Конарєва Вікторія Валентинівна

9 варіант

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Вимоги до роботи | Бали | Виконання |
| 1 | Реалізація компоненту та основної програми окремими проектами в одному рішенні (Solution) | 1 | Виконано. Є два проекти: бібліотека BinaryConverterLibrary та тестовий додаток. |
| 2 | Реалізація власних властивостей та подій компонента, що відображаються у вікні Properties | 1 | Виконано. Властивості DecimalValue, BinaryValue та подія Converted. |
| 3 | Реалізація функціональності компонента згідно завдання | 1 | Виконано. Компонент перетворює десяткове число у двійкове і виводить результат. |
| 4 | Пояснення у звіті щодо використання кожного з принципів SOLID | 2 | Виконано. Див. нижче пояснення. |
| 5 | Прив’язка піктограми до компонента для відображення на панелі інструментів | 1 | Виконано. Використано атрибут [ToolboxBitmap(typeof(BinaryConverterControl), "BinaryConverterControl.ico")]. |
| 6 | Прив’язка меж та пропорційна зміна внутрішніх елементів щодо розміру компонента | 1 | Виконано. Встановлено властивості Anchor для TextBox та Button. |
| 7 | Створення пакету NuGet з компонентом та тестування в окремому рішенні | 1/2 | Виконано частково. Описано процес створення пакету та встановлення в іншому рішенні (локальне джерело NuGet). |
| 8 | Своєчасне виконання базових вимог та звітності | 1 | Виконано. Завдання виконане в межах встановлених термінів. |

**Використання принципів SOLID**

1. **SRP (Single Responsibility Principle)**
   * Компонент **BinaryConverterControl** виконує єдину задачу: перетворення десяткового числа у двійковий формат та відображення результату. Уся логіка зосереджена в одному класі, який не займається сторонніми завданнями.
2. **OCP (Open/Closed Principle)**
   * Код компонента відкритий для розширення (наприклад, можна додати інші типи конвертацій або додаткові властивості), але закритий для модифікацій базової логіки. Щоб розширити функціональність, можна успадкувати цей клас або додати нові методи, не змінюючи основну реалізацію.
3. **LSP (Liskov Substitution Principle)**
   * Оскільки **BinaryConverterControl** наслідує UserControl, його можна замінити будь-яким іншим контролом, який успадковується від UserControl, не порушуючи роботи програми. Він поводиться так само, як і стандартні контролери WinForms.
4. **ISP (Interface Segregation Principle)**
   * Якщо в майбутньому з’явиться потреба підтримувати різні формати конвертації (наприклад, шістнадцятковий), можна виділити окремий інтерфейс IConverter і впроваджувати тільки потрібні методи. Зараз же компонента не змушує реалізовувати «зайвий» функціонал, оскільки має чітку вузьку відповідальність.
5. **DIP (Dependency Inversion Principle)**
   * Компонент не залежить від конкретної реалізації сторонніх сервісів: усі обчислення виконує в собі. Якщо знадобиться зовнішня логіка (наприклад, підключення іншого способу конвертації), можна буде підключити її через абстракцію (інтерфейс), і компонент працюватиме з цим інтерфейсом, а не з конкретним класом.