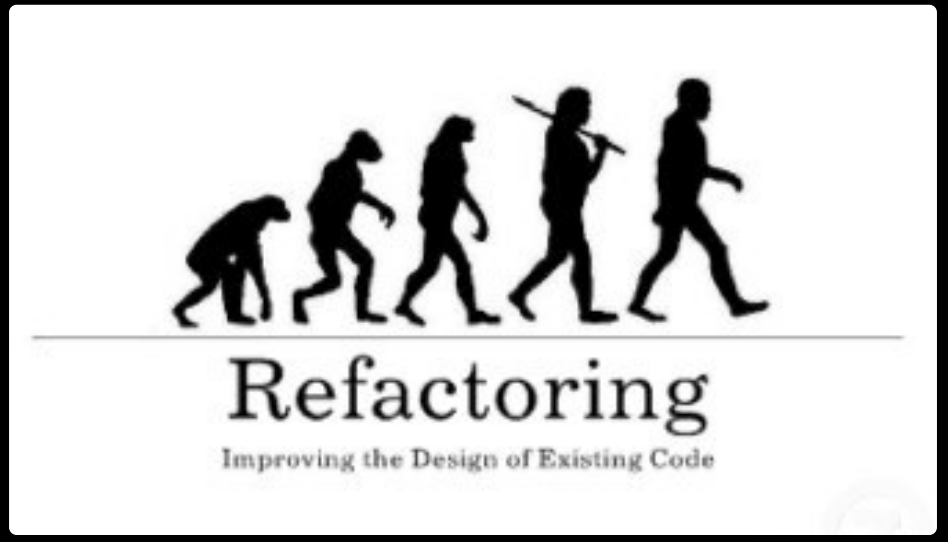
Качество на софтуера и преработка на кода

Учителски екип

Обучение за ИТ кариера

https://it-kariera.mon.bg/e-learning/



Съдържание

- Защо качеството на кода е важно?
- Външно и вътрешно качество на кода
- Що е това качествен код?
- Конвенции в кода
- Какво е преработка на кода?
- Принципи, които да следваме
- Процес на преработка и съвети

Защо качеството е важно?

- Какво прави този код? Коректен ли е?

```
static void Main()
{
  int value=010, i=5, w;
  switch(value){case 10:w=5;Console.WriteLine(w);break;case
9:i=0;break;
          case 8:Console.WriteLine("8 ");break;
     default:Console.WriteLine("def ");{
                Console.WriteLine("hoho ");      }
     for (int k = 0; k < i; k++, Console.WriteLine(k -
'f'));break;} { Console.WriteLine("loop!"); }
}
```

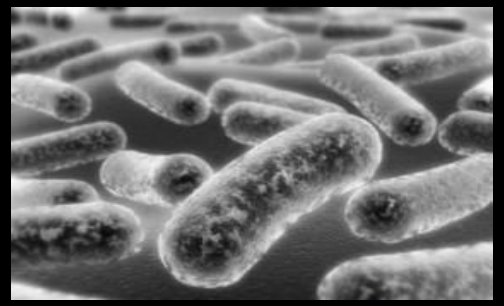
Защо качеството е важно? (2)

- Сега програмният код е форматиран, но все още е неясен

```
static void Main()
{
  int value = 010, i = 5, w;
  switch (value)
  {
    case 10: w = 5; Console.WriteLine(w); break;
    case 9: i = 0; break;
    case 8: Console.WriteLine("8 "); break;
    default:
      Console.WriteLine("def ");
      Console.WriteLine("hoho ");
      for (int k = 0; k < i; k++,
       Console.WriteLine(k - 'f')) ;
      break;
    }
    Console.WriteLine("loop!");
}
```

Качество на софтуера

- Външно качество
  - Дали програмата се държи коректно?
  - Дали връща очаквания резултат?
  - Дали софтуерът е бърз?
  - Дали неговия интерфейс е лесен за ползване?
  - Дали програмният код е достатъчно сигурен?
- Вътрешно качество
  - Дали кодът е лесен за четене и разбиране?
  - Дали кодът е добре структуриран?
  - Дали кодът е лесен за промяна?

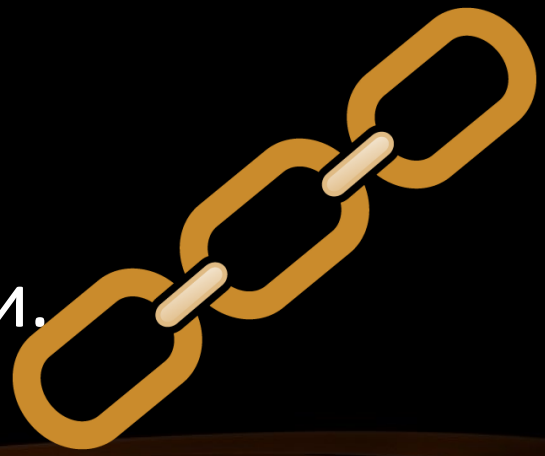


Какво е висококачествен програмен код?

- Висококачественият програмен код е:
  - Лесен за четене и разбиране
    - Лесен за промяна и поддръжка
  - Има винаги коректно поведение
    - Добре тестван е
  - Добре е проектиран и изграден
  - Добре е документиран
    - Самоописващ се (Self-documenting) код
  - Добре е форматиран

Какво е висококачествен програмен код? (2)

- Висококачественият програмен код има:
  - Силна специализация (strong cohesion) на всички нива: модули, класове, методи и т.н.
    - Един елемент отговаря за една-единствена задача
  - Слаба зависимост (loose coupling) между модули, класове и т.н.
    - Елементите са независими един от друг
  - Добро форматиране
  - Подходящи имена за класове, методи, величини.
  - Самоописващ се стил на кодиране



Конвенции в кода

- Конвенциите в кода са формализирани напътствия за стила на писане на програмен код:
  - Конвенции за форматиране на кода
  - Конвенции за именуване
  - Най-добри практики

- Официални препоръки на Microsoft за код на C#
  - Design Guidelines for Developing Class Libraries: http://msdn.microsoft.com/en-us/library/ms229042.aspx

Как да се справим със сложността в кода

- Справянето със сложността в кода има важна роля в конструирането на софтуера
  - Минимизиране на степента на сложност, с която нечий ум ще трябва да се справя в даден момент
- Предизвикателства пред архитектурата и дизайна
  - Проектиране на модули и класове, така че да се намали сложността
- Предизвикателства при изграждането на кода
  - Прилагане на добри практики при конструирането на софтуера: класове, методи, променливи, именуване, изрази, обработка на грешки, форматиране, коментари, компонентни тестове и т.н.

Как да се справим със сложността в кода (2)

- Ключово важно за това, да си ефективен програмист, е:
  - Максимизиране на частта от програмата, която можеш, без притеснение, да игнорираш
    - Докато работиш по коя да е част от кода
- Повечето практики, обсъждани тук, предлагат различни начини за постигане на тази важна цел

Основни характеристики на висококачествения код

- Коректно поведение
  - Спазващо изискванията
  - Стабилна работа, без увисвания и крашове
  - Без бъгове – работи според очакванията
  - Правилен отговор при неправилна употреба
- Четлив – лесен за разчитане
- Разбираем – себеописателен
- Ремонтопригоден – лесен за промяна, когато се наложи

Какво е преработка (Refactoring)?

Преработка означава „да се подобри организацията и качеството на програмен код без да се променя външното му поведение".

Martin Fowler

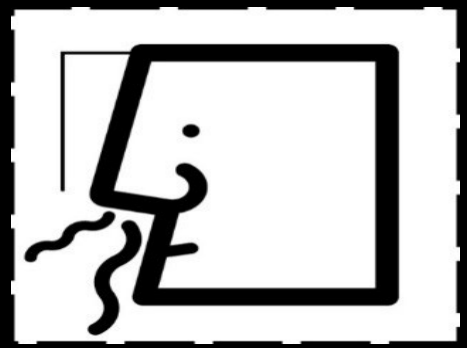
- Постепенен процес, превръщащ лошо написания код в качествен
  - Базира се на „шаблони за преработка" → добре известни рецепти за подобряване на кода

Преработка на кода

- Какво ще рече преработка на програмния код?
  - Подобряване на организацията и качеството на наличния програмен код, без да се променя неговото поведение
  - Това е постепенен процес, който превръща (ако е възможно)
  - лошо написания код в добре написан
- Защо бихме поискали преработка на кода?
  - Програмният код непрекъснато се променя и от това качеството му се влошава (ако не се преработи)
  - Изискванията често се променят и кодът трябва да се промени, за да отговори на новите изисквания

Кога е нужна преработка?

- Лоши практики в кода (code smells) показват, че трябва преработка
- Преработваме:
  - За да е по-лесно добавянето на нови функции
  - По време на оправянето на грешки
  - Когато преглеждаме чужд програмен код
  - Има технологични липси (или проблемен код)
  - По време на разработка чрез тестове (test-driven development)



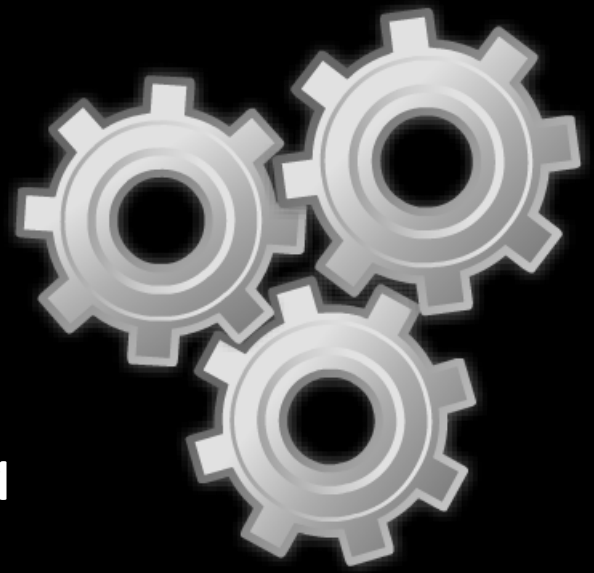
- Компонентните тестове (unit tests) гарантират, че преработката няма да промени поведението на кода
  - Ако нямате компонентни тестове, напишете!

Преработка: Основни принципи

- Нека да е по-просто (KISS принцип)
- Избягвайте повторения (DRY принцип)
- Нека да е ясно (говорящи имена, коментари и т.н.)
- По-малко количество код (KISS принцип)
- Разделяне на отговорностите (decoupling)
- Подходящо ниво на абстракция (използвайте абстракции)
- Правило на момчетата-скаути
  - Оставете кода в по-добро състояние от това, в което сте го заварили

Преработка: Типичен процес

1. Запишете програмния код в състоянието, от което започвате
   - Направете Check-in или архивирайте текущия програмен код
2. Подгответе тестове, за да се подсигурите, че кодът ще работи по същия начин след преработката
   - Компонентни / характеризиращи тестове
3. Правете преработките една по една
   - Правете малки преработки
   - Не подценявайте дребните промени
4. Изпълнете тестовете – те трябва да са успешни
   - Ако не са – върнете стария код
5. Направете check-in в системата за контрол на версиите

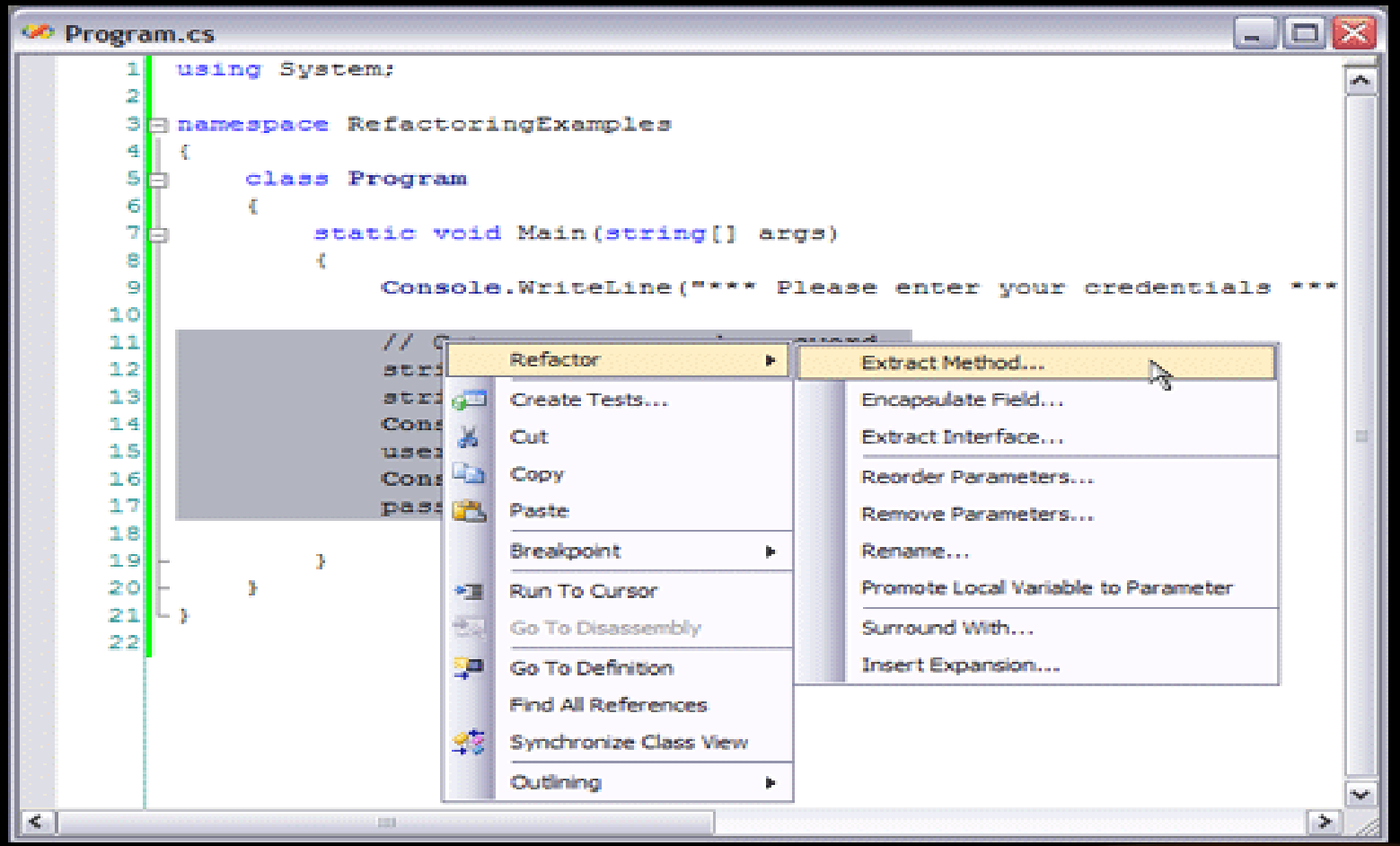


Преработка: Съвети

- Правете малки преработки
- Една по една
- Направете си списък
- Поддържайте „ПОСЛЕ“ TODO списък
- Правете check-in / commit често
- Добавете тестове за различните случаи
- Преглеждайте резултата
  - Програмирайте по двойки (pair programming)
- Използвайте инструментите (Visual Studio + добавки или други)

Преработка на кода във Visual Studio

- Преименуване на променлива / клас / метод / член
- Извличане на
  - метод
  - константа
  - интерфейс
- Капсулиране на поле



Обобщение

- **Качество на софтуера**
  - Външно – работи гладко, без бъгове и проблеми
  - Вътрешно – добре структуриран и разбираем код
- Аспекти на качеството на кода
  - Качествени класове, методи, условни изрази и цикли
  - Добро форматиране, коментари, силно специализиран, слабо зависим
  - Подлежи на компонентни тестове
- Преработка на кода – подобрява съществуващ код, без да променя поведението му

Качество на софтуера и преработка на кода

Въпроси?

https://it-kariera.mon.bg/e-learning/

Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**