiť počiatočnú hodnotu:

```
int maxVykon = 92;
```

- Ak sa premennej nepriradí žiadna hodnota, bude mať prednastavenú hodnotu `0`, `false`, `null`

Základné dátové typy - boolean

- Môže nadobúdať iba dve hodnoty, `true` a `false`
- Používa sa hlavne pri testovaní, je možné na neho aplikovať logické operátory: a &&, alebo ||, negácia !

```
boolean odpoved = true;
boolean vysledok = (!odpoved) || (10 < 20);
// vysledok = true
```

Základné dátové typy - čísla

- Celé čísla (so znamienkom): `byte`, `short`, `int`, `long`
- Desatinné čísla: `float`, `double`
- Základné operácie +, -, *, /, % (celočíselný zvyšok po delení)
- Skrátený zápis ++, --, +=, -=, *=, /=, %=
- Porovnanie ==, !=, <, >, <=, >= (výsledok je boolovská hodnota)

```
int otacky = 1000;
double moment = 100;
double vykon = moment * (otacky / 5000.0);
```

Čísla - pretypovanie

- V Jave sa desatinné čísla automaticky neprevedú na celé, hodnotu musíme explicitne *pretypovať*

```
double plyn = 80.5;
int otacky = (int)((plyn * 50) + 2000);
```

Pretypovanie na celé číslo

Keďže premenná plyn je desatinné číslo, výsledok je desatinné číslo

- Podobne je to aj pri prevode `double` na `float`, alebo `long` na `int`, atď.

Základné dátové typy – reťazce (1)

- Typ `String`
- Hodnoty ohraničené `"`, zapísané na jednom riadku
- Zástupné znaky pre nový riadok `\n`, tabulátor `\t`, `\\`, atď.
- Reťazce je možné spájať operátorom +

```
String sprava = "Ahoj" + "\nSvet...";
System.out.println(sprava);
/* vypíše sa:
Ahoj
Svet...
*/
```

Základné dátové typy – reťazce (2)

- Ak chceme previesť číslo, alebo iný typ na reťazec, môžeme použiť metódu `String.valueOf(hodnota)`

```
String s = "Moje šťastné číslo je " + String.valueOf(6);
```

  - Pri + s reťazcom sa však prevedie automaticky
- Naopak, reťazec na číslo môžeme previesť metódami `Integer.parseInt(reťazec)`, resp. `Float.parseFloat(reťazec)`, podobne pre `Long`, `Short`, a `Double`

```
int vykon = Integer.parseInt("92");
```

Základné dátové typy – reťazce (3)

- Reťazce sú objekty, ktoré majú definované rôzne metódy
- Metódy vrátia nový reťazec, pôvodný sa nezmení
- Napr.:

```
s.length()          // dĺžka reťazca
s.toUpperCase()     // prevedie reťazec na veľké písmena
s.toLowerCase()     // prevedie reťazec na malé písmena
s.trim()            // odstráni zo začiatku a konca
                    // prázdne znaky
```

- Volanie je možné zreťaziť

```
String s = "Ahoj  ".trim().toUpperCase();
s = s.toLowerCase();
```

Polia (1)

- typ[] názov, alebo typ názov[]
- prístup cez indexovanie od 0
- vytvárajú sa operátorom **new** (bez nastavenia majú prvky prednastavenú hodnotu `0`, `false`, `null`)

```
int cisla[] = new int[10];
cisla[1] = 10; //nastavíme druhý prvok
// posledný prvok
int posledny = cisla[cisla.length - 1];
```

Polia (2)

- inicializácia s hodnotami

```
String mena[] = {"Peter", "Lucia", "Elisabetta"};
```

- vnorené polia (dajú sa použiť na reprezentáciu matíc, vnorené polia však nemusia mať rovnaký počet prvkov)

```
int cisla[][] = {{1, 2, 3}, {4, 5}};
int cislo = cisla[1][0]; // cislo = 4
```

Príkazy - if

- **Vetvenie if**

```
if (podmienka)  {
    kód, ktorý sa vykoná keď je podmienka splnená
}
```

- Voliteľná `else` vetva, zreťazenie cez `else if`

```
if (cislo < 0)  {
    System.out.println("záporné");
} else if (cislo > 0) {
    System.out.println("kladné");
} else {
    System.out.println("nula");
}
```

Príkazy - switch

- **Vetvenie switch/case**
- Podmienky by mali byť ukončené **break**
- **default** vetva

```
String s;  // bez priradenia hodnoty = null
int cislo = 10;

switch (cislo) {
    case 0:  s = "žiaden"; break;
    case 1:  s = "jeden";  break;
    case 2:  s = "dvaja";  break;
    default: s = "veľa";
}
```

Príkazy - for

- **Cyklus for**
- `for (inicializácia; podmienka; inkrementácia)`

```
int cisla[] = {1, 2, 3, 6};
int suma = 0;
for (int i = 0; i < cisla.length; i++) {
    suma += cisla[i];
}
// suma = 12
```

Príkazy - while

- **Cyklus while**
- Vykonáva sa pokiaľ nie je podmienka splnená

```
int cisla[] = {1,2,3,6};
int suma = 0;

int i = 0;
while (i < cisla.length) {
    suma += cisla[i];
    i++;
}
// suma = 12
```

Príkazy – do/while

- **Cyklus do/while**
- Vykoná sa aspoň raz pokiaľ nie je podmienka splnená

```
int cisla[] = {1,2,3,6};
int suma = 0;

int i = 0;
do {
    suma += cisla[i];
    i++;
} while (i < cisla.length);
// suma = 12
```

Triedy, objekty, členské premenné

Triedy a členské premenné

- Triedy sa definujú kľúčovým slovom `class`

```
class Názov {
    definícia členských premenných
    definícia metód
}
```

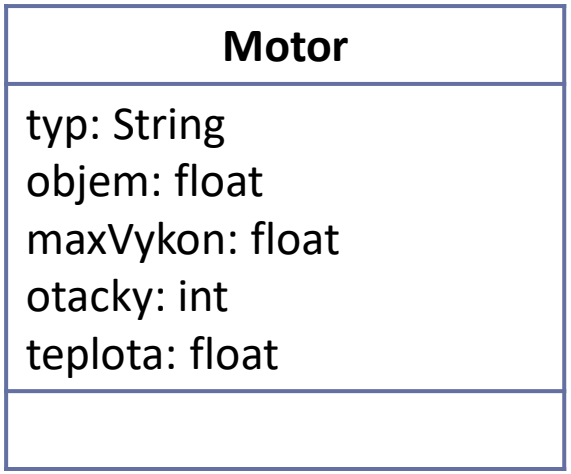
- Triedy sa zvyčajne označujú slovami s veľkými počiatočnými písmenami, napr. `MotoroveVozidlo`
- Zdrojový kód musí byť uložený v súbore s príponou .java, ktorý má rovnaký názov ako trieda (v jednom súbore sa zvyčajne definuje iba jedna trieda)

Triedy a členské premenné – príklad 1

```
class Motor {

    String typ;
    float objem;
    float maxVykon;
    int otacky;
    float teplota;

    // trieda Motor zatiaľ nemá žiadne metódy
}
```



Objekty/Inštancie

- Názov triedy definuje nový typ, ktorý môžeme použiť pri definícii premenných
- Objekty sa vytvárajú príkazom Trieda objekt = `new` Trieda`();`
- K členským premenným a k metódam sa pristupuje cez `.` (bodku)

- Napr.:

```
// vytvoríme novú inštanciu triedy Motor
Motor motor1 = new Motor();
// priradíme novú hodnotu členskej premennej maxVykon
motor1.maxVykon = 92;
System.out.println(motor1.otacky);   // = 0, prednastavené
```

Inštancie – príklad 2

```
class PrvyProgram {

    public static void main(String args[]) {
        Motor motor1 = new Motor();
        Motor motor2 = motor1;

        motor1.maxVykon = 92;
        motor2.otacky = 2000;

        System.out.println("motor2 max. výkon: " + motor2.maxVykon);
        // 92
        System.out.println("motor1 otáčky: " + motor1.otacky);
        // 2000
    }
}
```

Vytvorili sme iba jeden objekt. Obidve premenné ukazujú na ten istý objekt

Program v jave

Keďže v Jave nemôžeme priamo vytvoriť funkciu, pre program si definujeme novú triedu

Metóda main je preddefinovanou hlavnou funkciou programu

```
class PrvyProgram {

    public static void main(String args[]) {
        // hlavná funkcia programu
        System.out.println("Ahoj Svet...\n");
    }
}
```

- Iba triedy, ktoré majú *verejnú statickú* metódu `main` môžu byť spustené ako program

Inštancie – príklad 1

```
class PrvyProgram {

    public static void main(String args[]) {
        Motor motor1 = new Motor();
        Motor motor2 = new Motor();

        motor1.maxVykon = 92;
        motor2.otacky = 2000;

        System.out.println("motor1 max. výkon: " + motor1.maxVykon);
        // 92
        System.out.println("motor2 max. výkon: " + motor2.maxVykon);
        // 0 (neinicializované)
    }
}
```

Zhrnutie

- Programovacie paradigmy
  - Štrukúrované programovanie
  - Modulárne programovanie
  - Objektové programovanie
- Úvod do jazyka Java
  - Základné dátové typy – boolean, čísla, reťazce
  - Polia
  - Vetvenia if a switch
  - Cykly for, while a do-while
- Definovanie tried a vytváranie objektov
- Definovanie členských premenných