15. Polymorfismus

Polymorfismus v souvislosti se vztahem mezi objekty Předka a Potomka

- Týká se schopnosti objektů různých tříd vykonávat stejnou operaci nebo metodu.
- To znamená, že když máme například třídy "Předek" a "Potomek", které oba implementují stejnou metodu s názvem "vykonatOperaci()", můžeme s nimi pracovat stejně, jako by byly stejné třídy

```
// Vytvoření abstraktní třídy "Zvíře"
abstract class Zvíře {
    // Definice abstraktní metody "vydatZvuk"
    public abstract void vydatZvuk();
}

// Třída "Pes" dědící od třídy "Zvíře"
class Pes extends Zvíře {
    // Implementace metody "vydatZvuk" pro třídu "Pes"
    public void vydatZvuk() {
        System.out.println("Haf!");
    }
}

// Třída "Kočka" dědící od třídy "Zvíře"
class Kočka extends Zvíře {
    // Implementace metody "vydatZvuk" pro třídu "Kočka"
    public void vydatZvuk() {
        System.out.println("Mňau!");
    }
}
```

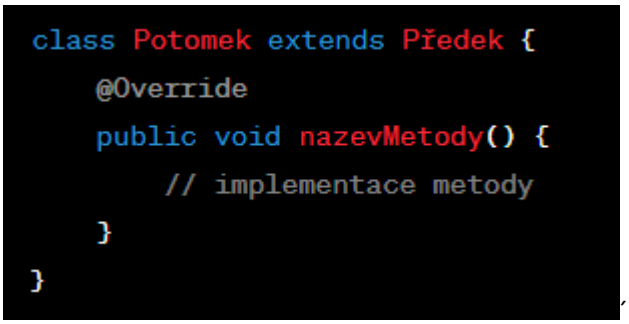
```
class Main {
    public static void main(String[] args) {
        // Vytvoření objektů tříd "Pes" a "Kočka"
        Pes pes = new Pes();
        Kočka kočka = new Kočka();

        // Volání metody "vydatZvuk" na objektech tříd "Pes" a "Kočka"
        pes.vydatZvuk();
        kočka.vydatZvuk();

        // Přiřazení objektu třídy "Kočka" do proměnné typu "Zvíře"
        Zvíře zvíře = kočka;

        // Volání metody "vydatZvuk" na objektu třídy "Kočka", uloženém
        zvíře.vydatZvuk();
    }
}
```

Jazyková konstrukce (zápis) přepsání metody Předka ve třídě Potomek



- Klíčové slovo **extends** značí, že třída **Potomek** je odvozena od třídy **Předek**. Pomocí anotace **@Override** označujeme, že metoda **nazevMetody** je přepsána z třídy **Předek**
- Poté následuje implementace metody v třídě **Potomek**. Při volání této metody na objektu třídy **Potomek** bude volána implementace v této třídě namísto implementace v třídě **Předek**

Rozdíl mezi přepsáním (redefinování) a přetížením metody

- Přepsání
  - Znamená vytvoření nové implementace metody ve třídě potomka (subtřídě), která nahrazuje implementaci stejnojmenné metody v třídě předka (supertřídě)
  - Přepsání se provádí tehdy, když chceme změnit chování metody v potomkovské třídě, ale ponechat stejné jméno, parametry a návratový typ
  - Při volání této metody na objektu potomka bude volána implementace metody v potomkovské třídě namísto implementace v předkovské třídě
- Přetížení
  - Znamená vytvoření více metod se stejným jménem, ale různými parametry v jedné třídě
  - Při volání metody se pak vybere ta, která nejlépe odpovídá typům argumentů předaných při volání
  - Při přetížení se změní počet nebo typ parametrů, ale návratový typ a jméno metody zůstávají stejné

Rozhraní a souvislost s přepsáním metody

- Rozhraní (interface) v Javě je abstraktní třída, která definuje sadu metod, které musí být implementovány v třídách, které implementují toto rozhraní
- Souvislost rozhraní s přepsáním metody
  - Spočívá v tom, že pokud třída implementuje rozhraní obsahující určitou metodu, musí tuto metodu implementovat, jinak tato třída není schopna implementovat toto rozhraní
- V Javě můžeme implementovat více rozhraní, ale dědit můžeme právě jednu třídu

Souvislost rozhraní s abstraktními datovými typy

- Abstraktní třídy jsou třídy, které obsahují alespoň jednu abstraktní metodu, tedy metodu bez konkrétní implementace. Tyto třídy nemohou být instanciovány přímo, ale slouží jako předek pro konkrétní třídy, které implementují jejich abstraktní metody. Abstraktní třídy mohou obsahovat i metody s konkrétní implementací a vlastnosti, které jsou zděděny do potomků
- Rozhraní jsou podobné abstraktním třídám, ale jsou zcela abstraktní, tedy neobsahují žádnou konkrétní implementaci metod. Rozhraní definují pouze signatury metod, které musí být implementovány v třídách, které rozhraní implementují

Prakticky:

- Implementace rozhraní, popsání významných rozhraní v Java (List, Comparable, ActionListener…)