## Aeroplane

Protože dolet letadla je závislý především na hmotnosti jeho nákladu, je vždy před letem zapotřebí zjistit jeho celkovou hmotnost včetně přepravovaného nákladu.

Vytvořte třídy **Person** (datové složky gender, height, age), **Animal** (má datovou složku type datového typu enum, která nabývá hodnot mammal, reptile, fish, bird nebo insect, při nedostatku času stačí datová složky typu string, která bude mít hodnoty „mammal“ atd. ), **Thing** (datové složky height, length, width). Datové složky by měly mít charakter automatizovaně generovaných vlastností.

Třídy mají metodu **getWeight**(). U osoby se váha vypočítá vzorcem výška mínus 100, u zvířete bude váha konstantní na základě typu (savci – 50 kg, plazi – 10 kg, ryby – 5 kg, ptáci – 3 kg, hmyz – 0,1 kg), u **věci** se spočte z objemu násobeno koeficientem 2.

Do Třídy TestAeroplane umístěte metodu Main. V ní vytvořte z každé třídy několik objektů a zařaďte je do kolekce. Kolekci projděte a zjistěte celkovou váhu nákladu.

## Tax

V příkladu budeme potřebovat tyto třídy:

**House**, **Flat**, **RealEstate** (nemovitost), **Vehicle**, **Car**, **Motorcycle**, **Trolleybus**.

Třídy House a Flat mají společného předka RealEstate. Třídy Car, Motorcycle a Trolleybus jsou odvozené od třídy Vehicle. Ze třídy RealEstate a Vehicle nelze vytvořit instanci.

Třídy mají bezparametrickou metodu **tax**()

U vozidel je situace jednoduchá. Všechna myslitelná vozidla (i třídy, které zatím ještě neexistují) mají hodnotu daně 1500. Pouze trolejbus, protože je ekologický, má sníženou sazbu na úroveň 0.9 základní sazby vozidel.

U nemovitostí je situace složitější. Neexistuje společný základ daně a nemá smysl ho jakkoli definovat, hodnota daně se určuje pro všechny nemovitosti samostatně. Dům má totiž hodnotu daně stanovenu jako 1.27 násobek pořadového dne v měsíci, byt jako 1.15 násobek pořadového čísla dne v roce. Je to divný svět :-) (pozn.: eventuelně si vymyslete jiný vzorec)

Do Třídy **TestTax** umístěte metodu Main. V ní naplňte pole, od každé třídy, kterou umíme instancovat, vytvořte jednu instanci. Ve smyčce pak u všech prvků kolekce vypište následující záznam:

název třídy - sazba daně: \_\_\_ hodnota daně. A samozřejmě spočítejte celkovou daň.

## Triangle

Volně dle P. Herouta: Vytvořte projekt Triangle se třídou Triangle (bude v souboru Triangle1.cs). Bude mít tři datové složky, a to délku jednotlivých stran trojúhelníka, budou typu **float**. Vytvořte tři konstruktory s třemi, dvěma (pro rovnoramenné tr.) a jedním parametrem (pro rovnostranné tr.). Dále pak bude obsahovat metodu (která vrací bool), která bude zjišťovat, zda trojúhelník existuje (tzn. jestli a+b>c a b+c>a a a+c>b). Dále bude mít třída metodu **perimeter()** počítající obvod. Pozor: program musí být řešen objektově, tedy musí být vytvořen objekt, jehož tři metody se pak budou volat. Tedy neřešit statickými metodami, nepředávat parametry v závorce.

Konstruktor pro rovnoramenný a rovnostranný trojhúhelník řešte nejprve tak, že v nich naplníte všechny tři datové složky. Tedy jednotlivé konstruktory budou na sobě nezávislé.

Co se stane, když v Main vytvoříme nějaký trojúhelník a některá strana nebude celé číslo? Proč chyba nastala?

*???*

V **druhém** kroku potom zakomentujte a předělejte na volání konstruktoru se třemi parametry. Takže např. v konstruktoru rovnostranného trojúhelníku zavoláte konstruktor s třemi parametry

**Triangle2**: Jak řešit, když chci zadávat i pravoúhlý trojúhelník a také k tomu chci použít konstruktor s dvěma parametry (odvěsny)?  
**Řešení**: přidejte zděděnou třídu **OrthogonalTriangle**, která bude mít konstruktor zadávající pouze odvěsny. Přepona se sama vypočítá v konstruktoru. V konstruktoru se **nejprve** naplní tři strany třemi přiřazovacími příkazy (příkaz pro přeponu bude obsahovat výpočet dle věty Pythágorovy). **Posléze** předěláme na volání nadřazeného konstruktoru (trojparametrického).   
**Třetí** možnost je volat nejprve dvojparametrický konstruktor a přeponu pak vypočítat ve vlastním konstruktoru

*???*

Při volání nadřazeného konstruktoru je kompilační chyba ..has some invalid arguments. Jak to napravit?

*???*

Ověřte ve funkci Main následovně:

vytvořte pět objektů T1 až T5 (poslední bude pravoúhlý) s následujícími hodnotami v závorce: (100,1,1), (3,4,5), (3,4), (3), prav.tr (3,4). Ověřte také konstruktor, kde některá strana **nebude celé číslo**.

Pak pro první dva objekty zavolejte metodu testující existenci (volejte ji v příkazu **WriteLine** s textem „Existuje trojúhelník T1? „. Odpověď pro T1 bude false, pro T2 bude true.

Dále vypište obvod T4 (vytiskne se 9) a obvod pravoúhlého (vytiskne se 3+4+5=12)

**Triangle3**: Jednoduché: Vyjdeme z Triangle1. Pravoúhlý trojúhelník nebude mít vlastní třídu, upravíme jen konstruktor pro rovnoramenný trojúhelník. Aby se odlišil od rovnoramenného, tak přidáme třetí parametr, který bude boolean (true: je pravoúhlý, false: je rovnoramenný). V kurzu pro pokročilé možno předělat na enum.

**Triangle4 (DÚ)**: variantní zadání: konstruktor bude mít vždy tři parametry. A bude obsahovat metodu getType, která vrátí řetězec „běžný“, „pravoúhlý“, „rovnoramenný“ nebo „rovnostranný“

**Triangle5** **(DÚ)**: Doplňte další konstruktor, bude mít tři parametry typu Point (viz dříve probírané cvičení). Tedy místo délek stran se budou zadávat souřadnice vrcholů. Vytvořte překrytí metody ToString, které bude využívat metodu ToString třídy Bod. Vypíše např: trojúhelník s vrcholy: A=(0,00; 3,00), B=(0,00; 0,00), C=(4,00; 0,00). A také metodu getType (pro kontrolu: pro uvedené vrcholy napíše, že se jedná o pravoúhlý trojúhelník).

## Vehicle and road

1. Vytvořte projekt Vehicle, v něm třídu **Vehicle** s celočíselnými datovými složkami **speed** a **typeOfRoad** (tedy po jakém typu komunikace právě jede: 1. pro obec – limit 50 km/h; 2 pro mimo město – limit 90 km/h; 3 pro dálnici – limit 130 km/h). Vytvořte konstruktor s oběma parametry. Metoda **writeInfo()** napíše informaci o rychlosti a typu komunikace (stačí zatím vypsat číslem 1-3). Pokud jede rychleji, než je příslušný rychlostní limit, pak metoda **writeInfo()** napíše navíc informaci o pokutě. V prvním chvíli asi všichni testují šest podmínek. Dokáže si někdo vystačit s pěti?  
   Řešit pomocí vnořených if a if else, tedy NEpoužívejte logické operátory && a ||  
   V tomtéž souboru vytvořte ještě třídy **Car**, **Motorcycle** a **Tractor**,nebudou mít žádné další datové složky. U traktoru bude metoda **writeInfo()** překryta, napíše info o pokutě vždy při překročení rychlosti 50 km/h (nezávisle na typu komunikace). A taky pokud se traktor ocitne na dálnici.  
   V tomtéž souboru vytvořte ještě třídu **TestVehicle** s metodou Main(). V ní bude vytvořeno několik aut, motorek a traktorů. Některé s překročenou rychlostí, některé bez. Po vytvoření každého vozidla spusťte metodu **writeInfo()** .  
   V **druhém** kroku předělat na složenou podmínku. Přičemž vnořený if se převede na &&, paralelní if na ||.
2. 30“ Vytvořte kopii předchozího souboru cs, pojmenujte Vozidlo2.cs, jmenný prostor přejmenujte na Vozidlo2. Doplňte do metody **pisInfo()** ve třídě **Vehicle** zobrazení třídy (Car, Motorcycle, Tractor), pro kterou je metoda volána.   
   Pokud by se nám nelíbilo, že název třídy začíná na velké písmeno (nebo chtěli použít diakritická znaménka, např. autíčko), tak v **druhém** kroku použijte operátor is.  
   V **třetím** kroku v závislosti na čísle 1-3 se vypíše „v obci“, „mimo obec“ a „na dálnici“. Řešte nejprve strukturou switch.   
   V **čtvrtém** kroku vytvořte pole řetězců s inicializací těmito třemi řetězci. **Program pak vypíše [i-1]. buňku pole.** Kde bude pole deklarováno?  
   ???

## Bod (samostatné)

Pozor: nesplést s cvičením Point (a Shape), proto raději nechávám v češtině.

**Procvičí se zde rozdíl mezi metodou instance a statickou. Dále zapouzdření. A getter, setter a Property**Vytvořte třídu **Bod** s datovými složkami **x**,**y** a tedy i konstruktorem udávajícím souřadnici x a y typu double.   
Vytvořte tři metody, které různým způsobem budou vypočítávat vzdálenost bodu od počátku. Souřadnice x, y jsou vlastně odvěsnami pravoúhlého trojúhelníka, vzdálenost je přeponou. Řešíme Pythagorovou větou. Metody budou výsledek zobrazovat pomocí **WriteLine**, jaký tedy bude jejich návratový typ? ???   
Začneme nejprve nejjednodušší verzí, tedy metodou **vzdXY()**, která nevyužívá objekt bod, předává jako parametr číselné hodnoty x a y. Bude metoda statická? ???   
Potom vytvoříme další metodu **vzdBod()**, té se však bude jako parametr předávat objekt typu bod. Bude metoda statická? ???.  
A jako poslední vytvoříme metodu bez parametru, tedy **vzd().** Bude metoda statická? ???.   
Metodu Main umístěte do téže třídy. V metodě Main():  
Nejprve vytvořte **Bod b1**se souřadnicemi 3,4  
Potom zavolejte (pro jaký objekt či třídu?) metodu **vzdXY(3,4)**.   
Pak zavolejte (pro jaký objekt či třídu?) metodu **vzdBod(b1)**.   
Do třetice zavolejte (pro jaký objekt či třídu?) metodu **vzd()**.   
Některé z těchto tří metod se volají pro třídu, tedy před tečkou je Bod. Ověřte, zda je nutné Bod před tečku uvádět. *???*Tatáž otázka na nutnost uvést před tečku b1.   
*???*Ověřte, že ve všech třech případech dostanete stejnou vzdálenost.  
Změňte datové složky na privátní. Bude to mít nějaký vliv na program?   
*???*Která (které) metoda lze zavolat ještě před vytvořením první instance bodu (ověřte)   
*???*Vypište na konci metody Main nějakou privátní datovou složku přímo pomocí **WriteLine**. Jde to?   
*???*

1. Pod třídu **Bod** do stejného souboru umístěte třídu **TestBod**, do které přesuňte metodu **Main**.   
   Bylo nutno metodu Main po přesunu nějak upravit?   
   *???*Projevilo se nyní nějak to, že jsou datové složky třídy Bod privátní?   
   *???*Vypište nějakou privátní datovou složku přímo pomocí **WriteLine**. Jde to?   
   *???*
2. Místo getteru a setteru naprogramujte vlastnost (property). Zajistěte navíc, aby souřadnice nemohla být záporná (představme si, že se jedná o souřadnice bodu obrazovky), a to ani při přiřazení ani v konstruktoru.
3. V rámci tréningu vytvořte ještě instanční metodu vzdKBodu se dvěma číselnými parametry a její přetíženou variantu, kdy parametrem bude instance bodu. Odzkoušejte v metodě Main. Pro kontrolu: pokud by jeden z bodů měl souřadnice (0,0), pak by tato metoda dávala stejný výsledek jako metoda vzdBod.  
   Pozn.: statickou metodu, kde by v závorce byly dva body, vytvářet nebudeme. V objektově orientovaném programování se statické metody používají jen zřídka. Jen když to jinak nejde. Tedy jen když pracují se statickými datovými složkami.