# Je to zákazka veľká

## 2. zadanie – deadline: 25. 4. 2025

Skvelé výsledky z prvého zadania vás povzbudili natoľko, že ste sa spolu s niekoľkými spolužiakmi rozhodli začať podnikať a založili ste firmu, ktorá vyvíja softvérové riešenia na zákazky. Keďže už teraz predpokladáte, že budete mimoriadne úspešní a klienti vám budú čochvíľa zadávať toľko objednávok, ktoré nebudete stíhať riešiť, chcete sa pripraviť na ťažké rozhodnutia a na výber z objednávok. Na vyriešenie problému – ako ináč – napíšete program v Pythone.

Cieľom zadania je, aby ste získali skúsenosti s objektovo orientovaným programovaním a modelovaním konceptov. Vytvoríte niekoľko tried, ktoré budú navzájom prepojené a budú reprezentovať našu riešenú situáciu. Konkrétne sa jedná o triedy reprezentujúce rôzne typy zamestnancov (všeobecná trieda `Employee`, konkrétne typy `Manager`, `Developer` a `Tester`), zákazky (`Order`), projekty (`Project`), firmu (`Company`) a rôzne prístupy k výberu zo zákaziek (všeobecný `Scheduler` a konkrétne typy `FirstScheduler`, `RandomScheduler`, `GreedyScheduler` a `CheapestEmployeeScheduler`).

Okrem toho projekt obsahuje súbor `config.py` s definovanými úrovňami seniority v rámci firmy (`SENIORITY`) a príslušnými platmi (`WAGES`). Tieto konštanty využijete počas riešenia v niekoľkých triedach, vždy si ich ale naimportujte priamo `config`u (nerátajte s tým, že pri testovaní budú použité iba hodnoty uvedené v ukážkovom konfiguračnom súbore).

**Poznámka: Triedy viete implementovať zväčša nezávisle, avšak pre implementáciu musíte pochopiť ich vzájomné prepojenia. Popis štruktúry tried v tomto súbore je minimálny, teda uvedené premenné a metódy musia existovať v triedach, môžete ale pridať ďalšie podľa potreby. Nemeňte názvy definovaných atribútov, ani parametre zadaných metód. Pri odovzdaní riešenia nahrajte všetky vypracované súbory (`company.py`, `employee.py`, `order.py`, `project.py`, `scheduler.py`) na Google Drive.**

**Pre lepšiu čitateľnosť zadania je kľúčové slovo `self` vynechané pri členských premenných a zo zoznamu parametrov metód.**

### Trieda `Employee` – 1 bod

Trieda `Employee` reprezentuje všeobecného zamestnanca, konkrétne typy budú vytvorené v podtriedach. V triede musíte zadefinovať nasledovné členské premenné:

- `name` – `string` – meno zamestnanca ako reťazec,
- `level` – `string` – úroveň seniority zamestnanca (musí byť hodnota z príslušného zoznamu v súbore `config.py`),
- `wage` – `int/float` – plat zamestnanca ako číslo z intervalu uvedeného v konštante `WAGES` v súbore `config.py`,
- `curr_project` – `Project` – projekt, na ktorom zamestnanec aktuálne pracuje.

V triede implementujte konštruktor, ktorý nastaví hodnoty členských premenných podľa parametrov. Aktuálny projekt nech ostáva prázdny (`None`). Ak sa konštruktor zavolá s nesprávnymi hodnotami, vygenerujte chybu podľa nasledovných pravidiel:

- (0,5b) ak úroveň seniority nie je platná, vygenerujte `ValueError` so správou *Unknown seniority level XY*, kde namiesto *XY* vypíšete hodnotu;
- (0,25b) ak zadaný plat zamestnanca nie je číslo (celé alebo desatinné), vygenerujte `TypeError` so správou *Incorrect wage type: XY*, kde namiesto *XY* vypíšete typ zadanej nesprávnej hodnoty;
- (0,25b) ak zadaný plat je správneho typu ale nie je z intervalu, ktorý korešponduje s úrovňou seniority zamestnanca, vygenerujte `ValueError` so správou *Incorrect wage for XY worker: AB*, kde namiesto *XY* vypíšete senioritu zamestnanca, a namiesto *AB* bude zadaný plat.

Ak hociktorá hodnota je neplatná, volanie konštruktora musí končiť chybou a nesmie sa vytvoriť inštancia triedy.

## Trieda `Tester` – 0,2 body

Trieda `Tester` reprezentuje testera, ktorý je konkrétnym typom zamestnanca v našom modeli. Implementujte konštruktor s rovnakými parametrami a funkcionalitou ako v triede `Employee`, a zabezpečte aj to, aby `Tester` dedila od triedy `Employee`.

## Trieda `Developer` – 0,3 body

Trieda `Developer` reprezentuje vývojára, ktorý je konkrétnym typom zamestnanca v našom modeli.

(0,2b) Implementujte konštruktor s rovnakou základnou funkcionalitou ako v triede `Employee`, a zabezpečte aj to, aby `Developer` dedila od triedy `Employee`. Trieda má mať jednu dodatočnú členskú premennú `languages`, ktorá je zoznamom programovacích jazykov a technológií, ktoré vývojár ovláda. Hodnotu tejto premennej dostanete ako dodatočný parameter konštruktora.

(0,1b) V triede implementujte aj metódu `can_program(language)`, ktorá vracia `True` alebo `False` v závislosti od toho, či vývojár ovláda programovací jazyk, ktorý metóda dostane ako parameter.

## Trieda `Manager` – 0,5 bodov

Trieda `Manager` reprezentuje projektového manažéra, ktorý je prvým typom zamestnanca v našom modeli.

(0,2b) Implementujte konštruktor s rovnakými parametrami a funkcionalitou ako v triede `Employee`, a zabezpečte aj to, aby `Manager` dedila od triedy `Employee`.

(0,3b) Okrem toho v triede potrebujete implementovať metódu `get_project_time(orig_time)`, ktorá vypočíta celkový počet hodín projektu, ktorý daný manažér vedie. Ako parameter `orig_time` dostanete odhadovaný čas potrebný na vyriešenie projektu ako celé číslo. Metóda vracia tiež celé číslo, pričom dostanete upravený počet hodín a to nasledovne:
- ak je manažér junior, tak odhad vynásobte konštantou 1,2 (vypracovanie projektu bude trvať dlhšie);
- ak je manažér medior, tak odhad ponechajte;
- ak je manažér senior, tak odhad vynásobte konštantou 0,8 (vypracovanie projektu bude rýchlejšie).

Dbajte na to, aby metóda stále vracala celé čísla, zaokrúhľujte vždy smerom hore.

## Trieda `Order` – 0,5 bodov

Trieda `Order` reprezentuje zákazku s členskými premennými:
- `client` – `string` – meno klienta, ktorý zákazku zadal;
- `offer` – `int/float` – cena zákazky (toľko nám zaplatí klient bez ohľadu na to, v akom čase ukončíme projekt);
- `active` – `int` – číslo dňa, od ktorého bude zákazka aktuálna (každá zákazka ostane aktuálna po dobu 3 dní);
- `total_hours` – `int` – celkový počet hodín potrebný pre vývoj zákazky;
- `stages` – `dict` – rozdelenie hodín pre riešenie jednotlivých fáz projektu – dovolené sú dva kľúče: `development` a `testing`;
- `required_technologies` – `list` – zoznam jazykov a technológií, ktoré musia ovládať členovia riešiteľského tímu pre riešenie zákazky (stačí ak každú technológiu ovláda jeden člen).

V triede potrebujete implementovať nasledovnú funkcionalitu:
- (0,1b) implementujte konštruktor triedy `Order`, ktorý inicializuje objednávku s údajmi o klientovi, cenovej ponuke, celkovom počte hodín a o dni kedy bola zákazka zadaná. Fázy projektu (`stages`) nastavte ako prázdny slovník, a zoznam potrebných technológií nech je tiež prázdny.
- (0,2b) napíšte metódu `set_tasks(development, testing)`, ktorá nastaví počet hodín fáz vývoja a testovania. Ak súčet hodín pre vývoj a testovanie neodpovedá celkovému času, metóda má vygenerovať výnimku `ValueError` s ľubovoľnou správou.
- (0,2b) implementujte metódu `add_required_technologies(technologies)`, ktorá pridá požadované technológie zákazky. Metóda musí správne spracovať prípad, keď parameter obsahuje jednu technológiu (hodnota typu `string`), alebo zoznam technológií (zoznam `string`ov). Aktualizujte členskú premennú `required_technologies`.

## Trieda `Project` – 2 body

Trieda `Project` reprezentuje vývojový projekt, ktorý vytvoríte vtedy, ak prijmete zákazku a priradíte na jej riešenie niekoľkých zamestnancov. Trieda má byť reprezentovaná nasledovnými členskými premennými:
- `company` – `Company` – firma, ktorá projekt rieši;
- `started` – `int` – deň v simulácii, kedy sa projekt začal riešiť;
- `order` – `Order` – zákazka, na základe ktorej vznikol projekt;
- `manager` – `Manager` – manažér projektu (vždy len jeden);
- `employees` – `list` – zoznam zamestnancov, ktorí riešia projekt (okrem manažéra);
- `remaining_hours` – `int` – počet hodín potrebných na dokončenie projektu.

V triede implementujte nasledovné metódy:
- (0,5b) implementujte konštruktor s definovanými parametrami, kde nastavíte hodnotu členských premenných. Zoznam zamestnancov `employees`, ktorý dostanete ako vstupný parameter, obsahuje aj manažéra, potrebujete ho teda zo zoznamu vybrať a pridať ho do príslušnej členskej premennej. V členskej premennej `employees` potom ostanú všetci ostatní. Všetkým zapojeným zamestnancom nastavte aktuálny projekt (`curr_project`) na práve vytvorenú inštanciu. Pri spracovaní zoznamu zamestnancov si skontrolujte, či máte presne

jedného manažéra v zozname. Ak manažér chýba, alebo je ich viac, vygenerujte `ValueError` s ľubovoľnou chybovou správou. Členskú premennú `remaining_hours` inicializujte vzhľadom na počet hodín uvedených v zákazke a na senioritu manažéra.

- (0,2b) napíšte metódu `update()`, ktorá nasimuluje jeden deň v simulácii a zníži počet hodín potrebných na dokončenie projektu. Na každého zamestnanca okrem manažéra rátajte s 8 odrobenými hodinami na daný deň, zabezpečte však, aby hodnota `remaining_hours` nikdy neklesla pod 0 (na konci riešenia projektu musí byť 0).
- (0,1b) implementujte metódu `is_finished()`, ktorá vracia `True`, ak bol projekt ukončený, v opačnom prípade vracia `False`. Vychádzajte z počtu ostávajúcich hodín.
- (0,2b) napíšte metódu `clear_project()`, ktorá uvoľní zamestnancov z projektu po jeho dokončení (vynulujte členskú premennú `curr_project`). Informáciu o projekte aktualizujte aj v triede `Company` (projekt presuňte zo zoznamu aktívnych projektov do zoznamu dokončených).
- (1b) implementujte metódu `get_income()`, ktorá vypočíta a vráti zisk firmy za daný projekt. Pri výpočte postupujte nasledovne:
    1. Berte do úvahy cenu zákazky, ktorú zaplatil klient (z triedy `Order`). Ak ste pre niektorého klienta urobili aspoň 3 zákazky, tak pre všetky vám zaplatí o 50 percent viac ako pôvodná cena (vrátane prvých troch). Metóda bude volaná iba na konci simulácie, preto môžete rátať s tým, že všetky projekty budú už ukončené.
    2. Vypočítajte mzdy všetkých testerov aj vývojárov. Mzdové náklady musíte vypočítať pre každú fázu zvlášť a to tak, že celkový počet hodín danej fázy rozdelíte rovnako medzi všetkými testermi resp. vývojármi a sčítate ich mzdy za všetky odrobené hodiny. Napríklad ak testovanie trvalo 100 hodín a na projekte ste mali 5 testerov, tak každý z nich odrobil po 20 hodín. Pri reálnych časoch jednotlivých fáz nezabudnite rátať aj so skúsenosťou manažéra. Počet hodín zamestnancov zaokrúhľujte smerom hore (platíte za začaté hodiny).
    3. Vypočítajte mzdové náklady projektového manažéra. Toho zaplatíte za toľko hodín, koľko odrobil priemerný zamestnanec na projekte (teda celkový čas delený celkovým počtom riešiteľov). Počet hodín zaokrúhľujte smerom hore (platíte za začaté hodiny).
    4. Váš zisk vypočítate ako rozdiel príjmu a mzdových nákladov.

## Trieda `Company` – 2 body

Trieda `Company` reprezentuje vývojársku firmu, trieda má byť reprezentovaná nasledovnými členskými premennými:

- `employees` – `list` – zoznam zamestnancov firmy;
- `active_projects` – `list` – zoznam práve riešených projektov;
- `finished_projects` – `list` – zoznam dokončených projektov;
- `scheduler` – `Scheduler` – firemná stratégia výberu zo zákaziek.

Konštruktor triedy je už hotový, nepotrebujete ho implementovať (doplniť ho však môžete podľa potreby). Do triedy pridajte nasledovné metódy:

- (0,2b) `calculate_total_income()` – metóda vypočíta celkový zisk firmy, teda súčet ziskov za jednotlivé ukončené projekty. Návratová hodnota môže byť celé alebo desatinné číslo.
- (0,1b) `get_available_employees(emp_type)` – metóda vracia zoznam zamestnancov zadaného typu (parameter `emp_type`), ktorí sú k dispozícii (momentálne nemajú pridelený projekt). Parameter bude niektorá z tried `Employee`, `Tester`, `Developer` a `Manager`.

- (0,1b) `get_lang_developers(language)` – metóda vracia zoznam vývojárov, ktorí ovládajú zadaný programovací jazyk (parameter `language`).
- (0,2b) `is_frequent_client(client_name)` – metóda vracia informáciu o tom, či je klient so zadaným menom (parameter `client_name`) častý zákazník, alebo nie. Zákazník je častý, ak ste preňho dokončili aspoň tri projekty (vrátane). Vstupný parameter je typu `string`.
- (0,2b) `close_project(project)` – metóda presunie projekt zo zoznamu aktívnych projektov do zoznamu dokončených projektov. Vstupný parameter je objekt typu `Project`. Ak sa zadaný objekt nenachádza medzi aktívnymi projektmi, metóda nič nerobí, nesmie sa však vygenerovať výnimka.
- (0,2b) `solving(order)` – metóda vracia informáciu o tom, či firma aktuálne pracuje alebo pracovala na zadanej zákazke (parameter `order` typu `Order`). Návratová hodnota je `True` alebo `False`. Zákazku hľadajte aj v zozname aktuálnych aj dokončených projektov.
- (0,5b) `can_solve_order(order)` – metóda zistí, či firma má dostatok zamestnancov pre riešenie zadanej zákazky (parameter `order` typu `Order`) a vracia `True` alebo `False`. Aby zákazka bola riešiteľná firmou, musia platiť nasledovné požiadavky:
  - musí byť dostupný aspoň jeden manažér;
  - pre každú potrebnú technológiu musí byť dostupný aspoň jeden vývojár;
  - počet dostupných vývojárov musí byť aspoň rovný počtu požadovaných technológií (jeden vývojár nemôže zastrešovať dve technológie);
  - ak zákazka vyžaduje testovanie, musí byť dostupný aspoň jeden tester.
- (0,5b) `simulate_day(day_no)` – metóda nasimuluje jeden deň v roku s poradovým číslom `day_no` (vstupný parameter typu `int`). Každý deň urobte nasledovné kroky:
  - aktualizujte bežiace projekty – aktualizujte počet ostávajúcich hodín a keď bol projekt dokončený, presuňte ho do zoznamu ukončených projektov;
  - vyberte nový projekt (kým je to možné) – výber realizuje `scheduler`, v ten istý deň môžete začať niekoľko nových projektov, tie pridajte do zoznamu aktívnych projektov.

## Trieda `Scheduler` – 0,5 bodov

Trieda `Scheduler` predstavuje všeobecnú funkcionalitu výberovej stratégie, konkrétna funkcionalita bude implementovaná v podtriedach. V triede máte dve členské premenné: `offers`, čo je zoznam všetkých zákaziek, ktoré firma dostane počas simulácie; a `company`, čo je referencia na firmu. Konštruktor nastaví len zoznam zákaziek, firma sa nastaví v ďalšej metóde.

V triede implementujte nasledovné metódy:
- (0,1b) `set_company(company)` – metóda nastaví hodnotu príslušnej členskej premennej;
- (0,2b) `get_active_offers(day_no)` – metóda vyberie zo zoznamu všetkých zákaziek tie, ktoré sú aktuálne v daný deň simulácie (parameter `day_no` typu `int`). Zákazka ostáva aktívna, ak na nej firma ešte nepracuje (ani nepracovala) a nie je staršia ako 3 dni. Ak zákazka začne byť aktívna v 10. deň simulácie, aktívna bude v 10., v 11. a v 12. deň simulácie. Začiatok zákazky nájdete v triede `Order`.
- (0,2b) `get_new_project(day_no)` – metóda reprezentuje výber a vytvorenie nového projektu v istý deň simulácie (parameter `day_no` typu `int`). Ak stratégia ešte nemá nastavenú firmu, metóda končí s návratovou hodnotou `None` (nevygeneruje sa chyba). Ak firma už bola nastavená, tak nový projekt vyberte nasledovne:
  1. získajte zoznam aktívnych ponúk;

2. vyberte najvhodnejšiu zákazku a priraďte jej zamestnancov (metóda `select_offer(offers)`, implementovaná bude v podtriedach);
3. ak nebola vybratá žiadna zákazka, metóda vracia `None`;
4. v opačnom prípade metóda vráti nový objekt typu `Project`, zavolajte konštruktor na základe zistených hodnôt.

Trieda okrem toho deklaruje metódy `select_offer(offers)` a `select_employees(order)`. Tie ostávajú v triede `Scheduler` neimplementované (generujú `RuntimeError`), napíšete ich v podtriedach. Za podtriedy získate body iba ak máte obe metódy správne implementované.

Metóda `select_offer(offers)` vyberá zo zoznamu ponúk jednu zákazku (podľa konkrétnej stratégie). Metóda má dve návratové hodnoty, prvá je vybraná zákazka a druhá je zoznam priradených zamestnancov (získate ho zo `select_employees()`). Ak žiadna zákazka nemohla byť vybratá napríklad pre nedostupnosť zamestnancov, metóda vracia dvojicu hodnôt `None`, `None`.

Metóda `select_employees(order)` priradí zamestnancov ku konkrétnemu projektu v rámci zákazky `order`. Návratová hodnota metódy je zoznam pridelených zamestnancov.

Pri výbere zamestnancov na projekt sa v každom prípade držte niekoľkými zásadami:
- vždy vyberajte iba jedného manažéra na projekt.
- počet testerov resp. vývojárov zistíte tak, že počet hodín na danú fázu predelíte 8 (maximálny počet hodín jedného zamestnanca na deň, projekt chcete dokončiť čo najskôr). Samozrejme aj tu zaokrúhľujeme smerom nahor. Pri počte hodín nerátajte so senioritou manažéra, vychádzajte z predpokladaného času. Po delení dostanete ideálny počet testerov/vývojárov, samozrejme sa ale môže stať, že ich nebudete mať toľko dostupných, v tomto prípade teda berte nižší počet.
- v prípade vývojárov musíte brať do úvahy aj počet požadovaných technológií – na projekte nesmiete mať menej vývojárov, ako je technológií. Teda ak máte zákazku, kde musíte pracovať s Pythonom a s HTML, aj keď máte vývojára, ktorý ovláda oba jazyky, na projekt musíte prideliť aspoň dvoch (pričom nemusia všetci ovládať oba jazyky).
- jeden zamestnanec môže pracovať v ľubovoľný deň iba na jednom projekte, aj keby končil skôr, nezačne ďalší projekt.

Konkrétne výbery implementujte v podtriedach s nasledovnou logikou:
- (0,5b) `FirstScheduler` – vyberiete prvú zákazku, ktorú viete riešiť (samozrejme len zo zoznamu aktuálnych zákaziek). Pri prideľovaní zamestnancov tiež zoberiete prvého dostupného manažéra, testerov a vývojárov vyberáte náhodne. Počet testerov a vývojárov sa riadi vyššie uvedenými pravidlami, pri vývojároch nezabudnite zabezpečiť, aby každá technológia bola pokrytá aspoň jedným z nich.
- (0,5b) `RandomScheduler` – vyberiete náhodnú zákazku zo zoznamu aktuálnych zákaziek, ktoré viete riešiť. Pri prideľovaní zamestnancov zoberiete náhodného dostupného manažéra, testerov a vývojárov vyberáte tiež náhodne. Počet testerov a vývojárov sa riadi pravidlami uvedenými vyššie, pri vývojároch nezabudnite zabezpečiť, aby každá technológia bola pokrytá aspoň jedným z nich.
- (1b) `GreedyScheduler` – zo zoznamu aktuálnych riešiteľných zákaziek vyberáte tú s najvyššou ponúkanou cenou. Pri prideľovaní zamestnancov zoberiete náhodného dostupného manažéra, testerov a vývojárov vyberáte tiež náhodne. Počet testerov a vývojárov sa riadi

pravidlami uvedenými vyššie, pri vývojároch nezabudnite zabezpečiť, aby každá technológia bola pokrytá aspoň jedným z nich.
- (1b) `CheapestEmployeeScheduler` – zo zoznamu aktuálnych riešiteľných zákaziek vyberáte tú s najvyššou ponúkanou cenou. Pri prideľovaní zamestnancov zoberiete dostupného manažéra s najnižším platom, testerov a vývojárov vyberáte tiež na základe ich platu – beriete tých, ktorí majú najnižšie mzdové náklady. Počet testerov a vývojárov sa riadi pravidlami uvedenými vyššie, pri vývojároch nezabudnite zabezpečiť, aby každá technológia bola pokrytá aspoň jedným z nich.

V súbore `main.py` máte pripravený ukážkový kód na generovanie a spustenie náhodnej simulácie. Kód môžete ľubovoľne upravovať, slúži len ako pomôcka pre lepšie pochopenie prepojenia jednotlivých tried. Parametre simulácie viete nastaviť v konštantách na začiatku súboru, resp. pri volaní funkcií `run_simulation()` a `generate_orders()`.