МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ В ТЕХНІЧНИХ СИСТЕМАХ

**Лабораторна робота № 3**

з дисципліни «Алгоритми та структури даних»

на тему «Введення в ООП. Класи, зв’язки між класами»

Виконав:

студент гр.ІТ-41

Ясько А.В.

Перевірив:

Март Б.А.

Київ-2015

**Завдання:**

Реалізувати класи у відповідності до варіанту.

Графічний інтерфейс не обов’язковий. Основне завдання реалізувати самі класи. Необхідно реалізувати невеличку програмку, яка демонструє роботу з даною моделлю. Можна консольну.

**Обов’язково** реалізувати в кожному класі метод **ToString**, який буде форматувати виведення інформації про об’єкт у вигляді рядку. Якщо в об’єкту є дочірні об’єкти, можна вивсти інформацію про них (наприклад під банківським рахунком – список транзакцій)

Advanced завдання робити не обов’язково, і за них не буде додаткових балів, але вони корисні в навчальних цілях.

Ви можете якимось чином покращити модель, і добавити новий функціонал. Цей проект створений виключно задля вашого навчання технологіям.

**Advanced:**

Логіку по роботі з цими класами можна винести з методу Main в статичний клас Manager.

В ньому реалізувати методи, які будуть створювати об’єкти.

**Варіант 10.**

1. Футбольний менеджер
   1. Класи: Команда, Гравець, Зіграний матч
      1. Зіграний матч містить:
         1. посилання на дві команди,
         2. рахунок
         3. дату проведення
      2. Команда містить
         1. список гравців
         2. список зіграних матчів
   2. **Advanced**:
      1. Зберігати історію переведень гравця з команди в команду (або в класі гравець, або в класі команда), ввести для цієї історії додатковий клас (Trnasfer), типу транзакції в банку
         1. Створений клас Гравец.

class Player

{

public Player(string firstName, string lastName, int salary)

{

FirstName = firstName;

LastName = lastName;

Salary = salary;

}

private string FirstName { get; set; }

private string LastName { get; set; }

private int Salary { get; set; }

public override string ToString()

{

return string.Format(

"FirstName: {0}, LastName: {1}, Salary: {2}",

FirstName, LastName, Salary);

}

}

1. Клас футбольної команди.

class Team

{

private string TeamName { get; set; }

public Team(string teamName)

{

this.TeamName = teamName;

Players = new List<Player>();

PlayedMatches = new List<PlayedMatch>();

}

public List<Player> Players { get; set; }

public List<PlayedMatch> PlayedMatches { get; set; }

public override string ToString()

{

StringBuilder sb = new StringBuilder();

sb.Append(TeamName);

sb.AppendLine();

sb.Append("Players Info");

sb.AppendLine();

foreach (var item in Players)

{

sb.AppendFormat(" {0} ", item);

sb.AppendLine();

}

sb.AppendLine();

sb.Append("Played Team Mathes Info");

sb.AppendLine();

foreach (var item in PlayedMatches)

{

sb.AppendFormat(" {0} ", item);

sb.AppendLine();

}

return sb.ToString();

}

}

1. Клас зіграного матчу.

class PlayedMatch

{

public PlayedMatch(string homeTeam, string guestTeam, string score, string dataTime)

{

HomeTeam = homeTeam;

GuestTeam = guestTeam;

Score = score;

DataTime = dataTime;

}

private string HomeTeam { get; set; }

private string GuestTeam { get; set; }

private string Score { get; set; }

private string DataTime { get; set; }

public override string ToString()

{

return string.Format(

"HomeTeam: {0}, GuestTeam: {1}, Score: {2}, DataTime: {3}",

HomeTeam, GuestTeam, Score, DataTime);

}

}

1. Клас для трансферу гравця.

class Transfer

{

public Transfer()

{

history = new List<Player>();

}

public List<Player> history { get; set; }

public void TransferAPlayer(Player namePlayer, Team oldTeam, Team newTeam)

{

oldTeam.Players.Remove(namePlayer);

newTeam.Players.Add(namePlayer);

history.Add(namePlayer);

}

}

1. Статичний клас Maneger для створення екземплярів потрібних класів.

static class Manager

{

public static Player GetInstancePlayer(string FirstName, string LastName, int Salary)

{

return new Player(FirstName, LastName, Salary);

}

public static PlayedMatch GetInstanceMatch(string HomeTeam, string GuestTeam, string Score, string DataTime)

{

return new PlayedMatch(HomeTeam, GuestTeam, Score, DataTime);

}

public static Team GetInstanceTeam (string Name)

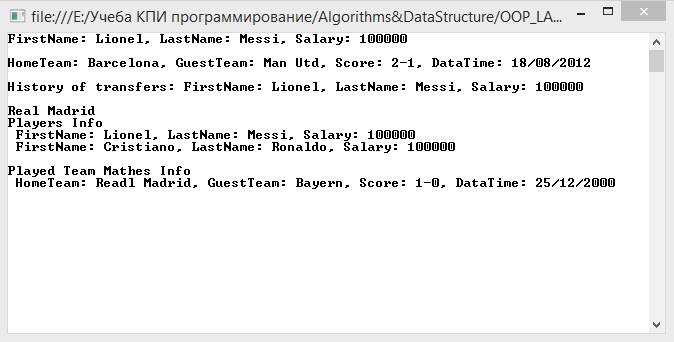
{

return new Team(Name);

}

}

Скріншот робочої програми.



**Висновок:** Клас визначає абстрактні характеристики деякої сутності, включаючи характеристики самої сутності (її атрибути або властивості) та дії, які вона здатна виконувати (її поведінки, методи або можливості). Наприклад, клас Собака може характеризуватись рисами, притаманними всім собакам, зокрема: порода, колір хутра, здатність гавкати. Класи вносять модульність та структурованість в об'єктно-орієнтовану програму. Як правило, клас має бути зрозумілим для не-програмістів, що знаються на предметній області, що, у свою чергу, значить, що клас повинен мати значення в контексті. Також, код реалізації класу має бути досить самодостатнім. Властивості та методи класу, разом називаються його членами.