

高级软件工程复习题

2024 版

2024/12/27

目录

一、概念解释	3
二、选择题	5
三、判断题	45
四、设计题	52
五、简答题	63

一、概念解释

1. 敏捷开发方法与 Scrum 方法
2. 基于计划-文档开发方法 (Plan-and-Document based Development)
3. DRY (Don't Repeat Yourself) 无重复代码
4. MVC (软件作为服务的开发框架)
5. SMART 用户故事
6. TDD and 红-绿-重构
7. FIRST 测试原则
8. 代码味道及类内方法 SOFA 原则
(说明 S、O、F、A 分别代表什么？违法该原则的代码的不好特征、重构和修复的方法)
9. 类间关系的 SOLID 原则 (说明每个原则的意义、违法该原则的代码的不好特征、重构和修复的方法)
 - (1) 单一责任原则
 - (2) 开闭原则
 - (3) 里氏替换原则
 - (4) 依赖注入原则
 - (5) 迪米特法则
10. 持续集成及开发
11. 文档对象模型 (DOM) 和 jQuery
12. JavaScript 函数特点
13. Ruby 面向对象
14. Ruby 中的数组
15. Ruby 哈希 (Hash)
16. Ruby 迭代器: each 和 collect
17. Ruby 的 Duck Typing
18. Ruby 的 Mix-ins
19. Git 工作流程
20. 分布式版本控制

21. 如何理解软件设计原则中的低内聚高耦合原则？
22. 请简述 RUP 与 UML 的含义，并指明两个的不同点。
23. 请简要描述 TDD 与 BDD 的发展理念，并说明二者的区别。
24. 请说出至少五种你知道的测试方法。
25. 结构化程序设计的主要内容是什么？它有什么优缺点？
26. 软件生命周期划分为哪几个阶段？
27. 什么是螺旋软件开发模型？它有什么特点？
28. Ruby 属于什么类型的语言？它有怎样的特性？有哪些优点？

二、选择题

1. 下面哪个 git 命令用来帮助跟踪谁修改了什么文件和什么时候修改的?
 - A. git list
 - B. git manage
 - C. git push
 - D. git blame
2. 如何比较两个文件或当前文件和以前版本的修订?
 - A. git diff
 - B. git compare
 - C. git clone
 - D. git checkout
3. 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?
 - A. 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。
 - B. 执行 git commit --allow-empty，允许空提交。
 - C. 执行 git commit -a，提交所有改动。
 - D. 执行 git commit --amend 进行修补提交。
4. 如果把项目中文件 hello.c 的内容破坏了，如何使其还原至原始版本?
 - A. git reset -- hello.c
 - B. git checkout HEAD -- hello.c
 - C. git revert hello.c
 - D. git update hello.c
5. 修改的文档 meeting.doc 尚未提交，因为错误地执行了 git reset --hard 导致数据丢失。丢失的数据能找回么?
 - A. 不能。执行硬重置使工作区文件被覆盖，导致数据丢失无法找回。
 - B. 能。可以通过 git checkout HEAD@{1} -- meeting.doc 找回。
 - C. 不确定。如果在重置前执行了 git add 命令将 meeting.doc 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。
 - D. 不能。因为未提交所以无法找回。

6. 仅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，用什么命令标记最快？
- A. git add -A
 - B. git add -p
 - C. git add -i
 - D. git add -u**
7. 下面哪一个命令不会改变提交历史？
- A. git reset --hard HEAD~1
 - B. git checkout HEAD^^ .**
 - C. git rebase -i HEAD^^
 - D. git commit --amend
8. 下面的描述中不属于 Ruby 特性的是？
- A. Ruby 是一种功能强大的面向对象的脚本语言。
 - B. Ruby 遵守 GPL 协议并且是免费的。
 - C. Ruby 具有自动内存管理机制。
 - D. Ruby 是基于 MVC 架构的。**
9. 下面的描述中不属于 Rails 特性的是？
- A. Rails 是一个使用 Ruby 语言写的开源网络应用框架。
 - B. Rails 具有强大的反射机制与后设编程。**
 - C. “不要重复自己”和“约定胜于配置”是 Rails 的设计原则。
 - D. Rails 具有实时映射技术和模板编程技术。
10. Ruby 提供了多种字符串的表示方法，下面哪一种是错误的？
- A. str=['在线学习乐园']**
 - B. str="\n{www.itzcn.com}"
 - C. str=%^t 免费教学和视频\n/
 - D. str="窗内网"
11. 要使用 RVM 选择某个已安装的 Ruby 的版本，应该使用以下哪个命令？
- A. rvm select
 - B. rvm install
 - C. rvm use**
 - D. rvm list

12. 在命令控制台要查看 Ruby 的版本应该使用命令_____。

- A. ruby -v
- B. ruby –version
- C. ruby -h
- D. ruby /?

13. 下列不属于 RubyGems 提供命令的是_____。

- A. gem list
- B. gem install rails
- C. gem –v
- D. gem –update rails

14. 下面关于 Ruby 编译器的使用方法，错误的是_____。

- A. ruby hi.rb
- B. ruby -e 'print "hi" '
- C. ruby –h hi.rb
- D. ruby –c hi.rb

15. Ruby 中所有类的基类为？

- A. Object
- B. BasicObject
- C. BasicClass
- D. ClassObject

16. 下面哪些用户故事描述是满足的 SMART 原则要求的？

- A. 用户能用影片名查找电影信息。
- B. 作为一个用户，我想看前 10 个按价格从低到高排序的影片列表，以便我买到最便宜的电影票。
- C. Rotten Potatoes APP 应该有好的用户界面。
- D. Rotten Potatoes APP 应该有好的用户界面和快的响应时间。

17. 下面哪个是对，就用户故事而言？

- (i) 它们应该描述该应软件是如何使用的。
 - (ii) 它们应该有业务价值。
 - (iii) 它们不必要是可以测试的。
 - (iv) 它们应该在敏捷软件开发生命周期的各阶段都能实施或实现。
- A. i only

- B. i and ii
C. i and iv
D. i, iii, and iv
18. 相对强调过程的场景描述而言，**声明性场景描述**特点是？
A. 有更复杂语句和步骤。
B. 不是 DRY。
C. 更段、易理解和维护。
D. 关注在低层步骤上，这些步骤需要初始化设置和执行测试。
19. 对隐性和显性需求而言，下面哪些说法是对的？
A. 无论隐性和显性需求，你都不能写用户故事。
B. 隐性需求更简洁，而显性需求更啰嗦。
C. 隐性需求是显性需求的逻辑结果，通常对应于综合性测试。
D. 显性需求通常可以用场景化过程性语言描述，而隐性需求通常用声明性语言描述。
20. 为什么很多软件项目失败？
(i) 软件不是用户所要的。
(ii) 延迟交付。
(iii) 软件超出预算。
(iv) 软件演化后容易维护和升级
A. i and ii
B. iii and iv
C. i, ii, and iii
D. i, ii, iii, and iv
21. 下面哪些是表达了 BDD 的不足。
(i) 与客户交互沟通的成本。
(ii) 客户满意不等于该软件有一个好的架构。
(iii) 编写功能代码前，先写测试代码。
(iv) 缺乏相关工具。
A. i and ii
B. iii and iv
C. ii and iii

- D. i, ii, iii, and iv
22. 需求分析最终结果是产生_____。
- A. 项目开发计划
 - B. 可行性分析报告
 - C. 需求规格说明书
 - D. 设计说明书
23. 需求分析中，开发人员要从用户那里解决的最重要的问题是_____。
- A. 让软件做什么
 - B. 要给软件供哪些信息
 - C. 需求软件工作效率怎样
 - D. 让软件具有何种结构
24. 需求规格说明书的内容不应包括对（）的描述。
- A. 主要功能
 - B. 算法的详细过程
 - C. 用户界面的运行环境
 - D. 软件性能
25. 软件需求分析阶段的工作，可以分成 4 个方面：需求获取，需求分析，编写需求规格说明书以及（）
- A. 用户
 - B. 需求评审
 - C. 总结
 - D. 都不正确
26. 在原型法中，开发人员根据（）的需求不断修改原型，直到满足客户要求为止
- A. 用户
 - B. 开发人员
 - C. 系统分析员
 - D. 程序员
27. 需求验证应该从下述几个方面进行验证：
- A. 可靠性、可用性、易用性、重用性

- B. 可维护性、可移植性、可重用性、可测试性
 - C.** 一致性、现实性、完整性、有效性
 - D. 功能性、非功能性
28. 敏捷开发者崇尚的价值点是
- A. 过程和工具胜于个人与团队交流
 - B.** 可工作的软件系统胜于全面的文档
 - C. 合同谈判胜于与客户协作
 - D. 遵循计划胜于响应需求变化
29. 瀑布软件开模型与螺旋软件开发模型的区别?
- A. 瀑布模型包括了大量的计划文档和时间跨度大的可工作软件版本,而螺旋模型小量的计划文档和时间跨度小的可工作软件版本。
 - B.** 瀑布模型从开始就写全所有的需求文档,而螺旋模型按迭代阶段写需求文档。
 - C. 瀑布模型有较长的迭代周期,而螺旋模型有较短和快的迭代周期。
 - D. 瀑布模型的在每个阶段完成后进行保障测试,最后验证阶段包括了验收测试;而螺旋模型的保障测试在每2个月内进行。
30. 哪种类型的代码是最坏的?
- A. 遗留代码 (legacy code)
 - B. 静态代码 (static code)
 - C.** 与期望不一致的短命代码 (unexpectedly short-lived code)
 - D. 优美代码 (beautiful code)
31. “系统中的每一个知识(功能或特性)必须有单一的、无二义和明确的表示。”是对下面的哪条原则的表达?
- A. REST
 - B. SAAS
 - C. SOA
 - D.** DRY
32. 如果一个项目延期后,为什么说增加人手不是一个好主意。因为:
- A. 程序员太贵
 - B. 不能用大团队来承建 SaaS 软件

- C. 交流成本会下降
 - D. 让新程序员上手需要较多时间
33. Model-View-Controller (MVC) 架构/设计模式有什么好处?
- A. 支持多用户存取和更新模型数据，对每个用户提供各自的视图。
 - B. 确保模型与视图有一对一的映射。
 - C. 为了调试目的，提供模型和控制器的调试窗口。
 - D. 像其他设计模式一样，遵循这类架构可以得到更简洁的代码。
34. 下面哪一种提高软件生产效率的方法比较能体现面向服务架构的软件?
- A. 通过简洁达到清晰 (Clarity via conciseness)
 - B. 代码合成 (Synthesis)
 - C. 可重用 (Reuse)
 - D. 自动化和工具 (Automation and Tools)
35. 一个服务直接访问另外一个服务的数据时可能出现不稳定的情况一般是以下哪一种情况的特点之一:
- A. Rails 应用框架
 - B. 面向服务的架构
 - C. 面向对象编程
 - D. 敏捷开发过程
36. 假设有两个 HTTP 请求的 URI 是一样，但调用的方法(GET 与 POST)不同。对于 Rails 路由而言，下面哪个说法是对的。
- A. 两个请求可以对应不同控制器的方法，但不是必须的。
 - B. 两个请求必须对应不同控制器的方法。
 - C. 两个请求必须对应相同控制器的方法。
 - D. Rails 将给该请求抛出一个错误信息
37. 下面的例子中哪个是属于表示层?
- A. Apache HTTP 服务器, Ruby and Rails, MySQL 数据库
 - B. Ruby and Rails, Django, Symfony, Catalyst
 - C. MySQL、PostgreSQL 、 Oracle 数据库
 - D. Microsoft IIS, Apache HTTP 服务器, lighttpd 服务器

38. 假设一个 Web APP 网站要开发一个移动 APP 版软件，若原有网站的架构设计采用了 MVC 架构，改版工作量大的地方是：
- A. models
 - B. views**
 - C. controllers
 - D. schema
39. 假设影评网站 RottenPotatoes.com 要增加'filming location' 属性值到 movie 模型中。修改 MVC 哪个部分的工作量大？
- A. Movie 模型**
 - B. 显示影片信息的 view 页面
 - C. 在控制器中增加新的 movie 实例
 - D. 以上所有都不是
40. 一个 HTTP 请求必须都包含哪两项
- A. CRUD 操作，数据库
 - B. header, cookie
 - C. URL, wildcard (e.g. :id)
 - D. URI, HTTP 请求方法**
41. 下面哪个说法是对的（根据 Ruby 语法要求）？
- (i) 局部变量以 \$开头
 - (ii) 实例变量以 @开头
 - (iii) 类变量以 @@开头
- A. i & iii
 - B. ii & iii**
 - C. i & ii
 - D. i, ii & iii
42. 如果执行下面代码的结果是什么？
- ```
a = SavingsAccount.new(100)
b = SavingsAccount.new(50)
c = SavingsAccount.new(75)
[a,b,c].sort
```
- A. 工作，因为存款余额是数字，可以比较。

- B. 不工作，但传入比较方法后也许可以。
- C. 不工作，但如果在 `SavingsAccount` 定义了`<=>`方法后也许可以。
- D. 不工作，`SavingsAccount` 不是 Ruby 基本类型不可以比较。
43. 当 Ruby 表达式 `foo + bar` 求值执行时，下面哪个说法对？
- A. 作为一个参数传给 `foo` 的方法 `+`。
- B. 作为一个参数传给 `bar` 的方法 `+`。
- C. 出错，因为`'+'` 只定义在字符串和数字对象上。
- D. `foo` 和 `bar` 相加。
44. 下面的哪个场景不应该用 HTTP GET 来实现。
- A. 在一个相片分享的网站上，用户点击一个指定的相册。
- B. 用户点击提交按钮发送一封邮件。
- C. 用户在下拉框中选择搜索查询条件。
- D. 用户在影评网 `RottenPotatoes` 上查看修改个人信息。
45. 下面代码中第 4 行做什么？
- ```
1 class PostsController < ActionController::Base
2   def add_post
3     # ...code to create new blog post...
4     flash[:notice] = "Post added successfully"
5     redirect_to posts_url
6   end
7 end
```
- (i) 写消息到日志文件。
- (ii) 这个方法执行后在 view 中提示。
- (iii) 保存信息直到下一个请到来。
- A. (ii) only
- B. (iii) only
- C. (i) and (iii)
- D. (ii) and (iii)
46. 如果在 controller 方法中设置了一个实例变量，该值可以被什么模块读取。
- A. views 不能读取。

- B. Views 可以用它，但仅限于这次请求处理。
 - C. Views 可以用它，仅限于这次请求和下次请求。
 - D. Views 可以用它，包括这次请求以后的所有请求。
47. 请阅读理解下面的代码

```
(i)
user = User.create(name: "David", age: 20,
                    email:'David@test.com')
(ii)
user = User.new
user.name = 'David'
user.age = 20
user.email = 'David@test.com'
user.save!
(iii)
user = User.create
user.name = 'David'
user.age = 20
user.email = 'David@test.com'
```

下面哪个方式可以正确地把一个用户的信息加入到数据库中？

- A. (i) and (ii)
 - B. (i) and (iii)
 - C. (ii) and (iii)
 - D. (i), (ii), and (iii)
48. 在课程管理系统中增加一个查询条件。除了按原有的 credit 列排序外，还增加一个域：search terms 在勾选框里。目的是让相应的 controller 的方法在 params[:search_terms] 里读到这个选择。参数格式为：

```
params = { :credit => 3, :search_terms => "software_engineering" }
```

请问下面哪种方式可以实现这样的效果？

- A. <input type="text" params="search_terms"/>
 - B. <input type="text" name="search_terms"/>
 - C. <input type="text" id="search_terms"/>
 - D. <input type="text" name="params[search_terms]" />
49. RESTful 路由：假设 index 方法接收到带有上一个请求中显示的参数哈希值的表单提交，那么下列哪种方法将被视为处理返回的文本搜索过的电影列表结果的最简单方法？

- A. 不渲染任何视图;而重定向到 movies index 页面, URI 如下
`/index?search_terms=world&rating[PG]=1`
- B. 渲染当前的 movies index 页, 使用匹配 “world”的结果, 过滤条件为 release date 和 rating "PG", 但仍然使用统一一个 URI /index
- C. 建立新的 controller 方法 MoviesController#search, 建立新的 route match '`/search/:ratings/:search_terms', 'MoviesController#search'` 然后重定向到 `/search/PG/world`
- D. 渲染 XML 响应表示 movies 匹配了“world”而且 ratings 过滤, “G”, “PG”, and “PG-13”
50. 下面哪个说法是错误的?
- A. 使用更多设计模式的软件不一定性能更好.
- B. 精心设计的软件可以发展到模式成为反模式的程度。
- C. 过早地尝试应用设计模式可能与应用它们太晚一样糟糕。
- D. 大多数设计模式特定于编程语言的特定子集。
51. 关于 P&D 维护的哪一项声明 (如果有的话) 是假的?
- A. 维护成本通常超过 P&D 的开发成本
- B. 与 P&D 变更请求相当的 Agile 是用户故事; 等价的变更请求成本估算 是积分; P&D 版本就像迭代一样
- C. Agile 生命周期类似于 P&D 维护生命周期: 增强工作软件产品, 与客户合作与合同谈判, 不断响应变化
- D. 以上所有都属实
52. 软件项目可能过度超出预算或延迟计划的最佳预测因素是:
- A. 项目的范围或规模非常大
- B. 项目使用 P&D 方法而不是 Agile
- C. 项目使用 Agile 方法而不是 P&D
- D. 项目的测试/ QA 由一个单独的团队而不是开发人员完成
53. 鉴于小型软件项目与课堂上提供的大型软件项目的成功率, 我们所涵盖的哪些技术将为我们提供从许多小型项目构建大型项目的最佳机会?
- A. 在计划和文档生命周期的设计阶段开发的软件体系结构
- B. 模型 - 视图 - 控制器设计模式

- C. 装饰设计模式
 - D. 遵循面向服务的体系结构
54. 关于类遵守单一责任原则（SRP）说法正确的是？
- A. 一般而言，我们预计会看到不良凝聚力得分与差的 SOFA 指标之间的相关性
 - B. 低内聚是分离类的可能指标
 - C. 如果一个类尊重 SRP，它的方法可能会尊重 SOFA
 - D. 如果一个类的方法尊重 SOFA，那么该类可能会尊重 SRP
55. OmniAuth 定义了一些应用程序必须提供的 RESTful 端点，以处理与各种第三方的身份验证。要添加新的身份验证提供程序，请创建与该提供程序一起使用的 gem。关于 OmniAuth，以下说法错误的是？
- A. OmniAuth 本身符合 OCP
 - B. 使用 OmniAuth 帮助您的应用程序遵循 OCP（关于第三方身份验证）
 - C. OmniAuth 是模板模式的一个例子
 - D. OmniAuth 是 Strategy pattern 的一个例子
56. 不恰当的亲密设计坏味，有时在单元测试中表现为 Mock Trainwreck，通常表示违反以下哪种 SOLID 原则：
- A. Single Responsibility
 - B. Open/Closed
 - C. Liskov Substitution
 - D. Injection of Dependencies
 - E. Demeter
57. 您希望创建相关对象的一个家族，以便可以互换地配置应用程序，哪种设计模式最适合使用？
- A. 抽象工厂模式
 - B. 组合模式
 - C. 观察者模式
 - D. 模板方法模式
58. 根据开闭原则，_____应该为_____开放但是____时应该关闭。
- A. class; extension; modification

- B. method; modification; polymorphism
 - C. class; delegation; polymorphism
 - D. method; stubbing; overriding
59. 下面关于 JavaScript 中函数的说法正确的是（多选）
- A. 它可以是匿名的
 - B. 它是该语言中的一级对象
 - C. 它可以与其他功能同时执行
 - D. 它可以是 JavaScript 对象中属性的值
 - E. 当没有参数调用时，括号是可选的（与 Ruby 函数一样）
60. 什么情况下可以只在客户端验证全部输入信息（而在服务端不做）
- A. 确认浏览器 javascript 启用
 - B. 确认浏览器 javascript 启用，且表单通过安全链接提交
 - C. 服务器必须总是执行验证，即使客户端有 javascript 验证。
61. 执行后台请求并根据响应更新适当位置的 html 页面的能力依靠的是：
- A. 浏览器内置 javascript 翻译器
 - B. 浏览器使 DOM 对 Javascript 可用
 - C. 一个对 XMLHttpRequest 的 Javascript 绑定
 - D. 浏览器中对 XML 的解析
 - E. 浏览器对 JSON 的解析
62. 最终莎莉意识到，她的网站的用户可能希望创建一个黑名单。用户不希望和他们的黑名单人员匹配。然而，她意识到此功能不是很重要，于是决定在几个月的后面实现它。最后她写了功能测试。莎莉显然没有遵循 FIRST 原则的哪一个？
- A. 快
 - B. 独立
 - C. 可重复
 - D. 自我检查
 - E. 及时
63. 莎莉希望她的网站在每个月的第一个星期二有一个特别的布局，她有以下的控制器和测试代码：

```

# HomeController
def index
  if Time.now.tuesday?
    render 'special_index'
  else
    render 'index'
  end
end

# HomeControllerSpec
it 'should render special template on Tuesdays' do
  get 'index'
  if Time.now.tuesday?
    response.should render_template('special_index')
  else
    response.should render_template('index')
  end
end

```

明显没有遵循的 FIRST 原则是什么?

- A. Fast
 - B. Independent
 - C. Repeatable
 - D. Self-checking
 - E. Timely
64. 以下哪一项保证你有详尽的单元测试一段代码?
- A. 100% C0 (statement) 覆盖率
 - B. 100% C1 (branch) 覆盖率
 - C. 100% C2 (path) 覆盖率
 - D. 以上都不是
65. 为了测试在代码中引入了故意错误, 以查看测试是否中断, 这是哪种测试?
- A. 变异测试
 - B. DU 测试
 - C. 黑盒测试
 - D. 模糊测试
66. 请按测试的代码量和复杂程度由低到高排序?

- A. 单元测试，功能或模块测试，集成测试
B. 集成测试，功能或模块测试，单元测试
C. 单元测试，集成测试，功能或模块测试
D. 功能或模块测试，单元测试，集成测试
67. 关于 TDD 哪一项是错误的?
- A. 视图无法使用 TDD 进行测试
B. 在编写要测试的代码之前编写测试用例
C. 可以将 TDD 与 BDD 结合使用
D. TDD 可以使代码更加可测试，模块化和可读性
68. 当探索一段遗留代码时，下列那些技术有可能会有帮助?
- (i) 创建一个应用的草稿分支
(ii) 观看用户如何使用该软件
(iii) 运行测试集合（如果有的话）
(iv) 检查非正式的设计文档（例如 wikis, 提交日志等）
- A. (iii) and (iv) only
B. (i) and (ii) only
C. (i), (ii), and (iv) only
D. (i), (ii), (iii), and (iv)
69. 在对遗留代码进行高级别的探索时，下列哪个表现可认为是正确的?
- A. 探索指的是在合理的期限内去修正代码中一些小的不够“漂亮的”问题
B. 当你发现了类之间重要的集合时，你将能轻松地为它们建立桩。
C. 你应该创建说明现在应用是如何工作的测试，即使这些测试是用于测试 bug 和一些丑陋的特性。
D. 如果设计文档是可获得的，你可以用他们来建立当前代码工作的“基准情况 ground truth”。
70. 关于鉴定/验收测试，下列哪个说法是正确的?
- A. 因为你还并不能理解类或方法级别的逻辑，所以你只能在集成测试的级别上创建它。
B. 你可以通过写你明知会失败的测试，然后根据观察到的应用的行为来代替测试中的预期结果。

- C. 因为你可以观察一个用户与该应用的交互，这个用户应该写代表他们交互的测试。
- D. 你只能在开发模式中运行鉴定测试。
71. 下列哪个不是准备修正遗留代码的挑战？
- A. 它可能能够在生成环境中运行，但是在开发环境中难以运行
- B. 在测试集合中，可能有失败或过时的测试。
- C. 用户可能并不清楚什么样的改进是最重要的。
- D. 可能存在没有被用户用例引发或者测试覆盖的 bug
72. 对比下面相同代码的两个版本，下面陈述正确的是：
- ```
version 1: # version 2:
def foo(x,y) def foo(x,y)
 if x
 if y
 z()
 end
 else
 if y
 w()
 end
 end
end

def check_y_z(val); z() if val; end
def check_y_w(val); w() if val; end
```
- A. 版本 1 有更高的循环复杂度。
- B. 版本 1 有更多的测试接缝。(通过所谓的接缝(Seam)，可以将具有逻辑的部分与承上启下的代码分离开，然后将单元测试注入到接缝中，形成对有逻辑代码的保护网)
- C. 版本 1 有更低的 ABC 得分
- D. 版本 1 不能通过 100% C2 覆盖测试。
73. 关于重构，下面哪个陈述是正确的？
- A. 重构提高测试覆盖率
- B. 在重构期间，永远不应该有失败的测试
- C. 改进代码结构是基本目标

- D. 重构总是减少代码量
74. 关于方法级的重构，\_\_\_\_\_代码味道是可能存在的在其他三个代码都存在的情况下。
- A. 长方法
  - B. 方法做了不止一件事
  - C. 方法有很多参数
  - D. 方法在抽象层次间来回跳跃**
75. 关于设计评审和会议的哪种表述是错误的？
- A. 旨在利用与会者的智慧来改进软件产品的质量
  - B. 他们提供技术信息交流，对年轻人有很高的教育价值
  - C. 设计评审对演讲者和参与者都是有益的
  - D. 会议成功的关键是 Serving food like Samosas**
76. Git 中回退到某个版本的相关命令为：
- A. git add
  - B. git reset**
  - C. git diff
  - D. git clone
77. 在软件工程中高质量的文档标准是：完整性，一致性和（）
- A. 统一性
  - B. 安全性
  - C. 无二义性**
  - D. 组合性
78. 面向对象中，对象之间的协作是通过（）来实现的。
- A. 依赖
  - B. 泛化
  - C. 消息**
  - D. 关联
79. 软件产品的质量是由（）阶段的工作决定的。
- A. 需求分析
  - B. 软件设计**

- C. 软件构建
  - D. 软件测试
80. 以下的 git 命令，哪一条用于将修改提交到远程版本库（）
- A. git clone
  - B. git add
  - C. git commit
  - D. git push
81. 软件设计中划分模块的一个准则是（）
- A. 低内聚低耦合
  - B. 低内聚高耦合
  - C. 高内聚低耦合
  - D. 高内聚高耦合
82. 以下不属于白盒测试技术的是（）
- A. 逻辑覆盖
  - B. 基本路径测试
  - C. 循环覆盖测试
  - D. 等价类测试
83. 意味着一个操作在不同的类中可以有不同的实现方式。（）
- A. 多态性
  - B. 多继承
  - C. 类的可复用
  - D. 信息隐蔽
84. 下面那个说法不属于设计准则
- A. 提高模块的内聚，降低模块间的耦合
  - B. 降低模块接口的复杂程度
  - C. 模块大小要适中
  - D. 模块要用重用性
85. 下面说法正确的是
- A. 模块的作用域在模块的控制域之内
  - B. 模块的控制域在模块的作用域之内

- C. 模块的作用域和模块的控制域有时相同
  - D. 模块的作用域和模块的控制域都是一种层次结构
86. 若要保证开发出来的软件布局优化，而且能尽早发现设计上的错误，应采用
- A. 自顶向下的开发方法
  - B. 自底向上的开发方法
  - C. 渐增式的开发方法
  - D. 非渐增式的开发方法
87. 需求分析的方法包括
- A. 结构化分析技术、面向对象的分析技术、原型开发技术
  - B. 结构化分析技术、面向对象的分析技术
  - C. 面向对象的分析技术、原型开发技术
  - D. 原型开发技术、结构化分析技术
88. 为了使项目在规定日期内上线，开发团队在项目中使用了效率较慢的暴力算法，该算法的性能能够满足目前的项目需求。但后续随着数据量的增大，该项目就达不到要求的性能表现。你认为上述情况属于：
- A. 一个异常（An exception）
  - B. 一个程序错误（An error）
  - C. 一个开发错误（An fault）
  - D. 一个失败的开发（An failure）
89. 某开发团队开发上线了一个系统 A，由于其他系统可能存在缺陷，导致向 A 提供了有缺陷的输入数据，系统 A 向系统管理员报告了这一情况，并不对这些有缺陷的数据进行处理。上述情况属于：
- A. 一个异常（An exception）
  - B. 一个程序错误（An error）
  - C. 一个开发错误（An fault）
  - D. 一个失败的开发（An failure）
90. 下列哪种属于发布-订阅的架构风格？
- A. 观察者模式
  - B. 策略模式

- C. 工厂模式
  - D. 代理模式
91. 某开发团队在合作开发某一项目时, 成员 A 认为某一功能由成员 B 负责的模块实现比在自己的模块中实现简单, 而成员 B 则认为该功能应该由成员 A 的模块进行实现, 针对上述情况, 你认为他们在软件工程九个重要任务中哪个任务没有合理完成?
- A. 软件需求和应用场景
  - B. 集成部署和系统运营
  - C. 软件架构和应用框架
  - D. 业务建模和数据模型**
92. 下列哪种图例强调的是系统式的建模?
- A. 活动图
  - B. 用例图
  - C. 类图**
  - D. 时序图
93. 下列哪个写法是  $a+x*y$  的显示 send 调用方法
- A.  $a.+(x.*y))$
  - B.  $a.send(:x, +.send(:x, y))$
  - C.  $a.send(:+, x.send(:x, y))$**
  - D.  $x.send(:*, y.send(:+, a))$
94. 根据依赖注入原则(DIP), 下列哪种写法更好
- A. # in view  
`@vips = User.find_vips`
  - B. # in controller**  
`@vips = User.find_vips`
  - C. #in view  
`@vips = User.where('group="VIP")`
  - D. 上述写法效果相同
95. 下列关于观察者模式 (Observer Pattern) 的叙述, 错误的为:
- A. 定义对象间的一种一对多的依赖关系, 当一个对象的状态发生改变时, 所有依赖于它的对象都得到通知并被自动更新。

- B. 一个对象状态改变给其他对象通知的问题，而且要考虑到易用和低耦合，保证高度的协作。
- C. 一个对象（目标对象）的状态发生改变，所有的依赖对象（观察者对象）都将得到通知，进行广播通知。
- D. 当一个对象被修改时，不会自动通知依赖它的对象，需要自行进行调用。
96. 下面匹配正则表达式 `^\d{1,3}\.\w{1,3}\.\d{1,3}\.\w{1,3}$` 的是
- A. "192.168.1.1"  
B. "192.abcd.1.1"  
C. "192.abc.1.abcd"  
D. "192.abc.1."
97. Ruby 采用诗歌模式，在没有歧义的前提下可以尽可能的简化表达，省略不必要的括号，下面调用函数 plus 的方法中，错误的是
- ```
def plus (a,b)
  a+b
end
```
- A. plus(3,5)
B. plus 3, b=5
C. plus b=3, c=5
D. plus a:3, b:5
98. 下面一段代码违背了哪一项原则
- ```
class Report
 def output
 formatter =
 case @format
 when :html
 HtmlFormatter.new(self)
 when :pdf
 PdfFormatter.new(self)
 end
 end
end
```

- A. 开闭原则
  - B. 里氏替换原则
  - C. 接口隔离原则
  - D. 依赖注入原则
99. 假如我们构建了一个选课系统，我们希望新增一个功能，让学生可以对一门课程进行评价、打分，其它学生也可以看到，那么整个系统中改动最多的部分是
- A. 路由
  - B. 数据库
  - C. 控制器
  - D. 视图\页面
100. 下面哪些任务不适用快速迭代开发的软件方式？ (i) 卫星控制系统 (ii) 手术机器人 (iii) 模拟火箭控制的网络游戏 (iv) 电脑的操作系统
- A. i 和 ii
  - B. i, ii 和 iii
  - C. i, ii 和 iv
  - D. 全部
101. "Walking on water and developing software from a specification are easy if both are frozen." - Edward V.Berard 这句话说明了软件工程三大问题中的那一个问题？
- A. 只有解决领域的问题的软件才有价值，因此必须对提出检验其质量的外化和内在标准。
  - B. 只有可升级的软件架构和模块是适应(领域)业务问题不断变化的要求。
  - C. 只有合适的开发管理模式和恰当的软件商业模式才能发挥软件效果和取得效益。
  - D. 无法确定
102. 以下（）不是 SCRUM MASTER 职责
- A. 尽可能提高团队影响力

- B. SCRUM MASTER 是公仆
- C. 负责 SCRUM 价值观与过程的实现
- D. 保护团队不受外来无端影响
103. 遵循测试中的“及时”原则，我们应该()
- A. 一边写测试一边开发 *PTP*
- B. 先写完测试然后再开发 *TDD*
- C. 先开发完然后再测试
- D. 以上三者均可
104. 软件设计中划分模块的原则是()
- A. 高内聚低耦合
- B. 低内聚高耦合
- C. 低内聚低耦合
- D. 高内聚高耦合
105. 问：在用 rails 开发一个项目时，要与页面布局中的链接进行交互式测试，  
测试文件一般应放在 test 的哪个文件夹中？
- A. controllers
- B. fixtures
- C. channels
- D. integration *GOF*
106. 下列模式中，属于行为模式的是()。
- A. 工厂模式
- B. 观察者
- C. 组合模式
- D. 以上都是
107. 下面哪些 UML 图描述系统行为()
- A. 用例图
- B. 类图
- C. 对象图
108. 软件测试的目的是()
- A. 尽可能的发现程序中的错误

- B. 发现并改正程序中的错误
  - C. 设计和执行测试用例
  - D. 诊断程序中的错误
109. 单元测试是在软件开发过程中的哪个阶段完成的？（）
- A. 需求分析
  - B. 概要设计
  - C. 实现/详细设计**
  - D. 使用和维护
110. 在 UML 的类图中，描述整体与部分关系的有（）。
- A. 泛化关系
  - B. 聚合关系**
  - C. 依赖关系
111. 软件测试的目的是（）。
- A. 试验性运行软件
  - B. 发现软件错误**
  - C. 证明软件正确
  - D. 找出软件中全部错误
112. 白箱测试方法重视（）的度量。
- A. 测试覆盖率**
  - B. 测试数据多少
  - C. 测试费用
  - D. 测试周期
113. 以下哪几个是软件的设计原则？
- A. 模块化**
  - B. 抽象**
  - C. 封装**
  - D. 松耦合与高内聚**
  - E. 接口和实现分离**
  - F. 充分性和完整性**
114. 以下设计模式中哪些属于创建型模式？

- A. 抽象工厂模式
- B. 生成器模式
- C. 工厂模式
- D. 原型模式
- E. 单例模式
- F. 适配器模式
- G. 修饰模式
- H. 外观模式
- I. 状态模式

115. 下面哪一个不属于 MVC 模式中的组成?

- A. Model
- B. View
- C. Collection
- D. Controller

116. ruby 是什么模式的编程语言?

- A. 基于逻辑的编程语言
- B. 面向对象编程语言
- C. 函数式语言
- D. 多范型语言

117. FIRST 原则中的 I 代表什么?

- A. informative
- B. imperative
- C. implemented
- D. isolated

118. 为了尽可能分离可能发生变化的代码, 编程时应当针对\_\_\_\_并且尽可能使用\_\_\_\_。

- A. 接口/组合与委派
- B. 实现/继承
- C. 接口/继承
- D. 实现/组合与委派

119. 下列哪一项是人类学习与机器学习软件系统的区别。

- A. 是否需要单元测试
  - B. 是否需要对程序中使用算法的准确率进行测试
  - C. 是否使用人类无法总结和解释的算法
  - D. 算法的实现是否依赖于有标注的数据
120. 下列选项中，哪一项对机器学习的智能程度要求最高：
- A. 识别
  - B. 记忆
  - C. 评价
  - D. 创造
121. 对观察者模式，以下叙述不正确的是（）。
- A. 必须找出所有希望获得通知的对象。
  - B. 所有的观察者对象有相同的接口。
  - C. 如果观察者的类型相同，目标就可以轻易地通知它们。
  - D. 在大多数情况下，观察者负责了解自己观察的是什么，目标需要知道有哪些观察者依赖自己。
122. 下列哪一个 git 命令用于将所有文件提交至暂存区？（）
- A. git add .
  - B. git push
  - C. git commit
  - D. git manage
123. 以下哪一条不是敏捷开发者崇尚的价值观？（）
- A. 流程和工具高于个体和互动
  - B. 工作的软件高于详尽的文档
  - C. 客户合作高于合同谈判
  - D. 响应变化高于遵循计划
124. 在 Ruby 中，类的构造函数是（）方法。
- A. create
  - B. initialize
  - C. new
  - D. class
125. 以下程序输出的结果是（）。

```
myStr = String.new("THIS IS TEST")
foo = myStr.downcase
puts "#{foo}"
```

- A. THIS IS TEST  
B. THISISTEST  
**C.** this is test  
D. This Is Test
126. 下列哪项不是软件需求阶段所要完成的任务？  
A. 需求获取  
**B.** 需求完善  
C. 需求定义  
D. 需求规约
127. 按照（），可以将软件生存周期过程分为基本过程、支持过程和组织过程  
A. 软件开发活动的层次关系  
**B.** 软件开发工作的主体  
C. 软件开发项目的结构  
D. 软件开发任务的重要程度
128. 下述情况分别最适合采取哪种需求发现的方式分别是什么？  
① 为解决生活中遇到的麻烦事而开发的软件  
② 有较多繁琐环节的社区医保系统的开发  
③ 某小型团体组织开发其内部人员管理系统  
④ 某大型连锁集团开发集团人员管理系统  
⑤ 某专业化软件外包公司接手烂尾的软件开发项目  
**A.** ①-自悟；②-观察；③-交流；④-小组会；⑤-提炼  
B. ①-观察；②-自悟；③-小组会；④-交流；⑤-提炼  
C. ①-自悟；②-交流；③-观察；④-提炼；⑤-小组会  
D. ①-提炼；②-自悟；③-交流；④-观察；⑤-小组会
129. 在 Ruby 中，下列关于 proc 和 lambda 的表述正确的是？  
A. lambda 不是 Proc 对象  
B. proc 和 lambda 都不会检查参数数量

- C. proc 和 lambda 对 return 关键字的含义一样
  - D. proc 的 return 只能在方法体中调用
130. 以下哪个不是软件工程的根本问题
- A. 构思创造发明有价值的软件系统或产品和找到检验其质量的理论和方法。
  - B. 寻找项目开发各项资源最优化分配和平衡效益成本的理论和方法。
  - C. 探讨测算软件设计实现和部署运营的效益模型的理论和方法。
  - D. 探寻应对领域问题和软件需求变化的可动态升级软件模块和架构的理论和方法。
131. 以下哪项不是 Ruby 的重要贡献
- A. 一切都是对象。
  - B. 每个操作都是调用某些对象的方法并返回一些值。
  - C. 变量不需要写类型。
  - D. 所有编程都是元编程。
132. 软件生命周期中花费最多的阶段是
- A. 软件编码。
  - B. 详细设计。
  - C. 需求分析。
  - D. 软件测试和维护。
133. ( ) 活动数量较大，占整个维护活动的 50%。
- A. 完善性维护。
  - B. 预防性维护。
  - C. 校正性维护。
  - D. 适应性维护。
134. 瀑布模型存在问题是 ( )
- A. 缺乏灵活性。
  - B. 用户容易参与开发。
  - C. 用户与开发者易沟通。
  - D. 适用可变需求。
135. 为了提高软件的可维护性，在编码阶段应注意 ( )

- A. 保存测试用例和数据
- B. 提高模块的独立性**
- C. 文档的副作用
- D. 养成好的程序设计风格

136. 软件测试的目的是（）

- A. 评价软件的质量
- B. 发现软件的错误**
- C. 找出软件的所有错误
- D. 证明软件是正确的

137. 瀑布模型的存在问题是（）

- A. 用户容易参与开发
- B. 缺乏灵活性**
- C. 用户与开发者易沟通
- D. 适用可变需求

138. 从结构化的瀑布模型看，在它的生命周期中的八个阶段中，下面的几个选项中哪个环节出错，对软件的影响最大（）

- A. 详细设计阶段
- B. 概要设计阶段
- C. 需求分析阶段**
- D. 测试和运行阶段

139. 以下哪一项不是面向对象的特征（）

- A. 多态性
- B. 继承性
- C. 封装性
- D. 过程调用**

140. 一个好的架构应该是（）

- A. 能不依赖于某个技术的实现**
- B. 是针对某一类业务而设计的
- C. 可以得到更简洁的代码
- D. 不需要用专业的业务术语来描述

141. solid 原则说法正确的是 ()

- A. I 代表接口隔离原则 ~~Viskov~~ 里氏替换  
B. 每一个类中可以包含多种相似功能符合 solid 原则  
**C.** 单一职责原则是指每个类应该只有一个职责  
D. 高凝聚是开放封闭原则的内容

142. MVC 架构是什么的简称 ()

- A. Model Vision Controller  
B. Manage View Condense  
C. Manage Vision Condense  
**D.** Model View Controller

143. 关于重构的说法正确的是 ()

- A. 只能在测试之后进行重构  
**B.** 重构可以提高代码的可读性，提高代码质量  
C. 重构不能找出 Bug  
D. 重构总是可以减少代码量

144. 在设计测试用例时，() 是用得最多的一种黑盒测试方法。

- A.** 等价类划分  
B. 边界值分析  
C. 因果图  
D. 判定表

145. 软件详细设计的主要任务是确定每个模块的()

- A. 功能  
B. 外部接口  
**C.** 算法和使用的数据结构  
D. 编程

146. Github 哪一年上线？

- A. 1991 年  
B. 2002 年  
C. 2005 年  
**D.** 2008 年

147. 假设有两个 HTTP 请求的 URI 是一样，但调用的方法(GET 与 POST)不同。对于 Rails 路由而言，下面哪个说法是对的。

- A. 两个请求可以对应不同控制器的方法，但不是必须的。
- B. 两个请求必须对应不同控制器的方法。
- C. 两个请求必须对应相同控制器的方法。
- D. Rails 将给该请求抛出一个错误信息

148. 软件测试的目的是()

- A. 证明软件的正确性
- B. 找出软件系统中存在的所有错误
- C. 证明软件系统中存在错误
- D. 尽可能多的发现软件系统中的错误

149. 使用白盒测试方法时，确定测试数据应根据()和指定的覆盖标准

- A. 程序的内部逻辑
- B. 程序的复杂程度
- C. 该软件的编辑人员
- D. 程序的功能

150. 开发软件所需高成本和产品的低质量之间有着尖锐的矛盾，这种现象称作()

- A. 软件工程
- B. 软件周期
- C. 软件危机
- D. 软件产生

151. 软件设计中划分模块的准则之一是()

- A. 高内聚低耦合
- B. 高耦合低内聚
- C. 高内聚高耦合
- D. 低耦合低内聚

152. 下列哪项不属于 SOLID 原则？()

- A. 单一职责原则
- B. 开放封闭原则

- C. 里氏替换原则
  - D. 依赖拒绝原则
153. 下列选项中属于产品的内部属性的是()。
- A. 模块耦合度
  - B. 软件可靠性
  - C. 软件有效性
  - D. 软件可维护性
154. 下面不属于软件工程的3个要素是()
- A. 工具
  - B. 过程
  - C. 方法
  - D. 环境
155. 设计模式通常不包括()
- A. 创建型模式
  - B. 结构型模式
  - C. 行为型模式
  - D. 操作型模式
156. ruby 变量前有一个@符号，意为：
- A. 实例变量
  - B. 类变量
  - C. 局部变量
  - D. 全局变量
157. 命令 `put :a.class` 得到? (ruby)
- A. Symbol
  - B. Integer
  - C. String
  - D. Object
158. ruby 的函数式语言中，`.map` 的作用是：
- A. 遍历
  - B. 改写

### C. 筛选

159. 下面关于 rake 的说法错误的是:

- A. rake 相当于 C 的 make
- B. rake 利用了 ruby 语言的 DSL 性质，完全用 ruby 语言书写
- C. make 不依赖于 shell**
- D. make 只更新修改过的文件，减少了大量的重复劳动

160. 下面关于 DevOps 描述错误的是: ()

- A. DevOps 突出重视了软件开发人员和客户之间的沟通合作。**
- B. DevOps 的核心是角色的分工，而不是组织架构变化。
- C. DevOps 涉及持续集成、持续部署、持续测试等流程。
- D. DevOps 一个好处就是会改善公司组织文化、提高员工的参与感。

161. 下面哪些用户故事的叙述是满足的 SMART 原则要求的: ()

- A. 用户可以翻译外语。
- B. 作为一个用户，我可以通过输入待翻译句子，选择要翻译的目标语言以及点击翻译按钮来获取翻译后的句子，以便我可以快速知道外语的意思，方便学习外语。**
- C. APP 应该有好的用户界面。
- D. APP 应该有好的用户界面和快的响应时间。

162. 下面哪个选项不是 ruby 的最重要支柱:

- A. 一切都是对象。
- B. 所有编程都是元编程。
- C. 每一个操作都是调用某些对象的方法，并且返回值。
- D. 具有内存管理机制。**

163. 根据下图以及自己对 MVC 模式的理解，当用户点击网页时，正确的代码调用顺序应该是: ()

- A. View>Controller>Model**
- B. Controller>Model>Controller>View
- C. Model>Controller>View
- D. Controller>View>Controller>Model

164. 软件开发的结构化生命周期方法将软件生命周期划分成 ()

- A. 计划阶段.开发阶段.运行阶段
  - B. 计划阶段.编程阶段.测试阶段
  - C. 总体设计.详细设计.编程调试
  - D. 需求分析.功能定义.系统设计
165. 需求分析是（）
- A. 软件开发工作的基础
  - B. 软件生存周期的开始
  - C. 由系统分析员单独完成的
  - D. 由用户自己独立完成的
166. 软件测试的目的是（）
- A. 避免软件中出现错误
  - B. 证明软件功能实现了
  - C. 解决测试中发现的错误
  - D. 发现软件中潜在的错误
167. 一个好的软件架构需要满足（）
- A. 不依赖技术实现
  - B. 用领域知识描述
  - C. 解耦合
  - D. 以上都需要满足
168. 白盒测试是根据程序的（）来设计测试用例
- A. 功能
  - B. 性能
  - C. 内部逻辑
  - D. 内部数据
169. 下面关于测试驱动开发(Test-Driven Development, TDD)的说法，哪一项是错误的？
- A. TDD 可以使代码更加可测试，提高项目模块化和可读性
  - B. TDD 通过减少代码量来提高效率
  - C. TDD 中不同代码的测试应该相互独立
  - D. TDD 在写测试用例之前澄清需求细节

170. 下列过于代码重构的说法正确的是\_\_。

- A. 对于过长的方法 (long method), 可以用 Extract Method (提炼函数) 进行重构。
- B. 代码重构能够提高测试覆盖率。
- C. 在重构期间, 永远不应该有失败的测试。
- D. 重构总是可以减少代码量。

171. 下面关于源代码管理的说法, 错误的是\_\_。

- A. 整个团队的代码托管应在一个平台, 不同的仓库有不同的权限。
- B. 目前发展起来的版本控制系统分为集中式和分布式。
- C. GitHub 不仅支持 Git 格式的版本库托管, 还对 CVS、SVN、Hg 等格式的版本库进行托管。
- D. 文档化和设计过程也通过 git 仓库管理, 且仓库默认有分支保护, 成员无权操作删除任何已托管内容。

172. 螺旋模型共有四个阶段, 这些阶段的顺序应该是 ()

- i. 确定本次迭代的目标和约束
- ii. 开发和验证本次迭代原型
- iii. 计划下一次迭代
- iv. 评估各种备选方案并加以确认, 化解风险

- A. i. iii. ii. iii.
- B. i. ii. iii. iv.
- C. i. ii. iv. iii.
- D. ii. i. iv. iii.

173. 针对接口编程, 不要针对实现编程, 是()的表述

- A. 开闭原则
- B. 接口隔离原则
- C. 里氏代换原则
- D. 依赖倒转原则

174. 在观察者模式中, 表述错误的是 ()

- A. 观察者角色的更新是被动的
- B. 被观察者可以通知观察者进行更新

- C. 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依据被观察者的状态进行
- D. 以上表述全部错误
175. 下列不属于三大关键问题的是哪一项？（）
- A. 只有解决领域业务问题的软件才有价值
- B. 只有可动态升级软件架构和模块是适应领域（业务）问题不断变化的要求
- C. 只有适合的开发管理模式和恰当的软件商业模式才能发挥软件效果和取得效益
- D. 只有使用了敏捷开发并以较快速度交付项目的软件才有价值
176. 以下哪个 Ruby 表达式无法匹配字符串中的 Email()
- A. "My email is 12345@qq.com." =~ /\w+([-.\w+]\*)@\w+\.\w+([-.\w+]\*)/
- B. "My email is 12345@qq.com." =~ /\w+([-.\w+]\*)@\w+([-.\w+]\*)\.\w+([-.\w+]\*)\w+([-.\w+]\*)/
- C. "My email is 12345@qq.com." =~ /\w+([-.\w+]\*)+@\w+([-.\w+]\*)+\.\w+([-.\w+]\*)+/\
- D. "My email is 12345@qq.com." =~ /\w+([-.\w+]\*)@\w+([-.\w+]\*)\.\w+([-.\w+]\*)/
177. String 类包含一个方法，给定一个字符串，判断它不包含单词 curvy 中的字母。该类定义如下：
- ```
class String
  def curvy?
    !("AEFHJKLMNTVWXYZ".include?(self.upcase))
  end
end
```
- 下面那个调用方法 curvy? 是正确的？
- A. String.curvy?("foo")
- B. curvy?("foo")
- C. self.curvy?("foo")
- D. "foo".curvy?
178. 以下哪种测试方法不属于白盒测试技术（）
- A. 基本路径测试

- B. 边界值分析测试
 - C. 循环覆盖测试
 - D. 逻辑覆盖测试
179. 一位软件工程师在网上求助：现在领导让我用测试代码对项目进行验收，请问我应该用编写项目时用过的算法再编写一个测试程序跑一遍吗？这样有什么意义？请问他没有正确理解掌握以下哪个概念？()
- A. 测试驱动开发
 - B. 用户需求
 - C. 软件设计原则
 - D. 老板的想法
180. 甲方要求乙方完成一个疫情期间能用于线上开会的在线视频软件，而且必须要有一定的安全性，即公司局域网外无法连入该软件，因为甲方可能用于商谈公司高层机密。乙方交付后甲方发现在家也能用这个软件。请问乙方在哪个环节出了问题()
- A. 软件架构设计
 - B. 安全问题与防御设计
 - C. 用户体验接口设计
 - D. 需求分析
181. 使得在多个类中能够定义同一个操作或属性名，并在每一个类中有不同的实现的一种方法是()
- A. 继承
 - B. 多态性
 - C. 约束
 - D. 接口
182. 下列关于 git 的操作错误的是：()
- A. 本地创建分支：git branch [分支名]
 - B. 从 github 上克隆项目到本地：git clone [github 源代码地址]
 - C. 将修改的代码提交到暂存区：git commit [文件名]
 - D. 查看本地所有分支：git branch -v
183. 要创建应用程序的 development 和 test 数据库，应使用下面命令()

- A. rails db:create
- B. rakes db:create
- C. rails new db
- D. rake db:create**

184. 软件测试的目的

- A. 评价软件质量
- B. 发现软件中的错误**
- C. 找出软件中的错误
- D. 证明软件是正确的

185. 需求分析是软件生存周期中的一个重要阶段，应该在（）进行。

- A. 维护阶段
- B. 软件开发全过程
- C. 软件定义阶段**
- D. 软件运行阶段

186. 在软件底层进行的测试称为（）

- A. 系统测试
- B. 集成测试
- C. 单元测试**
- D. 功能测试

187. 软件测试的目的是什么？（）

- A. 尽可能多的发现程序中的错误**
- B. 证明程序是正确的
- C. 调试程序
- D. 寻找程序中所有的缺陷

188. Ruby 中所有类的基类为

- A. Object**
- B. BasicObject
- C. BasicClass
- D. ClassObject

189. 以下对于 ruby 的说法正确的是

- A. 所有的编程都是元编程

- B. 一切都是对象
- C. 每一个对象的操作都是调用对象的某些方法并且返回值
- D. 以上都正确

190. 以下说法正确的是

- ①10 对 4 取余可用 `10.modulo(4)` 表示
 - ②ruby 不需要分号
 - ③SOLID 五项原则中 O 表示 Open-minded
- A. ①
 - B. ①②
 - C. ③
 - D. ②③

191. 以下说法正确的是

- ①使用 `attr_accessor` 可以节约代码量
 - ②ruby 中可以用；在同一行写多条语句
 - ③BDD 不可以与 TDD 结合使用
- A. ①
 - B. ①②
 - C. ③
 - D. ②③

192. 以下字符串输出方法正确的是

- A. `name="Yasuo"`
`print name`
- B. `print "Yasuo"`
- C. `name="Yasuo"`
`print name + "666"`

- D. 以上都正确

193. 下列哪个命令是“将本次所有修改提交到远程仓库”。

- A. `git clone`
- B. `git add`
- C. `git commit`
- D. `git push`

194. 下面说法正确的是()

- A. 经过测试没有发现错误说明程序正确
- B. 测试的目标是为了证明程序没有错误
- C. 成功的测试是发现了迄今尚未发现的错误的测试
- D. 成功的测试是没有发现错误的测试

195. 使用程序设计的控制结构导出测试用例的测试方法是()

- A. 黑盒测试
- B. 白盒测试
- C. 边界测试
- D. 系统测试

196. 软件维护困难的最主要原因是()

- A. 费用低
- B. 人员少
- C. 开发方法的缺陷
- D. 得不到用户支持

三、判断题

1. ruby 中一切都是对象。 ✓
2. 软件测试效果的取决如何选择高效的测试用例，以便用尽量少的测试用例覆盖尽可能多的测试情况，从而提高测试效率。 ✓
3. 白盒测试无需考虑模块内部的执行过程和程序结构，只要了解模块的功能即可。 ✗
4. 单一职责原则表示一个类只负责一个功能领域中相应的职责 ✓
5. 可以将 TDD 和 BDD 不可以结合使用。 ✗
6. 依赖注入原则表示将对象所依赖的外部对象以传递的方式注入。 ✓
7. 解耦合可以提高程序性能。 ✗
8. 如果一个项目延期，可以通过增加人手来解决。 ✗
9. BDD 和 TDD 可以结合使用 ✓
10. 在“红-绿-重构”的重构过程中，不应该出现失败的测试用例。 ✗
11. 自动测试代码工具不如人工测试，应当抛弃 ✗
12. 为了能够最快的得到软件投入使用，应该抛弃软件测试以及应用安全设计 ✗
13. Github 于 2008 年上线。 ✓
14. 需求验证应该从一致性、现实性、完整性、有效性这四个方面进行验证。 ✗
15. DU 测试是指在测试代码中故意引入错误，以查看测试是否中断的测试方法。 ✗
16. 需求规格说明书的内容不应包括对算法的详细过程的描述。 ✓
17. Git 是使用 C++ 语言开发的。 ✗
18. 可重用的软件构件在开发时都经过了很严格的测试，本身是无错误的，其构件的可靠性很高。 ✗
19. 模块化就是把程序划分成独立命名且可独立访问的模块，每个模块完成一个子功能。 ✗
20. 软件体系结构风格通过施加于构件上的限制及组成与设计规则来表现构件和构件间的关系。 ✓
21. 面向对象分析是提取和整理用户的需求，并建立问题域精确模型的过程，面向对象设计则是把分析阶段得到的需求转变成符合成本和质量要求的、抽象

的系统实现方案的过程。

- 22. 概要设计通过对系统的结构表示进行细化、得到软件的数据结构和算法。 X
- 23. 软件测试中，代码覆盖率达到 100% 说明该程序已被完全测试了。 X
- 24. 雁群由大雁组成属于聚合关系。 X
- 25. 大雁由翅膀等组成属于聚合关系。 X
- 26. 在系统开发中，要尽量实现一个实体的行为被拓展而不修改其源码。 X
- 27. 在聊天系统开发中，开发人员将对用户输入的判断（图片或文字）功能实现
在图片聊天模型中，是比较规范的开发。 X
- 28. 一个方法在类型 T 的实例上可以执行，它也应该在任何 T 的子类型上可以执
行。 VSP ✓
- 29. 设计代码架构的时候无须分离可能发生变化的代码，可以将多种功能写在同
一个类中。 X
- 30. Ruby 的主要贡献包括：一切都是对象，每一个“操作”都是调用某些对象的方
法，并且返回一个值，所有的编程都是元编程。 X
- 31. Ruby 的每一个操作都是一个方法调用。 X
- 32. 面向方面编程(AOP)不能提炼横切关注点 X
- 33. BDD 在开发之前和开发期间询问有关应用行为的问题，以减少误解 X
- 34. TDD：在代码本身之前，先为代码编写单元和功能测试 X
- 35. SOA 使得重用当前代码来创建新应用程序变得很繁杂 X
- 36. 敏捷强调测试驱动开发(TDD)来减少错误，记录用户故事来验证客户需求，
用速度来量进展 X
- 37. 只要测试过程中程序的每一行代码都执行了，那么测试覆盖率就是 100% X
- 38. 一个 HTTP 请求包含一个 HTTP 方法与 URI X 实践
- 39. Rails 中 Views 与 Controllers 之间的数据交互通过类变量实现。 X
- 40. 在 Class Movie 中声明了 attr_accessor title 后，使用 .title 就不是调用方法
了 X
- 41. 软件即服务，手机上的 app 都应该采用敏捷开发的方式 X
- 42. 一个好的软件设计，类间结构一定是符合 SOLID 原则的。 X

43. 一个好的软件必须有一套好的评判标准，因此测试在软件工程中的地位是极其重要的。
44. 软件实体（包括类、模块、函数等）都应该可扩展，而不用因为扩展而修改实体的内容。这符合 SOLID 中的开闭原则。
45. 好的单元测试是每个测试只关注逻辑的一个方面，这样有利于排错。
46. 模块化，信息隐藏，抽象和逐步求精的软件设计原则有助于得到高内聚，低耦合度的软件产品
47. 在 Ruby 中，self 被认为是默认的 receiver。
48. 在 Ruby 中，实例变量@var 是在 self（当前对象）中查找的。
49. Ruby 没有类型转换的概念，而且每个操作都是一个方法调用。
50. 在一个类中包含一个模块时，Ruby 创建了一个匿名类来封装这个模块，并将这个匿名类插入到祖先链中。
51. 在 Ruby 中，"good"+"day"是调用接收者"good"上的 String#+方法
52. Ruby 的三个重要支柱是：1、一切都是对象；2、每一个操作都是调用某些对象的方法，并且返回一个值；3、所有的编程都是元编程。
53. SOLID“五个首要原则”：单一职责、开闭原则、里氏替换、接口隔离、依赖反转
54. 若有问题：a 依赖 b，而 b 的接口或实现变化，即使功能是稳定的，则 Ruby 解决方法是：抽出这个多功能为独立模块来隔离这个接口。
55. 测试驱动开发 TDD 的原理是在开发功能代码之前，先编写单元测试用例代码，测试代码确定需要编写什么产品代码。先编写一个测试检查所需要的功
能，它失败了，然后我们编写代码、添加功能，最后确认测试能通过。
56. 工厂模式可以屏蔽产品的具体实现，调用者只关心产品的接口。
57. 为了减少编写代码的工程量，我们在设计类时应当使其具有多个目的。
58. 测试时应找到最小的测试用例覆盖所有可能的异常分支，测试代码量应少于用于实现功能的代码量。
59. FIRST 原则中 S 要求测试可以验证它们的结果，无需人工确认。
60. 在设计架构时，责任边界设计越细越好。
61. 开发时第一步是构思项目如何编写。

{
 FAST
 Independent
 Repeatable
 Self-checking
 Timely
}

62. 如果想要检测中测试人员难以发现的缺陷，可以采用模糊测试。
63. 自动回归测试的意思是新增代码在构建成功后执行自动测试。
64. ruby 中一切皆是对象。
65. 从是否关心内部结构来看，软件测试的类型只有白盒测试与黑盒测试。
66. 软件维护的本质是修改和压缩了的软件定义和开发过程。
67. 结构化设计方法一般使用文字来描述用户需求。
68. 软件计算价格时不需要把维护费用记入成本。
69. 需求规约是软件开发组织和用户之间的技术合同书，只有当需求规约完成后才能开始产品的设计。
70. 参与者一般可以表达与系统交互的人、硬件或系统等，因此实质上不是软件应用的一部分。
71. 人机交互的交互内容需求不仅与系统的功能需求有关，而且与人的主观意识也有很大关系。
72. 极限编程是采取必要的手段，充分挖掘软件开发团队人员的极限能力，在最短的时间内交付软件的开发方法。
73. 软件集成测试既可以将所有模块组装到一起然后进行测试，也可以在组装的过程中边连接边测试。
74. 在 ruby 访问控制只有 public、private 这两种
75. 为了测试在仓库中引入了故意错误，以便查看测试是否中断，被称作突变测试。
76. 可重用性比较能体现面向服务架构软件的生产效率。
77. 瀑布开发模型从一开始就写全所有的需求文档，而螺旋开发模型从迭代阶段写需求文档。
78. 需求分析最终的产物是项目开发计划。
79. 不能以工程类比软件工程，软件工程与传统工程的最大不同是几乎全部是智力劳动，产品也不是实体实物。
80. 观察者模式在被观察者和观察者之间建立抽象耦合，被观察者会向所有登记过的观察者发出通知。
81. 单元测试包括模块接口测试、路径测试、数据测试。

82. 为了加快软件维护作业的进度,应尽可能增加维护人员的数目.
83. 完成测试作业后,为了缩短源程序的长度应删除程序中的注解.
84. 当验收测试通过,软件开发就完成了.
85. 软件测试是要发现软件中的所有错误。
86. 在观察者模式中,当一个对象的状态发生改变时,所有依赖于它的对象的状态也会发生改变
被观察者
87. TDD 可以不用编写测试代码就进行功能开发
88. 在观察者模式中耦合的双方是具体依赖 *抽象*
89. 重构总是可以减少代码量
90. 开放封闭原则使实体的行为扩展而不修改其源代码
91. 在程序调试时,找出错误的位置和性质比改正该错误更难.
92. 修改数据表时可以直接在原来的迁移表中修改
93. 与应用代码相比,如果测试代码特别简短,倾向于先编写测试
94. 我们一般先编写控制器和模型测试,然后再编写集成测试
95. 代码 if(line1 =~ /Cats(.*)/) 中, =~ 判断 lines 中是否含有 Cats。
96. 每行 ruby 代码都会返回某个值。
97. HTML 定义了网页的内容, CSS 描述了网页的布局, JavaScript 书写的网页的行为。
98. rvm 有效的隔离了环境,使不同版本的 ruby 可以和谐共存、互不干扰。
99. BDD 根据现有的方法和类构造测试, TDD 只知道系统外部的行为,逐步向内测试
外→内 *内→外*
100. 每一个 Ruby 对象都是类指针和实例变量数组的组合。
101. 可以通过 user.send("hello") 实现方法的调用。
102. TDD 开发方式中,领域专家必需参与。
103. BDD 开发中,领域专家可以不参与。
104. 在面对一个软件开发需求时,只要前期调研结果和设计方案都是可实现的,并且预计收益可观,就可以着手实现。
105. 一个好的软件架构不需要考虑该软件面向的领域具体内容,只需要关注软件实现本身。

106. SOLID 设计理念是实现软件功能正确性的必要条件。
107. 软件模块之间的耦合性越弱越好
108. 测试驱动开发在编写某个功能的代码之前先编写测试代码，然后只编写使测试通过的功能代码，通过测试来推动整个开发的进行。
109. SOLID 不能使程序员更容易创造一个容易维护和随时间扩展的系统
110. 执行 bundle exec 后，可以让 ruby 使用 GEMFILE 中声明的对应版本的 gem 包。
111. ruby 中， $5+3$ 若改写为显式的 send 方式，可改为：5.send(:+, 3)。
112. ruby 中， attr_accessor 是元编程的一个实例，它会在运行时创建新的代码，使得实例变量可以执行 getter 和 setter 的操作。
113. 在一个 rails 项目中， Rakefile 文件用来保存项目运行时生成的日志文件以及 gem 信息。
114. 来自用户的 Web 请求由控制器处理， app/controllers 子目录是 Rails 查找控制器类的地方。
115. 观察者模式定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。
116. @@开头的是全局变量。
117. Ruby 遵守闭源商业协议。
118. 测试驱动开发最大的优点就是重构，不断迭代，不断地对现有代码进行重构，不断优化代码的内部结构，最终实现对整体代码的改进；
119. 测试驱动开发是现在(2020)应用最广泛的开发方式；
120. 软件的最终产品和递交物是可运行的应用系统(含软件和硬件)和相应的业务数据。
121. Ruby 是解释型语言，不需要编译，修改后马上就可以执行
122. 白盒测试法是将程序看成一个透明的盒子，不需要了解程序的内部结构和处理过程。
123. 如果通过软件测试没有发现错误，则说明软件是正确的。
124. ruby 可以修改内置的类
125. 通过代码重构，永远不会出现错误的测试

- 126. BDD 使用 assert 来测试功能
- 127. 使用更多设计模式的软件性能一定更好
- 128. GitHub 只支持 Git 格式的版本库托管，而不对 CVS、SVN、Hg 等格式的版本库进行托管。
- 129. Java 语言是强类型和静态类型，C 语言是弱类型，Ruby 语言是动态类型。
- 130. 软件的可维护性差是软件维护工作量和费用激增的直接原因。
- 131. 集成测试主要由用户来完成。
- 132. 软件错误可能出现在开发过程的早期，错误越早被修改越好。

3题 15

四、设计题

为下面的功能写一个可信的用户故事和两个场景，并确保你的答案足够 smart:

功能 1: 通过上课时间搜索课程 以便于我能快速找到并选择我适合的课程

用户故事: 作为一个学生, 我希望通过输入上课时间搜索到符合条件的课程

场景 1: 学生通过点击网页中的上课时间如周一~一节, 系统将自动将所有课程信息

场景 2: 键入时间关键字, 如周一, 将展示 All info. 按首字母排序展示
搜索框

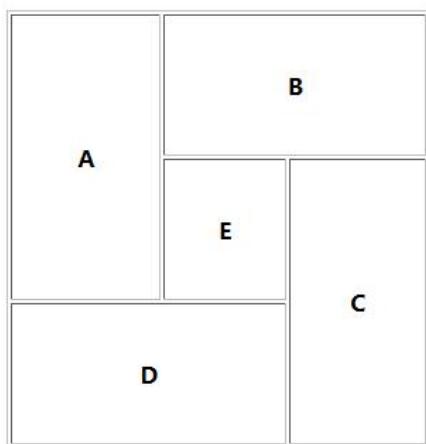
功能 2: 根据学生的成绩来进行排序 以便于

用户故事: 作为一名老师, 我想要根据学生成绩升序、降序排序, 这样我促进教学方法

场景 1: 升序

场景 2: 降序

2. 为了方便同学们找到上课教室的具体位置, 在选课系统中加入教室的分布图, 编写出实现如图所示页面效果的关键 html 代码。其中, A、B、C、D、E 均为默认字号和默认字体, 并且加粗显示, 它们都位于各自单元格的正中间, A 单元格的高度为 200 像素, B 单元格的高度为 100 像素, C 单元格的宽度为 100 像素, 高度为 200 像素。



3. 以下代码实现了一个选择框, 请修改代码使点击相应选项时网页出现弹框提示显示文本和值。

```

<!DOCTYPE html>
<html>
<body>
    <select onchange="_sel(this)">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
        <option value="opel">Opel</option>
        <option value="audi">Audi</option>
    </select>
</body>
</html>

```



4. 根据以下功能要求写出代码。

(a) `@users.sort_by! { |user| user.name }`
 @users.sort!{|a,b| a.name <=> b.name}

- 1) 假设我们有一个对象数组@users，我们需要让他对字段 name 排序，应该怎么做？如果在 ActiveRecord 中，又该如何做？(a) `users.order(:name)`
- 2) 用户与课程为多对多的关联关系，假设相应的字段已经具备，如何补全 Course 模型和 User 模型？

```

class User < ActiveRecord::Base
  has-many-and-belong-to :courses
end
has-and-belong-to-many :courses

class Course < ActiveRecord::Base
  has-many-and-belong-to :users
end
:users

```

- 3) 在用户模型 User 中新增一条用户记录并保存，具体信息姓名 David, 年龄 20, email:David@test.com, 密码 123456。
`user = User.create(name: "David", age: 20, email: "xxx", pw: "123456")`
- 4) 在 User 模型中找到姓名为 David 的用户，并更新用户的密码为 66666。

5) 找到 User 模型中最后一个用户，并将他删除(提示：代码需要有鲁棒性)。

`user = User.find_by(name: "David") / where
 user.update_attribute(:password, "66666") / user.save!`

- 5) 现有选课系统学生类的代码如下，请说明以下代码违背了 SOLID 原则中的哪一原则，并对代码进行修改，使其符合 SOLID 原则。单一职责

15) `user = User.last`
`if !user.nil? then`
 `user.destroy`
`end`

```

class Grade
  attr_reader :name, :home_work, :test, :paper
  def initialize(name)
    @name = name
  end
  class Student
    attr_accessor :first_term_home_work, :first_term_test, :first_term_paper
    attr_accessor :second_term_home_work, :second_term_test, :second_term_paper
  end
  def grade
    cal
  end
end

```

class student
class Student @ terms [Grade.new(:first), Grade.new(:second)]

```

def first_term_grade
  (first_term_home_work+first_term_test+first_term_paper)/3
end

def second_term_grade
  (second_term_home_work+second_term_test+second_term_paper)/3
end

```

def term
reference @ terms.find{|term| term.name == reference}

4. 补全以下代码，测试以下功能：

- 1) 一个学生用户在修改密码之后，新的 token 与原来的 token 不同；
- 2) 修改密码的时间
- 3) 发送修改密码的提醒邮件给用户

```

describe User do
  describe "#send_password_reset" do
    let(:user) { Factory(:user) }

    it "generates a unique password_reset_token each time" do
      user.send_password_reset
      last_token = user.password_reset_token
      user.send_password_reset
      ____(1)____ user.password_reset_token.should_not eq(last_token)
    end

    it "saves the time the password reset was sent" do
      user.send_password_reset
      ____(2)____ user.password_reset_sent_at.should be-present
    end
    it "delivers email to user" do
      user.send_password_reset
      last_email.to.should include(___(3)__)
    end
  end
end

```

user-email
(reload)

7. 请参考以下实现方式，修改为一组更优雅的、不依赖底层实现的设计，使得在增加一个 jsonprinter 时，无需修改原有代码中的 class printer 函数即可实现

增加一个格式的打印功能（不依赖于底层）

原来的实现方法

```
2  class Printer
3    def initialize(data)
4      @data = data
5    end
6
7    def print_pdf
8      PdfFormatter.new.format(@data)
9    end
10
11   def print_html
12     HtmlFormatter.new.format(@data)
13   end
14 end
15
16 class PdfFormatter
17   def format(data)
18     # format data to Pdf logic
19   end
20 end
21
22 class HtmlFormatter
23   def format(data)
24     # format data to Html logic
25   end
26 end
```

修改后的实现方法

```
class Printer
  def initialize(data)
    @data = data
  end

  def print(formatter: PdfFormatter.new)
    formatter.format(@data)
  end

end

class PdfFormatter
  def format(data)
    # format data to Pdf logic
  end
end

class HtmlFormatter
  def format(data)
    # format data to Html logic
  end
end
```



8. 现在要对用户表中的password进行数据验证，保证此字段非空，而且长度不小于6个字母，请使用validates方法补全：

```
class User < ActiveRecord::Base
  validates :password, presence: true, length: { minimum: 6 }
end
```

9. 给出如下一段 Ruby 代码，回答下面问题

```

1 class Customer
2   @@no_of_customers=0
3   def initialize(is_vip, name, price_list)
4     @cust_is_vip=is_vip
5     @cust_name=name
6     @goods_price_list=price_list
7   end
8   def display_details()
9     puts "Customer name: #{@cust_name}"
10    @a = @cust_is_vip?"Yes":"No"
11    puts "Customer is vip: #{@a}"
12    @b = @goods_price_list.sum
13    puts "goods_total_price: #{@b}"
14  end
15  def total_no_of_customers()
16    @@no_of_customers += 1
17    puts "Total number of customers: #{@no_of_customers}"
18  end
19  def Func(target, nums = @goods_price_list)
20    hash = Hash.new
21    nums.each.with_index do |num, index|
22      find_this = target - num
23      if hash.include? find_this
24        @ans_list = Array([hash[find_this] + 1, index + 1])
25        puts "#{@ans_list}"
26      else
27        hash[num] = index
28      end
29    end
30  end
31 end
32
33 class Vip_Customer < Customer
34 end
35
36 cust1=Vip_Customer.new(true, "John", [1,2,3,4])
37 cust2=Customer.new(false, "Poul", [5,6,7,8])
38
39 cust1.display_details()
40 cust1.total_no_of_customers()
41 cust2.display_details()
42 cust2.total_no_of_customers()

```

- (1) 理解上述代码片段，给出程序的输出结果
- (2) 说明代码中 19-30 行 func 函数的功能，并给出当调用 cust1.func(5)时函数的输出结果。该函数在 nums 上有一个明显的缺陷，请说出这个缺陷。
- (3) 在 Vip_Customer 子类中设计一个函数，使得当顾客为 vip 时，无需额外函数调用，使得该顾客 good_price_list 中的价格均打 9 折。

10. 补齐下方代码，以让 Rectangle 类支持多重初始化方法。

```
class Rectangle
```

```
def initialize(*args)
```

```
if args.size<2 || args.size>3
  puts 'Sorry. This method takes either 2 or 3 arguments.'
else
  puts 'Correct number of arguments.'
end
end
end

Rectangle.new([10, 23], 4, 10)
Rectangle.new([10, 23], [14, 13])
```

11. 编写一个 Ruby 观察者模式的代码，并给出预期的输出结果。

12. 下图给出了一个简单的登录界面，请给出该界面 html 部分的代码（只需要给出 form 部分）

欢迎登录！

用户名:

密码:

13. 设计一个 customer 类，包含析构函数、输出顾客信息函数、计算顾客总人数的方法，以及访问顾客 id 的访问器。

14. 使用 Ruby 编程，生成 n 个 0 到 n-1 之间的随机整数，找出它们之中的最大者和最小者，并统计大于 n/2 的整数个数。

要求：输入 n，输出最大值 max，最小值 min，大于 n/2 的整数个数 num。

```
a = []
sum = 0
n = gets
```

```

0.upto(n-1) {|i|
j = rand(i).to_i
a << j
sum +=1 if j > (n/2)
}
max = a.max
min = a.min
p a
puts max,min,sum

```

15. 2020 年新冠肺炎疫情突发，为了加强全国的疫情防控，加强个人的健康管理，拟设计一款软件辅助管理。如果你是甲方，你会给出怎样的设计需求？作为收到设计需求的乙方，你准备用什么样的方法解决这些需求？请给出你的设计。我们学校也有这样的疫情防控登记小程序，你觉得有什么可以改进的地方吗？

16. 使用观察者模式写出一些小男孩看电视的 ruby 代码，电视打开时会提示“TV is on!”并通知小男孩。小男孩接收到电视已经打开的通知后，便会宣布“I will go to watching TV!”

require 'observer' 导入模块/库

```

class TV
  include Observable
  # 包含模块
  # 具备被观察的能力
  def turn-on
    puts "TV is on"
    changed # TV state change
    # 改变状态
    notify_observers(self) ← TV 对象
  end
end

```

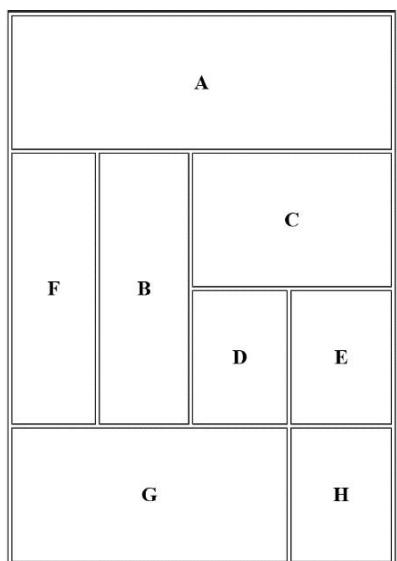
58

class Boy

```

def update(tv)
  puts "I will go to watch TV !"

```



`tv = TV.new`
`boy = Boy.new`
`tv.addObserver(boy)`
`tv.turnOn`

18. 基于 Rails 实现的选课系统中，用户通过点击「查看已选课表」按钮查看已选课数据，请画图并描述这一过程中的数据传输。(对 Rails 框架以及 MVC 结构的考察)。

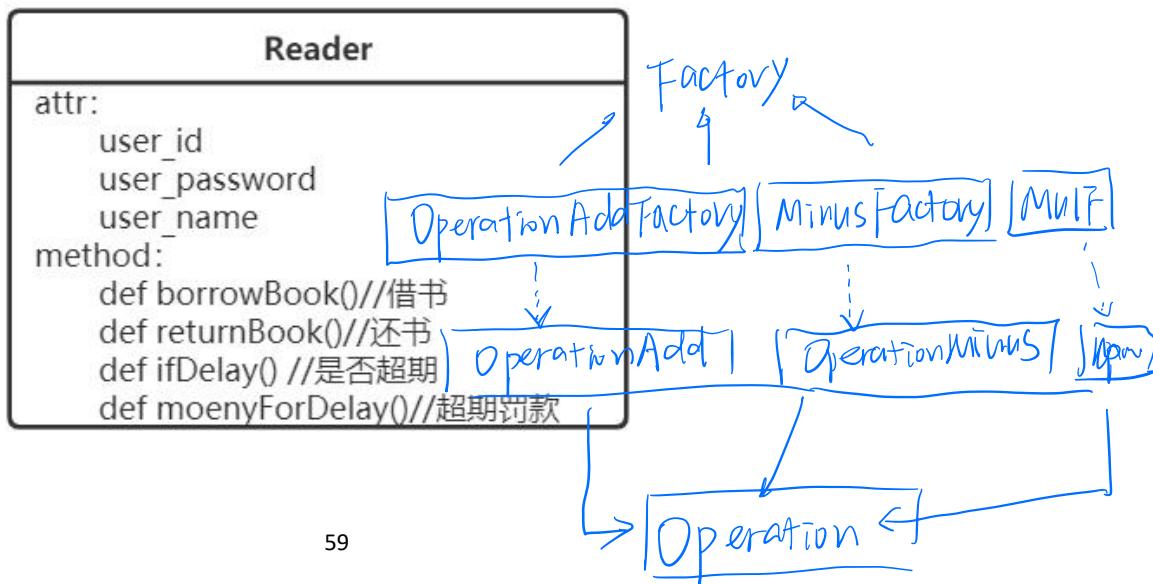
19. 工厂模式是最常用的设计模式之一，属于创建型模式，提供了一种创建对象的最佳方式。请完成以下问题：

- (1) 请完成简单工厂模式和工厂方法模式的结构图：以实现计算器为例。至少包含加减乘的操作

- (2) 请根据 SOLID 原则分析简单工厂模式和工厂方法模式

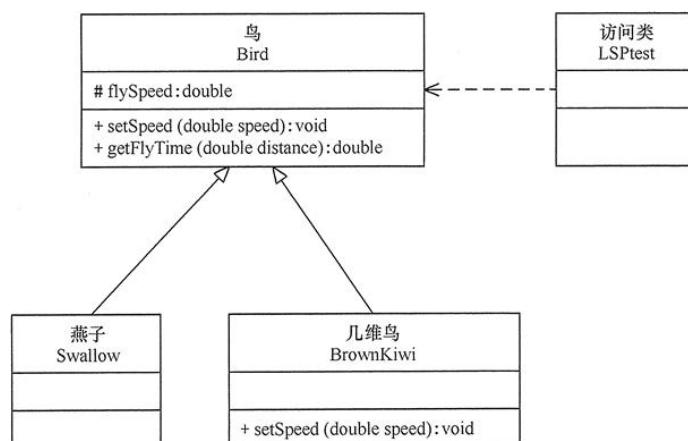
✓单一职责 不违反开闭原则

20. 下图是图书管理系统中读者的接口类，请分析该类是否违背了 SOLID 的原则，如果有，哪些违背了，哪些没违背，应该如何改正：



21. 假定教务管理包括以课程为中心进行资源(教师、教室、学生)配置，并根据各科考试成绩进行教学分析。在这一假定下，结合实际情况，充分考虑教务管理系统的需求后，试作出该系统的类图。

22. 鸟一般都会飞行，如燕子的飞行速度大概是每小时 120 千米。但是新西兰的几维鸟由于翅膀退化无法飞行。假如要设计一个实例，计算这两种鸟飞行 300 千米要花费的时间。显然，拿燕子来测试这段代码，结果正确，能计算出所需要的时间；但拿几维鸟来测试，结果会发生“除零异常”或是“无穷大”，明显不符合预期，其类图如下图所示：



请在遵循 SOLID 原则之一——LSP 的情况下，修改上述 UML 图并画出来。

23. 仔细查看以下代码，说说这段代码的用户场景，并分析其违背了 SOLID 原则中的哪一原则，试对其进行修改使符合 SOLID 原则。

```
class CoffeeMachineInterface
    def select_drink_type
        puts "select drink type"
    end
```

```
def select_portion
    puts "select portion"
end
```

```

def select_suger_amount
    puts "select suger"
end

def brew_coffee
    puts "brew coffee"
end

def clean_coffee_machine
    puts "clean coffee machine"
end

def fill_coffee_beans
    puts "fill coffee beans"
end

def fill_water_supply
    puts "fill water"
end

def fill_sugar_supply
    puts "fill sugar"
end

class Person
    def initialize
        @coffee_machine = CoffeeMachineInterface.new
    end

```

```

def make_coffee
    @coffee_machine.select_drink_type
    @coffee_machine.select_portion
    @coffee_machine.select_suger_amount
    @coffee_machine.brew_coffee
end

class staff
    def initialize
        @coffee_machine = CoffeeMachineInterface.new
    end

    def serv
        @coffee_machine.clean_coffee_machine
        @coffee_machine.fill_coffee_beans
        @coffee_machine.fill_water_supply
        @coffee_machine.fill_sugar_supply
    end
end

```

24. 某企业为了加强进出企业园区人员管理，希望设计一个智能管理系统，减轻保安人员的压力。每个职工在申请后凭个人的工卡就能实现进出园区大门、乘坐班车等功能。请你分析用户需求，并给出一个可行的方案，最好能够用上设计原则和设计模式。

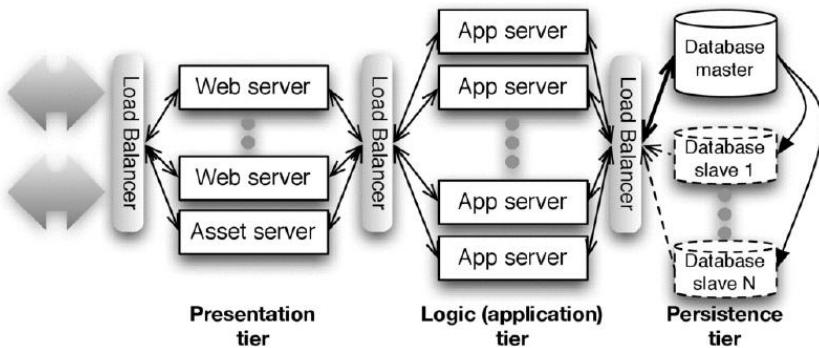
五、简答题

1. 现有如下输出 Fibonacci 序列的两种方法，请再给出一种改进方法。

```
# fib1: Naïve recursive algorithm
def fib1 n
  if n < 3
    1
  else
    fib2(n-1) + fib2(n-2)
  end
end

# fib2: Memoized recursive algorithm
$fib2 = [0,1]
def fib2 n
  $fib2.push($fib2[-1] + $fib2[-2]) until $fib2[n]
  $fib2[n]
end
```

2. 请从结合具体的代码从至少三个维度说明 Ruby 的类型。
3. 学生选课系统作为一种现代化的教学技术，越来越受到人们的重视。学生对选课系统的使用大多数在于选课及查询。如果让你来测试选课系统站内的搜索系统，请问你能想到从哪些方面来进行测试？
4. 结合 Cloud9 或者腾讯开发云的例子说明。
- 1) 相比传统自建数据中心而言，为什么云计算能以更低费用提供更多的计算资源？
 - 2) 采用云计算，任何一个应用都能“无限地横向扩展”吗？为何可以或不可以？
5. 三层无共享（Shared-Nothing）架构的解释（参照下面的图）。



- 1) 哪三层？为何叫“无共享”？
- 2) 该架构为何比较适合大规模的 SaaS 应用？
- 3) 相比其他层，哪一层是不容易横向扩展？为什么？

6. 软件作为服务（SaaS）

- 1) SaaS 与传统软件有什么不同？
- 2) 什么类型的软件不适合做成 SaaS？为何？

7. 考虑如下的应用场景。高级软件工程课的老师希望开发一个软件，希望设计一个学生课程项目的统计信息列表，列表包含了每个学生参与项目的关键信息。从三个独立使用网站获取，即，CodeClimate 网站的代码质量的 GPA、Travis CI 的测试覆盖率和 GitHub 代码修改情况等信息。所有这些信息都可以从这些相关网站提供的 API 中读取，并以 All JSON 对象方式保存。设计者想从下面的两个方案中选取一个：

- 1) 单一页面的 AJAX 应用：把主页 HTML 作为应用服务器，每个表单的单元格设计一个 AJAX 调用，动态从每个数据源获取数据。
- 2) 传统 Rails 应用：首先由 Rails 应用从三个数据源网站获取数据，然后，用户访问 HTML 页面，而不需要动态调用 AJAX。

请点评两种方法各自的优点和缺点。考虑的因素包括：相应时间、利用缓存、数据源限制每个 IP 的 API 访问次数、可扩展性（增加数据源）。

8. 给出下面这段代码的执行结果。

```

reg1=/hay/ =~ 'haystack'
reg2=y.match('haystack')

class Class
  alias old_new new
  def new(*args)
    print "Creating a new ", self.name, "\n"
    old_new(*args)
  end
end

class MyName
end

```

MyName.new

Creating a new MyName

9. 给出下面这段代码的执行结果。

```

class Interpreter
  def do_a() print "there, "; end
  def do_d() print "Hello "; end
  def do_e() print "!\\n"; end
  def do_v() print "Dave"; end
  Dispatcher = {
    "a" => instance_method(:do_a),
    "d" => instance_method(:do_d),
    "e" => instance_method(:do_e),
    "v" => instance_method(:do_v),
  }
  def interpret(string)
    string.each_char{|b| Dispatcher[b].bind(self).call}
  end
end
interpreter = Interpreter.new
interpreter.interpret('dave')

```

Hello there, Dave!

10. 下面程序段的输出是什么？

```
a = 0  
a += 1 if a.zero?  
p a
```

```
a = 0  
a += 1 unless a.zero?  
p a
```

1. 下面程序段的输出是什么？

```
selected = []  
0.upto 5 do |value|  
  selected << value if value==2..value==2  
end  
p selected
```

[2]

```
selected = []  
0.upto 5 do |value|  
  selected << value if value==2..value==2  
end  
p selected
```

[2, 3, 4, 5]

2. 写出下面程序执行后的结果。

```
class MegaGreeter
  attr_accessor :names

  # Create the object
  def initialize(names = "World")
    @names = names
  end

  # Say hi to everybody
  def say_hi
    if @names.nil?
      puts "..."
    elsif @names.respond_to?("each")
      # @names is a list of some kind, iterate!
      @names.each do |name|
        puts "Hello #{name}!"
      end
    else
      puts "Hello #{@names}!"
    end
  end

  # Say bye to everybody
  def say_bye
    if @names.nil?
      puts "..."
    elsif @names.respond_to?("join")
      # Join the list elements with commas
      puts "Goodbye #{@names.join(", ")}. Come back soon!"
    else
      puts "Goodbye #{@names}. Come back soon!"
    end
  end
end
```

```

if __FILE__ == $0
  mg = MegaGreeter.new
  mg.say_hi
  mg.say_bye

  # Change name to be "Zeke"
  mg.names = "Zeke"
  mg.say_hi
  mg.say_bye

  # Change the name to an array of names
  mg.names = ["Albert", "Brenda", "Charles",
              "Dave", "Engelbert"]
  mg.say_hi
  mg.say_bye

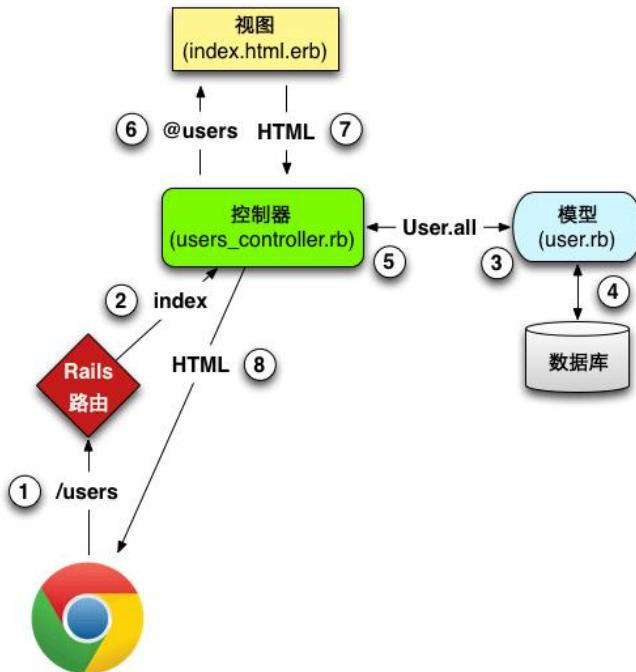
  # Change to nil
  mg.names = nil
  mg.say_hi
  mg.say_bye
end

```

13 在#后面填入该变量名的作用域（全局、局部、实例、类）。

price	#	全局	变量
@price	#	局部	变量
@@price	#	类	变量
\$price	#	全局	变量

14 根据以下的架构图和 Rails 应用框架的原理，填写下面 8 个步骤的缺失内容。



已知 users 对应于数据库中的一个表（table）；id、name 和 email 是表中的列（column）；

Users 资源中页面和 URL 的对应关系		
URL	动作	作用
/users	index	列出所有用户
/users/1	show	显示 ID 为 1 的用户
/users/new	new	创建新用户
/users/1/edit	edit	编辑 ID 为 1 的用户

当用户发出一个请求: <http://example.com/users> 后的如下步骤是:

1. 浏览器向 /users 发送请求；
2. Rails 的路由把 /users 交给 UsersController 的 index 动作处理；
3. index 动作要求 User 模型检索所有用户 User.all；
4. User 模型从数据库中读取所有用户；
5. User 模型把所有用户组成的列表返回给控制器；
6. 控制器把所有用户赋值给 @users 变量，然后传入 index 视图；
7. 视图使用嵌入式 Ruby 把页面渲染成 HTML；
8. 控制器把 HTML 送回浏览器。

15. 已知一个用户拥有多篇微博，一篇微博属于一个用户，数据模型 ER 图和相应代码如下：

microposts			users		
id	content	user_id	id	name	email
1	First post!	1	1	Michael Hartl	mhartl@example.com
2	Second post	1	2	Foo Bar	foo@bar.com
3	Another post	2			

```
# app/models/user.rb
class User < ApplicationRecord
  has_many :microposts
end

# app/models/micropost.rb
class Micropost < ApplicationRecord
  belongs_to :user
  validates :content, length: { maximum: 140 }
end
```

Validate :name, presence: true
presence = "微博内容不能为空"

请在上面的两个代码段的合适地方加入语句，（1）为 User 模型的 name 和 email 属性添加存在性验证；（2）确保微博的内容不能为空存在性验证。

16. 为什么要测试，什么时候测试以及如何测试。

17. 用 Ruby 写一个类和相应方法，统计用该类创建了多少个实例对象？调用方式如下：

```
p1=Person.new("John",27) #参数: name, age
p2=Person.new("Jane",37)

puts Person.how_many
```

18. 生成一个关于用户的数据迁移文件，字段有 name(string),age(int)和 email(string)，请写出 rails 的生成指令为：

\$ rails generate migration CreateUsers name:string age:int email:string

19. 现在若要对用户的字段进行扩充，新增加一个字段 password(string)，请补全新的数据迁移文件的内容：

```
class AddPasswordsToUsers < ActiveRecord::Migration
  def change
    #在此加入语句
    end
    add-column :users, password, :string
end
```

20. 结合上 18、19 两题，请简要叙述为什么在修改数据表时需要新建立数据迁移文件，而不是直接去原来的迁移表中修改？

- ~~21.~~ 在 config/routes.rb 使用了语句 resources :photos 创建关于用户的路由，并执行了 rake routes。请写在下表中的下划线上补齐出 rails 将会创建的 7 种标准的 Restful 路径(包括 HTTP 的方法)以及对应控制器和动作：

HTTP 方法	路径	控制器#动作
GET	/photos	photos#index
GET	/photos/new	photos#new
<u>POST</u>	/photos	photos#create
GET	/photos/:id	photos#show
GET	/photos/:id/edit	photos#edit
PATCH/PUT	/photos/:id	photos#update
<u>DELETE</u>	/photos/:id	photos#destroy

22. 现在要对用户表中的 password 进行数据验证，保证此字段非空，而且长度不小于 6 个字母，请使用 validates 方法补全：

```
class User < ActiveRecord::Base
  validates :password, presence: true, length: {minimum: 6}
end
```

23. 根据以下代码，请叙述 render 和 redirect_to 的效果分别是什么：

```

class ClientsController < ApplicationController
  def create
    @client = Client.new(params[:client])
    if @client.save
      redirect_to @client
    else
      render "new"
    end
  end
end

```

重定向到 show 页面
回到创建页面

24. 现在需要在视图文件`index.html.erb`中列举所有的书信息和后面的相关操作，请完成以下代码的下划线部分。

```

<h1>Listing Books</h1>


| Title                  | Summary             |                      |                      |
|------------------------|---------------------|----------------------|----------------------|
| <%= book.title %>      | <%= book.content %> | <a href="#">Show</a> | <a href="#">Edit</a> |
| <a href="#">Remove</a> |                     |                      |                      |

New book

```

25. 构造一个表单，提交用户的 name 字段和 email 信息，在视图文件`new.html.erb`完成以下代码中的下划线部分。

```
<%= __ @user do |f| %>
<%= __ :name %>
<%= f.email_field __ %>
<%= __ '提交' %>
<% end %>
```

26. 完成以下视图(控制器)测试文件'site_layout_test.rb'

```
class SiteLayoutTest < ActionDispatch::IntegrationTest
# test "the truth" do
#   assert true
# end

test "homepage layout links" do
  get root_path
  assert_template 'homes/index' # 判断目前渲染的视图
  assert_select "a[href=?]", root_path, count: 2
  # 判断当前视图下root_path这个link的个数
end
end
```

27. 完成以下模型测试文件 user_test.rb

```
class UserTest < ActiveSupport::TestCase
# test "the truth" do
#   assert true
# end

def setup
@user = User.new(name: "Example User", email: "user@example.com",
                 password: "password", password_confirmation: "password")
end

test "@user should be valid" do
  assert @user.valid?
end

test "name should be present" do
  @user.name = ""
  assert_not @user.valid?
end
end
```

28. 下面 3 个问题是基于如下代码。写了三个候选方法，返回电影的片名和发行年份，例如，“Star Wars - 1977”。假定电影发行日期是有效的，且 Ruby Time 实例返回年份是整数。

```
class Movie < ActiveRecord::Base
  # reminder: database columns are id, title, rating, release_date
  # method 1
  def title_with_year_1
    "#{self.title} - #{self.release_date.year}"
  end

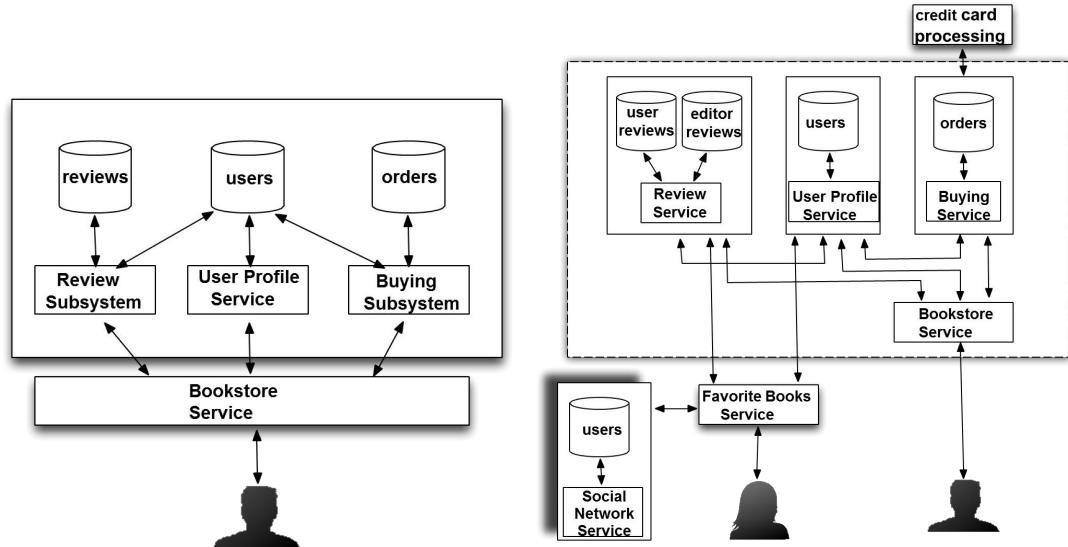
  # method 2
  def title_with_year_2
    "#{title} - #{release_date.year}"
  end

  # method 3
  def title_with_year_3
    "#{@title} - #{@release_date.year}"
  end
end
```

- (1) 下面哪句话准确描述了 method 1 的行为？
- A. 按设计意图工作。
 - B. 不按设计意图工作。
- (2) 下面哪句话准确描述了 method 2 的行为？
- A. 按设计意图工作。
 - B. 不按设计意图工作。
- (3) 下面哪句话准确描述了 method 3 的行为？
- A. 按设计意图工作。
 - B. 按设计意图工作，如果把 attr_accessor :title, :release_date 加到模型代码中。
 - C. 不按设计意图工作，即使把 attr_accessor 加到模型代码中。

29. 左下图和右下图分别是两种设计方案。左下图是一个单一系统架构，三个子系统都采用统一的 API（Bookstore Service）。右下图是一个面向服务设计架

构，三个子系统采用各自独立的 API，请解释两个方案的优缺点，从功能升级和横向扩展等方面比较。



30. 下面是一个有加和减功能的计算器，以及手工编写的测试程序。

- (1) 请用 Ruby 的 MiniTest 自动测试方法重写测试程序；
- (2) 若计算器要增加乘法和除法功能，如何修改 Calculator 和 自动测试程序。

```

class Calculator
  def add(x, y)
    x+y
  end

  def subtract(x, y)
    x-y
  end
end

def verify(expected, actual, message)
  if actual == expected
    print "\e[32m #{message} passed \e[0m"
  else
    print "\e[31m #{message} failed \e[0m"
  end
end

```

```
calculator = Calculator.new
result = calculator.add(1, 2)
verify(3, result, 'Addition')
result2 = calculator.subtract(2, 1)
verify(1, result2, 'Subtraction')
```

31. 使用 Ruby On Rails 搭建一个博客系统

- 1) 新建一个名为 blog 的 rails 项目

\$ rails new blog

- 2) 启动一个 web 服务器

\$ cd blog

\$ bin/rails server

- 3) 打开浏览器访问 http://localhost:3000 你将看到 Rails 默认信息页面：



Yay! You're on Rails!



- 4) 创建自己的欢迎页面

首先需要创建新的控制器：运行“controller”生成器，并告诉它您需要一个名为“Welcome”的控制器，并执行一个名为“index”的操作

\$ bin/rails generate controller Welcome index

输出如下：

```
create app/controllers/welcome_controller.rb
route get 'welcome/index'
invoke erb
create app/views/welcome
create app/views/welcome/index.html.erb
invoke test_unit
create test/controllers/welcome_controller_test.rb
invoke helper
create app/helpers/welcome_helper.rb
invoke test_unit
invoke assets
invoke coffee
create app/assets/javascripts/welcome.coffee
invoke scss
create app/assets/stylesheets/welcome.scss
```

Index.html.erb
Index.html.Rb

然后需要在视图 /views/welcome/ 中，写入如下代码：

```
<h1>Hello, Rails!</h1>
```

5) 设置主页

打开 config/routes.rb

```
Rails.application.routes.draw do
  get 'welcome/index'

  _____
```

加入 root 'Welcome#index'

6) 创建一个 resource。resource 是指类似物品的集合，如物品、人或动物。您可以为 resource 创建、读取、更新和销毁项，这些操作称为 CRUD 操作。

Rails 提供了一个可以用来声明标准 REST 资源的 resource 方法。将 article 资源添加到 config/routes 中需要在横线处填入 Resources=article

```
Rails.application.routes.draw do
  get 'welcome/index'

  _____
```

root 'welcome#index'

end

使加入的 resource 生效: bin/rails routes

7) 创建 form

New Article

Title

Text

在 app/views/articles/new.html.erb

```
<%= form_with scope: :article, local: true do |form| %>
  <p>
    <%= form.label :title %><br>
    <%= form.text_field :title %>
  </p>

  <p>
    <%= form.label :text %><br>
    <%= form.text_area :text %>
  </p>

  <p>
    <%= form.submit %>
  </p>
<% end %>
```

表单需要使用不同的 URL 才能到达其他地方。这可以通过 form_with 的:url 选项来完成。

```
<%= form_with scope: :article, url: articles_path,  
local: true do |form| %>
```

使 route 生效: \$ bin/rails routes

32. 简述下面代码的作用

```
describe "Home page" do  
  it "should have the content 'Sample App'" do  
    visit '/static_pages/home'  
    expect(page).to have_content('Sample App')  
  end  
end
```

RSpec 测试代码
Check 'static_pages/home'
路由中是否包含 "sample App"

33. 我们知道软件开发中最常见的两个问题分别是:

1. 需求和开发脱节: 用户想要的功能没有开发, 开发的功能并非用户想要。
2. 开发和测试脱节: 从开发到测试周期过长, 测试自动化程度低。

请回答 TDD 和 BDD 分别可以解决哪一个问题, 为什么?

TDD | BDD

34. Ruby 语言中, 以下分别对应的是什么类型的变量? 请连线对应

一般小写字母、下划线开头
\$开头
大写字母开头
@开头
@@开头

类变量 (Class variable)
全局变量 (Global variable)
实例变量 (Instance variable)
常数 (Constant)
变量 (Variable)

35. 问: 在软件开发中我们会进行版本控制。一般会使用 Git 做版本控制。假设当前软件项目中有个主分支名叫 master, 并且现在我们所做的改动在分支 fix1001(目前正处于)下。如何将 fix1001 下的改动合并到 master 下, 并推送到 Gitee 上。请写出命令。

git add -A

79 切换分支 git checkout master
git merge fix1001
git push -u origin master

36. 在阐述用户故事的时候，我们希望能满足SMART 的要求，即“Specific, Measurable, Achievable, Relevant, Timeboxed”，请根据这些原则对下面选课系统的用户故事做修改：
- ① 我登陆后能看到排序后的所有课程，② 在搜索栏能搜索找到自己想选的课程。
- ③ 搜索课程用时很短，在选择课程后能马上显示实时课表。并可以在另一个模块找到自己之前课程的得分。