Třídy, objekty

- Vlastní třída – `class`
- Deklarace, instance, použití
- Atributy, metody
- Strukturované typy
- Pole objektů

Datové typy v jazyce Java

- Primitivní datové typy

- Referenční datové typy
  - Pole
  - Objektové typy
    - `class`
    - `interface`
    - `enum`
  - Proměnná referenčního datového typu obsahuje referenci
  - Deklarace proměnné, vytvoření instance (alokace paměti a její inicializace)

Programování

- Strukturovaný přístup
  - Úloha
    - Postup řešení
    - Členění – jednotlivé operace

- Objektový přístup
  - Úloha
    - Textový popis úlohy
    - Identifikace objektů
    - Zodpovědnosti jednotlivých objektů
    - Návrh obecných šablon pro vytváření objektů
  - Jazyk Java – objektový
  - Základním prvkem objektového programování – návrh obecných šablon pro vytváření objektů – tj. tříd – v jazyce Java typ `class`

Typ `class`

- Objekt
  - Základní abstraktní jednotka v objektovém programování
  - Uchovává/pamatuje si svůj stav (v podobě dat čili atributů)
  - Poskytuje operace/služby, aby se s ním mohlo pracovat (nazývané metody).
  - Při používání objektu nás zajímá, jaké operace (služby) poskytuje, ale ne, jakým způsobem to provádí Zda to provádí sám nebo využije služeb jiných objektů, je celkem jedno. Vlastní implementaci pak lze (např. zefektivnit), aniž by se to dotklo ostatních objektů, které daný objekt používají.
- Třída
  - Abstrakce, která podchycuje na obecné úrovni podstatu objektů podobného typu
  - Třída je předpis (vzor), jak vyrobit objekt daného typu (instanci)
- Instance třídy (v některých programovacích jazycích také objekt)
  - Konkrétní datový objekt v paměti odvozený z nějakého vzoru (třídy)

- Třída – typ `class` – součástí třídy jsou
  - Atributy – uchovávají data – členské proměnné třídy, datové položky třídy, proměnné deklarované uvnitř třídy
  - Metody – slouží pro komunikaci mezi objekty – podprogramy (funkce) deklarované uvnitř nějaké třídy
    - Obsahují výkonný kód – příkazy, které se postupně provedou po zavolání metody
    - Pracují s atributy objektu, respektive třídy
    - Parametry metody – hodnoty, pro které se má příslušný kód provést
    - Vrací hodnotu svým jménem – návratová hodnota, specifikace typu návratové hodnoty (použití typu `void` – metoda nevrací hodnotu (vrací prázdnou hodnotu))

Deklarace třídy

- Deklarace třídy

  `<modifikátory> class <identifikátor> { ... }`

- Modifikátory
  - `public`, `private`, `final`, `abstract`, `static`
- Identifikátor
  - Jméno/název třídy – počáteční velké písmeno, každé další slovo v identifikátoru začínat velkým písmenem, neoddělovat „podtržítkem“
- Tělo třídy
  - Blok uvozený složenými závorkami
  - Obsahuje atributy a metody (deklarace atributů a metod)

- Veřejná třída
  - třída deklarovaná s modifikátorem `public` – umístění v samostatném souboru stejného jména jako je třída
  - Každá veřejné třída patří do nějakého balíku – klausule `package` – balíky vytvářejí jmenný prostor, pojmenování balíků musí korespondovat adresářovou strukturou, ve které jsou umístěné zdrojové soubory aplikace

Deklarace atributů a metod

- Deklarace atributů – proměnné deklarované ve třídě
  - Deklarace bez inicializace, s inicializací

```
<modifikátory> <typ> <identifikátor>;
<modifikátory> <typ> <identifikátor> = <hodnota>;
```

  - Modifikátory – `public, private, final, static, protected`
  - Rozsahu platnosti identifikátoru

- Deklarace metod

```
<modifikátory> <typ> <jméno> ( <formální parametry>) {
        <tělo metody>
}
```

  - Modifikátory – `static, public, final, private, protected, abstract`

Příklad

- Úloha – vytvořit aplikaci, která bude pro **sadu bodů** poskytovat **bod** s maximální vzdáleností od počátku.

- Objekty
  - Bod
    - Souřadnice
    - Umí poskytnout svou vzdálenost od počátku
  - SadaBodu

- Uvedená třída je v daném tvaru nepoužitelná

```
package bodytools;

public class Bod {
  private double x, y;

  public double vzd(){
    return Math.sqrt(x*x + y*y);
  }

}
```

Vytvoření instance objektu

- Deklarace proměnné – název třídy – typ

```
Bod b;
```

- Vytvoření instance – použití new a konstruktoru třídy – pokud třída nemá vlastní konstruktor má třída k dispozici konstruktor implicitní (implicitní konstruktor nemá parametry)

```
b = new Bod();
```

- Odstranění instance – Java nemá prostředky pro rušení instancí, které by mohl programátor použít v kódu – odstraňování instancí je prováděno automaticky – *Garbage collector* – pro zrušení je třeba zajistit, aby žádná proměnná neobsahovala referenci na danou instanci (stejně jako u pole).

```
b = null;
```

```
b = new Bod();
```

- Metoda konstruktoru
  - Implicitní konstruktor
  - Vlastní konstruktor, jeho deklarace a použití
    - Vytvoření instance – alokace paměti a její inicializace – inicializace atributů

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

b = new Bod(4.7, 9.6);
```

Příklad

```
package bodytools;

  public class Bod {
    private double x, y;

    public Bod() {}

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

    public double getVzd(){
      return Math.sqrt(x*x + y*y);
    }

    public double getVzd(Bod b){
      double dx = x – b.getX();
      double dy = y – b.y();
      return Math.sqrt(dx*dx + dy*dy);
    }

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

- Třída Bod
  - Dva atributy pro uchování souřadnic jednoho bodu – oba atributy jsou privátní
  - Konstruktor – parametry pro inicializaci atributů
  - Metoda, která poskytuje hodnotu vzdálenosti bodu od počátku
  - Metody, které poskytují hodnoty jednotlivých atributů
  - K atributům v metodách přistupujeme přímo
  - Platnost identifikátorů – blok, ve kterém je deklarován
  - Kolize identifikátorů – platnost identifikátoru deklarovaného na nejnižší úrovni
  - Lokální identifikátory, proměnné – v metodě
  - Lokální proměnné nejsou inicializovány
  - Pokud lokálním identifikátorem překryjeme atribut – můžeme k atributu objektu přistupovat pomocí reference `this`
  - Reference `this` – reference na instanci třídy (objekt) jejíž metoda je volána

Veřejné a privátní atributy a metody

- Deklarace atributů a metod s modifikátorem `private` – privátní položky
- Deklarace atributů a metod s modifikátorem `public` – veřejné položky

- Použití dříve deklarované třídy `Bod`

```
Bod b;
b = new Bod(4.7, 9.6);
System.out.println(b.getVzd());
```

- K veřejným položkám lze přistupovat i mimo třídu
- Pro přístup k položkám používáme „*tečkovou notaci*“
- Pokud používáme třídu a její prostředky v jiném balíku, nutno provést import (nebo používat plně kvalifikované jméno)
- K privátním položkám nelze mimo třídu přímo přistupovat

```
System.out.println(b.x + " " + b.y); // !takto ne
System.out.println(b.getX() + " " + b.getY());
```

Instanční a statické položky tříd

- Modifikátor `static`
- Atributy a metody deklarované **bez modifikátoru** `static`
  - Jedná se o tzv. instanční položky
  - Každý objekt má svoje instanční atributy – různé objekty mají obecně různé hodnoty instančních atributů
  - Při volání instanční metody – metoda má k dispozici implicitní parametr `this`
- Atributy a metody deklarované **s modifikátorem** `static`
  - Jedná se o položky třídy – statické položky (atributy, metody)
  - Lze je použít bez vazby na jakoukoli instanci třídy (objekt)
  - Statické atributy – atributy jsou společné všem objektům – například chceme-li uchovávat celkový počet doposud vytvořených bodů
  - Statické metody
    - Lze použít bez vytváření instance (objektu)
    - Ve statické metodě nelze použít `this`
    - Ve statické metodě lze přistupovat pouze ke statickým položkám třídy – používat statické atributy a volat statické metody
- **Knihovní třída**
  - Neslouží pro vytváření instancí, obsahuje pouze statické položky
  - Privátní konstruktor bez parametrů
  - Deklarace s modifikátorem `final` pro zamezení odvození jiných tříd od této třídy

Příklad

```
package bodytools;

  public class Bod {
    private static int pocetBodu = 0;
    private double x, y;
    private double id;
    private String name;

    public Bod(double x, double y) {
      this.x = x;
      this.y = y;
      id = pocetBodu++;
      name = “Bod” + id;
    }

    public double getX() {return x;}
    public double getY() {return y;}
    public String getName() {return name;}

    public static int getPocetBodu() {
      return pocetBodu;
    }
}
```

- Třída Bod
  - Statický atribut pro uchování celkového počtu již vytvořených bodů
  - Instanční atributy uchovávající číselný a jmenný identifikátor každého vytvořeného bodu
  - Třída poskytuje informaci o celkovém počtu již vytvořených bodů – příslušná metoda je statická – lze ji používat i bez objektů (aniž máme k dispozici konkrétní objekt)
  - Objekty mohou poskytnout svůj jmenný identifikátor
  - Všechny doposud uvedené varianty třídy Bod – instance (objekty) reprezentují konstantní body – nelze měnit hodnoty atributů – třída neposkytuje prostředky pro změnu atributů

Přístup k privátním atributům

- Čtení privátního atributu
  - Metoda poskytující hodnotu atributu „*getter*“
- Změna hodnoty privátního atributu
  - Metoda zajišťující přiřazení nové hodnoty atributu „*setter*“

```
package bodytools;

  public class Bod {
    private double x, y;

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

    public double getVzd(){
      return Math.sqrt(x*x + y*y);
    }

    public double getX() {return x;}
    public double getY() {return y;}

    public void setX(double x) {
      this.x = x;
    }
    public void setY(double y) {
      this.y = y;
    }
}
```

Pole objektů

- Výpočet obvodu n-úhelníka – využití instancí třídy Bod pro uchování jednoho bodu a pro výpočet vzdáleností mezi body

```
int n = sc.nextInt();
// deklarace promenne pro  pole bodu, prideleni pameti poli
Bod[] body = new Bod[n+1];       // vytvoreni pole

// nacteni souradnic bodu
// prideleni pameti objektum – instancim tridy Bod
for (int i = 0; i < body.length-1; i++) {
  body[i] = new Bod(sc.nextDouble(), sc.nextDouble());  // vytvoreni bodu
}
body[body.length-1] = body[0];

// vypocet obvodu n-uhelnika
double obvod = 0;
for (int i = 1; i < body.length; i++) {
  obvod += body[i].getVzd(body[i-1]);
}
System.out.format("Obvod: %f ",obvod);
```