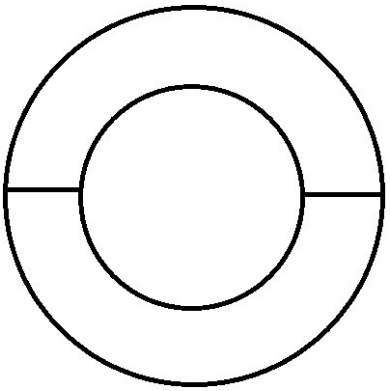
Téma 9: Principy OOP (dědičnost, zapouzdření, polymorfismus)

Výhoda objektově orientovaného programování spočívá v tak zvaném **zapouzdření**. Princip zapouzdření spočívá v tom, že k datovým složkám objektu není přístup od jinu, než z vlastní třídy. Ostatní třídy mohou s objektu získávat informace a některé i měnit pomocí getrů a setrů – tedy metod pro přístup k vlastnostem objektu. Vše funguje tak že datové složky nejsou viditelné pro ostatní třídy. Této neviditelnosti se dosáhne pomocí modifikátorů přístupu (viz dále).



Třídy a objekty tedy nemají k datovým složkám přístup a nemohou je tedy měnit. Měnit se dají jen pomocí setrů, a to tak že si novou hodnotu metoda nejprve zkontroluje a poté až hodnotu přiřadí. V případě že na hodnotě závisí nějaká jiná vlastnost, tak vyvolá metodu dané vlastnosti a pozmění ji. Vše ukazuje následující obrázek.

Znepřístupnění datových složek se dělá pomocí **modifikátoru přístupu**.
Zatím jsme se setkali jen s jedním modifikátorem a to public. 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 z OOP tak má modifikátor přístupu. Všechny čtyři zachycuje následující tabulka

Modifikátor přístupu	**Třída**	**Balíček**	**Podtřídy**	**Neomezeně**
public	Ano	Ano	Ano	Ano
private	Ano	Ne	Ne	Ne
protected	Ano	Ano	Ano	Ne
bez modifikátoru	Ano	Ano	N	N

V tabulce je výpis všech modifikátorů přístupu a jejich vlastností. Sloupec třída znamená, že metoda nebo proměnná označená daným klíčovým slovem je nebo není vidět vně vlastní třídy. Sloupec balíček určuje, zda je třída vidět vně svého balíčku, o balíčkách bude řeč jindy, zatím stačí vědět, že balíček udržuje několik tříd pohromadě, principem připomíná složku ve Windows. Sloupec podtřída uvádí, zda je metoda či proměnná vidět svými podtřídami, které nejsou ve stejném balíčku. No a poslední sloupec se stahuje pro ostatní balíčky. Nejčastěji používaný modifikátory přístupu jsou public a private, v tabulce jsou označeny tučně. Ostatní modifikátory se moc nepoužívají.

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

```
public class Potomek extends Rodič {
}
```

Kde potomek nebo také Dceřiná třída je třída, jenž dětí od Matčiny třídy (rodiče nebo předka). Dědí se všechny metody a proměnné, jenž jest v daný okamžik viditelné. Viz modifikátory

přístupu. Pokud chci v konstruktoru potomka zavolat konstruktor předka, použiji klíčové slovo super (parametry konstruktoru). Volání konstruktoru předka musí být vždy na začátku konstruktoru.

```
public class Potomek extends Rodič {
        public Potomek(params) {
        super(params_predka);
        inicializace_vlastnich_dat;
        }
}
```

Pokud chci přepsat metodu, jenž dědím od předka metodu normálně přepíši s tím že překladač upozorním na záměr pomocí anotace @Override.

```
@Override
public String toString {
return "Ahoj";
}
```

Polymorfismus neboli více násobné dědění v Javě není umožněno na rozdíl třeba od C++. V Javě je možno dědění zřetězit, ovšem nikdy nemohu dědit od více tříd zároveň. K tomuto účelu slouží interfaces.