**Упражнения: Качествени класове**

1. **Голям проект**

Разгледайте голям C# проект. Можете да погледнете в GitHub – потърсете проекти на езика C#, сортирайте ги по брой разклонения (forks, в низходящ ред) и изберете един от първите няколко проекта. Проектът, който сте избрали да прегледате, трябва да съдържа поне 30 класа. Попълнете таблицата по-долу, като документирате това, което намерите.

Документирайте всичко, което харесате (или не харесате) в кода.

|  |  |  |
| --- | --- | --- |
| **Клас / Метод** | **Линк** | **Ноти** |
| System.IO.File | <http://referencesource.microsoft.com/#mscorlib/system/io/file.cs> | Класовете са разделили своите задължения, нивата на абстракцията са постоянни… |
|  |  | Интерфейсите са малки, всеки интерфейс има собствени отговорности |
| … | … | … |

Някои неща, които да потърсите:

* OOP принципи
* Потвърждения и справяне с изключения
* Правилни нива на абстракция

Допълнително можете да потърсите и неща, които вече сте научили:

* Форматиране на кода
* Ясно именуване – променливи, методи, пространства от имена и т.н.
* Документация и коментари
* Използване на променливи
* Вмъкване на цикли и условни команди
* Праволинеен код
* Силна специализация и слаба зависимост

1. **\* Шаблони в дизайна**

Опитайте да намерите шаблони в дизайна на кода, който вече разгледахте в задача 1. Документирайте това, което откриете, в таблицата по-долу:

|  |  |  |
| --- | --- | --- |
| **Шаблон в дизайна** | **Пример** | **Описание** |
| Singleton | <https://github.com/apache/log4net/blob/trunk/src/log4net/LogManager.cs> | Класът **LogManager** използва Singleton защото трябва да има само една инстанция на log-мениджъра за дадено приложение |
| … | … | … |

1. **Абстракция**

Използвайте VS решението"**Abstraction**".

* Преправете кода му, за да има добра абстракция.
* Преместете свойствата и методите от класа Figure на правилните им места.
* Преместете общите методи в тялото на базовия клас.
* Махнете всичкия повтарящ се код (свойства / методи / друг код).

1. **Капсулиране**

Използвайте VS решението"**Abstraction**".

* Осигурете добро капсулиране във всички класове.
* Уверете се, че във вътрешното състояние на класовете не могат да бъдат зададени неверни стойности.

1. **Специализация и зависимост**

Използвайте VS решението"**Cohesion-and-Coupling**".

* Преправете кода му така, че да следва принципите на добра абстракция, слаба зависимост и силна специализация.
* Разделете класа Utils на други класове, които имат силна специализация и са слабо зависими вътрешно.

1. **Наследяване и полиморфизъм**

Използвайте VS решението"**Inheritance-and-Polymorphism**"

* Следвайте най-добрите практики за висококачествен програмен код.
* Проектирайте наново класовете и пренапишете кода.
* Извлечете абстрактен базов клас и преместете в него всички общи свойства.
* Капсулирайте полетата и се уверете, че задължителните полета не са оставени без стойност.
* Използвайте отново (reuse) дублирания код чрез методи в базовия клас.