Domácí úkoly pro denní studenty kurzy OOPR1

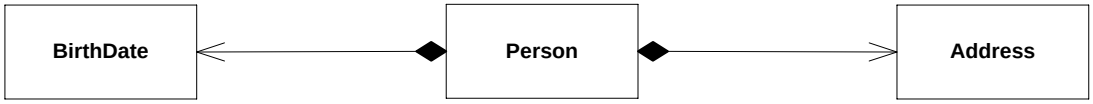
OOPR1 – první úkol

I) Vytvořte třídu ***datum narození*** [BirthDate], která bude reprezentována *dnem* [int day], *měsícem* [int month] a *rokem* [int year] (čtyři číslice). Nemusíte řešit přestupný rok, u měsíce provedete pouze kontrolu rozsahu 1 až 12 jednotlivých měsíců. U dnů je třeba provést kontrolu rozsahu dnů v rámci měsíců (bez zřetele na přestupný rok), tedy nepřipustit, aby např. květen měl jen 30 dni nebo červen měl naopak 31, či únor 30.

II) Vytvořte třídu ***adresa*** [Address address], která bude reprezentovaná atributy *stát* [String state], *město* [String city], *ulice s číslem* [String street] a *zip* [int ZIP].

III) Vytvořte třídu ***osoba*** [Person], která bude reprezentovaná atributy *jméno* [String name], *pohlaví* [char sex] a *rodné číslo* [String birthID] a adresa (použijte kopírovací konstruktor). Atribut rodné číslo vyžaduje, aby byl přístupný datum narození (použijte kopírovací konstruktor). Další potenciální atribut *věk* bude stanoven metodou getAge(), která k výpočtu využije třídu Calendar z balíčku java.util.Calendar. Atribut *pohlaví* [sex] je reprezentován jedním znakem – velkým písmenem {M, F} (male, female). Zařiďte, aby bylo pohlaví neměnné (immutable), tzn., aby nešlo změnit, jakmile je přiřazeno. Atribut *rodné číslo* [birthID] je řetězec o 10 pozicích a pro jednoduchost budete pouze pracovat s rodným číslem pro muže. Rodné číslo je odvozeno jednak z atributů třídy *datum* [birthDate] které následují čtyři číslice, které můžeme nazvat řídící číslice [int controlDigits]. První dvě pozice rodného čísla tvoří *rok* (např. 2017 tedy 17), další dvě pozice *měsíc*, další dvě pozice *den*. První dvě pozice řídících číslic zaplníte náhodným číslem (00 až 99) s využitím třídy Random. Poslední dvě pozice doplníte tak, aby výsledné číslo bylo dělitelné 11 bezezbytku (modulo 11). Blíže viz odkaz: rodné číslo na wikipedii. (Pozor na roky např. 2000 - 00 nebo 2006 - 06).

IV) Vytvořte třídu PersonRun ve které deklarujete tři instance třídy Person a vypíšete jejich datové atributy, i ty vypočítané.

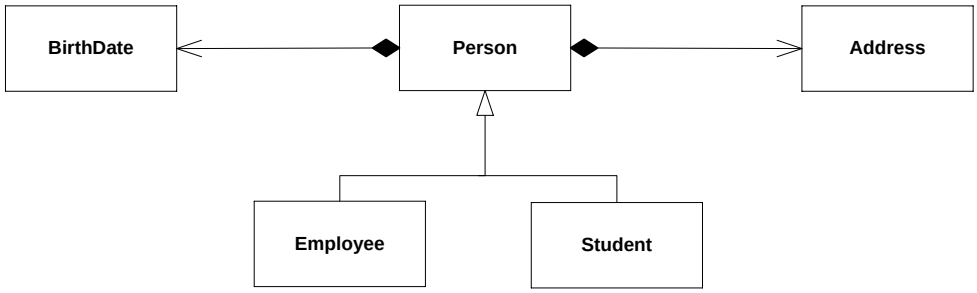


OOPR1 – druhý úkol

I) Ke třídě *osoba* [Person] doplňte podtřídy *pracovník* [Employee] a *student* [Student]. Třída [Employee] má datové atributy *pozice* [Position], která je daná výčtovým typem ADMINISTRATION, RESEARCH, PRODUCTION, PURCHASE, SALE (administrativa, výzkum, výroba, nákup a prodej). Dalším atributem třídy Employee je *plat* [Crowns salary]. Třídu koruny [Crowns] budeme probírat na cvičeních.

II) Vytvořte třídu *student* [Student] s datovými atributy *název university* [String universityName] a *obor studia* [String branchOfStudy].

III) Vytvořte třídu EmployeeStudentRun, kde v hlavní metodě vytvoříte dvě instance třídy Employee a dvě instance třídy Student.

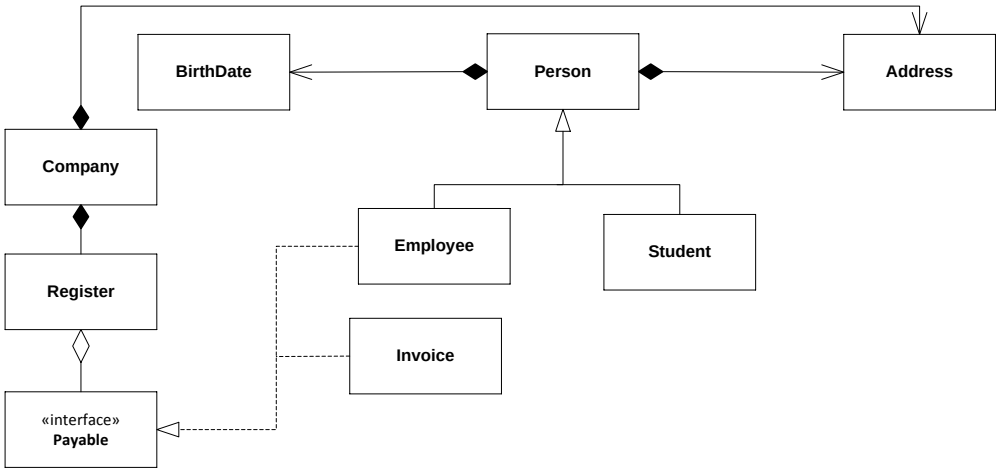


OOPR1 – třetí úkol

I) Vytvořte třídu *faktura* [Invoice], která má následující atributy: *číslo části* [String partNumber], *popis části* [String partDescription], *množství* [int quantity] a *cena za položku* [Crown pricePerItem]. Dílčím identifikátorem *položky* na faktuře je *číslo části*. Dále použijete již vytvořenou třídu *zaměstnanec* [Employee]. Identifikátorem zaměstnance je rodné číslo [birthID].

II) Vytvořte třídu *firma* [Company], která datové atributy *jméno* [String name] a již dříve vytvořenou třídu *adresa* [Address]. Dále má tato třída datový atribut *kolekce* [Register register], který bude uchovávat instance tříd *faktura* [Invoice] a *zaměstnanec* [Employee]. Obě tyto třídy jsou tzv payableObjects, tedy objekty, kterým firma musí platit. *Firma* platí *zaměstnanci* měsíční výplaty Crown salary a za fakturu *firma* musí platit požadovanou částku. Obě třídy (Invoice a Employee) proto implementují rozhraní Payable, které deklaruje hlavičku metodou Crown getPayableAmount().

III) Pro instance obou tříd musí být firma schopna vypočítat vyplacenou částku (paymentAmount). Metoda Crown getPayableAmount() je ve třídě [Invoice] implementovaná tak, že vynásobí datový atribut [int quantity] s datovým atrinutem [Crown pricePerItem] a vrací instanci třídy [Crown]. Metoda getPaymentAmount() je ve třídě [Employee] implementovaná tak, že pouze vrací atribut [Crown salary], což je rovněž instance třídy [Crown].



IV) Třídy Employee a Invoice zastiňují (override) metodu toString, kterou zdědily ze třídy Object. Vytvořte třídu CompanyRun, ve které deklarujete nejdříve instanci třídy Company (obsahuje objekty třídy Register a Address) a dvě instance třídy Employee a dvě instance třídy Invoice. Instance tříd Emploee a Invoice vložíte do instance třídy

Register. Pak projděte registrem a vypište informace o každé uložené instanci (employee, invoice) pomocí metody toString() a metody getPayableAmount().

Na co si dát pozor?

A) Hlavně na to, aby případné atributy nemohly mít nesmyslné hodnoty. Vytvořte vhodné konstruktory (např. Person nemůže existovat bez rodného čísla, takže podle toho volit vhodné konstruktory, aby uživatel musel zadat povinné údaje, nebo se údaje dopočítají se zadaných informací – viz rodné číslo birthID).

B) Bude se hodnotit i použití modifikátorů „static“, „final“, „private-public-no_modifier-public“.

C) Dejte si pozor na konvenci pojmenování proměnných, že proměnná (pokud není konstanta, která je celá velkým s podtržítkama jako mezery) začíná malým a každé další slovo začíná velkým (birthDate, birthID, numberID apod.). To samé s názvy metod. Metoda „getSomething“ bude vypadat např. „getDate()“, „getBirthID()“ apod.

D) Uvažujte o tom, jaké konstruktory použít. Co se týče konstruktorů, vždy budete mít konstruktor kopírovací + nějaký na vytvoření objektu (např. u Person musí nutně být zadané rodné číslo, věk se přepočítá z data narození s použitím třídy Calendar z balíčku java.util.Calendar., např. jméno musí být také povinné). V tomto případě je tedy bezparametrický k ničemu. Podobně uvažujte u všech tříd.