Mělo by zde být vše, jen tím rozhraním si nejsem moc jistý.

Principy objektově orientovaného programování

základní jednotky OOP:

Třída

vzor definován uživatelem, který může obsahovat metody i proměnné (např. třída clovek, obsahující proměnné, např:

```
class Clovek{
string jmeno;
int vek;}
```

a metody, např:

```
dychej(){...};
rekni(string co){...};
}
```

Objekt

instance třídy, vzájemně se liší svými vlastnostmi(atributy) a mají stejné metody jako třída

např .

```
clovek kaja{
jmeno = „Karel Novak“;
```

```
vek = 55; }
```

a můžeme použít metodu

```
kaja.rekni(„Mam hlad“);

kaja.dychej();
```

můžeme si vytvořit více instancí:

```
clovek honza{

jmeno = „Jan Cerny“;

vek = 10;}
```

a opět můžeme použít stejné metody…

Základní pilíře OOP

OOP stojí na třech základních pilířích:

Zapouzdření

umožňuje nám skrýt ty metody a atributy, ke kterým nechceme, aby bylo možné přistupovat z vnějšku. Např. u třídy clovek můžeme proměnnou datumNarozeni nechat na skrytou (private) a tím zabráníme, aby mohla být změněna z vnějšku. Tuto vlastnost definujeme pomocí viditelnosti, která může nabývat hodnot public, protected, private. Díky tomu můžou objekty fungovat jako tzv. černé skříňky: můžeme jim dát vstup, přičemž dostaneme nějaký výstup, ale nemusíme se vůbec zajímat o to, jak to vevnitř funguje. Na tomto principu funguje rozhraní(interface), jehož vlastnosti poté implementujeme do různých tříd.

Dědičnost

Usnadňuje vytváření podobných tříd. Z rodičovské třídy si vezme podtřída všechny metody a proměnné(resp. ty, které mají nastavenou viditelnost na public nebo protected) a může si navíc přidat svoje.

např:

```
class Zpevak extends Clovek {(říkáme, že třída pro zpěváky je potomkem člověka)

zpivej{};

tancuj{}; }
```

Polymorfismus

Umožňuje použít jednotné rozhraní pro práci s různými typy objektů např:

instance třídy *mladyClovek* bude metodu *presunSe();* vykonávat za pomocí metody *chod*, ale instance třídy *staryClovek* bude tu samou metodu *presunSe();* vykonávat za pomocí *chodOHoli*

To znamená, že i když každý objekt tuto metodu vykonává jinak, z vnějšího hlediska se tváří stejně a my nemusíme tedy přemýšlet, jak přesně toho u různých objektů docílit.

Pod pojmem polymorfismus můžeme také rozumět **Přetěžování metod** - to znamená že metoda může fungovat více různými způsoby, které se rozliší podle druhu a počtu parametrů. Např metoda *rekni(string vyrok);* umožní objektu clovek říct nějaký výrok jen tak do vzduchu, zatímco metoda *rekni(string vyrok; clovek prijemce; zpusobHlasitosti hlasitost)* umožní tomu samému objektu říct výrok konkrétnímu příjemci a zvolenou hlasitostí.

Vícenásobná dědičnost

Možnost potomka dědit z více tříd zároveň. Její použití se příliš neosvědčilo, kvůli riziku konfliktu jmen, konstruktorů, a dalším problémům. Lze ji využít např. v C++