**Оценочные материалы текущего контроля знаний и промежуточной аттестации по МДК.03.01 Моделирование и анализ программного обеспечения**

**Компетенция ПК 3.1**

|  |
| --- |
| 1 **Какие этапы входят в систему обеспечения качества программного продукта?**   1. **тестирование** 2. **ревьюирование проектной документации** 3. **обзоры кода** 4. кодирование |
| 2 **Этапы code review**   * + 1. **design review**     2. **Pre-commit review**     3. **Post-commit review** |
| 3 **Реорганизация кода**  —   * + - 1. **процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы**       2. систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки.       3. Программное обеспечение для автоматизированной инспекции кода |
| 4 **Систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки** –   * 1. **инспекция кода**   2. design review   3. Pre-commit review |
| 5 **Причины применения рефакторинга**   * 1. **необходимо добавить новую функцию, которая недостаточно укладывается в принятое архитектурное решение;**   2. **необходимо исправить ошибку, причины возникновения которой сразу не ясны;**   3. **преодоление трудностей в командной разработке, которые обусловлены сложной логикой программы.**   4. **Все варианты верны** |
| 6 **Способ организации хранения и контроля версий файлов в рабочем проекте** –   * 1. коммит   2. **Git**   3. бранч |
| 7 **Репозиторий** –   * 1. **облачное хранилище вашего проекта на сервере**   2. это ветка или копия проекта, в которую можно вносить любые изменения и они не повлияют на основной проект |
| 8 **CASE-средства высокой степени интегрированности называются**   1. **средствами с общий репозиторием** 2. локальными средствами 3. программно-аппаратными платформами |
| 9 **CASE-средства поддерживают**   1. **генерацию кода и тестирование** 2. **анализ и спецификацию требований** 3. производство вычислений |
| 10 **Основные функции CASE-средств включают**   1. **репозиторий** 2. удаленный вызов процедур 3. **оптимизацию кода продукта** |
| 11 **CASE-средства моделируют**   1. **структуру информационных систем** 2. **интерфейсы информационных систем** 3. концепцию проектирования |
| 12 **CASE-средства информационного моделирования основаны на диаграммах**   1. **сущностей и связей** 2. потоков данных 3. структурного анализа |
| **13 Репозиторий CASE-средства – это …**   1. **специализированная база данных, предназначенная для отображения состояния проектируемой информационной системы в каждый момент времени** 2. резервная база данных, предназначенная для отображения состояния проектируемой информационной системы 3. специализированный словарь терминов, применяющихся в предметной области разрабатываемой информационной системы |
| **14 Зачем нужны системы контроля версий?**   1. **Сохранение промежуточных версий проекта (истории изменений)** 2. **Обеспечение возможности командной работы над проектом** 3. Отправка своего кода на сервер |
| **15 Что такое индекс в системе контроля версий Git?**   * 1. **Специальная область, в которой хранятся изменения файлов, готовые к коммиту**   2. Специальная область, в которой хранятся отложенные изменения, которые разработчик не собирается коммитить   3. Специальная область, в которой хранятся закоммиченные файлы |
| 16 **Что такое коммит**?   * + - 1. **Это единица состояния проекта в Git**       2. Это результат вывода команды git diff       3. Это обобщающее название одного из статусов файла в выводе git status: untracked, new, deleted или modified |
| 17 Какие свойства есть у коммита?   1. **Хеш коммита** 2. **Дата и время коммита** 3. **Автор коммита** 4. **Комментарий коммита** 5. **Все перечисленные свойства** |
| 17 Что такое Git Hub?   1. Программа для работы с Git 2. **Веб-сервис для хостинга IT-проектов и их совместной разработки, основанный на Git** 3. Драйвер для Git |
| 19 Что такое репозиторий Git?   * + - * 1. Любая директория/папка в моей ОС         2. Любая папка, находящаяся внутри Git         3. **Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов, и хранилище, содержащее собственно файлы** |
| 20 Что такое ветка в репозитории Git?   * + - 1. Это то же самое, что и коммит       2. Это минимум два коммита с одинаковым коммит-сообщением       3. **Это разные пути развития проекта, по сути разные последовательности коммитов** |
| **21 Сколько всего веток может быть в репозитории?**   * + 1. **Сколько угодно**     2. Это число настраивается в конфиге     3. Не больше двух |
| **22 Что такое слияние двух веток?**   * + - 1. Когда одну ветку переименовывают в другую       2. Когда выполнили команду git fetch       3. **Когда все коммиты, сделанные для одной ветки, становятся видимыми во второй ветке** |
| 23 Чем директория с репозиторием отличается от любой другой?  Ничем, такая же директория  Правами доступа - у директории репозитория права доступа только того пользователя, который его "склонил" (git clone)  **наличием папки .git с настройками репозитория** |
| 24 Почему бывают конфликты при слиянии веток?   1. Потому что ветки были созданы в разное время 2. Потому что ветки были созданы от разных коммитов 3. **Потому что в обеих ветках есть изменения одних и тех же строк** |
| 25 Как добавить новую директорию в Git?  Добавить каждый файл из этой директории в Git  **Добавить хотя бы один файл из этой директории в Git**  Команда: git add -d |

**Компетенция ПК 3.3**

|  |
| --- |
| 1 **Какая команда создает репозиторий git для проекта**?  git new  **git init**  git start |
| 2 **Сканирование кода на ошибки в репозитории GitHub запускается со вкладки:**   1. **Security** 2. Discussions 3. Issues |
| 3 **Какая команда настраивает сканирование кода в репозитории GitHub:**   1. **Set up code scanning** 2. Start commit 3. Git init |
| 4 **Как можно отменить коммит в Git, если он уже был опубликован** ?   * 1. **с помощью команды git revert**   2. с помощью команды git fetch   3. с помощью команды git pull |
| 5 **Что делает команда git pull?**   1. **извлекает данные с сервера и автоматически делает слияние их с кодом текущей ветки** 2. связывается с удаленным репозиторием и получает данные, которые отсутствуют в локальном 3. создает коммит, отменяющий изменения указанного коммита |
| 6 **Что делает команда git fetch ?**   1. **связывается с удаленным репозиторием и получает данные, которые отсутствуют в локальном** 2. отправляет изменения в удаленный репозиторий 3. полностью удалит указанный коммит из истории |
| 7 **Отправить изменения в удаленный репозиторий можно с помощью команды:**   1. **git push** 2. git revert 3. git pull |
| 8 **Файл, содержащий информацию об изменениях, которые попадут в следующий коммит:**   1. drop 2. **staging area** 3. force |
| **9 Укажите правильные варианты анализа кода с помощью команды git bisect:**   1. **git bisect start – начало бинарного поиска по истории коммитов** 2. **git bisect good - указывает Git коммит в котором проблема не проявляется** 3. **git bisect bad — указывает Git коммит, в котором обнаружена проблема** 4. **все варианты верны.** |
| 10 **Как настроить Git-репозиторий для запуска инструментов проверки работоспособности кода непосредственно перед выполнением коммитов и предотвращения их в случае сбоя теста?**   * 1. **с помощью хука pre-commit**   2. с помощью флага stat   3. с помощью флага name-status |
| 11 **За что отвечает команда Git stash?**   1. отфильтровывает ветки, которые были слиты 2. **сохраняет измененное состояние рабочей директории или отдельного файла в хранилище незавершенных изменений** 3. объединяет несколько отдельных коммитов в один цельный коммит |
| 12 **Как найти список файлов, которые изменились в определенном коммите?**   1. с помощью команды git rebase 2. **с помощью команды git log** 3. с помощью команды git branch |
| **13 Какова минимальная длина SHA-1 хэша должна быть, чтобы можно было просмотреть информацию о коммите?**   * + - 1. 3 символа       2. **4 символа**       3. 5 символов |
| 14 **Перебазирование коммитов производиться командой**   1. **git rebase** 2. gitclone 3. gitrevert |
| 15 **Коммит в Git включает в себя:**   1. **объекты с измененными файлами** 2. **ссылка на предыдущий коммит** 3. **имя автора** 4. **метка времени** 5. **сообщения коммита** |
| 16 **За что отвечает команда “git config”?**  **позволяет просматривать и настраивать конфигурацию git репозитория**  объединяет несколько отдельных коммитов в один цельный коммит  выполняет поиск по истории коммитов |
| 17 Согласно модели ветвления *git flow* в репозитории должны быть две постоянные ветки:   * + 1. **master**     2. release     3. **develop**     4. feature     5. hotfix |
| **18 Какой командой можно загрузить с GitHub репозиторий на свой компьютер?**   * 1. git push   2. **git clone**   3. git fetch |
| 19 Какой текстовый редактор используется по умолчанию в git?   * + 1. Emacs     2. Notepad     3. **Установленный по умолчанию в систем**е |
| 20 Что делает команда git status?   * + - 1. **Показывает состояние проекта: кол-во untracked, deleted, new и прочих файлов, количество коммитов, на которое отличается локальная версия репозитория от удаленного и так далее**       2. Показывает имя и email нашего пользователя, а также является ли он авторизованным в системе GitHub или нет       3. Показывает место, занимаемое репозиторием на жестком диске и кол-во выделенного под репозиторий месте |
| 21 Что делает команда git stash?   * + - * 1. Отменяет все изменения         2. **Сохраняет все изменения в буфер**         3. Удаляет все измененные файлы |
| 22 При помощи какой команды можно посмотреть историю всех коммитов с сокращённым SHA-1 хэшем?  git log --short-hash  git log –short  **git log --abbrev-commit** |
| 23 **Как узнать, кто автор строчки в файле, используя систему Git?**   1. **Команда git blame** 2. Команда git status 3. Команда git show --author |
| 24 Что сделает команда "git branch" без какого либо параметра?  Переключится на последнюю используемую ветку  Выведет ошибку  **Выведет список локальных веток** |
| 25 В какой ситуации надо делать git status?   * + 1. Чем чаще, тем лучше     2. **Только если надо узнать, в каком статусе находится репозиторий, а так эта команда не является обязательной для любой манипуляции**     3. Всегда после команды git pull |
| 26 Для чего надо добавлять файлы в .gitignore?   * + - 1. Чтобы Git удалял их историю, храня только последнюю версию       2. **Чтобы Git не замечал их и любые команды Git не могли их заафектить**       3. Файл .gitignore не несет никакой смысловой нагрузки, так что этого не надо делать |
| 27 Как «спрятать» данные в git?   * + - * 1. git check –hide         2. **git stash**         3. git hide |
| 28 Как вывести список удалённых репозиториев с именем и url?  **git remote –v**  git show  git repository |
| 29 Чем отличается master и origin master?  Это просто два разных названия одной ветки  Это две разные ветки локального репозитория  **master принадлежит локальному репозиторию, a origin master - удаленному** |
| 30 Как исправить ошибку "fatal: The current branch mybranch has no upstream branch", возникающую при вводе git push?  Никак, придется создавать репозиторий заново  **Команда: git push -u my\_branch**  Ошибка означает, что fatal в коде, так что надо сначала исправить код проекта |
| 31 Как отменить действие команды "git add" на файл?  Команда git abort  Команда git stash  Команда git not-add  **Команда git reset** |
| 32 **Как отменить слияние веток, если произошел конфликт?**   * + 1. Команда git stash pop     2. **Команда git merge –abort**     3. Команда git remove repository |
| 33 Как перейти из ветки master в ветку dev?   * + - 1. **git checkout dev**       2. git change master dev       3. git branch master dev |
| 34 **Как получить список всех веток?**   * + - * 1. **git branch**         2. git branch –all         3. git tree –all |
| 35 **Как посмотреть id коммита?**  git info  git commit id  **git log** |
| 36 **Как посмотреть последний коммит у каждой ветки?**  git branch --last-commit  **git branch –v**  git checkout --last-commit |
| 37 Как привести измененный файл в начальное состояние (до изменения)?  Команда git abort path/to/file  **Команда git checkout path/to/file**  Команда git pull path/to/file  Команда git commit path/to/file |
| 38 Как применить патч в Git?  **Команда git apply path/to/file**  Команда git patch path/to/file  Команда git add path/to/file  Такого понятия все еще нет |
| 39 Как проиндексировать несколько файлов одной командой?   * + 1. git add TEXT1.txt, TEXT2.txt, TEXT3.txt     2. **git add TEXT1.txt TEXT2.txt TEXT3.txt**     3. git add TEXT1.txt ADD TEXT2.txt ADD TEXT3.txt |
| 40 Как проиндексировать файлы и сделать коммит одной командой?   * + - 1. git commit --add -m "Comment"       2. git commit -add -m "Comment"       3. **git commit -a -m "Comment**" |
| 41 Как просмотреть список меток?   * + - * 1. git show –labels         2. **git tag**         3. git labels |
| 42 Как решить конфликт в Git?   * + - * 1. **Руками поправить изменения там, где Git не смог это сделать автоматически и затем собрать все в коммит и запушить**         2. Никак, придется создавать репозиторий заново         3. Выполнить команду git commit merge please |
| 43 Как сделать ветку с названием my\_branch?  **Команда: git branch my\_branch**  Команда: git create branch my\_branch  Команда: git commit origin my\_branch |
| 44 Как сделать коммит для ветки my\_branch?  **Надо переключиться на нее и дальше сделать коммит по тем же правилам, что и для ветки master**  Надо в commit-сообщении прописать название ветки  Команда: git fetch origin my\_branch |
| 45 Как скачать ветку their\_branch, если она уже есть в удаленном (remote) репозитории, но нет локально?  Команда: git clone their\_branch  Команда: git get origin their\_branch  **Команда: git fetch origin their\_branch** |
| 46 Как создать новую ветку с именем dev?  git new dev  git create dev  **git branch dev** |
| 47 Как удалить ветку night?  git checkout --delete night  **git branch -d night**  git delete night |
| 48 Как удалить все untracked файлы?  **Команда: git clean –f**  Команда: git delete  Команда: git stash |
| 49 Как удалить локальную ветку my\_branch?   * + 1. Команда: git delete branch my\_branch     2. **Команда: git branch -D my\_branch**     3. Команда: git reset my\_branch |
| 50 Как узнать, какие изменения мы сделали локально относительно последнего состояния нашего репозитория?   * + - 1. Команда git show       2. **Команда git diff**       3. Команда git izmeneniya |

**Компетенция ПК 3.4**

|  |
| --- |
| **1 Качество ПС отражается тремя группами показателей, характеризующими:**  **внутреннее, внешнее, качество при использовании**  требуемое, обусловленное, реальное  номинальное, идеальное, реальное |
| **2 На чем основано определение ошибки?**  **на эталонном состоянии объекта**  на случайном обнаружении ошибки  на поисковой деятельности |
| **3 Какие факторы влияют на степень качества программного средства?**   * + - 1. **качество технологий проектирования**       2. **качество разработки ПС**       3. **качество сопровождения**       4. **качество документирования** |
| **4 Выделите особенности процесса тестирования программ по отношению к тестированию аппаратуры:**   1. отсутствие эталонной программы, которой должны точно соответствовать все результаты тестирования 2. принципиальная невозможность использования полных тестовых наборов для исчерпывающей проверки функционирования сложных ПС 3. относительно невысокая степень формализации критериев качества результатов тестирования и достигаемых при этом корректности и надежности функционирования испытуемых ПС 4. **все ответы верны** |
| **5 Совокупность организационных структур, методик, технологий и ресурсов, необходимых для осуществления общего руководства качеством – это:**   1. **система качества** 2. стандартизация 3. сертификация 4. метрология |
| **6 С точки зрения пользователя программного обеспечения качество последнего заключается в**   1. **легкости эксплуатации** 2. модификации 3. **Безотказности** 4. **Производительности** 5. воспроизводимости |
| **7 К качествам характеризующим информационные системы относят**   1. планирование времени выполнения запросов 2. **поддержку целостности данных** 3. **доступность данных** 4. **производительность транзакций** 5. наличие сетевого сервиса 6. **безопасность работы с огромными массивами данных** |
| **8 UML – это**   1. оболочка высокоуровневого языка программирования 2. группа разработчиков программного обеспечения 3. **язык моделирования программных систем** 4. формат общения «разработчик» — «заказчик» 5. методика построения модулей |
| **9 Часть процесса изготовления программного обеспечения, связанная с поддержкой и контролем взаимосвязей рабочих продуктов различных версий конечного продукта называется**   1. управлением коллективом 2. управлением качеством 3. управлением продажами 4. **управление конфигурацией** 5. управлением данными |
| **10 Программное сопровождение подразделяют на три категории**   1. изменяющее 2. **Корректирующее** 3. формирующее 4. **Настраивающее** 5. **Совершенствующее** |
| **11 Недостаток использования оценки работы по размеру кода связан с**   1. квалификацией разработчиков 2. сложностью подсчета 3. сложностью реализации 4. **его субъективностью** 5. **Относительностью** |
| **12 Предусмотрение изменений – это принцип, который влияет на такие качества программного обеспечения как**   1. детерминированность реализации 2. понятность 3. **повторную применимость** 4. прозрачность 5. **способность модификации** |
| **13 С точки зрения разработчика программного обеспечения качество последнего заключается в**   1. **расширяемости** 2. **тестируемости** 3. производительности 4. **переносимости** 5. легкости применения |