**Наследяемост в ООП . Дефиниране на йерархията. Достъп до методи и данни**

**на различни нива**

Характерно за наследяемостта е, че полетата и методите (без private) се копрат и в

наследяващия клас, което предотвратява повторното писане на телата им в

наследяващите класове.

Когато един клас наследява друг се използва обшата форма:

<сlass име-производен-клас> : <тип-достъп> <име-базов-клас>

*{ … };*

Тук тип-достъп е една от трите ключови думи: public, private или protected.

Спецификаторът за достъп определя по какъв начин елементите на базовия клас се

наследяват от производния клас. Когато спецификаторът е public, всички public членове

на базовия клас стават public членове и на производния клас. Ако спецификаторът е

private, всички public членове на базовия клас стават private за производния клас. И в

двата случая всички private членове на базовия клас си остават private за него и не са

достъпни от производния клас.

Спецификаторът за достъп protected е еквивалентен на спецификатора private с едно

единствено изключение – protected членовете на базовия клас са достъпни за членове на

всеки на всеки клас, който е производен на базовия клас. Извън базовия клас или

неговите производни, protected членовете са недостъпни. Когатоедин protected член на

клас се наследи като public от производен клас, той става protected член на производния

клас. Ако базовия клас се наследи като private, един protected член на базовия клас

става private член на производния клас. Един базов клас може да бъде наследен и като

protected от един производен клас. В такъв случай public и protected членовете на

базовия клас стават protected членове на производния клас.

Ако отсъства спецификатор, то по подразбиране той е private за клас и public за

структора.

Тъй като конструкторът не се наследява автоматично, трябва да го предефинираме

наново или да го наследим от базовия клас. Синтаксисът за наследяване е:

<производен-конструктор>(<списък-аргументи>): <базов-клас>(<списък-аргументи>)

{ …}

Същото се отнася и за деструкторите.

Съществуват два начина, по които един производен клас може да наследи повече от

един базов клас. Първо, един производен клас може да бъде използван като базов за

друг производен клас, като по този начин се създава многостепенна класова йерархия.

В този случай за оригиналния базов клас се казва, че е индиректен базов клас на втория

производен клас. Второ, един производен клас може директно да наследява повече от

един базов клас. В този случай два или повече базови класа се използват за създаването

на производен клас.

Когато един производен клас директно наследява множество базови класове, се

изреждат със запетайки така:

**сlass** <име-производен-клас>: <тип-достъп> <база1>, <тип-достъп> <база2>, ... N

{ ... };

В случая когато производния клас наследява директно множество базови класове е

възможно да възникне проблем, а именно да се наследи два или повече пъти даден

базов клас, намиращ се по-нагоре в йерархията. Проблемът може да се реши чрез

наследяване на базовия клас като виртуален. Това предотвратява присъствието на две

или повече копия на базовия клас във всеки следващ индиректен производен клас.

Когато се създава подобен производен клас, ключовата дума virtual се поставя пред

спецификатора за достъп до базовия клас.