1. Pole objektů

Ve stávajícím solution Csharp_OOP_cv vytvořte projekt Rozhrani typu Console App. Vytvořte soubor Rozhrani1.cs (namespace změňte z Rozhrani na Rozhrani1). V tomto jednom souboru vytvořte abstraktní třídu Animal, která bude mít jen datovou složku name a konstruktor s parametrem typu name. Dále podřízené třídy Dog, Cat, Turtle, každá z nich bude mít navíc nějakou datovou složku, například Boolean isPedigree (s rodokmenem) u psa i kočky, int speed u želvy. Tyto dodatečné datové složky nebudeme využívat (jsou jen pro větší podobnost s reálnými programy, a taky, aby se třídy trochu lišily), proto konstruktory těchto tří tříd budou mít jen parametr name, budou volat konstruktor třídy Animal. Dále bude pes a kočka mít metodu sound(), která bude vracet řetězec („haf“ a „mňau“). Na konec souboru napište třídu Rozhrani, kde bude jen metoda Mainx. V ní vytvořte několik instancí daných tříd, pak vytvořte pole nazvané animals a instance do pole vložte. Protože pole musí být homogenního typu, musí se jednat o pole typu, který je všem společný, tedy typu Animal.
Potom ve smyčce vypište všechna jejich jména.
Tento první krok projektu je velmi podobný etapě 5 v projektu První. Akorát není tříúrovňová dědičnost a metoda (zde sound) není v bázové třídě, protože želva by ji nemohla podědit. A také si povšimněte, že třídy pes, kocka a zelva jsou až na pár písmenek totožné, takže poté co vytvoříte první z nich, tak ostatní zkopírujete.
Druhý krok: aby byl výpis přehledný, tak u každého jména uveďte datový typ, tedy zda se jedná o psa, kočku či želvu.
Použijeme k tomu jednu ze čtyř metod zděděných od třídy Object (už jsme pracovali s metodou ToString), a to GetType(). Metoda však vrací objekt. Abychom dostali čitelný výsledek, musíme vypsat jen datovou složku Name, tedy dohromady GetType().Name.

1b. V druhé fázi pole zdokonalte tak, aby si program zjistil počet vytvořených instancí a podle toho si nastavil rozměr pole. K tomu ale potřebujeme mít datovou složku countOfAnimals, kterou budeme inkrementovat při vzniku každého zvířete.
Jakého typu tato datová složka bude? A kde se bude inkrementovat?
???
A smyčku for předělejte na foreach.
Druhý krok: vytvořené instance vložte do pole při jeho deklaraci, budou ve složené závorce. v tom případě nepotřebujeme datovou složku countOfAnimals
Třetí krok: Nevyužijeme již vytvořené instance, vytvoříme je (pomocí new) až v {} při deklaraci pole (předchozí způsob deklarace pole zakomentujeme)

2. Protože v zadání není žádná další práce s vytvořenými instancemi, tak není nutné instance nejprve vytvářet a poté vkládat do pole. Lze je vytvářet přímo v poli.
Protože však již nebudou instance zvířat vytvořeny ještě před vznikem pole, tak není v okamžiku vzniku pole zřejmé, jak má být velké. (ani to není třeba vědět kompilátor si to zjistí sám) Proto bod 2 bude založen na bodě 1 (ne 1b).
2. krok: předělat na foreach. Vytvořte a vložte do buněk pole instance přímo při deklaraci pole.

2b Předělat na foreach, místo pole použijte kolekci. Vyjde se z 1b.

3. A nyní modifikace: Kromě třídy a jména máme vypsat ve smyčce i zvuk (sound), který zvířata vydávají. Pokud pouze doplníte ve smyčce volání metody sound(), tak program nepůjde skompilovat, protože třída Animal tuto metodu nemá. A nemůžeme ji do této třídy doplnit, protože třída Turtle ji nemůže podědit.
Takže tudy cesta nepovede, ve smyčce nám sound vypsat nepůjde.
2. krok: zkuste tedy aspoň v metodě Mainx vypsat sound pro konkrétní buňku pole, o které víte, že je v ní např. Pes. Zkuste přetypování a potom operátor as
3. krok: zkuste výše uvedené (přetypování a as) ve smyčce, ale jen pro psa. Tedy před WriteLine musíte doplnit if. Testujte pomocí operátoru is. Pak zkuste i testování pomoci GetType
4. krok: upravit, aby se metoda sound spouštěla i pro kočky: nejprve pomocí elseif a potom pomocí if plus ternárním výrazem

4. Interface
Nezbývá tedy, než použít rozhraní ISoundable (tedy vydává zvuk) které bude metodu sound (bez těla) obsahovat. Rozhraní pak budou implementovat jen zvířata, která nějaký sound vydávají, tedy Dog a Cat. Typ pole pak změníme na ISoundable. Ověřte si, že pak již do pole nepůjde vložit želvu, překladač si to hlídá. Poté již ve smyčce můžeme všechna „zvučící“ zvířata vypsat. Nepůjde ale vypsat jméno (protože rozhraní nesmí mít žádné datové složky).

5. Zkusme tento problém vyřešit. Vždyť pole obsahuje reference na instance tříd Cat a Dog, které tuto datovou složku mají! Jak se k ní ale dostat? Tak, že vytvoříme pomocnou referenci typu Animal, které předáme referenci z buňky pole typu ISoundable. A pro tuto pomocnou referenci pak vypíšeme jméno.
Ale zřejmě to nestačí, kompilátor u přiřazovacího příkazu hlásí
Cannot implicitly convert type ISoundable to Animal. An explicit conversion exists (are you missing a cast)?
A má pravdu, na levé straně je proměnná třídy Animal, a na pravé je buňka pole typu rozhraní ISoundable. Jak to udělat, aby typy na obou stranách byly stejné?
???
2. způsob: nevytváříme dočasnou proměnnou (důležité!!).
A záludný dotaz: jakto, že se nám povedlo vytvořit referenci typu Animal, když je přece tato třída abstraktní?!
???

6. Rozhraní nás již osvobozuje od pout dědičnosti. Tedy můžeme vypsat všechny objekty, které vydávají zvuk. Nikde není řečeno, že to mohou být pouze zvířata. Proto vytvoříme třídu Car, která samozřejmě nebude dědit z třídy Animal. Bude mít datové složky name a speed, bude implementovat rozhraní ISoundable a proto bude mít i metodu sound(). Ověříme si, že proto auto můžeme zařadit do pole a potom zvuky vypsat.
Program se v pořádku skompiloval. Když ale program spustíme, vypíšou se informace o prvních čtyřech objektech (samí psi a kočky), při pokusu o vypsání jména auta však dojde k havárii. Proč, když třída Car tuto datovou složku má?

7. My však nechceme na vypsání jména rezignovat. Zvlášť když víme, že každý objekt v poli jméno má. Rozhraní sice nesmí mít datové složky, může však mít metody. Takže do rozhraní doplníme metodu getName().

A aby byl výpis „celou větou“, chceme dostat výpis např. takovýto: Kočka Micka vydává zvuk mňau. Pes Azor vydává zvuk haf. Auto Škoda vydává zvuk tů. (kdo má čas, vytvoří si pro ten účel ve třídách metody ToString).
2. krok: místo getteru použijeme vlastnost. Využijte vygenerování vlastnosti pomocí Implement interface z místní nabídky interface ISoundable v hlavičce třídy Dog atd.