Фитнес система

* Основен клас ***“User”***
  + Интерфейс “IUser“
  + Потребителя има режим (IRegimen), който ще може да си избере след регистрация(създаване на инстанция на User).
* Основен абстрактен клас ***“Exercise”***
  + Клас наследник на Exercise - *“UpperBody” – конкретен клас, упражнения за горна част*
  + Клас наследник на Еxercise =*“LowerBody” – конкретен клас, упражнения долна част*
  + *Интерфейст “IExercise”*
  + *Енумерация - „ExerciseType”*
  + *Енумерация – “MuscleGroup”*
* Основен абстрактен клас ***“Diet”***
  + *Интерфейст „IDiet”*
  + Клас наследник *“RookieDiet”*
  + Клас наследник *“WeightLossDiet”*
  + *Клас наследник „StrengthDiet”*
* Основен абстрактен клас ***“Regimen”***
  + *Интерфейст „IRegimen”*
  + *Клас наследник – “Rookie”*
  + *Клас наследник – „WeightLoss”*
  + *Клас наследник – „ Strength”*
* Основен абстрактен клас ***“TrainingProgram”***
  + *Интерфейст „ITrainingProgram”*
  + *Енумерация „Intensity”*
  + Клас наследник *“RookieProgram”*
  + Клас наследник *“WeightLossProgram”*
  + *Клас наследник „StrengthProgram”*

Плюсове:

* 4 абстрактни класа
* 5 интерфейса
* 12 класа
* 2 нива дълбочина на наследяване
* Използваме шаблона “Singleton”
* 3 Енумерации
* 1 polymorphism – ще се направи за наследниците на Diet или някъде другаде, не е проблем

Минуси:

* 1 structure
* 1 Custom Exception
* 1 Event

Предстои да се имплементира:

В библиотеката „Models”

* Class Diet и конкретните му инстанции
* Class Exercise и конкретните му инстанции

В библиотеката “Engine”

* Factory for TrainingProgram
* Factory for Diet
* Factory for Regimen
* Factory for Exercise

В библиотеката „ Data”

* Да се направят колекции от Exercises за различните режими

В клиента „Console“

* Да се направи UserManager, за регистриране/логване на потребител
* Да се направи ConsoleRenderer

*The End …*