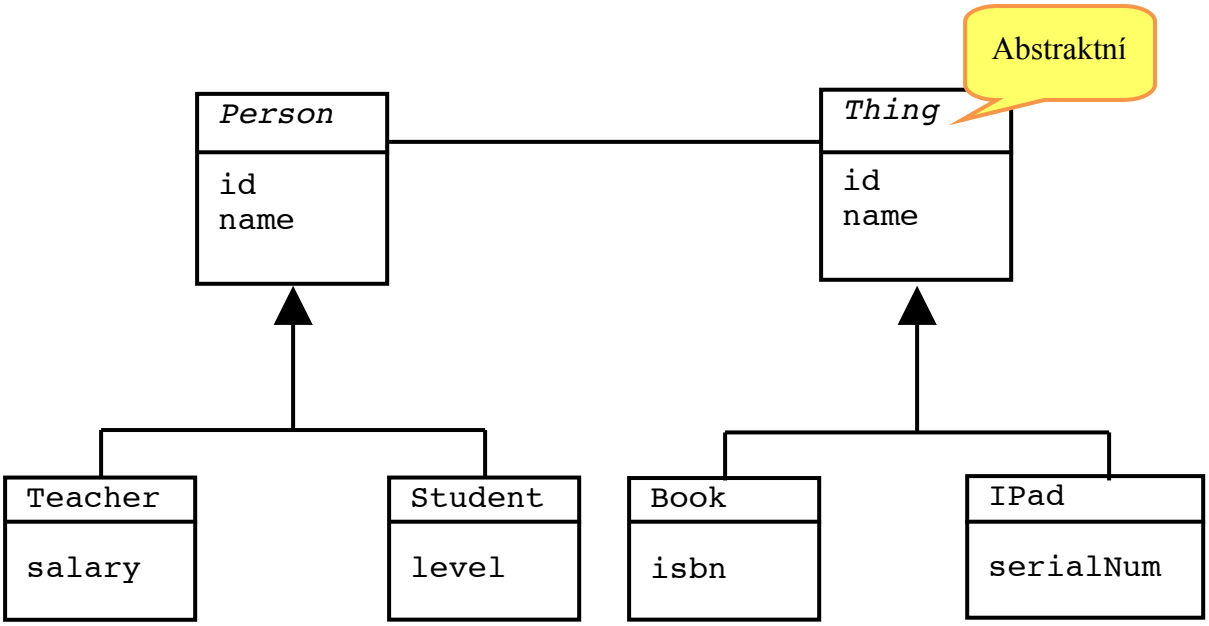
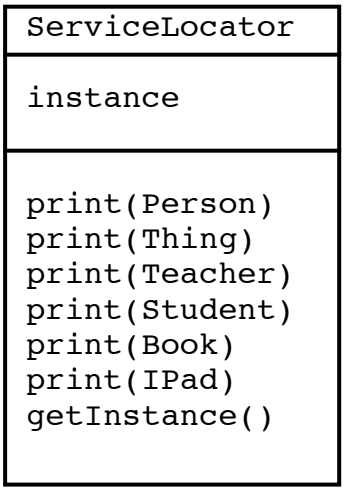
Singleton

Mějme systém, kde existují dvě skupiny osob, studenti a učitelé. Dále mějme skupinu předmětů, které mohou být drženy osobou. Tyto předměty mohou být například Kniha nebo IPad. Viz obrázek níže.

Vytvořme třídu `ServiceLocator`, tato třída umí vytisknout detaily o osobách a věcech.

Třídu `ServiceLocator` implementujte jako singleton.





Factory Method

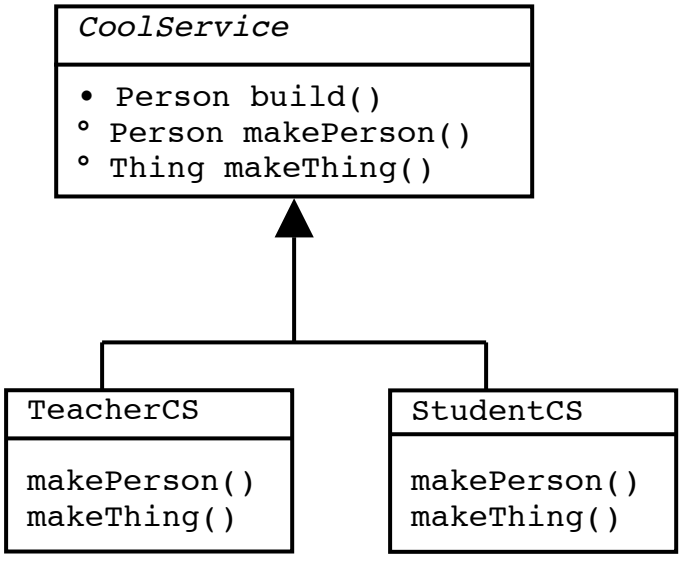
1. Vytvořte třídu `CoolService`, ta bude mít methody pro tvorbu instancí z přechozího obrázku.

2. Vytvořte metody pro `CoolService`
`abstract protected Person makePerson()` a
`abstract protected Thing makeThing()`

`CoolService` dale implementuje metodu `build()`

```
final public Person build() {
     Person p = makePerson();
    Thing t = makeThing();
     p.setThing(t);
     t.setPerson(p);
     return p;
}
```

3. Dále vytvořte `TeacherCoolService`, který dědí od `CoolService` a specializuje se na `Teacher` a `Book`.
4. Podobně vytvořte `StudentCoolService`, který dědí od `CoolService` a specializuje se na `Student` a `IPad`.



Abstract Factory

(a) `ServiceLocator` a `CoolService` jsou oddelene, ale zkusme je integrovat.

1. metoda build se presune do `ServiceLocatoru`
2. `ServiceLocator` bude mit v konstruktoru parameter `CoolService` a ten bude hrát roli AbstractFactory
3. Dle volby `CoolService` se bude medota `ServiceLocator #build()` chovat jinak

(b) Zobecněme `CoolService` na `GenericCoolService<A,B>`, který dědí od `CoolService` a je generický.

Prototype

Umožněte klonování objektů `Person a Thing`, vytvořte hluboké kopie. A tiskněte je na obrazovku. Nezapomeňte, použít metodu `initialize` na inicializaci.