Beháme po úradoch

Cieľom druhého zadania je vytvoriť jednoduchý model fungovania úradu, kde klienti prídu vybavovať rôzne záležitosti a musia si vybrať medzi ochotnými aj menej ochotnými úradníkmi. Pri riešení úlohy vytvoríte výpočtové modely, a precvičíte si prácu s dedičnosťou a abstraktnými triedami. Tiež získate skúsenosti s rôznymi údajovými štruktúrami v C#. Pri riešení využijete aj náhodnosť.

V projekte potrebujete implementovať niekoľko tried, pričom musíte dodržať štruktúru projektu ako aj namespacy:

`Program.cs` (namespace `Assignment2`) – hlavný súbor, implementujete tu metódu pre spustenie simulácie

`Forms` (namespace `Assignment2.Forms`) – adresár so súbormi modelujúcimi rôzne formuláre a agendy

- `Agenda.cs` – agenda, ktorú klient chce vybaviť
- `AgendaType.cs` – enumerácia s kategóriami agend
- `Form.cs` – trieda reprezentujúca formulár, ktorý zákazník musí vyplniť pre vyriešenie istej agendy

`Clerks` (namespace `Assignment2.Clerks`) – adresár s triedami rôznych typov úradníkov

- `AbstractClerk.cs` – abstraktná trieda so spoločnou funkcionalitou úradníkov
- `RudeClerk.cs` – trieda reprezentujúca nepríjemného úradníka, ktorý klientovi nepomôže, ak niektorý formulár vyplnil nesprávne
- `KindClerk.cs` – trieda reprezentujúca nápomocného úradníka, ktorý klientovi pomôže opraviť zle vyplnené formuláre
- `ImpatientClerk.cs` – trieda reprezentujúca netrpezlivého úradníka, ktorý sa správa podľa toho, koľkí čakajú v jeho rade

`Clients` (namespace `Assignment2.Clients`) – adresár s triedami rôznych typov klientov

- `Client.cs` – trieda reprezentujúca základného klienta, ktorý si vyberá náhodne medzi úradníkmi
- `CarefulClient.cs` – trieda reprezentujúca klienta, ktorý primárne vyberá správneho nápomocného úradníka
- `EagerClient.cs` – trieda reprezentujúca klienta, ktorý sa ponáhľa a ide tam, kde je rad najkratší
- `OptimizingClient.cs` – trieda reprezentujúca klienta, ktorý sa snaží nájsť rad, v ktorom končí čo najskôr

Kostru riešenia nájdete v predpripravenom projekte. Keďže vaše riešenia prejdú automatizovanými testami, je dôležité, aby ste dodržali štruktúru projektu, najmä čo sa týka menných priestorov. Ak programujete v IDE, ktoré vytvára inú štruktúru ako vidíte v kostre riešenia, nezabudnite upraviť štruktúru pred finálnym odovzdaním.

Všetky členské premenné musia byť privátne, a nesmiete k nim pristupovať priamo mimo triedy. Členské premenné v nadtriedach môžu byť `protected`. Vaše riešenie môžete rozšíriť o premenné a metódy okrem tých uvedených v zadaní.

Uvedené poradie tried reprezentuje odporúčané poradie implementácie, avšak niektoré metódy sa spoliehajú na metódy iných tried, práve preto je dôležité, aby ste si na úvod prečítali celé znenie zadania.

`AgendaType`

Súbor obsahuje enumeráciu s možnými kategóriami agend. Súbor nepotrebujete nijak upravovať. Ak niečo zmeníte, tak dbajte na to, aby naďalej boli dostupné všetky pôvodne zadefinované kategórie: `Personal, Housing, Documents, Vehicles`.

`Form` – 0,5 bodov

Trieda definuje model formulára, ktorý klient musí vyplniť pri vybavovaní istej agendy. Slúži to iba na reprezentáciu základných vlastností, ktoré budú využité počas simulácie, nepotrebujete definovať žiadnu špecifickú funkcionalitu okrem konštruktora. Konštruktor máte deklarovaný, potrebujete ho doplniť tak, aby ste nastavili hodnoty členských premenných:

- `difficulty` (`double`) – hodnota od 0 po 1, vyjadruje obťažnosť formulára, resp. pravdepodobnosť toho, že ho klient správne vyplní;
- `filledOut` (`bool`) – vyjadruje, či bol formulár (správne) vyplnený – na začiatku má byť formulár nevyplnený;
- `formTitle` (`string`) – názov formulára;
- `handlingTime` (`int`) – čas potrebný na spracovanie vyplneného formulára – rátame s diskrétnym časom, vyjadruje to teda počet kôl.

V konštruktore vykonajte kontrolu hodnoty `difficulty`, ak nebude z určeného intervalu, vygenerujte `ArgumentException` so správou *Invalid form difficulty*.

`Agenda` – 1 bod

Trieda definuje agendu, ktorú chce klient vybaviť na úrade. Každá agenda je daná nasledovnými hodnotami:

- `type` (`AgendaType`) – hodnota z enumerácie, vyjadruje kategóriu, do ktorej agenda patrí;
- `forms` (`List<Form>`) – zoznam formulárov, ktoré klient musí vyplniť pre úspešné vyriešenie agendy;
- `handled` (`bool`) – vyjadruje, či bola agenda vybavená.

V triede implementujte nasledovné metódy:

- v konštruktore nastavte typ agendy podľa parametra, inicializujte zoznam formulárov ako prázdny, a nastavte agendu na zatiaľ nevybavenú.
- metóda `AddForm(Form form)` pridá do zoznamu formulár, ktorý dostane ako argument. Zabráňte pritom duplikátom. **Pozor:** duplicitu neriešte na úrovni objektov, ale na úrovni názvu formulára. Dve inštancie formulára s rovnakým názvom by nemali byť v jednej agende.
- metóda `RemoveForm(Form form)` vymaže zo zoznamu formulárov inštanciu, ktorá má rovnaký názov ako argument. Kontrolu teda zase vykonajte na základe názvu a nie na základe inštancie (podobne ako pri pridávaní). Ak agenda neobsahuje formulár s daným názvom, nesmie sa vygenerovať chyba, ale nedôjde k žiadnym zmenám.
- metóda `CalculateTime()` vráti celkový čas potrebný pre vybavenie agendy. Je to súčet časov, ktoré potrebujete na spracovanie jednotlivých formulárov spätých s agendou. Ak agenda nevyžaduje žiadne formuláre, výsledok má byť 1.

`AbstractClerk` – 1 bod + 3×0,5 bodov za podtriedy

Abstraktná trieda definujúca spoločnú funkcionalitu všetkých úradníkov spolu s ich modelom a vlastnosťami. Každý úradník je definovaný nasledovnými premennými:

- `agendas` (`List<AgendaType>`) – zoznam typov agend, ktoré daný úradník vybavuje;
- `speed` (`int`) – rýchlosť úradníka, vyjadruje, koľko kôl vie vykonať v jednom kroku simulácie pri spracúvaní formulárov a agend;
- `queue` (`List<Client>`) – zoznam klientov, ktorí stoja v rade úradníka.

Do triedy pridajte nasledovnú funkcionalitu:

- v konštruktore inicializujte členské premenné podľa atribútov, rad klientov nastavte ako prázdny zoznam;
- metóda `FilterAgenda(Agenda agenda)` nech vráti `true` alebo `false` v závislosti od toho, či agenda, ktorú dostane ako parameter je typu, ktorý daný úradník vybavuje;
- metóda `GetWaitingCount()` nech vráti počet čakajúcich ľudí v rade úradníka;
- metóda `HandleNext()` nech vybaví prvého čakajúceho klienta v rade úradníka. Vybavovanie môže byť neúspešné, klienta však každopádne odstráňte z radu (ostatní sa posunú o pozíciu vpred);
- metóda `HandleAgenda(Agenda agenda)` vybaví agendu, ktorú dostane ako argument, ak tá je správneho typu (úradník ju vybavuje). Vybavenie agendy reprezentujte vhodným nastavením príslušnej premennej v triede `Agenda`.

Okrem toho trieda obsahuje abstraktnú metódu `HandleClient(Client client, Agenda agenda)`, ktorá simuluje vybavovanie klienta úradníkom. Metódu implementujte v podtriedach s nasledovnou logikou (získate pol boda za každú správnu implementáciu):

- `RudeClerk` – agendu nevybaví, ak je nesprávneho typu, alebo aspoň jeden z potrebných formulárov bol nesprávne vyplnený. V opačnom prípade bude agenda vybavená. Metóda vráti počet krokov, ktorý úradník potrebuje na spracovanie formulárov. Túto hodnotu dostanete ako hornú hranicu celkového času potrebného na spracovanie všetkých formulárov deleného rýchlosťou úradníka. Napríklad ak pre spracovanie formulárov je potrebných celkovo 15 kôl a úradník má rýchlosť 4, metóda vracia 4 (vždy teda zaokrúhľujte hore). Zaokrúhľujte iba celkový počet krokov, nie kroky za spracovanie jednotlivých formulárov. Ak agenda nebude vybavená, metóda vráti 1.
- `KindClerk` – agendu určite vybaví, ak je správneho typu. Ak niektorý formulár bol nesprávne vyplnený, úradník ho opraví. Metóda vráti počet krokov potrebných na spracovanie formulárov podľa rovnakého pravidla ako `RudeClerk`.
- `ImpatientClerk` – úradník sa správa ako `RudeClerk` ak v jeho rade stojí minimálne `limit` počet ľudí, ak je ich menej, správa sa ako `KindClerk`. Limit je nastavený priamo v konštruktore, návratová hodnota metódy sa riadi rovnakými pravidlami ako v prípade predošlých tried.

`Client` – 1 bod + 3×1 bod za podtriedy

Trieda definuje základného klienta, ktorý si vyberá medzi úradníkmi náhodne, a zároveň slúži ako nadtrieda pre implementáciu ďalších typov klientov. Každý klient bude definovaný nasledovnými premennými:

- `ability` (`double`) – hodnota od 0 po 1, vyjadruje schopnosť klienta správne vyplniť formuláre;
- `agenda` (`Agenda`) – agenda, ktorú chce klient vybaviť na úrade;
- `visited` (`List<AbstractClerk>`) – zoznam úradníkov, u ktorých klient už skúsil vybaviť svoju agendu.

Do triedy pridajte nasledovnú funkcionalitu:

- v konštruktore inicializujte členské premenné podľa atribútu, rad navštívených úradníkov nastavte ako prázdny zoznam a klient zatiaľ nebude mať žiadnu agendu;
- metóda `FillOutForms(Agenda agenda)` simuluje vyplnenie formulárov istej agendy klientom. Úspešnosť vyplnenie nasimulujete pomocou náhodnosti podľa nasledovných pravidiel:
    - ak schopnosť klienta vypĺňať formuláre je aspoň dvakrát taká aká je obťažnosť daného formulára, vyplnenie bude určite správne (ak obťažnosť je *0,01* a schopnosť klienta je *0,1*, formulár vyplní s istotou správne);
    - v opačnom prípade vygenerujte číslo od 0 po schopnosť klienta; ak táto hodnota bude vyššia ako obťažnosť vyplnenia formulára, formulár bude správne vyplnený.

  Nezabudnite nasimulovať vyplnenie všetkých formulárov danej agendy.
- metóda `SelectClerk(List<AbstractClerk> clerks)` vráti vybraného úradníka zo zoznamu `clerks`, ktorý dostanete ako argument. Agendu, ktorú chce klient vyriešiť budete mať stále v členskej premennej `agenda`. Základný klient (`Client`) vyberá z úradníkov náhodne, pričom neberie do úvahy, aké typy agend riešia. Klient však nenavštívi dvakrát toho istého úradníka, ak nie je k tomu nútený. Primárne teda vyberá náhodne z tých úradníkov, ktorých ešte nenavštívil. Ak už bol neúspešný u všetkých, znova vyberá zo zoznamu všetkých úradníkov.
- metóda `SolveAgenda(Agenda agenda, List<AbstractClerk> clerks)` vráti dvojicu hodnôt, a to konkrétne úradníka, ktorého si klient vybral, ako aj počet krokov spracovania jeho agendy (údaj dostane od úradníka). V metóde nech klient najprv vyplní formuláre, následne si vyberie úradníka, ktorého navštívi (nezabudnite aktualizovať príslušný zoznam), a následne skúsi vybaviť svoju agendu u vybraného úradníka.

Trieda `Client` má tri podtriedy, v ktorých potrebujete predefinovať metódu pre výber úradníka podľa nasledovných kritérií:

- `EagerClient` – klient sa ponáhľa a práve preto sa najprv pozrie na to, ktorí úradníci vybavujú jeho agendu (aspoň jeden taký určite bude). Následne si vyberie toho úradníka, pri ktorom stojí najmenej ľudí. Ak pred dvomi úradníkmi je rovnako dlhý rad, môže si z nich vybrať hociktorého. `EagerClient` neberie do úvahy zoznam navštívených úradníkov, kľudne sa vráti k tomu istému úradníkovi dvakrát, ak rad je najkratší práve uňho.
- `CarefulClient` – opatrný klient si tiež vyberá primárne z tých úradníkov, ktorí majú na starosti jeho agendu, zároveň ale uprednostňuje nápomocných úradníkov. Ak nenájde žiadneho úradníka `KindClerk`, ktorý vybavuje jeho agendu, vyberá náhodne zo všetkých úradníkov. Podobne ako `EagerClient`, vyberá bez použitia histórie.
- `OptimizingClient` – klient sa ponáhľa podobne ako `EagerClient`, k výberu úradníka však pristupuje vedeckejšie. Najprv sa pozrie na to, ktorí úradníci vybavujú jeho

agendu a následne z nich vyberie toho najrýchlejšieho. Ak dvaja úradníci majú rovnakú rýchlosť, môže si z nich vybrať ľubovoľne. Zoznam navštívených úradníkov neberie do úvahy.

V triede `Program` implementujte metódu `RunSimulation(Dictionary<Client, Agenda> clientAgendas, List<AbstractClerk> clerks)`, ktorá nasimuluje vybavovanie agend na úrade. Ako prvý argument dostanete slovník, kde kľúčmi sú klienti a hodnoty pod klientom sú agendy, ktoré chcú vybaviť. Druhý parameter je zoznam úradníkov na úrade. V metóde nasimulujte výber všetkých klientov až dovtedy, kým všetci nevybavia svoju agendu. Môžete predpokladať, že pre všetky typy agend bude existovať úradník, a aj ten najmenej schopný klient, ktorý by vyberal náhodne skôr či neskôr narazí na nápomocného úradníka, ktorý mu pomôže s vyplnením formulárov. Metóda vráti počet krokov, po ktorých simulácia končí (daný úradníkom, ktorý končí najneskôr).

V metóde `Main()` nájdete ukážku niekoľkých formulárov, agend, klientov aj úradníkov.