软件工程测试题

学堂在线选择题汇总

1. **初识软件工程**

1. 软件工程方法是（ ）。
    1. 为了获得高质量软件而实施的一系列活动
    2. 为开发软件提供技术上的解决方法
    3. 为支持软件开发、维护、管理而研制的计算机程序系统
    4. 为了理解问题和确定需求而采取的一些技术和方法
2. 下面的（ ）是正确的。
    1. 运行正确的软件就是高质量的软件。
    2. 软件质量是在开发过程中逐渐构建起来的。
    3. 软件产品质量越高越好，最理想的情况是达到“零缺陷”。
    4. 软件质量是由产品的功能、性能、易用性等外在特性决定的。
3. 在 Garvin 多维度模型中，可靠性是指（ ）。
    1. 软件产品提供了让用户产生惊喜的特性
    2. 软件实现了用户需要的功能和性能
    3. 软件在规定时间和条件下无故障持续运行
    4. 软件符合国家或行业的相关标准
4. （ ）是软件从一个硬件或软件环境转换到另一环境的容易程度。
    1. 易用性
    2. 可维护性
    3. 可移植性
    4. 性能
5. 下面的（ ）说法是正确的。
    1. 由于软件是产品，因此可以应用其他工程制品所用的技术进行生产
    2. 购买大多数计算机系统所需的硬件比软件更昂贵

3. 大多数软件系统是不容易修改的，除非它们在设计时考虑了变化
4. 一般来说，软件只有在其行为与开发者的目标一致的情况下才能成功

6. 造成大型软件开发困难的根本原因在于（ ）。
    1. 开发人员缺乏足够的开发经验
    2. 对软件开发的资金投入不足
    3. 项目开发进度不合理
    4. 软件系统的复杂性
7. 软件会逐渐退化而不会磨损，其原因在于（ ）。
    1. 软件通常暴露在恶劣的环境下
    2. 软件错误在经常使用之后会逐渐增加
    3. 不断的变更使组件接口之间引起错误
    4. 软件备件很难订购
8. “软件工程”术语是在（ ）被首次提出。
    1. Fred Brooks 的《没有银弹：软件工程中的根本和次要问题》
    2. 1968 年 NATO 会议
    3. IEEE 的软件工程知识体系指南（SWEBOK）
    4. 美国卡内基·梅隆大学的软件工程研究所
9. Ariane 5 火箭发射失败的事例告诉我们（ ）。
    1. 系统环境的变化可能影响软件采集数据的精度、范围和对系统的控制
    2. 软件后备系统可以通过复制生成
    3. 软件重用必须重新进行系统论证和系统测试
    4. 选项 A 和 C
    5. 选项 A、B 和 C
10. 软件工程的基本目标是（ ）。
    1. 开发足够好的软件
    2. 消除软件固有的复杂性
    3. 努力发挥开发人员的创造性潜能
    4. 更好地维护正在使用的软件产品

1. **编写高质量代码**

1. 下面的（ ）不是良好编码的原则。
    1. 在开始编码之前建立单元测试
    2. 建立一种有助于理解的直观布局
    3. 确保注释与代码完全一致
    4. 保持变量名简短以便代码紧凑
2. 下面的（ ）是错误的。
    1. 在程序设计中使用括号以改善表达式的清晰性
    2. 不要修补不好的程序，要重新写
    3. 在程序设计中应尽可能对程序代码进行优化
    4. 不要在注释中重复描述代码
3. 为了保证软件的质量，使其具有较好的可维护性，关键在于（ ）。
    1. 选择合适的程序设计语言
    2. 选择好的程序设计风格
    3. 具有好的数据结构
    4. 选择好的运行环境
4. 下面的（ ）是对提高程序编码效率没有影响的。
    1. 变量名的使用
    2. 选择良好的设计方法
    3. 选择良好的算法
    4. 选择良好的数据结构
5. 下面的（ ）不是一种好的做法。
    1. 好的注释应解释为什么，而不是怎么样。
    2. 好的命名应一目了然，不需要读者去猜，甚至不需要注释。
    3. 如果项目中原有代码不符合新的规范，应允许其存在，同时在新的代码中要延续原有的风格。
    4. 如果项目中原有代码不符合新的规范，应允许其存在，但不应在新的代码中延续旧的风格。
6. 下面的（ ）不是模块化设计的目的。

    1. 降低程序设计的复杂性
    2. 清楚地描述系统的功能和性能
    3. 易于维护和功能扩展
    4. 提高模块的可靠性和复用性
7. 下面的（）说法是错误的。
    1. 代码审查用于检查源代码是否达到模块设计的要求
    2. 代码在审查之前必须要成功地编译通过
    3. 代码审查比运行程序进行测试的效率低
    4. 代码审查可以发现不符合团队代码规范的地方
8. 关于代码性能优化，下面（）是错误的。
    1. 任何优化都不能破坏代码的正确性
    2. 应以提高程序的全局效率为主，局部效率为辅
    3. 应先通过测试找出限制效率的真正瓶颈
    4. 要优先改进耗时最多的部分
9. 下面的 Python 语句中，（）是没有错误且写得最规范的。
    1. import os, sys, random, math
    2. n += 1; m += n; print(m)
    3. class = Class()
    4. return [i ** 2 for i in range(n)]
10. 下面的（）语句风格是最不利于维护的。
    1. return s['name'] if s['age'] >= 18 else s['nickname'] if s['age'] > 14 else 'anonymous'
    2. main(sys.argv[1:])
    3. from my_module import (Class1, Class2, Class3, Class4)
    4. a, b = b, a

1. **单元测试**

1. 在单元测试中，（）是用来代替被测模块的子模块的。
    1. 驱动模块
    2. 桩模块
    3. 通讯模块
    4. 代理模块
2. 在下面列举的测试覆盖中，（）是最强的逻辑覆盖准则。

1. 语句覆盖
2. 条件覆盖
3. 判定覆盖
4. 条件组合覆盖

3. 一个判定中的复合条件表达式为（A>2）or（B≤1），为了达到 100%条件覆盖率，至少需要设计（ ）测试用例。
    1. 1
    2. 2
    3. 3
    4. 4
4. 条件覆盖要求（ ）。
    1. 每个判定中每个条件的所有取值至少满足一次
    2. 每个判定至少取得一次“真”值和一次“假”值
    3. 每个判定中每个条件的所有可能取值组合至少满足一次
    4. 每个可执行语句至少执行一次
5. （ ）要求每个判定中所有条件的可能取值至少执行一次，而且每个判定的可能结果也至少执行一次。
    1. 判定覆盖
    2. 条件覆盖
    3. 判定条件覆盖
    4. 条件组合覆盖
6. 单元测试内容不包括（ ）。
    1. 出错处理
    2. 全局数据结构
    3. 独立路径
    4. 模块接口
7. 下面的（ ）是错误的。
    1. 静态测试是不运行被测程序，仅通过检查和阅读等手段来发现程序中的错误
    2. 动态测试是实际运行被测程序，通过检查运行的结果来发现程序中的错误
    3. 动态测试可能是黑盒测试，也可能是白盒测试

    4. 白盒测试是静态测试，黑盒测试是动态测试

8. 关于等价类划分，下面的（ ）说法是正确的。
    1. 等价类划分是将输入域划分成尽可能少的若干子域
    2. 同一输入域的等价类划分是唯一的
    3. 用同一等价类中的任意输入对软件进行测试，软件都输出相同的结果
    4. 对于相同的等价类划分，不同测试人员选取的测试用例集是一样的
9. 白盒测试是根据程序的（ ）来设计测试用例。
    1. 功能
    2. 性能
    3. 内部逻辑
    4. 内部数据
10. 关于测试覆盖率，下面的（ ）说法是错误的。
    1. 测试覆盖率是度量代码质量的一种手段
    2. 测试覆盖率是度量测试完整性的一种手段
    3. 测试覆盖率意味着有多少代码经过测试
    4. 不要盲目地追求 100%测试覆盖率

1. **软件开发过程**

1. 下面的（ ）决策是在需求分析时做出的。
    1. 自动售票机系统的开发时间预计是 6 个月
    2. 自动售票机系统由用户界面子系统、价格计算子系统以及与中心计算机通信的网络子系统组成
    3. 自动售票机系统已经达到交付的要求
    4. 自动售票机系统将为使用者提供在线帮助
2. 下面的（ ）决策是在系统设计时做出的。
    1. 自动售票机系统的开发时间预计是 6 个月
    2. 自动售票机系统由用户界面子系统、价格计算子系统以及与中心计算机通信的网络子系统组成
    3. 自动售票机系统已经达到交付的要求
    4. 自动售票机系统将为使用者提供在线帮助

3. 下面的（ ）是软件构造活动的任务。
    1. 构建软件组件
    2. 设计用户界面
    3. 实施组件的单元测试
    4. 评估组件的质量
    5. 选项 A 和 C
    6. 选项 A、B、C 和 D
4. 瀑布模型是（ ）。
    1. 适用于需求被清晰定义的情况
    2. 一种需要快速构造可运行程序的好方法
    3. 一种不适用于商业产品的创新模型
    4. 目前业界最流行的过程模型
5. 增量模型是（ ）。
    1. 适用于需求被清晰定义的情况
    2. 一种需要快速构造核心产品的好方法
    3. 一种不适用于商业产品的创新模型
    4. 已不能用于现代环境的过时模型
6. 原型化模型是（ ）。
    1. 适用于客户需求被明确定义的情况
    2. 适用于客户需求难以清楚定义的情况
    3. 提供一个精确表述的形式化规格说明
    4. 很难产生有意义产品的一种冒险模型
7. 开发一个支持 3D 打印的操作系统最适合采用（ ）。
    1. 瀑布模型
    2. 原型化模型
    3. 增量开发
    4. 可转换模型
8. 开发一个铁路信号控制系统最适合采用（ ）。
    1. 瀑布模型

2. 原型化模型
3. 增量开发
4. 可转换模型

9. 下面的（ ）不是敏捷开发方法的特点。
    1. 软件开发应该遵循严格受控的过程和详细的项目规划
    2. 客户应该和开发团队在一起密切地工作
    3. 通过高度迭代和增量式的软件开发过程响应变化
    4. 通过频繁地提供可以工作的软件来搜集人们对产品的反馈
10. 关于 Scrum 的每一次冲刺（Sprint），下面的（ ）是正确的。
    1. Sprint 是一个不超过 4 周的迭代，其长度一旦确定，将保持不变。
    2. Sprint 的产出是一个可用的、潜在可发布的产品增量。
    3. Sprint 在进行过程中，其开发目标、质量验收标准和团队组成不能发生变化。
    4. 以上所有选项

1. **团队开发管理**

1. 在软件开发的各种资源中，（ ）是最重要的资源。
    1. 开发工具
    2. 方法
    3. 硬件环境
    4. 人员
2. 在攻克技术难题时，最佳的开发团队组织模型是（ ）。
    1. 民主式结构
    2. 主程序员式结构
    3. 矩阵式结构
    4. 以上所有选项都不是
3. 在选择开发团队组织结构时应考虑（ ）因素。
    1. 沟通的复杂程度
    2. 最终程序的规模大小
    3. 发布日期的严格程度
    4. 项目预算的多少

5. 选项 A，B 和 C
6. 选项 A，B 和 D

4. 下面的（ ）很有可能会促进高效项目团队的建设。
    1. 团队成员超过 20 人
    2. 团队成员部分时间参与项目
    3. 团队成员向多个经理汇报
    4. 团队成员被指派到项目中
    5. 以上选项都不是
5. 下面的（ ）沟通方式最利于协助解决复杂的问题。
    1. 口头
    2. 书面
    3. 电子邮件
    4. 即时通讯工具
6. 下面的（ ）方法最不适合你向团队成员解释他或她为什么表现不合格。
    1. 个人谈话
    2. 项目团队会议
    3. 正式报告
    4. 电子邮件
7. 软件开发团队的每一个成员都应该参与计划活动，以便（ ）。
    1. 降低计划的粒度
    2. 深入地分析需求
    3. 所有成员同意该计划
    4. 开始设计
8. 功能点估算技术需要以（ ）为基础进行问题分解。
    1. 信息域
    2. 项目进度
    3. 软件功能
    4. 过程活动

9. 某大型化工产品公司计划开发一个新的计算机应用，用以跟踪原材料的使用情况。这个应用由公司内部组成的开发团队进行开发，已有多年开发类似应用的经验。假设初始估计的程序规模是 32000 行源代码，使用基本 COCOMO 模型进行估算，开发工作量大约是（ ）人月。
    1. 32
    2. 91
    3. 230
    4. 146
10. 经验估算模型是基于（ ）。
    1. 专家基于过去项目经验的判断
    2. 期望值估计的细化
    3. 来自历史项目数据的回归模型
    4. 反复试验决定参数和系数

1. **敏捷开发与配置管理**

1. 下面的（ ）是有效的软件配置项。
    1. 软件工具
    2. 文档
    3. 可执行程序
    4. 测试数据
    5. 以上所有选项
2. 敏捷开发方法通过（ ）管理不可预测性。
    1. 非常仔细地收集和定义需求
    2. 制定详细的开发计划
    3. 软件增量必须在较短周期内发布
    4. 软件过程必须逐渐适应变化
    5. 选项 A 和 B
    6. 选项 C 和 D
3. 关于 Sprint，下面的（ ）是错误的。
    1. 一个 Sprint 通常是一个 1-4 周的迭代
    2. Sprint 长度在开发过程中是可以调整的
    3. 需求在一个 Sprint 中是不允许变化的

    4. Sprint 的产出是“完成”的、可用的、潜在可发布的产品增量

4. 在每日站立会议上，下面（ ）不是每个团队成员需要回答的主要问题。
    1. 从上次 Scrum 站立会议后你做了什么?
    2. 你遇到哪些障碍或困难?
    3. 你所遇到问题的原因是什么?
    4. 你打算到下次 Scrum 站立会议完成什么?
5. 下面的（ ）不属于产品负责人（Product Owner）的职责范围。
    1. 组织每日站立会议
    2. 定义产品需求
    3. 确定需求优先级
    4. 验收迭代结果
    5. 负责产品的投资回报
6. 在敏捷开发方法中，用户故事（User Story）的作用是（ ）。
    1. 定义需要发布给最终用户的软件特性和功能
    2. 确定发布每一次增量的日程表
    3. 用于代替详细的活动计划
    4. 用于估算构建当前增量所需要的努力
    5. 选项 A 和 C
    6. 选项 A 和 D
7. 下面的（ ）是正确的。
    1. 故事点是一个绝对度量单位
    2. 故事点估算一定要做到非常精确
    3. 故事点表示开发一个用户故事或特性的复杂度
    4. 故事点表示开发一个用户故事或特性所要付出的工作量
8. 软件配置管理的目的是（ ）。
    1. 降低开发成本
    2. 控制软件修改
    3. 减少混乱
    4. 提高软件开发效率

5. 提高正确率

9. 小图所在的某校信息学院有一位程老师，他对生命游戏特别感兴趣，正巧他看到小图最近在研究生命游戏。程老师想了一些生命游戏的新规则，他想检验一下那些规则是否有效，于是拍了拍小图的肩膀，语重心长地说：“生命游戏能不能成为游戏界的主流，能不能在游戏史上留下浓墨重彩的一笔，就靠你们年轻人了！”然后程老师就把实现那些规则的任务交给小图了。什么？程老师为什么不自己实现？程老师这么多年的编程经验，什么样的程序没见过！这是给小图一个掌握编程经验的机会！
小图自己已经编写了一个有 bug 的生命游戏。目前的代码中存在一些 bug，想到这小图就更慌了。幸好，小图有个从小穿一条裤子长大的好朋友。相信屏幕前那闪耀着智慧光芒的你已经猜到了，这位好朋友就是你！快来帮小图这个忙吧！小图想请你帮他一起去完成程老师提出的那些新规则。
为了更好地进行协作，你们决定用 git 版本库来管理代码。
1.小图首先在 Github 上创建了一个版本库（https://github.com/ThssSE/MOOC_LifeGame），然后马上在本地通过 git clone 将它克隆到了本地，这时 git status 中会提示 On branch （ ）。
    1. dev
    2. master
    3. release
    4. Branch
10. 小图首先把他已经编写好的代码文件全部放入了本地版本库中，然后想通过 git commit 提交，但提示提交内容为空，不允许提交，于是你通过（ ）帮他解决了这个问题。
    1. git commit --amend，进行修补提交
    2. git commit -a，提交所有改动
    3. git commit --allow-empty，允许空提交
    4. git status 查看状态，再执行 git add 命令选择要提交的文件，然后提交
11. 提交后，你才发现不小心把一些临时文件（*.xxx）也提交进去了，以下最好的解决方式是（ ）。
    1. 编辑.gitignore 文件，增加*.xxx 条目，然后 git commit -a 把.gitignore 提交到版本库
    2. git rm 删除*.xxx，然后 git commit 提交
    3. git rm 删除*.xxx 后再编辑.gitignore 增加*.xxx 条目，最后 git commit --amend 进行修补提交
    4. 幸好还没进行 git push，重新 clone 然后重新添加文件后提交即可
12. 折腾了半天，终于把小图的现有代码完整、干净地提交并 push 到了服务器上，你也本地 clone 了同一个版本库。请将版本库（https://github.com/ThssSE/MOOC_LifeGame）clone

到本地，可以看到有多个分支，请切换到 git-demo 分支，然后先把 git-demo-1 合并至 git-demo 分支，会发生（ ）。

1. 合并成功，且不产生新的提交
2. 合并成功，且产生一个自动 merge 的新提交
3. 合并失败，因为出现了冲突（Conflicts）
4. 合并失败，因为 git-demo 分支是 git-demo-1 的子分支

13. 接着，请继续把 git-demo-2 分支合并至上述合并后的 git-demo 分支，会发生（ ）。
    1. 合并成功，且不产生新的提交
    2. 合并成功，且产生一个自动 merge 的新提交
    3. 合并失败，因为出现了冲突（Conflicts）
    4. 合并失败，因为 git-demo-2 分支是 git-demo-1 的子分支
14. 接下来你们将进行开发工作，做了分工，每人实现几个功能。你在本地新增了一个名为 feature1 的 branch，关于这个 branch，以下说法正确的是（ ）。
    1. 在本地新增 feature1 后，远端服务器的版本库中也会自动出现一个同名分支
    2. feature1 分支在本地被删除后，远端服务器的版本库中不会自动删除同名分支
    3. 可以指定将 feature1 分支提交到远端服务器的其他分支，但这会导致其他分支被覆盖，因此一般不这么做
    4. 可以指定将 feature1 分支提交到远端服务器的另一分支 branch2，提交后可以手动再把 branch2 之前的最新提交 merge 到 branch2 去，因此不会丢失原 branch2 的提交
15. 终于开始愉快地编程了，你们先进行 bug 的修复工作。你在修复某个 bug 时，非但没有完成修复，还导致了更严重的 bug，这时你想把一个名为 game 的文件恢复至原始版本，只要执行（ ）就可以了。
    1. git checkout --reset game
    2. git rm --revert game
    3. git checkout HEAD -- game
    4. git ignore game
16. 你又继续了 bug 修复，这回成功完成了修复。又过了一会儿，你在修复另一个 bug 时又不小心改乱了想恢复，于是用 git reset --hard 来把另一个文件恢复，但小手一抖不小心把已经完成修复的 game 文件恢复了导致数据丢失。丢失了的数据还能找回吗？（ ）
    1. 不能，硬重置使工作区文件被覆盖，导致数据丢失无法找回
    2. 不能，因为尚未提交
    3. 能，可以通过 git checkout HEAD@{1} -- game 找回

4. 不确定，如果在重置前执行了 git add 命令将 game 加入了暂存区，则可以在对象库中处于悬空状态的文件中找到

17. 你对一些文件进行了修改后，通过执行（ ）就可以把当前工作区的所有被修改的文件都添加到暂存区且不添加新增的文件。
    1. git add .
    2. git add -A
    3. git add -m
    4. git add -u
18. 你和小图都进行了一些开发工作，然后进行合并时，提示了出现 conflicts，该怎么处理呢？（ ）
    1. 这是因为你和小图同时 push 到服务器导致的，只要你们俩轮流 push 就不会有问题
    2. 这是因为合并时你们修改了不同的文件，导致无法合并，只要将被修改的文件在另一个待合并的分支也相应修改就能解决
    3. 这是因为合并时你们修改了同一个文件，因此合并后只能保留其中一个分支的该文件，要顺利解决可以先将文件备份，然后合并后再手动将变动同步过来
    4. 这是因为你们对同一个文件的同一个位置进行了不同的改动，合并后会有特殊标记标明冲突的部分，进行处理后再提交即可解决冲突
19. 以下文件片段，表示该片段发生代码冲突的是（ ）
    1. <<<<<<< feature-2 if __name__ == '__main__': ======= if os.path.basename(__file__) == 'main.py': >>>>>>> HEAD
    2. <<<<<<< HEAD if __name__ == '__main__': ======= if os.path.basename(__file__) == 'main.py': >>>>>>> feature-2
    3. ======= HEAD if __name__ == '__main__': <<<<<<< if os.path.basename(__file__) == 'main.py': ======= feature-2
    4. ======= feature-2 if __name__ == '__main__': >>>>>>> if os.path.basename(__file__) == 'main.py': ======= HEAD

1. **需求获取**

1. 下列哪项需求描述属于业务需求描述?
    1. 我们的任务是无缝集成有竞争力的软件信息服务来解决商业问题
    2. 我们的目标是让客户将我们的品牌和高质量联系在一起
    3. 我们公司的主营业务是销售飞机票
    4. 公司网站上销售的产品必须满足所有食品药品监管需求
2. 下面哪项是百货店收银系统的非功能性需求?
    1. 提供新鲜的蔬菜和水果

    2. 买 10 个或 10 个以下商品的客户可以走特殊通道
    3. 设有存包处
    4. 为雇员发工资
3. 以下哪种方法最适用于身处多个不同地点的人在各自方便的时间参与并围绕同一个主题表达自己的观点?
    1. 问卷调查
    2. 面谈
    3. 群体诱导
    4. 文档分析
4. 在一个列车控制软件的需求文档中，我们发现了以下两条需求描述：“列车车门在两个停靠站之间要保持关闭”；“列车发生紧急停车时，要打开车门”。这里出现的需求问题是什么?
    1. 无法测试的需求
    2. 不完整的需求
    3. 含糊的需求
    4. 矛盾与不一致的需求
5. 获取软件系统需求不包括以下的哪个来源?
    1. 系统相关领域的法律法规
    2. 系统的质量控制团队
    3. 系统的业务流程描述
    4. 其他类似系统产品
6. 软件需求工程师的职责不包括以下的哪一项?
    1. 撰写需求规格说明书
    2. 与用户持续沟通，了解用户对产品的期望
    3. 控制项目的风险
    4. 对需求的优先级进行排序
7. 在选择软件需求获取技术的时候，以下哪种策略最优?
    1. 考虑尚不了解的那部分需求的特点
    2. 考虑需求工程师本身对各种获取技术的驾驭能力
    3. 考虑目前系统所属的行业及应用领域的现状

    4. 综合考虑上述因素

8. 以下哪种需求获取方法是面向创新型产品的?
    1. 竞争性需求分析
    2. A/B 测试
    3. 用户行为数据采集
    4. 可用性分析
9. 在敏捷开发方法中，用户故事（User Story）的作用是什么?
    1. 定义需要发布给最终用户的软件特性和功能
    2. 确定发布每一次增量的日程表
    3. 用于代替详细的活动计划
    4. 用于估算构建当前增量所需要的努力
    5. 选项 A 和 C
    6. 选项 A 和 D
10. 下面的哪一种说法是正确的?
    1. 故事点是一个绝对度量单位
    2. 故事点估算一定要做到非常精确
    3. 故事点表示开发一个用户故事或特性的复杂度
    4. 故事点表示开发一个用户故事或特性所要付出的工作量

1. **用例建模**

1. 我们在为一家互联网电商开发订单处理软件，该公司从供应商那里购买产品，然后销售给客户。这家公司在线发布商品目录，并将其推送给客户和其他感兴趣的人。
   客户以提交商品列表并向电商付费的方式购买商品。电商填写帐单，并委托快递公司把商品运送到客户的地址。订单处理软件记录从收到订单直到商品被运送给客户的整个过程。电商将提供快捷的服务，以最快、最有效的方法来发送客户订购的产品。
   客户可以退货，但有时要付运费。
   （1）电商订单处理软件系统的参与者不包括:
    1. 网络电商
    2. 客户
    3. 其他感兴趣的人
    4. 快递公司

2. （2）在网络电商客户定单处理应用中与客户有关的用例不包括：
    1. 退货
    2. 计算运费
    3. 浏览商品
    4. 订单查询
3. （3）进入订购商品用例的前置条件是：
    1. 客户对商品感兴趣
    2. 客户安装了与系统兼容的浏览器版本
    3. 商品已经放入购物车
    4. 客户通过合法账户登入系统
4. （4）取消订单用例与查询订单用例建模为以下哪种关系最合适？
    1. 关联关系
    2. 依赖关系
    3. 包含关系
    4. 扩展关系
5. （5）使用订单处理系统一段时间以后，电商希望增加一种功能——为老顾客提供折扣。以下哪种方法比较合适？
    1. 建立老顾客折扣新用例
    2. 扩展订购商品用例
    3. 在订购商品用例中包含老顾客提供折扣用例
    4. 为订购商品用例建立两个子用例：普通顾客订购商品和老顾客订购商品
6. （6）需求说明文档通常不会采用以下哪种方式组织撰写？
    1. 用户手册
    2. 用户故事
    3. 用例模型
    4. 测试用例
7. （7）以下哪个关于用例建模的说法是正确的？
    1. 用例可以定义系统功能性需求的优先级
    2. 用例建模是对系统进行功能分解的过程

3. 用例能够描述非功能性的需求
4. 用例的参与者只能是系统用户

8. （8）以下哪种关于用户故事和用例描述的说法是不正确的?
    1. 用户故事用于敏捷过程；用例描述用于统一建模过程
    2. 用户故事作为开发者与用户交互面对面交互时的提示；用例作为项目文档保存
    3. 用户故事可以用于估算；用例描述则不能用于估算
    4. 用户故事采用自然语言文本描述；用例主要采用图形化的模型表示
9. （9）可以选用以下哪种工具进行用例建模?
    1. Microsoft Project
    2. Enterprise Architect
    3. Enterprise Architecture
    4. IBM Rational DOORS
10. （10）用例图中，当一个用例只在一定条件下比另一个用例增加少数步骤时，用哪种关系建模最合适?
    1. extends
    2. includes
    3. uses
    4. Inherits

1. **面向对象分析与设计**

1. 下列哪项关于面向对象分析来源的说法不正确?
    1. 面向对象分析的思路部分源于面向对象的程序设计
    2. 面向对象分析的思路部分源于数据库领域的实体关系图
    3. 面向对象分析的思路部分来源于面向对象设计
    4. 面向对象分析的思路部分来源于人工智能领域的知识表示方法
2. 下列哪项关于对象服务的说法是不正确的?
    1. 创建新对象、撤销就对象，修改对象属性等瞬时完成的服务
    2. 为其他对象完成各种计算服务
    3. 持续检查预设条件是否满足的监控服务
    4. 对象只有属性和行为，不对外提供服务
3. 采用 CRC 卡片分拣法的分析过程不含以下哪个步骤?
    1. 建立系统的类设计模型

    2. 定义每个类的职责
    3. 确定类之间的交互关系
    4. 识别对象类
4. 面向对象技术中，封装的含义是
    1. 用状态机图来描述对象的行为
    2. 将对象的状态锁定，使之不能被修改
    3. 保证对象内部的数据只能通过操作来访问
    4. 将对象放入集合
5. 面向对象设计中，“设计抽象的接口”的含义是?
    1. 向用户暴露尽可能多的系统实现细节
    2. 向用户暴露尽可能少的实现细节
    3. 不仅仅考虑用户的业务需求，还要考虑设计约束
    4. 让用户决定接口的定义，减少开发人员的决策负担
6. 面向对象设计方法中，开闭原则的含义是?
    1. 软件实体在更改性方面应该是开放的，在扩展性方面应该是封闭的
    2. 要尽可能多地使用接口进行封装，利用多态技术，扩展时不需修改源代码
    3. 尽可能多定义类的继承关系，运用抽象机制
    4. 采用契约式设计
7. 面向对象设计方法中，LSP 替换原则要求“子类可以替换父类出现在父类能出现的任何地方”，下面正确的说法是?
    1. 将正方形定义为矩形的子类，符合 LSP 替换原则的要求
    2. 要尽可能多地使用接口进行封装，利用多态技术，扩展时不需修改源代码
    3. 子类中方法的前置和后置条件不能弱于父类中相应方法的前置和后置条件
    4. 采用契约式设计
8. 类定义不会要求其对象实例具有以下哪个特征?
    1. 相同状态
    2. 相同属性
    3. 相同行为
    4. 相同的对象关系

9. UML 类图中对以下几种类型关系的使用频度从高到低应为?
    1. 实现>关联>依赖>泛化>聚合>组合
    2. 泛化>聚合>组合>关联>实现>依赖
    3. 关联>泛化>聚合>组合>依赖>实现
    4. 依赖>泛化>聚合>组合>泛化>关联
10. 关于 UML 类图中泛化关系建模，不正确的说法是?
    1. 定义泛化关系的好处是当环境发生变化时，便于添加新的子类
    2. 当某个类中，存在属性和操作略有不同的子类时，应定义泛化关系
    3. 当现有的多个类具有公共属性和方法时，可以定义一个父类让它们共同继承
    4. 关联关系也可以用于定义分类关系，可以替代泛化关系

1. **行为建模**

1. UML2.0 中对行为建模的图不包括：
    1. 对象图
    2. 状态图
    3. 顺序图
    4. 时间图
2. 顺序图中带条件消息的发送，不能采用以下哪种方式进行?
    1. 用文字说明，作为注释添加
    2. 添加条件控制框
    3. 分成多个顺序图子图来描述
    4. 在消息名字前加条件子句
3. 顺序图的组合控制框（Frame）中，用于表达分支选择关系的控制符是哪个?
    1. opt
    2. alt
    3. par
    4. loop
4. 当一个顺序图过大时，最好的处理方法是?
    1. 添加注释，说明顺序图的各部分之间的接续关系
    2. 用不完整的箭头，指明本页的顺序图未完待续
    3. 去掉无关的细节，保持图的简洁性

    4. 添加 ref 框，建立顺序图间的引用关系

5. 对顺序图与用例之间关系的阐述，错误的是：
    1. 顺序图表达单个情景实例的行为，每个用例对应一个顺序图
    2. 用例分析阶段的顺序图要包含设计对象，并关注消息参数
    3. 顺序图用于表示为完成用例而在系统边界输入输出的数据以及消息
    4. 顺序图可帮助分析人员对用例图进行扩展、细化和补遗
6. 下面关于对象状态建模的说法中，正确的是：
    1. 大部分对象的状态空间都是有限的
    2. 对象状态建模要穷举对象能够到达的所有状态，保证完整性
    3. 大部分对象的状态空间大小是由它的属性取值决定的
    4. 对象的状态数量是由它对外提供的操作的数量决定的
7. 状态迁移的发生不会受到哪个因素的影响？
    1. 目标状态
    2. 外部事件
    3. 警戒条件
    4. 迁移动作
8. 下面关于 UML 状态图的说法正确的是：
    1. UML 状态图中的状态可以分解为“与”状态，以及“或”状态，但是都可以转化为基本状态机来表示
    2. UML 状态图中的状态是原子的，不可再分
    3. UML 状态图中的状态只可以分解为“与”状态，表示可以两种状态并存
    4. UML 状态图中的状态只可以分解为“或”状态，表示只能选择其中之一
9. 关于状态图与其他 UML 图的关系，说法不正确的是：
    1. 状态图中的事件为顺序图中该对象的输入消息
    2. 状态图中每个动作对应于其他类的一个操作
    3. 状态图中的动作定义等价于类图中的操作定义
    4. 状态图应针对类图中所有的类进行建模
10. 以下状态迁移上的警戒条件定义中，哪一组是最合适的？
    1. x 大于 0，x 等于 0， X 小于 0

2. x 大于等于 0，x 小于等于 0
3. x 大于 0，x 小于 0
4. x 大于 0，x 等于 0

1. **软件系统设计**

1. 随着软件系统的规模和复杂性越来越大，（）变得更加重要。
    1. 算法的选择
    2. 数据结构的设计
    3. 数据库的构造
    4. 系统的全局结构设计
2. 下面的说法（）是错误的。
    1. 软件体系结构的最佳表示形式是一个可执行的软件原型
    2. 软件体系结构描述是不同项目相关人员之间进行沟通的使能器
    3. 良好的分层体系结构有利于系统的扩展与维护
    4. 设计模式是从大量成功实践中总结出来且被广泛公认的实践和知识
3. 良好设计的特征是（）。
    1. 模块之间呈现高耦合
    2. 实现分析模型中的所有需求
    3. 包括所有组件的测试用例
    4. 提供软件的完整描述
    5. 选项 B 和 D
    6. 选项 B、C 和 D
4. Word、Excel 等应用系统适合采用（）结构风格。
    1. 层次系统
    2. 事件系统
    3. 解释器
    4. 管道-过滤器
5. 与 C/S 架构的信息系统相比，B/S 架构的信息系统的优势是（）。
    1. 具备更高的安全性
    2. 更容易部署和升级维护

3. 具备更强的事务处理能力，易于实现复杂的业务流程
4. 用户界面友好，具有更快的响应速度

6. 对于观察者模式，下面的（ ）说法是错误的。
    1. 观察者的更新是被动的
    2. 被观察者可以通知观察者进行更新
    3. 观察者可以改变被观察者的状态，再由被观察者通知所有观察者
    4. 以上所有选项
7. 设计目标可分成性能、可靠性、成本、维护和最终用户等类型，下面（ ）描述的是性能目标。
    1. 当用户发出任何命令后，系统必须在 1 秒内将信息反馈给用户。
    2. 即使在网络失败的情况下，火车票发售系统也必须能够成功地提交火车票。
    3. 火车票发售系统的机器外壳必须允许安装新按钮以便增加新的不同票价。
    4. 系统用户界面应该防止用户以错误的顺序执行命令。
8. 下面的（ ）架构可以更好地实现 Web 应用的前后端分离。
    1. MVC
    2. Restful API
    3. RPC
9. 一个创业团队想要开发一款社交 App，但是他们对于可能的发展的业务仍不十分明确，此时希望能够开发一个相对简单的版本进行原始的需求验证与测试，这种情况比较适合选择（ ）数据库。
    1. Mysql
    2. Mongo
    3. Redis
10. 如今社交文化横行，大数据分析遍地，几乎所有的应用与产品都多少会与社交网络或大用户量、大数据相关联。如果准备开发一个较完整的社交网络应用，支持 10 万以上日活跃用户进行各种点赞、评论等交互活动，应该采用（ ）数据库或数据库组合。
    1. Mysql
    2. Mongo
    3. Mysql + Redis
    4. Mongo + Redis

假设你所在的开发团队负责系统的升级改造，请结合以下描述回答问题：

1. 为了提升抢票性能，开发团队提出了下面的数据库优化方案，其中（ ）方案是不合理的。
    1. 对频繁检索的键增加适当的索引（Index）

    2. 对频繁检索的表移除外键（Foreign Key），改为通过代码层面保证安全性
    3. 对于已结束一段时间的活动，将票的信息归档后即可从数据库中删除相关数据项
    4. 将票的信息存储于 Redis 等内存型数据库，而不再存储于 MySQL 等关系型数据库中
2. 对于一些频繁查询且不易变动的信息，可以通过 Redis 等内存型数据库进行缓存，下面的（）信息不需要进行缓存。
    1. 活动详情
    2. 近期活动列表
    3. 剩余活动票数
    4. 已抢到票的信息
3. 当 Redis 缓存的数据过期时，下面的（）更新方案是比较合理的。
    1. 清空 Redis 数据库
    2. 删除所有受影响的 Redis 数据
    3. 计算得到所有受影响数据的新数据，缓存至 Redis 替换已有数据
    4. 无需处理，当缓存有效时间过期后自然会更新数据
4. 开发团队决定对已有系统进行重构，即将后端改写为 RESTful，这样做的好处是（）。
    1. 易于优化数据库访问
    2. 减少前后端耦合，方便分离开发
    3. 易于维护与测试
    4. 易于开放第三方接口
    5. 选项 B 和 D
    6. 选项 B、C 和 D
    7. 选项 A、B、C 和 D
5. 原有系统只支持单人抢单张票，而且是自动分配座位。新的升级系统希望允许用户在一次活动中可以抢不超过设定最大票数的任意张票，而且可以为每张票选择座位。针对这个需求，下面的（）实现方式是比较合理的。
    1. 在抢票开始前，将所有票及其对应座位在数据库中生成好。用户抢票时，根据其请求抢票的张数，分配电子票。
    2. 设计电子票的数据表和座位的数据表，用户抢票时的逻辑与原有系统类似，只是增加对一次抢多张票的支持。抢票完成后用户可以进行选座，选座即将电子票与座位建立对应关系。

3. 以上方式均不合理

6. 当管理员创建一个活动后，希望之前参加过同类活动的用户能收到该活动的推送。当用户抢到票后，活动开始当天早 9 点和活动开始前 30 分钟，用户都能收到一条推送消息提醒及时检票入场。针对这个需求，下面的（ ）实现方式是不合理的。
    1. 使用 Linux cron job，在每天早 9 点运行脚本向所有抢到票的用户推送提醒消息
    2. 使用异步队列的定时任务，在每天早 9 点向所有抢到票的用户推送提醒消息
    3. 使用异步队列，在用户抢到票后启动异步任务，阻塞至活动开始前 30 分钟向用户推送提醒消息
    4. 使用异步队列的定时任务，每隔 1 分钟检查是否需要向用户推送活动开始前 30 分钟的提醒消息

1. **软件交互设计**

1. 输出一列数值时，需要考虑对齐方法，你会选择（ ）方法。
    1. 左对齐
    2. 右对齐
    3. 两端对齐
    4. 小数点对齐
2. 用 KLM 自己计算课件上的两个温度转换器软件界面的操作时间。后者交互效率高，（ ）原因更本质。
    1. 没有在两种输入设备间切换
    2. 界面没有需要用户额外表达的信息
    3. 用户操作的更快
    4. KLM 的计算结果小
3. 访问 http://fww.few.vu.nl/hci/interactive/fitts/ 并对其中实验 5 的结果进行分析，下面的（ ）是正确的。
    1. 所采用的两种输入设备都是目标越大访问时间越长
    2. 在所采用的两种输入设备上，只有其中一个是目标越大访问时间越长
    3. 在所采用的两种输入设备上，实验时间均与 Distance/Width 呈正相关
4. 访问 http://www.asktog.com/columns/022DesignedToGiveFitts.html 并为每个问题答案寻找实例界面，其中环形菜单的设计是（ ）。
    1. 不知如何启动，毫无意义

2. 对美工的设计能力要求很高
3. 到达每个菜单项时手的移动距离短且一样

5. 阅读下面给出的文献 “A Brief History of Human-Computer Interaction Technology”，其中第一个装载 GUI 操作系统的计算机是（ ）。
    1. 苹果公司的 Macintosh
    2. 微软公司的 Windows
    3. 施乐公司的 Alto
    4. 施乐公司的 Star
    5. 施乐公司的 Lisa

阅读文献：Brad A. Myers. A Brief History of Human-Computer Interaction Technology. ACM interactions. Vol. 5, no. 2, March, 1998. pp. 44-54

1. 思考一下，乔布斯为苹果公司制定的手机上的黄金法则是什么？为什么？采用现状及其原因？这里的黄金法则是指（ ）。
    1. 单手操作手机，屏幕尺寸 3.5 英寸
    2. 单手操作手机，屏幕尺寸 4.7 英寸
    3. 双手操作手机，屏幕尺寸 4.7 英寸
    4. 双手操作手机，屏幕尺寸 4.0 英寸
2. 同学们上网用一下谷歌界面、雅虎界面和 bing 界面，都是检索任务的软件，差异在哪里？这几款产品目前用户量的差异很大程度上是界面在注意力设计上的差异，请进行分析。其中，最能使用户集中注意力的界面设计是（ ）。
    1. 谷歌
    2. 雅虎
    3. bing
3. 视频（电影、电视、动画、数字视频，等）帧律的设置依据是（ ）。
    1. 根据机器的处理能力
    2. 根据个人偏好
    3. 根据人的视觉暂留时间
4. 分析课件中数字软键盘布局上的不一致现象，为什么在同一款手机上，不同应用中的数字键盘的布局不一致？其中（ ）应用中的数字软键盘布局是一致的。
    1. 电话拨号和计算器
    2. 通讯录和电话拨号

3. 计算器和通讯录

5. 人们不会把 [ ]( )[ ] 中的“](”认作一对，原因是视觉认知中的（ ）定律在起作用。
    1. 连续律
    2. 接近律
    3. 相似律
    4. 对称律

1. **软件系统测试**

1. 软件测试的目的是（ ）。
    1. 避免软件中出现错误
    2. 证明软件的正确性
    3. 解决测试中发现的错误
    4. 发现软件中潜在的错误
2. 下面（ ）说法是错误的。
    1. 测试应该尽早不断地执行
    2. 软件错误具有聚集性，对存在错误的部分应重点测试
    3. 软件测试是提高软件质量的决定性因素
    4. 测试用例需要定期评审和修改，并且要不断增加新的测试用例
3. 下面的（ ）不是集成测试的内容。
    1. 对软件中最小可测试单元进行检查和验证
    2. 把各个模块连接在一起时，穿越模块接口的数据是否会丢失
    3. 一个模块的功能是否会对另一个模块的功能产生不利的影响
    4. 若干子功能组合在一起是否能产生预期的主功能
4. 下面的（ ）是错误的。
    1. 功能测试是根据需求规格说明验证产品的功能实现是否符合要求
    2. 压力测试是检测在极限环境中使用系统时施加在用户上的压力
    3. 安全测试是检测系统中的保护机制是否可以保护系统免受非正常的攻击
    4. 安装测试是保证应用程序能够被成功地安装
5. （ ）是为了有效地发现软件缺陷而精心设计的少量测试数据。
    1. 测试计划

2. 测试用例
3. 缺陷报告
4. 测试报告

6. 错误推测法是（ ）。
    1. 将输入数据划分成若干个等价类，从中选取有代表性的数据作为测试用例
    2. 将所有可能的输入数据作为测试用例
    3. 运用场景对系统的功能点或业务流程进行描述，对应不同的业务场景生成相应的测试用例
    4. 根据经验或直觉推测程序中可能发生错误的情况，编写检查它们的测试用例
7. Web 链接测试不包括（ ）。
    1. 客户端与服务器端的连接速度
    2. 无链接指向的页面
    3. 错误的链接
    4. 不存在的页面
8. （ ）是检测 Web 应用系统提供信息的正确性、准确性和相关性 。
    1. 表单测试
    2. 链接测试
    3. 内容测试
    4. Cookies 测试
9. 下面的（ ）不是软件性能的指标。
    1. 响应时间
    2. 并发进程数
    3. 吞吐量
    4. 资源利用率
10. 下面的（ ）不是性能测试的目的。
    1. 达到百分之百的语句覆盖
    2. 验证软件系统是否能够满足预期的性能要求
    3. 发现软件系统中存在的性能瓶颈
    4. 评估软件系统的稳定性和可靠性

1. **软件交付与维护**

1. 下面的（ ）是错误的。
    1. 软件交付的主要工作是将程序代码和相关文档交给用户
    2. 用户培训是帮助用户理解产品并掌握系统的使用和操作
    3. 软件部署是通过配置、安装和激活等活动保证软件系统的正常运行
    4. 持续集成是频繁持续地将团队成员的工作进行集成
2. 下面的（ ）是正确的。
    1. 只有质量差的软件产品才需要维护
    2. 软件的维护成本通常比开发成本低
    3. 软件的不断修改将导致系统结构的恶化
    4. 重新开发一个新系统通常要比再工程的成本要低
3. （ ）是由于计算机软件和硬件环境变化而修改软件的过程。
    1. 改正性维护
    2. 适应性维护
    3. 完善性维护
    4. 预防性维护
4. 下面的（ ）不是软件再工程活动。
    1. 增加新的功能
    2. 逆向工程
    3. 程序结构改善
    4. 数据再工程
5. 逆向工程通常用在软件生命周期的（ ）阶段，它是从源代码或目标代码中提取设计信息。
    1. 需求分析
    2. 软件设计
    3. 软件测试
    4. 软件维护

1. **期末考试与总结**

1. 在选择软件需求获取技术时，下面的（ ）策略是最优的。
    1. 考虑尚不了解的那部分需求的特点

    2. 考虑需求工程师本身对各种获取技术的驾驭能力
    3. 考虑目前系统所属的行业及应用领域的现状
    4. 综合考虑上述因素
2. 下面的（ ）需求获取方法最适用于身处多个不同地点的人在各自方便的时间参与，围绕同一个主题表达自己的观点。
    1. 问卷调查
    2. 面谈
    3. 群体诱导
    4. 文档分析
3. 我们在为一家互联网电商开发订单处理软件，该公司从供应商那里购买产品，然后销售给客户。这家公司在线发布商品目录，并将其推送给客户和其他感兴趣的人。客户以提交商品列表并向电商付费的方式购买商品。电商填写帐单，并委托快递公司把商品运送到客户的地址。订单处理软件记录从收到订单直到商品被运送给客户的整个过程。电商将提供快捷的服务，以最快和最有效的方法来发送客户订购的产品。客户可以退货，但有时要付运费。在下面所列的用户故事中，（ ）是与客户无关的。
    1. 退货
    2. 计算运费
    3. 浏览商品
    4. 订单查询
4. 在一个校园微信抢票系统中，需要对活动的相关信息进行维护并支持抢票。具体包括发布活动通知、管理抢票过程（定义抢票起止时间、电子票数量等）以及活动现场检票。这时，应该采用下面的（ ）策略进行用户故事建模。
    1. 为每个主要活动分别定义一个用户故事，并为每个故事撰写相应的测试场景。
    2. 建立一个“团委活动抢票”用户故事，为该故事定义若干个场景，分别对应主要的交互过程。
    3. 定义四个用户故事，添加活动信息，修改活动信息，查询活动信息，删除活动信息。
    4. 以上所有选项
5. 需求说明文档通常不会采用（ ）方式组织撰写。
    1. 用户手册
    2. 用户故事
    3. 用例模型

    4. 测试用例

6. 关于软件测试，下面的（ ）说法是正确的。
    1. 软件测试的目的是证明软件的正确性
    2. 穷举测试是不现实的
    3. 如果单元测试做得足够好，就不用进行集成测试
    4. 自动化测试一定比手工测试的效果好
7. 下面的（ ）不属于单元测试的内容。
    1. 模块接口
    2. 局部数据结构
    3. 独立路径
    4. 用户界面
8. 下面的（ ）不是单元测试原则。
    1. 单元测试应该是可以重复执行的，并且结果是可以重现的。
    2. 单元测试应相互独立，某个测试不应为下一个测试设定条件。
    3. 单元测试可以通过查看日志文件或人工分析结果来确认是否通过。
    4. 单元测试应该快速运行。
9. 黑盒测试是根据程序的（ ）来设计测试用例。
    1. 功能
    2. 需求规格说明
    3. 内部逻辑
    4. 内部数据
    5. 性能
10. 下面的（ ）不是测试用例的设计要求。
    1. 具有代表性和典型性
    2. 寻求系统设计和功能设计的弱点
    3. 只需选取合理的输入数据
    4. 考虑用户实际的诸多使用场景
11. 在下面列举的测试覆盖中，（ ）是最弱的逻辑覆盖准则。
    1. 语句覆盖

2. 条件覆盖
3. 判定覆盖
4. 判定条件覆盖
5. 条件组合覆盖

12. 关于等价类划分，下面的（ ）说法是正确的。
    1. 等价类划分是一种常用的白盒测试方法
    2. 等价类划分是将输入域划分成尽可能少的若干子域
    3. 同一输入域的等价类划分是唯一的
    4. 为了提高测试效率，一个测试用例可以覆盖多个有效等价类
13. （ ）是在强负载下对系统进行测试，观察系统在峰值使用情况下的表现行为，从而有效地发现系统可能存在的隐患。
    1. 压力测试
    2. 负载测试
    3. 疲劳强度测试
    4. 可靠性测试
    5. 大数据量测试
14. 下面的（ ）不属于系统总体设计的任务。
    1. 明确系统设计目标
    2. 确定子系统或模块
    3. 设计数据结构与算法
    4. 选择系统部署方案
15. 某游戏公司欲开发一个大型多人即时战略游戏，游戏设计的目标之一是能够支持玩家自行创建战役地图，定义游戏对象的行为及其之间的关系。针对该目标，最适合采用（ ）体系结构风格。
    1. 管道-过滤器
    2. 隐式调用
    3. 主程序-子程序
    4. 解释器
16. 网站系统是一个典型的（ ）。
    1. 仓库体系结构

2. 胖客户机 / 服务器结构
3. 瘦客户机 / 服务器结构
4. 以上选项都不是

17. 设计目标可分成性能、可靠性、成本、维护和最终用户等类型，下面（ ）描述的是最终用户目标。
    1. 当用户发出任何命令后，系统必须在 1 秒内将信息反馈给用户。
    2. 即使在网络失败的情况下，火车票发售系统也必须能够成功地提交火车票。
    3. 火车票发售系统的机器外壳必须允许安装新按钮以便增加新的不同票价。
    4. 系统用户界面应该防止用户以错误的顺序执行命令。
18. （ ）是选择合适的解决方案策略，并将系统划分成若干子系统，从而建立整个系统的体系结构。
    1. 系统总体设计
    2. 软件详细设计
    3. 数据库设计
    4. 用户界面设计
19. 耦合表示一个模块的（ ）程度。
    1. 可以被更加细化
    2. 联接其他模块和外部世界
    3. 仅关注在一件事情上
    4. 能够适时地完成其功能
20. 某创业团队打算开发一个简单的社交网站，要求实现用户登录、内容分享等功能，每秒同时在线人数（并发请求）大约为 100 人。在这种情况下，选择（ ）数据库比较合适。
    1. MySQL
    2. Redis
    3. Mongo
    4. MySQL+Redis
21. 下面的应用场景采用（ ）数据库更合适：有一个大型的新闻网站，新闻的内容编辑完成后就不会再修改，同时因为时事热点，一些新闻会被成千万的用户在短时间内访问。
    1. Mysql
    2. Redis
    3. Mongo

22. 今年初，配合微信公开课 PRO，微信发布了“我和微信的故事”。在 HTML5 页面中，用户可以看到自己的微信注册时间、第一个朋友、第一条朋友圈以及 2015 年的朋友圈、红包、地理位置、好友、运动等活跃情况。在测试状态下，该网页链接就已经传播开来，导致访问量暴涨，所有用户需要等待几分钟甚至几十分钟才能完成载入。考虑到微信大约有 5 亿的月活跃用户，希望以尽量小的投入提高该页面的加载速度，下面的（ ）方案是最不合理的。
    1. 通过缓存，避免用户再次打开页面时的重复计算
    2. 预先计算出所有用户的“微信故事”存储于数据库中，用户打开页面只需要从数据库中读取即可
    3. 将静态文件服务器与计算服务器分离（例如将静态文件放在 CDN 即内容分发网络上）
    4. 当用户请求数据时，通过异步任务队列将计算分散至大量服务器集群，提高并行计算能力
23. 下面的（ ）界面设计原则不允许用户保持对计算机交互的控制。
    1. 允许交互中断
    2. 允许交互操作取消
    3. 对临时用户隐藏技术内部信息
    4. 只提供一种规定的方法完成任务
24. 下面的（ ）不是一种好的做法。
    1. 建立一种有助于理解的代码布局
    2. 变量命名应该一目了然
    3. 对每一行代码都要进行注释
    4. 修改代码的同时，也要维护代码周围的所有注释
25. 下面的（ ）是正确的。
    1.
```
arr = [1,2,3]
def foo():
    arr.append(4)
```
foo 函数的作用是给全局变量 arr 的末尾添加一个元素 4
26. （ ）是增加或修改系统功能，使其适应业务的变化。
    1. 改正性维护
    2. 适应性维护
    3. 完善性维护
    4. 预防性维护

27. （）是从现有的程序代码中抽取有关数据、体系结构和处理过程的设计信息，以便恢复设计结果。

    1. 代码重构
    2. 逆向工程
    3. 数据重构
    4. 正向工程

28. 下面的（）是正确的。

    1. 运行正确的软件就是高质量的软件。
    2. 软件工程会导致开发人员产生大量的无用文档，降低工作效率。
    3. 向一个进度延迟的软件项目中增加人员可能会使其进度更加推迟。
    4. 对于一个成功的软件项目，可执行程序是唯一的交付制品。

29. （）是为了获得高质量软件而实施的一系列活动，包括问题定义、需求开发、软件设计、软件构造、软件测试等。

    1. 软件工程
    2. 软件过程
    3. 软件配置管理
    4. 软件项目管理

30. （）是将一个复杂问题分解成若干个简单问题，然后逐个解决。

    1. 分而治之
    2. 软件复用
    3. 逐步演进
    4. 优化折中

31. 在 ISO 9126 模型中，下面的（）不属于易用性的质量属性。

    1. 软件显示的信息要清晰、准确且易懂，使用户能够快速理解软件。
    2. 软件使用户能学习其应用的能力。
    3. 软件产品避免因软件中错误发生而导致失效的能力。
    4. 软件产品使用户能易于操作和控制它的能力。

32. 瀑布模型的主要问题在于（）。

    1. 过于灵活

2. 用户参与开发过程
3. 强调文档的作用
4. 难以适应需求的动态变化

33. 下面的（ ）不是敏捷开发的基本原则。
    1. 尽早和持续地交付有价值的软件
    2. 要面对面进行交流
    3. 严格遵循计划和流程
    4. 坚持不懈地追求技术卓越和良好设计

34. 下面的（ ）是 Scrum 方法的优点。
    1. 降低变更对软件开发造成的风险
    2. 提高投入产出比
    3. 持续快速地发布可用的软件产品
    4. 以上所有选项

35. 下面的（ ）不是 Scrum 主管的职责。
    1. 定义产品需求
    2. 组织每日站立会议
    3. 引导团队正确应用敏捷实践
    4. 促进团队紧密协作

36. 关于每日站立会议，下面的（ ）说法是错误的。
    1. 它是一个简短的团队会议，由所有团队成员在每天固定的时间和地点进行。
    2. 站立的目的是为了让会议高效并且让每个人都集中精力。
    3. Scrum 主管应该对成员所描述的任务内容进行评价。
    4. 它不是一个汇报会议，而是开发团队内部的沟通会议，以便快速发现问题。

37. 下面的（ ）是错误的。
    1. 故事点是衡量产品特性规模的定量估算单位
    2. 故事点一般使用直接的小时或人天等时间单位来表示
    3. 故事点估算不需要考虑个人能力因素
    4. 故事点的大小表示开发一个产品特性所需要投入的工作量

38. 在选择开发团队组织结构时应考虑（ ）因素。

1. 沟通的复杂程度
2. 最终程序的规模大小
3. 发布日期的严格程度
4. 项目预算的多少
5. 选项 A、B 和 C
6. 选项 A、B 和 D

39. 民主式结构团队的特点是（ ）。
    1. 开发人员以志愿者形式参加，每个人参与自己感兴趣的项目，完全无人管理。
    2. 以主程序员为核心，团队其他人员的职能进行专业化分工。
    3. 技术经理负责技术决策，项目经理负责非技术性事务的管理决策和绩效评价。
    4. 团队成员完全平等，享有充分的民主，成员之间通过协商做出决策。
40. 经验估算模型是（ ）。
    1. 专家基于过去项目经验的判断
    2. 期望值估计的细化
    3. 来自历史项目数据的回归模型
    4. 反复试验决定参数和系数
41. 在软件配置管理中，基线的目的是（ ）。
    1. 合理控制变更
    2. 合理分配存取权限
    3. 保证配置项的完整与正确
    4. 保证配置项的依赖性
42. 在使用 Git 进行代码文件提交时，如果提示提交内容为空、不能提交，则最为合适的处理方式是（ ）。
    1. 执行 git status 查看状态，再执行 git add 命令选择要提交的文件，然后提交。
    2. 执行 git commit --allow-empty，允许空提交。
    3. 执行 git commit -a ， 提交所有改动。
    4. 执行 git commit --amend 进行修补提交。
43. 如果项目中文件 hello.c 的内容被破坏，执行（ ）使其还原至原始版本。
    1. git reset -- hello.c
    2. git checkout HEAD -- hello.c

3. git revert hello.c
4. git update hello.c

44. 如果只是将工作区中修改的文件添加到暂存区（新增文件不添加）以备提交，使用下面的（ ）命令标记最快。
    1. git add -A
    2. git add -u
    3. git add -p
    4. git add -i

45. 关于 git clone，下面的（ ）是错误的。
    1. 克隆时所有分支均被克隆，但只有 HEAD 指向的分支被检出。
    2. 可以通过 git clone --single-branch 命令实现只克隆一个分支。
    3. 克隆出的工作区中执行 git log、git status、git checkout、git commit 等操作不会去访问远程版本库。
    4. 克隆时只有远程版本库 HEAD 指向的分支被克隆。

46. 软件需求工程师的职责不包括下面的（ ）。
    1. 撰写需求规格说明书
    2. 与用户持续沟通，了解用户对产品的期望
    3. 控制项目的风险
    4. 对需求的优先级进行排序

47. 下面的（ ）描述了“滴滴打车”的业务需求。
    1. 我们的任务是提供叫车与车辆信息服务来解决司机与乘客的问题。
    2. 我们的目标是让客户将我们的品牌和低价格联系在一起。
    3. 我们公司的主营业务是提供在线打车服务，如：出租车、专车、快车等。
    4. 公司的服务必须满足所有租车服务行业监管条例。

48. 下面的（ ）是学生微信抢票系统的非功能性需求。
    1. 组织丰富多样的课外文艺活动
    2. 定期推送让同学们了解最新活动情况
    3. 不支持活动票的转让
    4. 要确保抢票系统的 7/24 可用性

49. 下面的（ ）非功能性需求是面向软件过程的。
    1. 安全性

2. 可靠性
3. 可维护性
4. 用户友好性

50. 获取软件系统需求不包括以下的（ ）来源。

1. 系统相关领域的法律法规
2. 系统的质量控制团队
3. 系统的业务流程描述
4. 其他类似系统产品