**třída:**

**šablona:**na řešení určitého problemu

**Dědění:**

schopnost vytvořit třídu z jiné třídy a rozšířít chování

nebo stavovou složku této třídy

možnost přetížení metod předka

důležité vlastnosti dědění:

potomek rozšiřuje předka – obsahuje vše co předek a něco

navíc

odvozená třída nemůže měnit definici poděděného prvku

dědí se vše kromě konstruktorů a destruktorů

přístupnost prvku v odvozené třídě záleží na definici v třídě

základní

prvky odvozené třídy mohou být předefinovány (override)

**kdo v dědění vystupuje a jak se to nazývá:**

-nadtřída a podtřída

**Rozhraní Interface**

množina definicí metod a vlastností

název začíná I

pokud definuje jen jednu metodu, tak pojmenováno I…able

implementace rozhraní – děděním a implementací

jeho metod a vlastností

rozhraní může dědit z rozhraní, ale ne z třídy

třída využívající rozhraní musí implementovat vše co

je jím definováno

metody mohou být virtuální

**IF** podmíněný výraz

if ( „podmínka“ ) if-tělo [else else-tělo]

**Abstraktní třída**

něco mezi rozhraním a klasickou třídou

definuje stav a chování

na rozdíl od rozhraní však může pro libovolnou část

definovaných prvků obsahovat i implementaci

**Namespace**-utváří „kontext“ pro pojmenovávání

umožňují vytvářet třídy stejných názvů jako jsou

obsaženy v .NET Frameworku

**co je interval**

**z ceho se skládá objekt** - 3 složky=stav,chování identita

**co je interface**

**pojmem interface rozumíme**:

**definice příkazu While,cyklu DO a FOR**

Iterace-DO-podmínka je na konci přikazu,While podmínka tělo, For („inicializace“, „podmínka“, „iterace“) tělo

**POLE**

množina objektů stejného typu

délka je dána hodnotou nebo proměnnou při

deklaraci

po deklaraci nelze délku měnit

int[] celaCisla = new int[20];

int delka = 32;

string[] texty = new string[delka];

**přidání odkazem v c#**

**spravné dědění**

**C#**

**Třídy:** šablony pro tvorbu objektů

obsahují implementaci funkcionality programu

klíčové slovo „class“

vytváření instancí

vlastnosti: zprostředkovávají funkcionalitu metod pomocí

syntaxe pro proměnné

statické vs. instanční

klasická vs. zjednodušená syntaxe pro

zpřístupňovací metody

**Konstruktor:**

slouží k řízení inicializace objektu

většinou nastavují výchozí stav vlastností

lze využít k naplnění konkrétních hodnot

přetěžování konstruktorů

více konstruktorů s různými parametry

**Delegáti**

základní kameny událostmi řízeného programování

delegát

definuje která metoda je volána pokud je odpálena událost

událost

oznámení, že byla provedena nějaká akce

**Delegát**

datový typ umožňující referencovat metodu

umožňují „multicasting“

definice delegáta určuje signaturu přiřazené

metody

lze přiřadit pouze metody se shodnou signaturou

**Generické typy**

umožňují využít jednu definici pro několik datových

typů

 maximalizují znovupoužitelnost kódu, typovou

bezpečnost a výkon