# Упражнения: Абстрактни класове и интерфейси – допълнителни задачи

Можете да проверите решенията си в **Judge системата**: <https://judge.softuni.org/Contests/Practice/Index/4067>

## Граничен контрол

Пренасяме се в бъдещето и вие сте владетел на тоталитарно дистопично общество, обитавано от **граждани** и **роботи**. Тъй като се страхувате от потенциални бунтове, решавате да имплементирате стриктен контрол върху това кой влиза във вашия град. Вашите войници проверяват **Id**-тата на всеки, който влиза и излиза от града.

До получаване на команда "**End**" на всеки ред ще имате информация или за **гражданин**, или за **робот**, който иска да влезе в града, в следния формат: "**{name} {age} {id}**" за **граждани** и "**{model} {id}"** за **роботи**. След получаване на команда "**End**", ще получите **едно число**, което представлява **последните цифри на фалшивите id**-та. **Всички граждани** и **роботи**, чиито **id**-та завършват с тези цифри, трябва да бъдат **задържани**.

Изходът от програмата ви трябва да се състои от **id**-тата на всички **задържани** граждани и роботи **на отделен ред** в последователността, в която сте ги приели.

Моделирайте чрез **класове** и **интерфейси** входните данни (например IPerson, Citizen, Robot). Прочетете входа в списък от IPerson и го обработете, за да получите резултата на изхода.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Pesho 22 9010101122  MK-13 558833251  MK-12 33283122  End  122 | 9010101122  33283122 |
| Toncho 31 7801211340  Penka 29 8007181534  IV-228 999999  Stamat 54 3401018380  KKK-666 80808080  End  340 | 7801211340 |

## Рожден ден

Известен факт е, че хората празнуват рождените си дни, а понякога празнуват рождените дни и на своите домашни любимци.

Разширете програмата от предишната задача, като добавите **рождените дни** на гражданите и създадете клас **Pet**. Всеки домашен любимец има **name** и **birthdate**. Разпределете повтарящата се функционалност в интерфейси и ги имплементирайте в класовете си.

До получаване на команда "**End**", всеки ред ще съдържа информация в един от следните формати:

* "**Citizen <name> <age> <id> <birthdate>**" за **гражданин**
* "**Robot** **<model> <id>**" за **робот**
* "**Pet <name> <birthdate>**" за **домашен любимец**

След получаване на команда "**End**" ще получите **едно число**, което представлява **конкретна година**. Вашата задача е да отпечатате **всички** **рождени дни** (и за граждани, и за домашни любимци) от тази година във формата **ден/месец/година** в реда, в който сте ги получили.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| Citizen Pesho 22 9010101122 10/10/1990  Pet Sharo 13/11/2005  Robot MK-13 558833251  End  1990 | 10/10/1990 |
| Citizen Stamat 16 0041018380 01/01/2000  Robot MK-10 12345678  Robot PP-09 00000001  Pet Topcho 24/12/2000  Pet Kosmat 12/06/2002  End  2000 | 01/01/2000  24/12/2000 |
| Robot VV-XYZ 11213141  Citizen Penka 35 7903210713 21/03/1979  Citizen Kane 40 7409073566 07/09/1974  End  1975 | <empty output> |

## Недостатъчна храна

Вашето тоталитарно общество страда от недостиг на храна и се появяват много бунтовници. Разширете кода от предишната задача с нова функционалност, за да разрешите този проблем.

Дефинирайте клас **Rebel**, който има **name (име)**, **age (възраст)** и **group** (**група**). Имената са **уникални** – няма да има двама бунтовници/граждани с едно и също име.

Дефинирайте интерфейс **IBuyer**, който дефинира метод **BuyFood()** и свойство **int Food**. Имплементирайте интерфейса **IBuyer** в класовете **Citizen** и **Rebel**. И бунтовниците, и гражданите започват с **0 Food**. Когато бунтовник купи храна, **Food** се увеличава с **5**, а когато гражданин купи храна, **Food** се увеличава с **10**.

### Вход

1. Първи ред – цяло число **N** – общият брой на хората
2. На следващите **N** реда – информация в един от следните формати:
   * "**<name> <age> <id> <birthdate>**" за гражданин (**Citizen**)
   * "**<name> <age><group>**" за бунтовник (**Rebel**)
3. След получаване на **N** реда, до получаване на команда "**End**", ще получавате **имена на хора**, които са **купили храна**, всеки на нов ред.

**Важно**: Не всички имена ще бъдат валидни – в случай че получите невалидно име, игнорирайте данните.

### Изход

Изходът се състои от **един ред**, на който трябва да отпечатате **общото количество купена храна**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 2  Pesho 25 8904041303 04/04/1989  Stancho 27 WildMonkeys  Pesho  Gosho  Pesho  End | 20 |
| 4  Stamat 23 TheSwarm  Toncho 44 7308185527 18/08/1973  Joro 31 Terrorists  Penka 27 881222212 22/12/1988  Jiraf  Jo ro  Jiraf  Joro  Stamat  Penka  End | 20 |

## Експлицитни интерфейси

Създайте клас **Citizen**, който имплементира два интерфейса **IResident** и **IPerson**, описани по-долу.

Създайте два интерфейса **IResident** и **IPerson**.

* **IResident** трябва да има **name** **(име)**, **country** **(държава)** и метод **GetName()**.
* **IPerson** трябва да има **name** **(име)**, **age** **(възраст)** и метод **GetName()**.

Създайте клас **Citizen**, който имплементира едновременно **IResident** и **IPerson**.

* Методът **GetName()** на **IResident** трябва да **връща** "**Mr/Ms/Mrs** " преди името
* Методът **GetName()** на **IPerson** трябва да връща **само името**.

Най-разумно ще е да ползвате експлицитна имплементация на интерфейсите, за да решите конфликта със съвпадащите имена на някои от техните методи.

### Вход

До получаване на команда "**End**" ще получавате информация за **Citizen** (всеки на отделен ред). Всеки гражданин ще бъде във формата "**<name> <country> <age>**". Създайте съответната инстанция на **Citizen** и отпечатайте неговия **GetName()** от **IPerson** и неговия **GetName()** от **IResident**.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| PeshoPeshev Bulgaria 20  End | PeshoPeshev  Mr/Ms/Mrs PeshoPeshev |
| JoroJorev Bulgaria 33  EricAnderson GreatBritain 28  PeterArmstrong USA 19  End | JoroJorev  Mr/Ms/Mrs JoroJorev  EricAnderson  Mr/Ms/Mrs EricAnderson  PeterArmstrong  Mr/Ms/Mrs PeterArmstrong |