Programovanie v jazyku C#

Abstraktné triedy, rozhrania, hierarchia tried

prednáška 4

Ing. Ján Magyar, PhD.

ak. rok. 2024/2025 ZS

Problémy so základným dedením

Čo ak nadtrieda predstavuje iba koncept, z ktorého nepotrebujeme vytvoriť inštanciu?

Čo ak nechceme zadefinovať celú funkcionalitu nadtriedy?

Čo ak v nadtriede nechceme implementovať žiadnu funkcionalitu?

Abstraktná trieda

popis abstraktných konceptov

z abstraktnej triedy nemôžeme vytvoriť inštanciu

garantuje, že neabstraktné podtriedy prepíšu abstraktné metódy

v C# nemusí mať ani jednu abstraktnú metódu (ale mala by)

Abstraktná metóda

nemá telo

iba deklarovaná

musí byť implementovaná v neabstraktných podtriedach

iba v abstraktných triedach

implicitne `virtual`

Kľúčové slovo `abstract`

v C# použiteľné pre triedy, metódy, indexery, udalosti (eventy) a properties

súvisiace kľúčové slovo - `sealed`

Používanie abstraktných tried

pri dedení, definícia hierarchie tried

definícia spoločnej funkcionality v rámci hierarchie

nesmie byť statická

iba priame dedenie

definícia typov parametrov

Konštruktor v abstraktnej triede

abstraktná trieda môže definovať niekoľko konštruktorov

konštruktor podtriedy by vždy mal zavolať konštruktor nadtriedy

konštruktory abstraktnej triedy by mali byť `protected`

Hierarchia tried

umožnená cez dedenie a implementáciu rozhraní

umožňuje jednotný pohľad na objekty rôznych tried

podpora polymorfizmu

zvyčajne hlavná trieda, ktorá je nadtriedou všetkých ostatných tried

v C# je to `System.Object`

Výhody hierarchie tried

polymorfické referencie

znovapoužitie kódu

umožňuje definíciu univerzálnych atribútov a správaní ako porovnávanie, vytvorenie kópií, hašovanie

System.Object

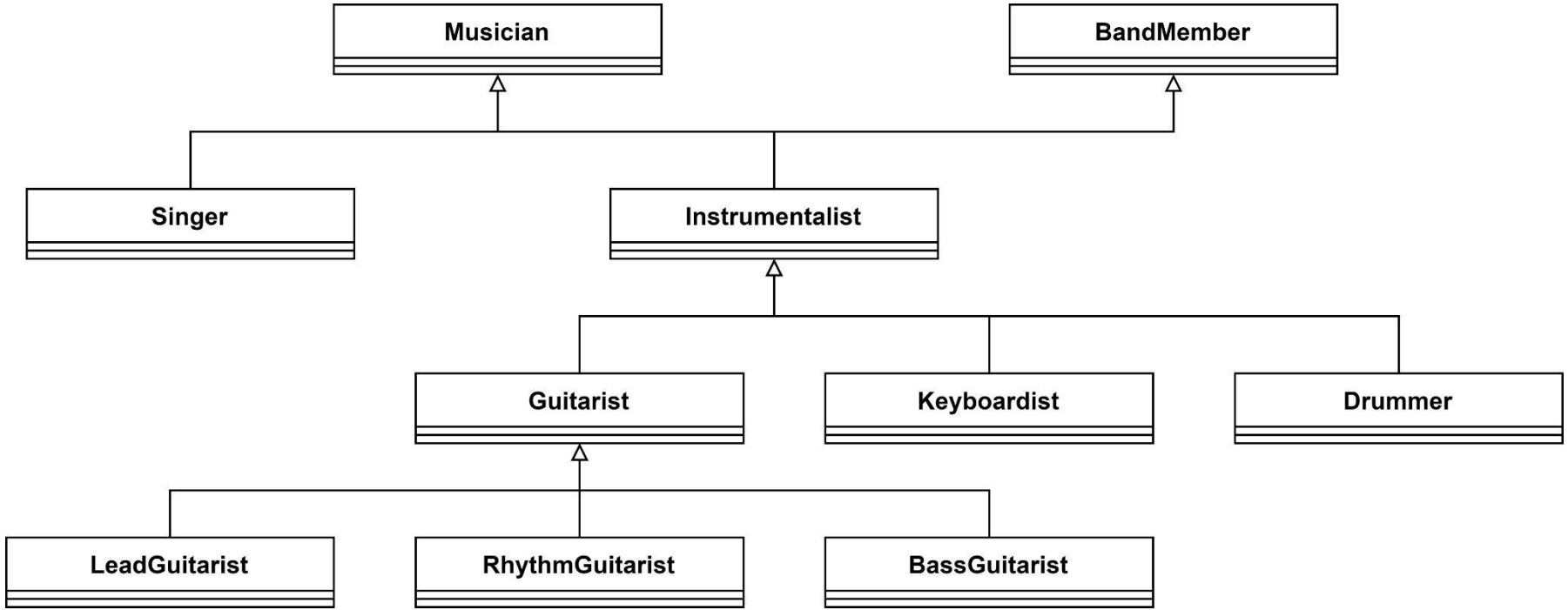
```
Equals(Object)

Finalize()

GetHashCode()

GetType()

MemberwiseClone()

ReferenceEquals(Object, Object)

ToString()
```

Vytvorenie hierarchie tried

cez dedenie (`:`)

každá trieda je automaticky podtriedou `System.Object`

Hierarchia tried - ukážka

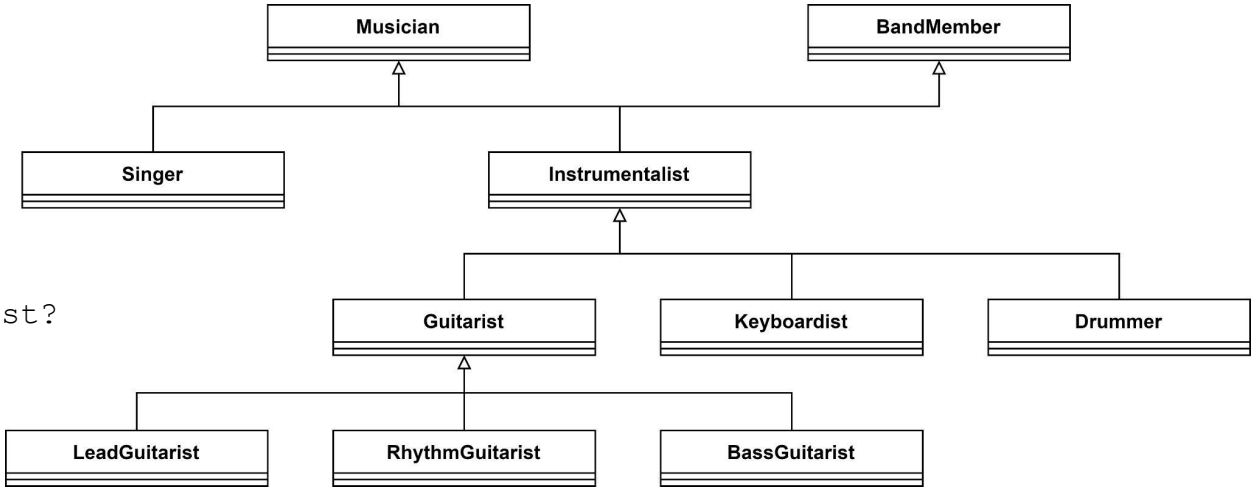


Viacnásobné dedenie

```
public class Musician {
     ...
     public bool PlaysInstrument() {
          return false;
     }
     ...
}
```

```
public class Instrumentalist {
     ...
     public bool PlaysInstrument() {
          return true;
     }
     ...
}
```

SingerGuitarist?



Vyriešenie problému viacnásobného dedenia

umožní viacnásobné dedenie - C++, Python, ...

neumožní viacnásobné dedenie - Java, C#, …

Rozhranie

rozhranie (**interface**) je definícia kontraktu: každá trieda implementujúca rozhranie musí zadefinovať metódy rozhrania

podobné abstraktnej triede

rozhranie obsahuje iba abstraktné metódy a konštanty

Rozhrania v C#

definícia cez kľúčové slovo `interface`

od C# 8.0 môže obsahovať defaultné implementácie (neodporúča sa)

môže definovať statické prvky pre jednu implementáciu spoločnej funkcionality

môže obsahovať metódy, atribúty, indexery, eventy, konštanty, operátory, statické konštruktory, vnorené typy, statické prvky, explicitné modifikátory viditeľnosti

Defaultné metódy rozhrania

nie sú dedené

dostupné iba cez rozhranie

prepísané v implementujúcej triede iba pri explicitnej implementácii

statické prvky sú dostupné cez rozhranie, defaultné implementácie vyžadujú objekt

Aplikácia rozhraní

označenie ako s dedením

rozhranie je súbor všetkých metód, ktoré trieda musí implementovať

trieda môže implementovať viacero rozhraní

rozhranie môže dediť od iného rozhrania (aj od viacerých)

z rozhrania nemôžeme vytvoriť inštanciu

Rozhranie vs typ

typ je rozhranie objektu - definuje vlastnosti a schopnosti

viaceré objekty môžu byť rovnakého typu a ten istý objekt môže mať niekoľko typov - polymorfizmus

objekt je prístupný cez svoje rozhranie

rozhranie je implementácia subtypingu v OOP

Rozhranie vs abstraktná trieda

C# umožňuje dedenie iba od jednej triedy

rozhranie nemôže obsahovať neabstraktné metódy

triedy implementujúce rozhranie musia zadefinovať všetky metódy

abstraktná trieda môže obsahovať abstraktné aj neabstraktné prvky

Rozhranie vs abstraktná trieda

pre rozhranie nemôžeme vytvoriť inštanciu

rozhranie je aj typom

rozhranie je spôsob podpory polymorfizmu

Rozhranie v štruktúre tried

jedna trieda môže implementovať niekoľko rozhraní

rozhranie nie je nadtriedou, pridáva garanciu funkcionality

dedenie je silný vzťah IS-A

implementácia je slabý vzťah IS-KIND-OF

Rozhrania best practice

zvyčajne malý počet metód

názov sa začína na `I`

názov je zvyčajne prídavné meno: `IGroupable`

alebo podstatné meno: `IShape`

Programming to interface

paradigma pre voľnejšie prepojenie prvkov

rozhranie garantuje prítomnosť potrebnej funkcionality

možnosť písať kód bez ohľadu na skutočné typy

prepojenie dvoch tried, ktoré nemajú podobnú štruktúru, ale majú spoločné vlastnosti

otázky?