# 09. Principy OOP (dědičnost, zapouzdření, polymorfismus)

## Dědičnost

Dědičnost se v Javě provádí pomocí klíčového slova extends v deklaraci třídy.

#### public class Potomek extends Rodič {}

Potomek je třída, jenž dědí od jiné třídy (rodiče či předka). Dědí se všechny metody a proměnné, jenž jsou v daný okamžik viditelné.

Pokud se neuvede předek, tak je třída potomek java.lang.Object, dědí jeho defaultní metody (equals(), toString(), finalize()…)

Pokud se má v konstruktoru potomka zavolat konstruktor předka, použije se klíčové slovo super(parametry). Volání konstruktoru předka musí být vždy na začátku konstruktoru.

#### public class Potomek extends Rodič {

#### public Potomek(params) {

#### super(params\_predka); //konstruktor předka

#### super.toString(); //lze volat metodu předka

#### inicializace\_vlastnich\_dat;

#### }

#### }

Při přepisování metody, která se dědí od předka, metoda se normálně přepíše s anotací **@Override** 🡪 upozornění překladače, že se daná metoda přepisuje.

#### @Override

#### public String toString {

#### return “Ahoj”;

#### }

Nelze dědit (přepisovat) metody, které mají v deklaraci třídy final.

### Specializace

Jedním z nejčastějších důvodů pro použití dědičnosti je specializace existujících tříd a objektů. Při inicializaci získává třída nové datové atributy a chování proti původní třídě 🡪 Vytvářejí se potomci z rodičovské třídy.

### Generalizace

Opak specializace, při generalizaci jsou potomci (určité objekty), pro které se vytváří společný rodič.

## Zapouzdření

Princip zapouzdření (Encapsulation) chrání data proti přístupu zvenčí.

Ostatní třídy mohou z objektu získávat informace pomocí getterů a některé i měnit pomocí pomocí setterů. Vše funguje tak, že datové složky nejsou viditelné pro ostatní třídy. Této neviditelnosti se dosáhne pomocí private modifikátorů přístupu. Třídy a objekty nemají k datovým složkám přístup a nemohou je měnit.

Měnit se dají jen pomocí setterů a to tak, že si setter novou hodnotu nejprve zkontroluje a až poté hodnotu přiřadí. V případě že na hodnotě závisí nějaká jiná vlastnost, tak vyvolá metodu dané vlastnosti a pozmění ji.

## Polymorfismus

Zastoupení. Vlastnost, která umožňuje zastoupit objekt, kde je očekáván rodič, potomkem. (Tam, kde se očekává rodič, lze použít potomka 🡪 umí vše co rodič).

Podstatou polymorfismu je metoda (metody), kterou mají všichni potomci definovanou se stejnou hlavičkou, ale jiným tělem. V Javě nikdy nelze dědit od více tříd zároveň (na rozdíl od C++). K tomuto účelu slouží rozhraní (interface).

## Modifikátory přístupu

Znepřístupnění datových složek se provádí pomocí modifikátoru přístupu. Celkem existují čtyři modifikátory. Modifikátory přístupu se umísťují hned na začátek deklarace statických a instančních datových složek, na začátek deklarace tříd i metod. Prakticky vše co má něco společného s OOP, má modifikátor přístupu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Modifikátor | Třída | Balíček | Podtřídy | Neomezeně |
| Public | Ano | Ano | Ano | Ano |
| Protected | Ano | Ano | Ano | Ne |
| Bez modifikátoru | Ano | Ano | Ne | Ne |
| Private | Ano | Ne | Ne | Ne |

Sloupec třída znamená, jestli je metoda (proměnná) vidět z jiné třídy.

Sloupec balíček určuje, zda je třída vidět z jiného balíčku.

Sloupec podtřída uvádí, zda je metoda (proměnná) vidět svými podtřídami, které jsou v jiném balíčku.

Poslední sloupec se vztahuje pro ostatní balíčky.