## General Thoughts

loloto:

Да променим точкуването на цялата игра – да не се базира на брой грешки, а на нещо друго (брой грешки спрямо дължината на думата, брой опити спрямо дължината на думата (вж. по-долу) и т.н).

„mistackes“ и деривативи да се преименуват до “mistakes” или нещо по-добро.

Да добавим възожност думата да се познава директно. След като има отворена поне една буква потребителят да може да въвежда предположение за това как изглежда цялата дума. За всяка неправилно поставена буква (може да я има в думата, но да не я е сложил на рпавилното място) да се брои по 1 грешен опит. Букви, които не са на местата си да не се покзават.

guessedLetters хич не ми стана ясно какво пази – да направим едно поле Ilist – да държи вече познатите букви и едно TriedLetters да държи всички, които вече са пробвани (за да не се брои една грешка 2 пъти -> съобщение към потребителя -> промяна на метод GuessLetter).

Има бележки и по интерфейсите.

## Interfaces

IScoreBoard

bool AddScore(string name, int mistakesCount);

void Reset();

string ToString();

Бележки

Методът AddScore да приема само score (double?) и сам да преценява дали ще записва резултата в себе си и ако да да пита тогава потребителя за име.

int GetWorstScore() – отпада от интерфейса, при приключване на играта винаги ще се пита за име и ще се прави опит да се добавят точките през AddScore. Ако резултата е true значи точките са добавени в класацията, ако е false – не са били достатъчно добри. Целта е да се намали coupling-а между отделните елементи като се премахне един от публичните методи. Минус е, че винаги питаме играча за име, независимо дали ще го впишем или не... Въведеното име може да се ползва при извеждането на последващото съобщение за това дали са добавени точките в класацията или не...

public void Print() – по-добре е да се имплементира като ToString(). Не е работа на класирането да печата по конзолата. Функционалността остава същата...

IHangman

int Mistakes { get; }

bool HelpUsed { get; }

bool IsOver { get; }

void Reset();

char RevealLetter();

int GuessLetter(char letter);

string GetCurrentProgress();

Бележки

bool IsOver става на пропърти. Реално показва състояние, не някакво действие.

int NumberOccuranceOfLetter - резултата, връщан от този метод се ползва като булева променлива – дали е нула или по-голямо от нула. Преименуваме го на bool GuessLetter, защото реално това прави. В зависимост от върнатия резултат преценяваме дали буквата съществува в думата или не. По този начин ще трябва да се откажем от изписването на броя познати букви(не видях такова изискване в заданието).

Update: 3 часа по-късно се сетих, че е по-добре връщаната стойност да си остане int и да се запази функционалността непокътната... сменяме само името...

Виж общите бележки в частта за guessedLetters.

void PrintCurrentProgress би трябвало да връща стринг а не да печата на конзолата. Може да се казва string GetCurrentProgress – не ми харесва това име, ако някой има идея за по-добро нека каже ☺

По отношение на IScoreBoard смятам, че има смисъл да остане в крайния проект – има логика да се сменят начините на записване на точките – във файл, в облака...

За IHangman обаче имам известни съмнения, може да го ползваме, докато рефакторираме и в последствие да го премахнем. Има прекалено голям coupling и не виждам как може имплементацията на Hangman да се подмени с друга...

Напрактика това си е game engine-а – да се инстанцира просто. Интерфейс няма да е нужен май.

## Преименуване

Besenica.cs -> Hangman.cs

Hangman.cs -> GameEngine.cs

## Директории

doc – текстовите файлове на заданието, списъка със стъпките по рефакторирането , този файл с идеи и предложения

source – сорс кода на проекта

tests – юнит тестовете

originalFiles – името е малко тъпо, идеята е тук да се пази оригиналния проект.

## Други

В класа ScoreBoard.cs не ми харесва как се съхраняват точките. Може да се направи допълнителен клас ScoreEntry.cs и в ScoreBoard да има List<ScoreEntry> - така изглежда по-подредено. По този начин по-лесно ще се ресетва ScoreBoard, винаги ще се знае колко записа има в списъка, добавяне, сортиране, триене ще става през List. ScoreEntry трябва да имплементира IComparable.

Добре звучи ☺

static void ExecuteCommand(string command, ScoreBoard scoreBoard, besenica game) мисля, че трябва да връща стринг вместо да пише по конзолата... така ще се направят по-лесно юнит тестовете в последствие.

Да се отдели логиката на играта от presentation layer-а -> още един интерфейс и отделен клас?