Домашна работа - 1

Обектно-ориентирано програмиране

1 Дати

Реализирайте клас за работа с дати. Класът трябва да има 3 член-данни: ден,месец и година. Класът трябва да има:

1. Конструктор по подразбиране, който прави датата на 01.01.2000

2. Конструктор, който приема като аргумент ден, месец и година.

3. Метод, който приема като параметър число n и увеличава датата с n дни.

3. Метод, който приема като параметър число n и намалява датата с n дни.

5. Метод, който връща истина, ако годината е високосна и лъжа, ако не е.

6. Метод, който връща колко дни остават до коледа.

7. Метод, който приема друга дата и връща истина само ако другата дата е в по-късен момент от текущата.

```
{

   Date d1(31, 3, 2014); // 31.03.2014
   Date d2; //01.01.2000

   d1.addDays(1);// 01.04.2014
   d1.isLeapYear();//false (2014 is not a leap year)
   d1.isLaterThen(d2);//true

}
```

2 Battle City tournament

Сето Кайба организира турнир по Ю-Ги-О!, но е съкратил бюджета в компютърния отдел и помолил нас да организираме базата данни с всички дуелисти в турнира.

Трябва да реализирате клас Ю-Ги-О! карта (Card).Ще работим с 2 типа карти - "Карта чудовище"и "Магическа карта". Всяка карта има:

1. Име (Низ с дължина до 25 символа)

2. Атакуващи точки (ако е чудовище)(цяло число между 0 и 5000)

3. Защитни точки (ако е чудовище)(цяло число между 0 и 5000)

ВАЖНО: Тук трябва да работим само с един клас карта(а НЕ 2 отделни класа (за всеки тип)). Вие трябва да прецените как ще правите разлика м/у магическа карта и чудовище в класа.

Реализирайте клас тесте (deck).Тестето трябва да има:

1. Масив от карти (винаги да са точно 40).

2. Конструктор по подразбиране, който прави тесте с 40 карти чудовище "Token"с 0 атака и 0 защита.

3. Функция, която променя дадена карта от тестето. Ако като аргумент се подаде само име - това е магическа карта, ако не - карта чудовище.

4. Функция, която връща като резултат броя на магическите карти в тестето.

5. Функция, която връща като резултат броя на картите чудовища в тестето.

6. Функция, която запазва тестето в двоичен файл, като името на файла го приема като аргумент.

7. Конструктор, който приема като аргумент име на файл и ако файлът съществува, зарежда тестето, ако не - тестето отново става с 40 карти "Token".

Реализирайте клас дуелист (Duelist). Той трябва трябва да има:

1. Име (Низ с произволна дължина).

2. Тесте

3. Дуелистът трябва да има същите методи като в deck, с които да променяме тестето му.

```
{

   Deck d;//40 cards "Token" ATK: 0 DEF: 0


   // 4-th card in the deck becomes "Blue-eyes white dragon" with
   //ATK: 3000 DEF: 2500
   d.changeCard(4, "Blue-eyes white dragon", 3000, 2500);

   // 5-th card becomes the magic card "Monster reborn"
   d.changeCard(5, "Monster reborn");

   d.getMagicCardCount(); //1

   d.getMonsterCardCount(); //39

   d.saveToFile("MyDeck.bat"); //Saves the deck in to a file

   Deck d2("MyDeck.bat"); //Loads the deck from the file
}
```

```
{

   Duelist player1("Seto Kaiba");
   player1.changeCardInDeck(2, "Blue-eyes white dragon", 3000, 2500);

   Duelist player1("Yugi Muto");
   player1.changeCardInDeck(2, "Dark magician", 2500, 2100);

}
```

ВАЖНО. Реализирайте задачите спазвайки добрите ООП практики (валидация на данните, подходяща капсулация и тн.) и реализирайте "голямата 4-ка"само ако е нужно.