B-OOP 2025: Semestrálne zadanie

Viackrát si pozorne prečítajte **celé** zadanie a až potom ho začnite riešiť! Odporúčame vytvoriť si rozumnú reprezentáciu znalostí, aby ste na nič nezabudli (použite napr. mind mapu).
Očakávame, že v prípade nejasností budete konzultovať cez niektorý z komunikačných kanálov (emaily, discord - ak nemáte prístup, napíšte na [redacted]).

1 Štruktúra zadania

Stiahnite si súbor k semestrálnemu zadaniu. Nájdete v ňom priečinok `test`. Súbor má nasledujúcu štruktúru:

```
zadanie.zip
└── test
    └── RequiredTests.java
```

Priečinok test umiestnite na rovnakú úroveň ako priečinok src vašej implementácie. Nesmiete meniť umiestnenie triedy `RequiredTests`, ani jej názov, ani názvy testov.

2 Prehľad zadania

Vašou úlohou je implementovať veľmi jednoduchú verziu poisťovacieho systému pre malé poisťovne. Váš systém bude podporovať tri zjednodušené typy poistných zmlúv:

- `SingleVehicleContract` - povinné zmluvné poistenie (PZP) pre vozidlo
- `MasterVehicleContract` - povinné zmluvné poistenie (PZP) pre flotilu vozidiel, tzn. súborová PZP zmluva
- `TravelContract` - cestovné poistenie

Vlastnosti a atribúty týchto zmlúv budú inšpirované reálnymi poisťovacími systémami, no vo viacerých prípadoch sa pre jednoduchosť od nich odkloníme. Zmluvy obsahujú platobné údaje, údaje o poistníkovi, poisťovni, poistených objektoch, atď.

Tieto zmluvy bude spravovať poisťovňa (`InsuranceCompany`). Poisťovňa zmluvy vytvára a modifikuje. Poisťovňa pravidelne vyhodnocuje splatnosť zmlúv. Na každej zmluve, kde už čas splatnosti nastal, nastaví poisťovňa nedoplatok vo výške poistnej sumy.

Platby na zmluvách orchestruje `PaymentHandler`. Má za úlohu korektne vykonať úhradu zmluvy (rámcovej alebo inej) a ukladať históriu úhrad.

Systém bude rozlišovať dva typy poistených objektov:

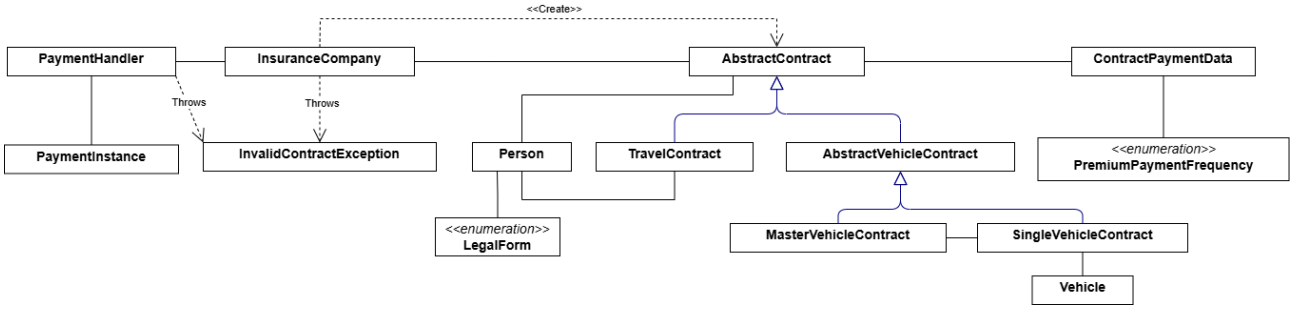
- `Person` - Osoba (právnická aj fyzická, pretože nám je to z hľadiska poistných zmlúv jedno).
- `Vehicle` - vozidlo

V neposlednej rade poisťovňa podporuje vyplatenie poistného plnenia v prípade úspešne vyriešenej poistnej udalosti.

Na obrázku 1 je vidieť zjednodušený prehľad tried, ktoré budete implementovať a vzťahy medzi nimi. Tento prehľad má informatívny charakter.

3 Slovník pojmov

V tabuľke 1 nájdete stručný prehľad pojmov, ktoré vám budú užitočné na implementáciu zadania, a ich približných prekladov. Niektoré pojmy v zahraničných poisťovniach neexistujú, v týchto prípadoch používame preklady, ktoré vytvorili slovenské poisťovne. Pojmy vysvetľujeme najmä v kontexte zadania.



Obr. 1: Prehľadový UML diagram tried

Tabuľka 1: Tabuľka pojmov z domény poisťovníctva a ich približných prekladov

Slovenský termín	**Anglický termín**	**Poznámky**
Poistná zmluva	Insurance contract	Zmluva o poskytovaní poisťovacích služieb. Poistná zmluva v tomto zadaní musí obsahovať poistníka, poisťovateľa, poistené objekty a dodatočné platobné údaje.
Poisťovateľ	Insurer	Subjekt, ktorý poskytuje poisťovnícke služby, tzn. poisťovňa.
Poistník	Policy holder	Subjekt, ktorý uzavrel zmluvu o poistení s poisťovateľom. Vo väčšine prípadov je poistník tá osoba, ktorá uhrádza predpisy poistného na zmluve. To, že je osoba poistníkom, nemusí znamenať, že je aj poistená. Napríklad, osoba, ktorá je poistníkom na cestovnom poistení, nemusí nevyhnutne byť poistenou osobou na tomto poistení.
Oprávnená osoba	Beneficiary	Osoba, ktorej je prednostne vyplatené poistné plnenie v prípade likvidácie poistnej udalosti. Napríklad, ak dôjde ku nehode vozidla, ktoré má uzatvorené povinné zmluvné poistenie, na ktorom je uvedený aj poistník, aj oprávnená osoba, tak sa poistné plnenie vypláca oprávnenej osobe. Ak nie je uvedená oprávnená osoba, poistné plnenie sa na PZP vypláca poistníkovi. Môže ísť napríklad o nákup vozidla na lízing. Poistníkom je osoba, ktorá spláca lízing, a oprávnenou osobou je lízingová spoločnosť. Nakoľko vlastníkom vozidla je lízingová spoločnosť, škoda na majetku vznikla jej, nie poistníkovi. Preto sa prípadné poistné plnenie vypláca jej.
Poistné plnenie	Coverage amount	Suma, ktorú poisťovateľ vyplatí oprávnenému príjemcovi (poistníkovi, oprávnenej osobe, poistenej osobe...), v prípade pozitívneho vyhodnotenia nahlásenej poistnej udalosti. Napríklad, poistník má uzatvorené PZP na svoje vozidlo. Dôjde ku havárii (poistná udalosť). Poistník poistnú udalosť nahlási. Poisťovňa vyhodnotí všetky náležitosti hlásenia a rozhodne sa, či má klient nárok na "vyplatenie poistenia". Ak áno, vyplatí mu financie vo výške poistného plnenia. (Veľmi zjednodušene.)
Poistné	Insurance premium	Suma, ktorú je potrebné uhradiť za dané splatné obdobie za poskytovanie poisťovacích služieb poisťovni. Napríklad, za poskytovanie PZP každoročne platíte poisťovni nejakú sumu. Táto suma sa nazýva poistné.
Poistná udalosť	Claim	Akákoľvek udalosť, ktorá môže viesť k vyplateniu poistného plnenia. Napríklad, v prípade PZP môže ísť o haváriu.

Tabuľka 1: Tabuľka pojmov z domény poisťovníctva a ich približných prekladov

Slovenský termín	**Anglický termín**	**Poznámky**
Nedoplatok	Outstanding balance	Neuhradená suma, ktorá je asociovaná s nejakou zmluvou. Napríklad, ak máte uzatvorené PZP, ktoré každoročne platíte, ale minulý rok ste ho neuhradili, vznikne na vašej zmluve nedoplatok vo výške poistného. Nedoplatok sa navyšuje po každom uplynutom zúčtovacom období. V kontexte tohto zadania je nedoplatok na zmluve kladné číslo. Ak existuje preplatok (uhradili ste viac, než poistné), tak je nedoplatok záporné číslo.
Poistený objekt	Insured object	Objekt, ktorý je predmetom poistnej zmluvy. Poistených objektov môže byť na jednej zmluve viacero. V prípade PZP ide o auto. V prípade cestovného poistenia ide o osobu alebo osoby, ktoré vycestujú.
Fyzická osoba	Natural person	Osoba, ktorú identifikujeme pomocou rodného čísla.
Právnická osoba	Legal person	Osoba, ktorú identifikujeme pomocou IČO.

4 Dátový model

V tejto sekcii nájdete UML diagram tried (viď obrázok 2), ktorý musí vaša implementácia spĺňať. To znamená:

- Vaše riešenie musí obsahovať všetky triedy, enumerácie a výnimky, ktoré sú v tomto diagrame. Mená týchto objektov vo vašom riešení sa musia zhodovať s ich pomenovaniami v diagrame. Štrukturálne vzťahy medzi týmito objektami (dedičnosť, asociácie, závislosti...), ktoré diagram predpisuje, musíte tiež rešpektovať. Musíte dodržať umiestnenie tried v balíčkoch a názvy balíčkov. Všetky balíčky, ktoré sú uvedené v diagrame, musíte implementovať do balíčku src (viď 6).
- Vaše riešenie musí obsahovať všetky atribúty a metódy, ktoré majú jednotlivé objekty v tomto diagrame. Mená týchto atribútov a metód vo vašom riešení sa musia zhodovať s ich pomenovaniami v diagrame. Musíte splniť modifikátory viditeľnosti (private, protected, public, default), ktoré sú predpísané diagramom. Musíte dodržať dátové typy atribútov a signatúry metód, ktoré sú predpísané diagramom.
- Vo vašom riešení môžete pridať do jednotlivých tried dodatočné metódy a/alebo atribúty, ak to vaše riešenie vyžaduje. Ak pridáte metódy a/alebo atribúty, ktoré v diagrame nie sú, musíte vedieť odôvodniť, prečo ste tak vykonali. Aj v prípade, keď nepridáte metódy či atribúty, budete musieť vedieť odôvodniť, prečo ste sa tak rozhodli.
- Musíte porozumieť uvedenému diagramu. Počas odovzdávania sa vás cvičiaci môžu opýtať aj otázky týkajúce sa tohto diagramu.

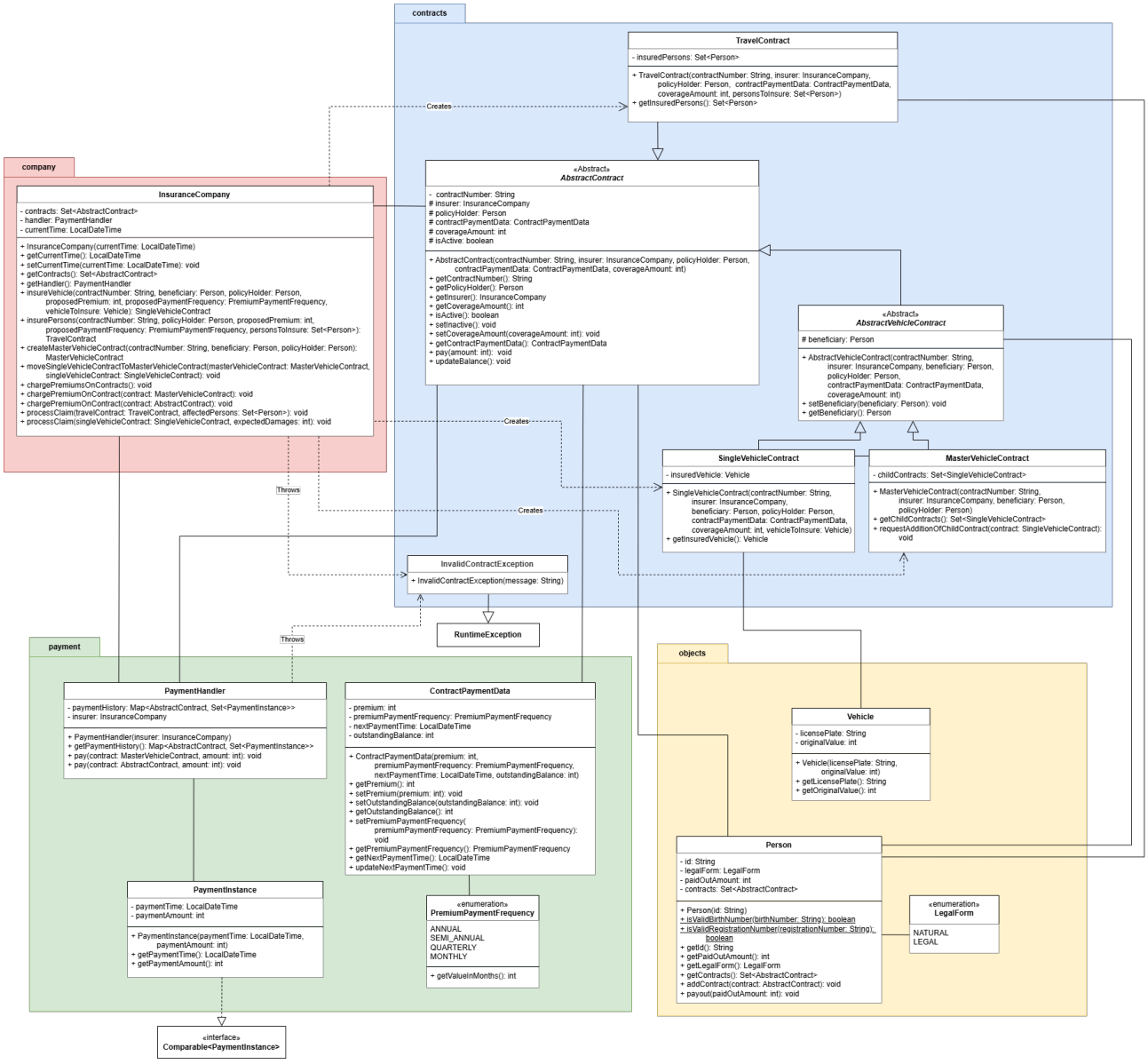
V uvedenom UML diagrame tried je zachytených relatívne veľa detailov, ale v závislosti od vašej implementácie je pravdepodobné, že neobsahuje všetky potrebné metódy a atribúty.

5 Špecifikácia

5.1 Spoločné informácie

Pre všetky triedy platí nasledujúce:

- Ak nejaká metóda triedy vyhadzuje výnimku, ktorá akceptuje ako parameter správu, môžete použiť ľubovoľnú správu. V testoch je kontrolovaný typ vyhodenej výnimky, nie správa. Pozor! To neznamená, že správy môžu byť nezmyselné. Nezabudnite, že hodnotená je o. i. čistota vášho kódu (viď časť 7).
- Naopak, ak v nižšie uvedenej špecifikácii nie je explicitne uvedené, že metóda vyhadzuje výnimku, tak metóda nesmie vyhodiť výnimku.
- Ak nejaká metóda triedy realizuje výpočet so sumami peňazí, zaokrúhľuje sa **nadol** (rovnako ako tomu bolo v prípade malých zadaní). To znamená:
  - Ak je suma 4, polovica zo sumy je 2.



Obr. 2: UML diagram tried semestrálneho zadania

  - Ak je suma 5, polovica zo sumy je 2.
  - Ak je suma 5, 70% sumy je 3.

- Ak je v špecifikácii uvedené, že nejaký atribút triedy sa po jej skonštruovaní nebude meniť, znamená to, že tento atribút má byť `final`. To však nevyhnutne neznamená, že sa nemení jeho obsah. Napríklad, atribút deklarovaný ako `private final List<Integer> list;` znamená, že `list` je konštantná referencia, ale obsah tohto zoznamu sa môže meniť.
- Okrem použitia v metóde `equals`, nesmiete nikde vo vašom zadaní použiť operátor `instanceof`.

5.2 Zmluva - Contract

Náš jednoduchý poisťovací systém obsahuje 4 triedy zmlúv tvoriace hierarchiu. Na vrchole tejto hierarchie je trieda `AbstractContract`.

5.2.1 AbstractContract

Táto trieda má nasledujúce atribúty:

- číslo zmluvy - `contractNumber`. Ide o neprázdny reťazec odlišný od `null`. Tento reťazec slúži ako identifikátor zmluvy. Číslo zmluvy musí byť unikátne v rámci jednej poisťovne. Po nastavení sa číslo zmluvy nesmie meniť.
- poisťovateľ - `insurer`. Ide o odkaz na poisťovateľa (typ `InsuranceCompany`), ktorý zmluvu uzavrel s poistníkom (typ `Person`), tzn. ide o poisťovňu, ktorá vytvorila danú inštanciu zmluvy. Poisťovateľ nesmie mať hodnotu `null` a po nastavení sa nesmie meniť.
- poistník - `policyHolder`. Ide o odkaz na osobu, ktorá uzavrela zmluvu s poisťovňou. Poistník nesmie mať hodnotu `null` a po nastavení sa nesmie meniť.
- platobné údaje zmluvy - `contractPaymentData`. Tieto údaje obsahujú výšku poistného, frekvenciu platby, čas nasledujúcej splatnosti a stav úhrady (tzn. existencia nedoplatku či preplatku). Atribút sa po nastavení nesmie meniť.
- výšku poistného plnenia - `coverageAmount`. Výška poistného plnenia musí byť nezáporná.
- atribút `isActive`, ktorý indikuje, že zmluva je živá a že je na nej možné realizovať platbu a hlásiť poistné udalosti. Po vytvorení zmluvy je tento atribút nastavený na hodnotu `true`.

Trieda vyhadzuje:

- v konštruktore výnimku `IllegalArgumentException`, ak je nesplnené akékoľvek validačné kritérium.
- v set metóde `setCoverageAmount` výnimku `IllegalArgumentException`, ak je nová suma nevalidná.

5.2.2 TravelContract

Ďalším typom zmluvy je `TravelContract`. Predstavuje jednoduché cestovné poistenie. Jeho konštruktor akceptuje tie isté parametre ako konštruktor `AbstractContract` a navyše parameter `personsToInsure`. Ide o množinu, ktorá nesmie byť `null` a musí byť neprázdna. Pomocou nej sa nastaví atribút `insuredPersons`, ktorý sa potom nesmie meniť. Poistník môže, ale nemusí byť súčasťou poistených osôb. Poistiť sa v prípade cestovného poistenia môžu iba fyzické osoby, ale poistníkom môže byť aj právnická osoba (napríklad zamestnávateľ poisťuje zamestnancov pred vycestovaním na zahraničné sympózium). Ak nie sú splnené uvedené obmedzenia množiny `personsToInsure` alebo ak je `contractPaymentData` `null`, konštruktor vyhodí výnimku `IllegalArgumentException`.

5.2.3 AbstractVehicleContract

Poistenie vozidiel je mierne komplikovanejšie. Trieda `AbstractVehicleContract` pridáva atribút oprávnená osoba - `beneficiary`, ktorý môže byť `null`. Ak je `beneficiary` `null`, zmluva nemá oprávnenú osobu a prípadné poistné plnenie sa vypláca poistníkovi. Ak je `beneficiary` rovnaká osoba ako `policyHolder`, konštruktor vyhodí výnimku `IllegalArgumentException`. Ak zmluva má oprávnenú osobu, prípadné poistné plnenie sa vypláca jej.

5.2.4 SingleVehicleContract

Od `AbstractVehicleContract` dedí zmluva `SingleVehicleContract`, ktorá slúži na poistenie konkrétneho vozidla (`Vehicle`). Táto zmluva približne reprezentuje PZP. Vozidlo, ktoré je poisteným objektom na tejto zmluve, je parametrom konštruktora a nesmie byť `null`. Ak je tento parameter `null` alebo ak je `contractPaymentData` `null`, konštruktor vyhodí výnimku `IllegalArgumentException`. Vozidlo nastavené v konštruktore sa nesmie meniť.

5.2.5 MasterVehicleContract

Poslednou zmluvou je `MasterVehicleContract`. Ide o rámcovú zmluvu, ktorá slúži na poistenie flotily vozidiel. Obsahuje množinu zmlúv `SingleVehicleContract` pre jednotlivé vozidlá. Ako príklad z reality môžeme uviesť Dopravný podnik Bratislava, ktorý má rámcovú zmluvu na PZP a táto rámcová zmluva obsahuje PZP zmluvy pre jednotlivé vozidlá vozového parku. Poistíkom na `MasterVehicleContract` musí byť právnická osoba. `MasterVehicleContract` musí nastaviť platobné dáta (`contractPaymentData`) na hodnotu `null` a `coverageAmount` na hodnotu 0. Ak neplatia uvedené obmedzenia, konštruktor vyhodí výnimku `IllegalArgumentException`. Konštruktor tiež inicializuje množinu `childContracts`. Po inicializácii sa táto množina nesmie meniť. Zmluvy sú v tejto množine ukladané v takom poradí, v akom sa do nej pridali (tzn. použite vhodnú implementáciu množiny).

Zmluvy vozidiel sa do `MasterVehicleContract` pridávajú podaním požiadavky na poisťovateľa, tzn. zavolá sa metóda `InsuranceCompany::moveSingleVehicleContractToMasterVehicleContract`.
`MasterVehicleContract` sa považuje za neaktívny, ak sú neaktívne všetky jeho dcérske zmluvy. Ak žiadne dcérske zmluvy nemá, tak sa jeho aktivita vyhodnocuje podľa atribútu `isActive`. Metóda `setInactive` musí nastaviť ako neaktívne všetky jeho dcérske zmluvy, aj svoj atribút `isActive`.

5.2.6 Úhrada zmlúv

Zmluvy je možné zaplatiť. Platí sa volaním metódy `pay` vo výške `amount`. Parameter `amount` môže byť ľubovoľná kladná hodnota. To znamená, že klient nemusí uhradiť celý nedoplatok, alebo naopak, môže uhradiť viac, než mu predpisuje `premium` v `contractPaymentData`. V zmluve sa ale priamo nič nenastavuje. Metódu `pay` môžete brať ako API, ktoré je vystavené používateľovi. Metóda `pay` na zmluve musí interne zavolať korešpondujúcu metódu `pay` z triedy `PaymentHandler`. Inak povedané, postupnosť operácií, keď používateľ chce uhradiť nejakú svoju zmluvu, je nasledujúca:

1. používateľ zavolá na zmluve jej metódu `pay`
2. metóda `pay` tejto zmluvy zavolá vhodnú metódu `pay` z triedy `PaymentHandler`
3. metóda `pay` z triedy `PaymentHandler` vykoná logiku platby

Vaša implementácia musí docieliť, aby sa volal vhodný variant metódy `pay` z triedy `PaymentHandler` pre danú zmluvu, tzn. mala by sa zavolať:

- metóda `pay(MasterVehicleContract, int)` pre `MasterVehicleContract`.
- metóda `pay(AbstractContract, int)` pre inú zmluvu.

Vymyslite, ako je možné takéto správanie dosiahnuť. (Pomôcka: Bude potrebné vhodne prepísať metódu `pay`.)

Na zmluve je tiež možné požiadať o aktualizáciu nedoplatkov volaním metódy `updateBalance`. Táto metóda je opäť len žiadosť na vykonanie operácie, zmluva samotná by si nemala svojvoľne modifikovať nedoplatky. Metóda `updateBalance` interne volá vhodnú metódu `chargePremiumOnContract` z triedy `InsuranceCompany`, pričom opäť musíte zaistiť, aby bol zavolaný správny variant.

5.3 Poistné objekty - Insured objects

V našom jednoduchom poistnom systéme je možné poistiť iba 2 druhy objektov: osoby a vozidlá.

5.3.1 Osoba

Osoba je reprezentovaná triedou `Person`. Má nasledujúce atribúty:

- `id` - reťazec odlišný od `null`, ktorý je neprázdny. Keď konštruktor nastaví `id`, už sa nesmie meniť. `id` musí byť buď validné rodné číslo alebo IČO.
- `legalForm` - typ osoby. Ak je `id` rodné číslo, ide o fyzickú osobu (`NATURAL`). Ak je `id` IČO, tak ide o právnickú osobu (`LEGAL`). Po inicializácii sa tento atribút nesmie meniť.

- `paidOutAmount` - ide o súhrnnú sumu vyplatenú zo všetkých spracovaných poistných udalostí. Konštruktor ju inicializuje na hodnotu 0.
- `contracts` - množina zmlúv, na ktorých je daná osoba uvedená ako `policyHolder`. Táto množina ukladá zmluvy v poradí, v akom sa do nej vkladali. Po inicializácii v konštruktore sa táto množina nesmie meniť.

Za validné rodné číslo považujeme reťazec, ktorý:

1. nie je `null` a má dĺžku 9 alebo 10 znakov a všetky jeho znaky sú číslice (tzn. rodné číslo je bez symbolu lomeno).
2. je vo formáte RRMMDDNNN alebo RRMMDDNNNN.
3. Mesiac (MM) musí byť v rozsahu 1-12 (vrátane) alebo 51-62 (vrátane; tento rozsah značí, že RČ patrí žene, pričom mesiac jej narodenia sa dá získať odčítaním čísla 50 od MM).
4. Ak má RČ 9 znakov, tak rok (RR) musí byť menší alebo rovný 53 (tzn. RČ bolo vydané do roku 1953 vrátane). Ak je dátum narodenia spočítaný z RČ existujúci historický dátum, tak je toto RČ validné. (Existenciu dátumu môžete overiť použitím triedy `LocalDate`.)
5. Ak má RČ 10 znakov (tzn. RČ bolo vydané od roku 1954 vrátane), tak musí platiť kontrolná suma. Označme c_i číselnú hodnotu i-tej cifry RČ. Kontrolná suma platí, ak platí vzťah $\sum_{i=0}^{9}(-1)^i c_i \bmod 11 = 0$. (Poznámka: existujú aj valídne RČ, pre ktoré táto kontrolná suma neplatí. Také RČ ale vaša implementácia nemusí brať do úvahy.) Ak platí kontrolná suma, musíte ešte overiť, či je dátum narodenia spočítaný z RČ existujúci historický dátum. (Existenciu dátumu môžete overiť použitím triedy `LocalDate`.) Ak áno, ide o validné RČ.

Za validné IČO považujeme reťazec, ktorý nie je `null`, a skladá sa zo 6 alebo 8 znakov, ktoré sú číslice.

Ak `id` nie je validné (je `null` alebo nie vyhodnotené ani ako validné RČ, ani ako validné IČO), tak konštruktor triedy `Person` vyhodí `IllegalArgumentException`. Trieda `Person` navyše vyhadzuje výnimku `IllegalArgumentException` v metóde `payout`, ak je jej argumentom nekladná suma, alebo v metóde `addContract`, ak je jej argumentom hodnota `null`. V metóde `payout` sa do celkovej vyplatenej sumy osoby `this.paidOutAmount` pripočíta parameter metódy `paidOutAmount`.

5.3.2 Vozidlo

Druhým poistným objektom je vozidlo (`Vehicle`). Vozidlo obsahuje EČV (`licensePlate`) a cenu (`originalValue`). Atribút `licensePlate` je reťazec, ktorý nesmie byť `null` a musí mať dĺžku 7 znakov. Všetky jeho znaky musia byť veľké písmená A-Z alebo číslice. Atribút `originalValue` musí byť pozitívny. Ak uvedené obmedzenia neplatia, konštruktor vyhodí `IllegalArgumentException`. Ani jeden z týchto atribútov sa po nastavení nesmie meniť.

5.4 Platby - Payments

5.4.1 ContractPaymentData, PremiumPaymentFrequency

Platobné údaje na zmluve sú reprezentované triedou `ContractPaymentData`. Obsahuje atribúty:

- `premium` - suma, ktorú je potrebné zaplatiť za dané obdobie splatnosti. Musí byť kladná.
- `premiumPaymentFrequency` - frekvencia platieb (tzn. frekvencia úhrad zmluvy), nesmie byť `null`. Možné hodnoty tohto atribútu sú:
  - `ANNUAL` - ročná platba (`getValueInMonths` vracia hodnotu 12)
  - `SEMI_ANNUAL` - polročná platba (`getValueInMonths` vracia hodnotu 6)
  - `QUARTERLY` - štvrťročná platba (`getValueInMonths` vracia hodnotu 3)
  - `MONTHLY` - mesačná platba (`getValueInMonths` vracia hodnotu 1)
- `nextPaymentTime` - čas, kedy má nastať nasledujúca úhrada zmluvy. Nesmie byť null.
- `outstandingBalance` - nedoplatok.

Ak nie sú splnené uvedené obmedzenia, konštruktor vyhodí výnimku `IllegalArgumentException`. Pre lepšie pochopenie významu atribútov uvedieme ešte nasledujúce:

- Ak je `premium` rovné 10 a `premiumPaymentFrequency` je nastavená na `SEMI_ANNUAL`, znamená to, že na tejto zmluve sa majú vykonať 2 platby ročne (zmluva sa uhrádza každých 6 mesiacov), a teda očakávame, že poistník za rok uhradí 2*10=20.
- `outstandingBalance` zachytáva stav úhrady zmluvy. Ak je `outstandingBalance` kladný, tak je na zmluve nedoplatok. Ak je nulový, zmluva je uhradená. Ak je záporný, na zmluve je preplatok. Napríklad, ak je `outstandingBalance` rovný -10, klient zaplatil o 10 viacej, než mal. To je v poriadku, pri ďalšej platbe môže (ale nemusí) vďaka tomu zaplatiť menej.
- `nextPaymentTime` je čas, kedy poisťovňa navýši nedoplatok o hodnotu `premium`. Napríklad, ak je `premium` 10, `outstandingBalance` 5, aktuálny dátum podľa poisťovne je 01. február 2025 a zmluva má dátum v `nextPaymentTime` 31. január 2025, znamená to, že pri volaní `updateBalance` na zmluve by poisťovňa mala navýšiť `outstandingBalance` na hodnotu 5+10=15.

Trieda `ContractPaymentData` navyše vyhodí výnimku v set metódach pre atribúty `premium` a `premiumPaymentFrequency`, ak sú ich argumenty nevalidné. Metóda `updateNextPaymentTime` nastaví čas `nextPaymentTime` na nový dátum splatnosti pripočítaním takého počtu mesiacov, koľko indikuje frekvencia platby. Napríklad, ak je frekvencia platby polročná, nový dátum platby je `nextPaymentTime` plus 6 mesiacov.

5.4.2 PaymentHandler, PaymentInstance

O realizáciu a spracovanie platieb sa stará `PaymentHandler`. `PaymentHandler` si ukladá inštanciu poisťovne, ktorej patrí. Ak nie je, konštruktor vyhodí výnimku `IllegalArgumentException`. Po nastavení je atribút `insurer` nemožné meniť. Konštruktor zároveň inicializuje `paymentHistory`. Po inicializácii ani tento atribút nie je možné meniť.

Metóda `PaymentHandler::pay(AbstractContract, int)` je realizovaná nasledujúcim spôsobom:

- Ak je zmluva `null` alebo je `amount` nekladný, tak vyhodí výnimku `IllegalArgumentException`.
- Ak je zmluva neaktívna alebo nejde o zmluvu poisťovateľa, ktorý prevádzkuje tento `PaymentHandler`, tak vyhodí výnimku `InvalidContractException`.
- V opačnom prípade zníži `outstandingBalance` zmluvy o hodnotu `amount`.
- Potom vytvorí záznam o realizácii platby (inštanciu `PaymentInstance`) s aktuálnym časom podľa poisťovne a zaplatenou sumou a uloží ju do `paymentHistory` ku uhradenej zmluve.

Metóda `PaymentHandler::pay(MasterVehicleContract, int)` je realizovaná nasledujúcim spôsobom:

- Ak je zmluva `null` alebo je `amount` nekladný, tak vyhodí výnimku `IllegalArgumentException`.
- Ak je zmluva neaktívna alebo nejde o zmluvu poisťovateľa, ktorý prevádzkuje tento `PaymentHandler`, alebo neobsahuje žiadne dcérske zmluvy, tak vyhodí výnimku `InvalidContractException`.
- V opačnom prípade iteruje cez všetky dcérske zmluvy (`childContracts`) a pokúsi sa vynulovať ich nedoplatky (pričom vždy spotrebované financie odčíta od `amount`). Ak ostanú nejaké financie, tak opäť iteruje cez zmluvy a vytvára v nich preplatky vo výške `premium` (alebo vo výške `amount`, ak už neostáva dostatok financií). Ak ešte ostali financie, iteruje znova. Iteruje dovtedy, kým nespotrebuje celý amount.
- Potom vytvorí záznam o realizácii platby (inštanciu `PaymentInstance`) s aktuálnym časom podľa poisťovne a zaplatenou sumou a ulož ju do `paymentHistory` ku uhradenej `MasterVehicleContract` zmluve (nie ku jednotlivým dcérskym zmluvám). História platieb k danej zmluve sa ukladá v poradí podľa času realizácie platby, od najstaršieho po najmladšieho. Toto je možné docieliť vhodnou implementáciou rozhrania `Comparable`.

Proces platby `MasterVehicleContract` je zachytený v nasledujúcom pseudokóde:

```
pay(MasterVehicleContract contract, int amount):
    Pre každú aktívnu dcérsku zmluvu:
        Ak má zmluva nedoplatok:
            Ak mám dostatok financií amount:
                amount -= zmluva.nedoplatok
                zmluva.nedoplatok = 0
            Inak:
                zmluva.nedoplatok -= amount
                amount = 0
```

```
Pokým platí amount > 0:
    Pre každú aktívnu dcérsku zmluvu:
        Ak mám dostatok financií amount:
            zmluva.nedoplatok -= zmluva.premium
            amount -= zmluva.premium
        Inak:
            zmluva.nedoplatok -= amount
            amount = 0
```

Uveďme si príklad. Majme rámcovú zmluvu, ktorá obsahuje 4 dcérske zmluvy:

- contract1 - má `premium` rovné 30 a je atívna.
- contract2 - má `premium` rovné 50 a je aktívna.
- contract3 - má `premium` rovné 75 a je aktívna.
- contract4 - má `premium` rovné 20 a je neaktívna.

	Pred uhradením	**Po uhradení nedoplatkov**	**Prvý cyklus po uhradení**	**Druhý cyklus po uhradení**
contract1	30	0	-30	-60
contract2	50	0	-50	-85
contract3	100	0	-75	-75
contract4	0	0	0	0
Amount	400	220	65	0

Tabuľka 2: Ukážka priebežných hodnôt pri úhrade rámcovej zmluvy.

Trieda `PaymentInstance` zachytáva jednu vykonanú platbu - kedy sa udiala a v akej hodnote. Jej atribút `paymentTime` nesmie byť `null` a suma `paymentAmount` musí byť kladná. Ak tieto obmedzenia neplatia, konštruktor vyhodí výnimku `IllegalArgumentException`. Po nastavení týchto atribútov v konštruktore ich už nie je možné meniť.

5.5 Poisťovateľ - InsuranceCompany

Posledným komponentom zadania je trieda `InsuranceCompany`. Táto trieda vytvára a spravuje zmluvy, aktualizuje ich platobné údaje a v neposlednej rade spracúva poistné udalosti. Konštruktor `InsuranceCompany` berie ako parameter aktuálny čas - `currentTime` (Poznámka: Napriek názvu `currentTime`, tento čas bude v testoch nastavený podľa potreby, tzn. nie je garantované, že platí `currentTime.isEqual(LocalDateTime.now())`. Ak je `currentTime` `null`, tak konštruktor vyhodí `IllegalArgumentException`. Konštruktor tiež inicializuje množinu zmlúv, ktoré poisťovňa uzatvorila, a vytvorí správcu platieb `handler`. Po inicializácii nie je možné tieto dva atribúty meniť. Zmluvy sú v množine zmlúv uložené v takom poradí, v akom ich poisťovňa uzatvárala.

Atribút `currentTime` je možné nastaviť príslušnou set metódou. Ak je nový `currentTime` rovný `null`, tak táto metóda vyhodí výnimku `IllegalArgumentException`.

5.5.1 Vytvráranie nových zmlúv

Nové zmluvy poisťovňa uzatvára použitím metód:

- `insureVehicle` - Pre parameter `contractNumber` musí platiť, že v danej poisťovni nejestvuje iná zmluva s týmto číslom. Celková ročná čiastka, ktorú poistník zaplatí, musí byť väčšia alebo rovná 2% z ceny vozidla. Ak tieto obmedzenia nie sú splnené alebo ak je niektorý z argumentov potrebných na výpočet nevalidný, tak metóda vyhodí `IllegalArgumentException`. Ak naopak sú splnené, metóda vytvorí nový `SingleVehicleContract`, ktorý má `coverageAmount` nastavený na polovicu hodnoty vozidla. V platobných dátach sa nastaví `premium` a `premiumPaymentFrequency` podľa navrhovaných hodnôt, nedoplatok sa nastaví na 0 a dátum ďalšej platby sa nastaví na `currentTime` poisťovne. Následne poisťovňa zavolá metódu `chargePremiumOnContract` s touto zmluvou. Potom sa novovytvorená zmluva uloží do množiny zmlúv poisťovne, množiny zmlúv poistníka a metóda ju vráti.
- `insurePersons` - Pre parameter `contractNumber` musí platiť, že v danej poisťovni nejestvuje iná zmluva s týmto číslom. Celková ročná čiastka, ktorú poistník zaplatí, musí byť väčšia alebo rovná päťnásobku počtu poistených osôb. Ak tieto obmedzenia nie sú splnené alebo ak je niektorý z argumentov potrebných

na výpočet nevalidný, tak metóda vyhodí `IllegalArgumentException`. Ak naopak sú splnené, metóda vytvorí nový `TravelContract`, ktorý má `coverageAmount` nastavený na desaťnásobok počtu poistených osôb. V platobných dátach sa nastaví `premium` a `premiumPaymentFrequency` podľa navrhovaných hodnôt, nedoplatok sa nastaví na 0 a dátum ďalšej platby sa nastaví na `currentTime` poisťovne. Následne poisťovňa zavolá metódu `chargePremiumOnContract` s touto zmluvou. Potom sa novovytvorená zmluva uloží do množiny zmlúv poisťovne, množiny zmlúv poistníka a metóda ju vráti.

- `createMasterVehicleContract` - Pre parameter `contractNumber` musí platiť, že v danej poisťovni nejestvuje iná zmluva s týmto číslom. Ak toto obmedzenie nie je splnené, tak metóda vyhodí `IllegalArgumentException`. Ak naopak je splnené, metóda vytvorí nový `MasterVehicleContract`, ktorý zatiaľ neobsahuje žiadne dcérske zmluvy. Potom sa novovytvorená zmluva uloží do množiny zmlúv poisťovne, množiny zmlúv poistníka a metóda ju vráti.

Za účelom pridania dcérskej zmluvy do rámcovej zmluvy je potrebné najprv vytvoriť samostatnú dcérsku zmluvu (tzn. `SingleVehicleContract`) a potom požiadať poisťovňu o jej presun do rámcovej zmluvy. Presun realizuje metóda `moveSingleVehicleContractToMasterVehicleContract`. Ak je ktorýkoľvek z jej parametrov `null`, vyhodí výnimku `IllegalArgumentException`. Obe zmluvy, ktoré sú jej parametrami, musia byť aktívne a musia byť uzatvorené poisťovňou, u ktorej žiadame o presun. Navyše, obe zmluvy musia mať rovnakého poistníka. Ak tieto obmedzenia neplatia, je vyhodená výnimka `InvalidContractException`. Ak sú obmedzenia splnené, tak sa `singleVehicleContract` odstráni z množín zmlúv poisťovne aj poistníka a pridá sa do množiny dcérskych zmlúv `masterVehicleContract`. Platobná história zmluvy `singleVehicleContract` sa ponecháva bez zmeny.

5.5.2 Aktualizácia nedoplatkov na zmluvách

Metóda `chargePremiumsOnContracts` iteruje cez všetky zmluvy, ktoré uzatvorila poisťovňa (nachádzajú sa v množine zmlúv), a na každej zmluve, ktorá je aktívna, sa zavolá metóda `updateBalance`. Metóda `updateBalance` zasa zavolá niektorú implementáciu metódy `chargePremiumOnContract`:

- `chargePremiumOnContract(AbstractContract)` - metóda nevaliduje argument. Overí, či daná zmluva má termín splatnosti pred časom `currentTime` alebo je zhodný s časom `currentTime` (`isBefore`, `isEqual`). Ak áno, navýši nedoplatok na zmluve o hodnotu `premium` z `contractPaymentData`. Zároveň aktualizuje termín splatnosti na tejto zmluve (podľa `premiumPaymentFrequency` v `contractPaymentData`). Tento postup opakuje, pokým neplatí, že `currentTime` je skôr ako termín splatnosti `nextPaymentTime`.

- `chargePremiumOnContract(MasterVehicleContract)` - metóda nevaliduje argument. Iteruje cez všetky dcérske zmluvy danej rámcovej zmluvy a pre každú z nich zavolá `chargePremiumOnContract`.

5.5.3 Spracovanie poistných udalostí

Na poistných zmluvách je možné hlásiť poistné udalosti. Proces hlásenia poistných udalostí je však nesmierne komplikovaný a len jeho popis by zabral niekoľko desiatok strán. Vo vašej implementácii vytvoríte dve metódy `processClaim`, ktoré predpokladajú, že nejaký proces riešenia poistnej udalosti už prebehol, a majú na starosti iba vyplatenie poistného plnenia. Fungujú nasledovne:

- `processClaim(SingleVehicleContract, int)` - parameter `singleVehicleContract` musí byť odlišný od `null`. Parameter `expectedDamages` musí byť kladný. Ak tieto podmienky nie sú splnené, tak metóda vyhodí `IllegalArgumentException`. Navyše, `singleVehicleContract` musí byť aktívna zmluva. Ak nie je, metóda vyhodí výnimku `InvalidContractException`. Ak existuje oprávnená osoba, je jej vyplatené plnenie vo výške `coverageAmount` (zavolaním jej metódy `payout`). Ak neexistuje, je táto suma vyplatená poistníkovi. Ak je parameter `expectedDamages` väčší alebo rovný 70% hodnoty vozidla, tak sa to považuje za totálnu škodu a zmluva samotná sa zmení na neaktívnu.

- `processClaim(TravelContract, Set<Person>)` - parameter `travelContract` musí byť odlišný od `null`. Parameter `affectedPersons` musí byť odlišný od `null` a musí ísť o neprázdnu množinu, ktorá je podmnožinou poistených osôb v `travelContract`. Ak tieto podmienky nie sú splnené, tak metóda vyhodí `IllegalArgumentException`. Navyše, `travelContract` musí byť aktívna zmluva. Ak nie je, metóda vyhodí výnimku `InvalidContractException`. Ak podmienky sú splnené, tak sa vypočíta výška plnenia ako `coverageAmount / affectedPersons.size()` a toto plnenie sa vyplatí všetkým osobám v množine `affectedPersons`. Zmluva samotná sa potom zmení na neaktívnu.

6 Odovzdanie

V AIS je otvorené miesto odovzdania `OOP - semestrálne zadanie` do `2025-05-09T23:59:00+01:00`. Do tohto miesta odovzdania nahráte **zip** archív (to znamená nie `rar`, ani `tar`...), ani žiadny iný formát než `zip`. Váš archív bude mať nasledujúcu štruktúru:

```
zadanie.zip
└── src
    ├── company
    │   └── InsuranceCompany.java
    ├── contracts
    │   ├── AbstractContract.java
    │   ├── AbstractVehicleContract.java
    │   ├── InvalidContractException.java
    │   ├── MasterVehicleContract.java
    │   ├── SingleVehicleContract.java
    │   └── TravelContract.java
    ├── objects
    │   ├── LegalForm.java
    │   ├── Person.java
    │   └── Vehicle.java
    └── payment
        ├── ContractPaymentData.java
        ├── PaymentHandler.java
        ├── PaymentInstance.java
        └── PremiumPaymentFrequency.java
```

7 Hodnotenie

Hodnotenie bude prebiehať nasledujúcim spôsobom:

1. Automatické hodnotenie:
   (a) Overenie štruktúry odovzdaného archívu. V prípade nesprávnej štruktúry bude zadanie ohodnotené 0 bodmi.
   (b) Overenie splnenia UML diagramu tried. Vaše riešenie musí spĺňať všetko, čo je v diagrame tried uvedené. Ak nie je splnený diagram tried, zadanie bude ohodnotené 0 bodmi. Vaše zadanie môže obsahovať dodatočné metódy a atribúty, ktoré sa na UML diagrame tried nenachádzajú.
   (c) Ohodnotenie funkcionálnej stránky odovzdaného riešenia unit testami, ktoré ste dostali (`RequiredTests`). Ak vaše riešenie nesplní ktorýkoľvek z týchto testov, bude ohodnotené 0 bodmi.
   (d) Ohodnotenie funkcionálnej stránky odovzdaného riešenia privátnymi unit testami. Na základe počtu testov (a významnosti) testov bude vypočítaný návrh bodového ohodnotenia zadania.
   (e) Overenie vášho riešenia antiplagiátorským systémom. Ak vaše riešenie nesplní kontrolu, bude automaticky ohodnotené 0 bodmi. V tomto prípade bude ďalší postup prebiehať v súlade s článkom 13 odst. 5 študijného poriadku FEI STU.
2. Ústny pohovor:
   (a) Ak vaše riešenie nedostalo v procese automatického hodnotenia 0 bodov, zúčastníte sa ústneho pohovoru ku zadaniu. Cvičiaci vám položí niekoľko otázok, ktoré budú mať za cieľ detailne preveriť znalosť vášho riešenia a dizajnové princípy, ktoré ste použili, ako aj znalosť jazyka Java a základných princípov OOP. Cvičiaci vaše odpovede vyhodnotí a na základe nich bude korigovať navrhnuté bodové hodnotenie z automatizovaného testovania. Ak cvičiaci zistí, že nemáte znalosť odovzdaného kódu alebo máte fundamentálne nedostatky v prebranom učive, vaše hodnotenie bude 0 bodov. Hodnotí sa aj čistota a čitateľnosť kódu, dodržiavanie konvencií jazyka Java, atď.