УДК 004.415.3

МЕТОДИКИ ЧИСТОГО КОДУ ТА ЇХ КОРИСТЬ ДЛЯ ПРОГРАМІСТІВ

Лисенко Д.О.
Науковий керівник – к.т.н., доц. Руденко Д.О.
email: danylo.lysenko@nure.ua
Харківський національний університет радіоелектроніки, каф. ІНФ
м. Харків, Україна

Clean code techniques help you write code that is easy to understand and maintain. Their implementation allows you to extend the code without any problems, reduce the number of errors, and increase the reliability of the product. Developers' understanding of these principles helps to unify their work so that it is as clear as possible for other programmers. We consider three clean code techniques that provide significant advantages when creating a software product.

Методики чистого коду сприяють написанню простого для розуміння та легко підтримуваного коду. Їх впровадження дозволяє розширювати код без особливих проблем, зменшувати кількість помилок і підвищувати надійність продукту. Розуміння розробниками цих принципів допомагає уніфікувати їх роботу так, щоб вона була максимально зрозуміла для інших програмістів. Розглядається три методики чистого коду, які дають значні переваги при створенні програмного продукту.

При розробці будь-якого рішення програміст стикається з необхідністю найменування змінних. Цей легкий на перший погляд процес може стати головною проблемою в розумінні коду в майбутньому. Незрозумілі імена, які не відображають суть змінної, змушують програміста шукати по різних частинах коду, щоб зрозуміти сенс цієї змінної. Саме це пропонує змінити автор книги «Чистий код» Роберт Мартін [1].

Потрібно використовувати імена, котрі точно описують сенс. Ця назва повинна відповідати на найпоширеніші запитання, які можуть виникнути при першому погляді на змінну. За допомогою імені треба чітко визначити, для чого ця змінна, що вона робить і як вона використовується. Зауважимо, якщо для пояснення змінної вам треба написати коментар до неї, тоді ця назва не відповідає концепту чистого коду [2].

Під час формування назв необхідно уникати можливих дезінформаційних неточних найменувань. Слова, які на перший погляд можуть бути доречними у контексті конкретного рішення, можуть також мати інакший сенс в іншій темі. Це може призводити до незрозумілості змінних іншими програмістами, що зменшить його підтримуваність [3].

Кожен код має у собі функції, які можуть використовуватися майже всюди. Вони рухають код, тому написання зрозумілих і неперевантажених функцій – ключ до успіху. Перевантаженням може бути дублювання інформації, якого можна уникнути, витративши на створення функції трохи більше часу.

Правильне написання функцій підвищує читабельність коду. Кожен програміст хотів би розуміти сенс функції, лише подивившись один раз на неї, тому слід писати їх якомога меншими.

Роберт Мартін у книзі «Чистий код» ділиться своїм досвідом про написання функцій, в якому підтверджує тезу, наведену вище [1]. Він зазначає, що функції повинні робити одну справу, задля якої їх створюють. Не треба намагатися включити в одну функцію великий перелік дій. Вона повинна добре виконувати все, що від неї очікують.

Такий підхід до створення функцій значно спрощує їхнє розуміння та вигляд. Це економить час розробників при першому ознайомленні з кодом, завдяки чому вони можуть швидко почати працювати з ним [3].

При написанні коду важливу роль відіграє форматування. Неохайний, неструктурований код може створити негативне враження у читача, ніби його писала некомпетентна людина. Це може відвернути увагу від важливих деталей і, зрештою, вплинути на розуміння коду. Потрібно визначити для себе чіткий набір правил, яких потрібно дотримуватися під час всього написання коду.

Для кращого розуміння можна навести приклад того, як побудована інформація у газетах. Зверху є заголовок, який поверхово повідомляє про те, що ми будемо читати. Проте він також надає достатньо інформації, щоб можна було зрозуміти, що ми читаємо саме те, що нас цікавить. У тексті статті ми отримуємо все більше фактів і деталей, які дають уже повне розуміння. Це знову ж доводить ідею про те, що правильно написане ім’я, це вже половина справи.

Такий самий метод треба використовувати і в написанні коду. Це допомагає програмістам швидше отримувати інформацію та не втрачати уваги, намагаючись прочитати код [3].

Підсумовуючи все вище сказане, можна зробити висновок, що імплементація цих методів допомагає швидкому розумінню коду іншими розробниками та надає можливість для довготривалої підтримки й оновлення проєкту. Це дозволяє споживачам, які навіть не знайомі з особливостями продукту, швидко розібратися й почати працювати з ним.

Список використаних джерел:

1. Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2009. 464 p.

2. Latte B., Henning S., & Wojcieszak M. Clean Code: On the Use of Practices and Tools to Produce Maintainable Code for Long-Living Software. *Software Engineering*. 2019. 96-99.

3. Koller H.G. Effects of Clean Code on Understandability: An Experiment and Analysis. 2016.