



# ЧИСТЫЙ КОД



# Зачем нужен чистый код?

# Несколько фактов о разработке



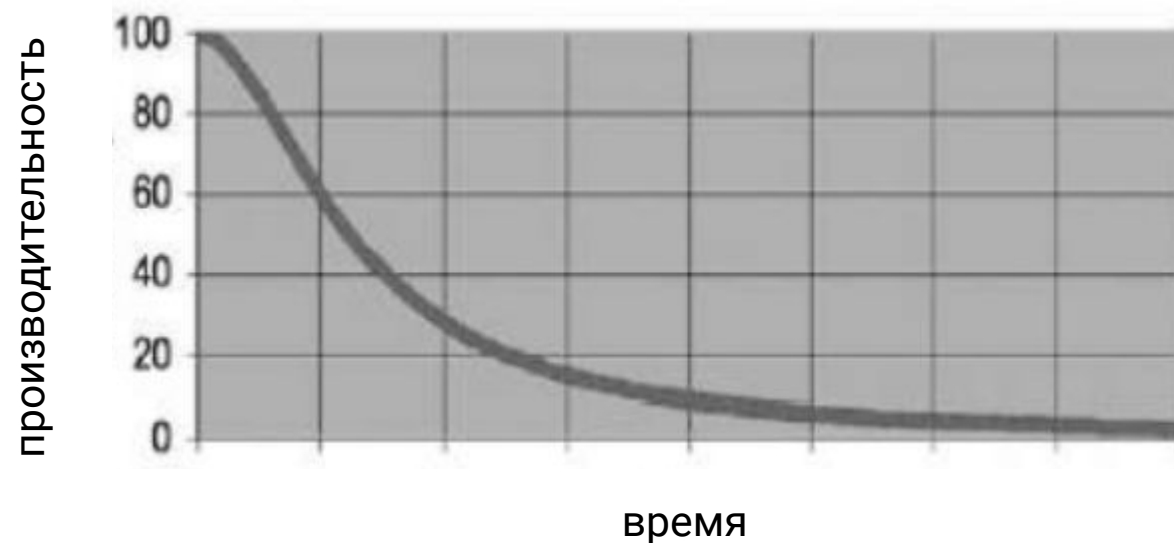
Настоящая стоимость кода – это его поддержка



# Несколько фактов о разработке



- Над чтением кода мы проводим в 10 раз больше времени, чем над его написанием
- Чем больше загрязняется наш код, тем больше мы теряем в производительности





# Что такое ЧИСТЫЙ КОД?



# *Чистый код*

Нет конкретного определения «Чистого кода». Не во всех средах программирования есть всеми признанный («единственно верный») кодекс аккуратности, иногда его просто нет или существует несколько конкурирующих.



# *Признаки плохого кода*

- Дублирование кода;
- Длинный метод;
- Большой класс;
- Длинный список параметров;
- Избыточные временные переменные;
- Классы данных.



# Как написать красивый и чистый код?





# ***Используйте понятные идентификаторы***

```
team.Select( p => p.Age > 18);
```

VS

```
teams.Select(team => team.Age > 18);
```



```
select bi.BusinessIdentifir, bia.SSAFA, IsNull(bia.Bullshit, 'Bullshit'), bis1.*, bis2.*, bis.*
from businessItems bi
    inner join businessItemsArtifacts bia on ...
    inner join businessItemsSegment bis1 on ...
    inner join businessItemsSegment bis2 on ...
    inner join businessItemsSegment bis3 on ...
where
bia.Date = @creationDate
and bi.Staus = 'RFW'
AND
(
    (bis1.SignIn = 'Europe' and ss2.Limit = 42 and bis3.Connection not in ('Towel', 'Galaxy'))
OR
    (bis1.SignIn = 'USA' and ss3.Limit = 146 and bis2.Connection not in ('Trump', 'Klinton'))
    OR
(bis1.PNH = 'SP' and ss2.Limit = 21 and bis3.Connection not in ('Stan', 'Kyle', 'Cartman'))
)
```

# Пишите краткие и понятные КОММЕНТАРИИ



```
public void msg(string s)
{
    Console.WriteLine("Вывод из объекта tst1: " + s);
}
```

VS

```
/// <summary>
/// Выводит строку текста
/// </summary>
/// <param name="s">Строка текста для вывода</param>
public void msg(string s)
{
    Console.WriteLine("Вывод из объекта tst1: " + s);
}
```

# ***Используйте комментарии без фанатизма***



- Написание комментариев для себя (пример: /\* Закончу как-нибудь потом... \*/).
- Ни о чём не говорящие выражения (e.g. /\* Это очередная математическая функция. \*/).
- Также иногда люди не уверены в какой-то функциональности и просто комментируют фрагмент кода.

# Используйте стандарты именования переменных и функций



## Is vs GetIs





# ***Не забывайте о признаках плохого кода***

- Не делайте дублирование кода, лучше вынесите в отдельный компонент;
- Не делайте длинные методы, лучше разделите на несколько мелких;
- Не делайте большие классы, лучше разделите на несколько мелких;
- Не делайте длинный список параметров, лучше разбейте на несколько групп;
- Не делайте избыточные временные переменные, лучше создайте несколько переменных под каждую задачу;
- Не делайте классы данных, лучше вынесите данные в отдельную структуру.



# Рефакторинг



# Когда рефакторить?

- Правило трёх:
  - 1) Делая что-то в первый раз, вы просто это делаете.
  - 2) Делая что-то аналогичное во второй раз, вы морщитесь от необходимости повторения, но все-таки повторяете то же самое.
  - 3) Делая что-то похожее в третий раз, вы начинаете рефакторинг.
    - Когда делаете новую фичу
    - Когда исправляете баги
    - Во время код-ревью





# *Как рефакторить?*

- Код должен стать чище.
- В процессе рефакторинга не создаётся новая функциональность.
- Все существующие тесты должны успешно проходить.



# *УТИЛИТЫ*





# УТИЛИТЫ

```
var user = Membership.GetUser(model.UserName);
if (null == user) {
    ModelState.AddModelError("", "The user name or password is incorrect.");
}
```

Lightbulb icon: Move to resource

Lightbulb icon: Convert to verbatim string

Lightbulb icon: Convert all values of parameter 'errorMessage' to verbatim string

Wrench icon: Insert format argument

Wrench icon: Split string

Wrench icon: To verbatim string

Wrench icon: Make regular expression here

Wrench icon: Add argument name 'errorMessage'

Wrench icon: Inspection "Element is localizable"

```
MethodInfoForKey(CodeCleanupOptionDescriptor key, string name)
Type.
entTy
anupP
(Code
{
    return GetMethodInfoForKey(
}
public void SetSetting(CodeCl
{
    GetMethodInfoForKey(key, "S
}
public T GetSetting<T>(CodeCleanupOptionDescriptor<T> key) where T : new()
```

Refactor This

- Change Signature Ctrl+R, S
- Inline Method Ctrl+R, I
- Move Instance Method Ctrl+R, O
- Rename Ctrl+R, R
- Safe Delete Alt+Del
- Extract Interface
- Extract Superclass
- Extract Class
- Push Members Down
- Convert Method To Indexer
- Make Static
- Extract Class From Parameters



Подготовил  
Владимир Нарожный

Харьков, 2020