Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Домашно 2

курсове *Обектно-ориентирано програмиране и Обектно-ориентирано програмиране-практикум*

специалности *Информатика и Компютърни науки (1-ви поток)*

летен семестър 2020/21 г.

Редакции

2021-05-12. В описанието на ACQUIRE е добавено пояснение какво да се случва, ако командата се изпълни за кола, която вече има собственик.

Съдържание

Условие на задачата

В тази задача трябва да проектирате проста програма, в която да може да се пази база с информация за хора и превозни средства.

Когато решавате задачата, покрийте класовете, които разработвате с подходящи unit test-ове.

В програмата ще работим със следните типове:

`Registration`

Представя регистрационен номер на превозно средство. За целите на задачата ще считаме, че всеки номер е поредица от символи, подредени както следва:

1. Един или два символа от латинската азбука, които указват населено място;
2. Четири цифри;
3. Два символа от латинската азбука.

Например C1234AB, XY1111YX са валидни регистрационни номера, а 111145, ABC34DEF, ABCDEF и C11D не са.

`Person`

Представя информация за човек. Всеки обект от тип Person трябва да има следните свойства:

- Име (`std::string`)
- Уникален идентификатор (`unsigned int`)
- Превозни средства, които притежава.

`Vehicle`

Представя превозно средство. То има следните свойства:

- Уникален регистрационен номер (`Registration`)
- Описание (`std::string`)
- Собственик.

За типовете са в сила следните ограничения:

- Не може да има двама човека с еднакъв идентификатор.
- Не може да има две превозни средства с еднакъв регистрационен номер.
- Едно превозно средство може да има или нула, или един собственик.
- Един човек може да притежава нула, едно или повече превозни средства.
- Всеки обект от тип `Person` съдържа препратки към превозните средства, които човекът притежава (ако има такива). Тоест по обект от тип `Person` може да се намерят обектите, които представят превозните средства, които той притежава

- Всеки обект от тип `Vehicle` има препратка към собственика му (ако има такъв). Тоест по обект от тип `Vehicle` трябва да може да се намери обектът, който представя собственика му.

- Обектите от тип `Person` и/или `Vehicle` съществуват независимо едни от други. Например дадено превозно средство може да стане собственост на даден човек, а по-късно да се укаже, че то не принадлежи на никого. Това обаче не довежда до унищожаване на превозното средство.

Програмата ви трябва да позволява на потребителя:

- да добавя или премахва записи за хора и превозни средства.

- да указва, че дадено превозно средство принадлежи на даден потребител.

- да указва, че дадено средство вече НЕ принадлежи на даден потребител.

При стартиране на програмата потребителят може да укаже име на файл, от който да се зареди информация за хората и превозните средства. В такъв случай програмата трябва да зареди информацията от файла. В противен случай се започва с празна база и потребителят може да започне да въвежда информация "от нулата".

Информацията за файл, който трябва да се зареди се подава като аргумент от командния ред на програмата. Например ако тя се стартира по дадения по-долу начин, да се зарежда съдържанието на файла `Info.txt`

```
program.exe "C:\Temp\My Program\Info.txt"
```

Респективно, ако не се подаде такъв аргумент, започваме с празна база.

По време на работа потребителят може да избере да запише цялата информация за хората и превозните средства във файл в текстов формат. Ако потребителят укаже име на вече съществуващ файл, програмата трябва да изведе предупреждение. Ако потребителят потвърди, тя трябва да презапише старото съдържание на файла.

Програмата трябва да работи в интерактивен режим. В него потребителят въвежда команди, а програмата ги интерпретира и изпълнява. Имената на командите могат да се въвеждат с малки или големи букви (например `Save`, `SAVE`, `SaVe` и `save` са валидни изписвания на командата за запис във файл).

Командите, които трябва да се поддържат са описани по-долу.

`VEHICLE <registration> <description>`

Добавя ново превозно средство с регистрационен номер `<registration>` и описание `<description>`. Ако регистрационният номер се дублира, да се изведе съобщение за грешка. Новодобавеното средство все още е без собственик.

`PERSON <name> <id>`

Добавя данни за човек с име `<name>` и идентификатор `<id>`. Новодобавеният обект все още не притежава нито едно превозно средство.

`ACQUIRE <owner-id> <vehicle-id>`

Задава, че човекът с идентификатор `<owner-id>` притежава превозното средство с регистрационен номер `<vehicle-id>`. Ако към момента на изпълнение на командата превозното средство вече има друг собственик, ефектът от командата е все едно старият собственик престава да притежава колата (`RELEASE`), а новият я придобива.

`RELEASE <owner-id> <vehicle-id>`

Задава, че човекът с идентификатор вече НЕ Е собственик на превозното средство с регистрационен номер `<vehicle-id>`

`REMOVE <what>`

`<what>` може да бъде или идентификатор на човек, или регистрационен номер на кола. Програмата трябва да определи сама какво е получила. След това тя трябва да премахне съответния обект от базата. Ако в нея няма запис за обект с идентификатор `<what>`, да се изведе съобщение и да не се прави нищо. Ако се опитва да се премахне кола, която има собственик или собственик, който притежава коли, да се изведе съобщение и да се попита потребителят дали е сигурен. Ако той потвърди, обектът да се изтрие, но да се коригират останалите, които са свързани с него.

`SAVE <path>`

Записва базата във файла с път `<path>`.

`SHOW [PEOPLE|VEHICLES|<id>]`

Показва информация за един или повече обекти на екрана. Ако потребителят укаже опциите `PEOPLE` или `VEHICLES` се извеждат съответно всички записи за хора или всички записи за превозни средства съхранени в програмата. Ако потребителят укаже `<id>` на обект (може да е човек или превозно средство), програмата показва подробна информация за това с кои други обекти е свързан.

Например ако се подаде идентификатор на даден човек, тя трябва да покаже кои превозни средства притежава той.

На някои от командите може да се подаде аргумент, който съдържа в себе си интервали. В тези случаи трябва да се използват кавички. Например за създаване на запис за човек с име “Ivan Petrov”, може да се използва командата:

```
PERSON "Ivan Ivanov" 123
```

Упътвания и допълнителни уточнения

1. Изнесете обработката на командите в отделен клас (или класове). За обработката тръгнете от един символен низ. Разделете го на части (команда и аргументи). Проверете дали е подадена валидна команда и дали са подадени нужният брой аргументи. Ако нещо се обърка хвърлете изключение.

2. Използвайте класа `std::vector`, за да пазите обекти от тип `Person` и `Vehicle`. Недейте да реализирате сами контейнер за динамичен масив.

3. Когато записвате и зареждате информация от файл, трябва да успеете да запазите идентификаторите на отделните обекти.

4. Вместо да измисляте нов файлов формат, можете да използвате синтаксиса на командите. Когато записвате базата във файл, изведете съдържанието ѝ под формата на една или повече команди, всяка на отделен ред, които ако се обработят, ще доведат до възпроизвеждане на базата в паметта. Респективно, когато четете съдържанието на базата от файл, четете и изпълнявайте командите.

5. Идея за това как да моделирате връзката между собственик и превозни средства можете да намерите например в:
   Martin Fowler (2003) UML Distilled, глава *"Chapter 3: Class Diagrams: the essentials"*, раздел *"Bidirectional associations"*.
   На български книгата е издадена като:
   Мартин Фаулър (2004) UML основи; ISBN: 9546853062.