|  |
| --- |
| EPAM Systems, RD Dep. |
| Контрольный тест  NET.C# Контрольные вопросы по Рефакторингу |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial version | Oleg | 11.07.2014 |  |  |

ФИО Студента\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата опроса\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Пояснения:**

1. Студент заполняет колонки:
2. «Верно?». Если верно, то укажите «Да», если нет – «нет».
3. «Объяснения и комментарии»:
   * ответы на вопросы «Почему не правильно? Почему правильно?»;
   * при необходимости пошаговое описание с указанием состояния переменных на каждом шаге.
   * нужно изложить объяснение, которое покажет понимание теории и умение применять знания на практике
   * в некоторых вопросах и вариантах ответах нужно исследовать практическую задачу, разобраться и написать объяснение.
   * в некоторых вариантах желательно, указать как исправить код, чтобы он успешно компилировался или выполнялся без ошибок.
4. Используя стиль «EDU\_Student».
5. Преподаватель заполняет колонки «Верный ответ», «Правильное объяснение», «Зачтено».
6. А также задает вопросы и даёт свои комментарии в секции «Объяснения и комментарии».
7. Используя стиль «EDU\_Reviewer».
8. При выполнении можно и нужно пользоваться:
   1. Лекцией;
   2. MSDN;
   3. Учебниками вроде Шилда и т.п.
9. **Что такое Рефакторинг? В чем его суть?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Рефакторинг – процесс улучшения кода без написания новой функциональности. Суть рефакторинга – упростить дизайн и сделать код легким для понимания. |  |  |  |

Сложность: [1]

1. **Почему появилась такая «штука» как Рефакторинг?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | В процессе разработки может появится технический долг, который замедляет разработку, т.к. за него приходится платить. Рефакторинг появился, чтобы упростить жизнь другим людям, читающим код, а также, чтобы не срывать сроки разработки. |  |  |  |

Сложность: [1]

1. **Зачем делают Рефакторинг? Какие реальные задачи решает Рефакторинг?**
   1. Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Если необходимо добавить новую функциональность в код, но код недостаточно ясный, делают рефакторинг, чтобы добавление функциональности произошло более гладко и заняло меньше времени. Если логика программы слишком сложна, то делают рефакторинг, чтобы сделать код понятнее. Рефакторинг проводится, когда исправляются существующие ошибки в коде.  Рефакторинг нужен для того, чтобы сделать код проще для чтения и понимания другими программистами команды. |  |  |  |

Сложность: [1]

1. **Чем полезно знанием и умение выполнять Рефакторинг?**

**Рефакторинг – это догма для неукоснительного выполнения?**

* 1. Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Знание и умение выполнять рефакторинг полезно тем, что в последствии нужно будет потратить на поддержание кода меньше накладных расходов. Во время разработки код будет легче читать, отлаживать и модернизировать.  Рефакторинг – не догма, но необходим для упрощения работы. |  |  |  |

Сложность: [2]

1. **Когда можно делать Рефакторинг?**
   1. Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Рефакторинг можно делать в любом случае, когда есть явное понимание, что код плохо пахнет, но при этом нет никаких ошибок. Также, рефакторинг можно проводить, когда самому разработчику становится трудно добавить в код что-то новое и т.д. |  |  |  |

Сложность: [3]

1. **Когда нужно делать Рефакторинг?**
   1. Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Рефакторинг нужно проводить при добавлении новой функциональности в код, чтобы это добавление прошло легко и не заняло слишком много времени. Рефакторинг нужно проводить при наличии ошибок и багов в коде. Также, рефакторинг нужно проводить во время код-ревью, т.к. это может быть последним шансом почистить код перед тем, как окажется доступным публично. |  |  |  |

Сложность: [2]

1. **Рефакторинг и Ошибки:**
2. **После Рефакторинга изменяется работа «приложения»?**
3. **После Рефакторинга появляются Ошибки в приложении?**
4. **Кто должен проверять работу приложения после рефакторинга кода на наличие новых/проверку старых ошибок?**
5. **Как они (проверяющие) узнают какие части программы нужно проверять?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | После рефакторинга работа приложения не изменяется.  После рефакторинга ошибки в приложении не появляются.  Работу приложения после рефакторинга проверяют тесты. Все существующие тесты должны успешно проходить. Если тесты были слишком низкоуровневыми, их необходимо переписать, сделав более высокоуровневыми.  Т.к. тесты были написаны до выполнения рефакторинга, то известно, какие части программы они должны проверять. |  |  |  |

Сложность: [3]

1. **Простота и сложность Рефакторинга:**
2. **Какие сложности в освоении Рефакторинга?**
3. **Как научиться и привыкнуть его применять?**
4. **Почему легко выполнять Рефакторинги?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | a) Множество приемов рефакторинга, иногда неочевидных для начинающих. Так же упрямство и лень программистов, т.к. код писать легче, чем читать.  b) Научиться и привыкнуть можно только постоянной практикой.  c) Рефакторинг легко выполнять, т.к. применяются одни и те же методы для его проведения и с опытом будет сразу понятно, что требует рефакторинга. |  |  |  |

Сложность: [3]

1. **Бывает идеальный код? Какие разумные рамки (ограничители) применения Рефакторинга?**
   1. Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Идеального кода не бывает. Рефакторинг нужен для упрощения чтения и понимания кода, поэтому нет необходимости переписывать весь код приложения. |  |  |  |

Сложность: [2]

1. **Code Smell:**
   1. **Что такое Code Smell?**
   2. **Зачем они нужны?**
   3. **Чем отличаются от Багов?**
   4. **Что делать с Code Smells?**
   5. **Зачем нужен список наиболее распространённых Code Smell?**

* Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Code smell – использование принципов, которые загрязняют код, делая его сложным для понимая.  В отличие от багов не представляют собой непредвиденного поведения программы, но при изменении чего-то, может привести к ряду новых дефектов.  Code smell необходимо исправлять в соответствии с принятыми принципами.  Список наиболее распространенных Code Smell нужен, чтобы автоматически было понятно, как исправлять данную часть кода. |  |  |  |

Сложность: [2]

1. **Что такое Code Debt? Что с ними можно делать? Что следует делать, чтобы они не накапливались?**
   1. Нужно описать своими словами как сам понимаешь, а не копи-паст.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Code debt – та работа, которая должны быть выполнена, но по разным причинам ее выполнение отложено (осознанно или нет).  Любой технический долг, как и реальный, рано или поздно нужно возвращать. Для возврата технического долга используется рефакторинг.  Чтобы долг не накапливался необходимо бороться с жесткой связанностью компонентов, писать тесты, документировать код. Также, отсутствие рефакторинга и контроля за соблюдением стандартов порождают технический долг. |  |  |  |

Сложность: [2]

1. **А может взять и «всё переписать заново»? Какие плюсы и минусы у этого подхода?** 
   1. Нужно описать своими словами как сам понимаешь, а не копи-паст.
   2. Нужно указать минимум по пять плюсов и по пять минусов.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Верно? | Объяснение и комментарии | Верный ответ | Правильное объяснение | Зачтено |
|  | Минусы:  - увеличение сроков разработки проекта;  - увеличение накладных расходов;  - в большинстве случаев результат будет тем же;  - если не известны требования, то качество приложения будет определяться самым слабым звеном, и вряд ли будет высоким;  - часто проще исправить часть кода, чем переписывать все заново.  Плюсы:  - можно учесть прошлые ошибки;  - возможность заново спроектировать систему:  - можно потратить больше времени на исправление ошибок в существующем проекте, чем на переписывание заново;  - возможность взглянуть по-новому на решение задачи; |  |  |  |

Сложность: [2]