# 5 - Objektové programování I

# Strukturované a objektové programování

* = paradigmata (programovací styly)
* Strukturované: Využívá skoků, podmínek, cyklů, podprogramů – Pascal, C
* Objektové: Založeno na objektech obsahujících data (atributy/vlastnosti) a kód (metody) – C#, Java

# Objekt

* => instance
* Místo v paměti alokované a nakonfigurované třídou/strukturou
* Program může vytvořit několik objektů stejné třídy/struktury
* Může být uložen v proměnné/kolekci
* V C# je třída Object nejvyšší základní třída

## Dědičnost

* Tvoření nových datových struktur na základě starých
* Dědící třída používá, rozšiřuje nebo modifikuje chování rodičovské třídy
* V C# lze mít pouze jednoho rodiče
* Konstruktory a destruktory se nedědí, každá třída si musí definovat svůj vlastní (zavolání původního konstruktoru za pomocí :base())
* Podporuje koncept znuvupoužitelnosti, méně redundance
* Ze sealed třídy nelze dědit

## Zapouzdření

* = Zabalení dat a metod a restrikce přístupu k nim
* Skrytí dat ve třídě
* private – pouze vlastnosti z dané třídy
* private protected - ke členům třídy mají přístup třídy, které z ní dědí a nachází se ve stejném sestavení (stejné dll/exe)
* protected – vlastnosti třídy a potomků
* internal - ke členům třídy mají přístup všechny třídy ve stejném sestavení
* protected internal – Ke členům třídy mají přístup všechny třídy ve stejném sestavení nebo třídy, které z ní dědí
* public – Ke členům třídy mají přístup všechny třídy

## Polymorfismus

* Podstatou jsou metody, které mají všichni potomci definované se stejnou hlavičkou, ale jiným tělem (jiná implementace "stejných" metod)
* V C# virtual + override

## Abstrakce

* Abstraktní třída = nejde od ní vytvářet instance, předek

# Třída

* Referenční datový typ
* Vzor pro tvorbu objektů

## Abstraktní třída

* Modifikátor abstract může používat třída, metoda, vlastnost
* Abstraktní třída je myšlena pouze jako základ pro ostatní třídy, nelze z ní vytvořit instanci
* Abstraktní členy lže použít pouze v abstraktní třídě, mají pouze hlavičku, tělo je definováno v dědících třídách (pomocí override)

## Statická třída

* Nemohou z ní být tvořeny instance, nelze z ní dědit
* Modifikátor static u třídy vyžaduje, aby měla všechny členy také statické
* U této třídy nelze tedy použít klíčové slovo new, ke členům třídy se přistupuje přes jméno třídy samostatně

## Statický konstruktor

* Jakákoli třída může mít statický konstruktor (i vedle „normálního“ konstruktoru)
* Nemůže mít vstupní parametry
* Volá se automaticky (před vytvořením první instance nebo použitím statického členu), aby inicializoval statické atributy (nebo vlastnosti) dané třídy; k nestatickým atributům přístup nemá

## Zapečetěná třída

* Slovo sealed
* Nelze ze třídy dědit
* Modifikátor lze také použít na override metody, aby již nešly přepisovat v dalších potomcích

## Částečná třída

* Slovo partial
* Definice tříd, struktur a rozhraní lze rozdělit do více zdrojových souborů
* Všechny části se kombinují při kompilaci

## Privátní konstruktor

* Pokud má třída privátní konstruktor, nemohou z ní ostatní třídy (kromě nested potomků) vytvářet instance
* Použití, pokud má třída metodu, která vrací její instanci (factory metod)
* Nebo pokud třída nemá žádné instance methods (~metody vyžadující vytvoření instance pro své volání), příkladem je knihovna Math
* (Pokud má třída všechny členy statické, doporučuje se i třídu samotnou udělat statickou)

## Rozhraní

* Neboli jak je objekt viditelný zvenku
* Obsahuje výčet metod, vlastností, indexerů, událostí (pouze hlavička bez těla (od C# 8.0 může mít i tělo)
* Všechny členy jsou automaticky veřejné
* Třída implementující rozhraní musí obsahovat všechny jeho členy
* Nelze z něj vytvořit instanci
* V C# může třída dědit jen z jedné třídy, může ale implementovat více rozhraní. Rozhraní také mohou implementovat rozhraní.