

# Objektově orientované programování v Javě

Učební pomůcka od [ITnetwork.cz](http://ITnetwork.cz) - Přehled OOP syntaxe

## Třída

```
class Uzivatel {  
    public String jmeno;  
    private int vek = 42;  
  
    public void pozdrav() {  
        System.out.println("Ahoj");  
    }  
}
```

// Vytváříme novou instanci  
Uzivatel jan = new Uzivatel();  
  
// Používáme instanci  
jan.jmeno = "Jan Nový";  
jan.pozdrav();

Do třídy píšeme proměnné jako její **atributy** a funkce jako její **metody**.  
Ty se pak ale vztahují jen k této konkrétní třídě.

## Konstruktor

```
public Uzivatel(String jmeno) { // Použití  
    this.jmeno = jmeno;  
} // Vytvoření instance volá konstruktor  
Uzivatel jan = new Uzivatel("Jan Nový");
```

Metoda bez návratového typu, která se jmenuje vždy stejně jako třída.

## Gettery a Settery

```
// Getter  
public String getJmeno() {  
    return jmeno;  
}  
  
// Setter  
public void setVek(int vek) {  
    this.vek = vek;  
}
```

Jakmile potřebujeme vystavit **atribut veřejně**, uděláme ho **vždy privátní** a **tvoříme gettery/settery**. Atributy jen pro **vnitřní účely** děláme **privátní**.

## Dědičnost

```
class Programator extends Uzivatel {  
    public String jazyk;  
}
```

Dědičností přejímáme atributy a metody předka dle modifikátorů přístupu.

## Modifikátory přístupu

- **public**
  - Neomezený přístup
- **protected**
  - Přístup omezen na třídu obsahující daný prvek, její potomky a **třídy ve stejném balíčku**
- **private**
  - Přístup omezen na třídu obsahující daný prvek
- **(default) - bez ničeho (výchozí)**
  - Přístup omezen pouze na třídu obsahující daný prvek a třídy ve stejném balíčku

## Převod na text

```
@Override  
public String toString() {  
    return jmeno;  
} // Použití  
System.out.println(jan);
```

## Statika (static)

```
private static int delkaHesla;  
public static double stupneNaRadiany(double stupne) {  
    return 3.14 * stupne / 180;  
}  
Matematika.stupneNaRadiany(360); // Použití
```

Patří třídě jako takové, jsou **společné** pro všechny její instance.

## Konstanty

```
public static final PLNOLETOST = 18;  
Uzivatel.PLNOLETOST; // Použít
```

## Rozhraní (interface)

```
// Implementace  
interface Vykreslitelny {  
    void vykresli();  
}  
class Obdelnik implements Vykreslitelny {  
    public void vykresli() { /* ... */ }}
```

Předpisuje nám metody, která by měla třída implementovat.

Třída může implementovat i více interface, ale dědit smí jen z jedné třídy.

## Abstraktní třídy (abstract)

```
abstract class OvladaciPrvek {
```

```
    public abstract void kliknuti();  
    // Zbytek můžeme implementovat...  
}
```

Kombinace klasické třídy a rozhraní.

## Výčtové typy (enum)

```
enum BarvaOci { CERNA, HNEDA, ZELENA, MODRA }  
BarvaOci barva = BarvaOci.ZELENA; // Použít
```

## Balíčky (package)

```
package clanky.administrace; // Použít  
import clanky.administrace.Editor;  
public class Editor { /* ... */ }
```