主管领导审核签字

哈尔滨工业大学 2017-2018 学年秋季学期

软件工程　　试题

题号	一	二	三	四	五	六	总分
得分							
阅卷人							

片纸鉴心 诚信不败

授课教师＿＿＿＿＿＿　密　姓名＿＿＿＿＿＿　封　学号＿＿＿＿＿＿　线　院系＿＿＿＿＿＿

本试卷满分 100 分，按 40%计入总成绩。

微信是一款非常流行的社交 App。一个用户可以与多个用户建立好友关系，不妨认为该关系是双向的，意即若 A 在 B 的通讯录里，则 B 必定在 A 的通讯录里。用户的通讯录里保存着与他有社交关系的所有其他用户和他所加入的所有“群”。用户有微信号（唯一标识，与其他用户区分，一旦确定后不可修改）和名字（可随时修改），通过修改名字展现个性。

一个群最少由 3 个用户构成。群内的任一用户在群内发送一条文字消息，群内的其他用户均可收到消息并做出反馈（反馈的本质仍然是向群内发送消息），从而实现 1 对多的消息传递。用户发出消息之后，该消息无法被修改。一个群内的所有消息按发送时间从早到晚的次序，形成群的“会话”，该会话包含了该群自创建时刻起到当前时刻群内所有用户向该群发出的所有消息。群也有名字和唯一标识的 ID。

为实现特定人群之间的多对多社交，用户可主动创建某个群（创建群时应至少邀请其他 2 个用户加入，否则无法创建）。创建群的用户称为该群的“群主”。若某个群已存在，群内的任何用户可继续邀请其他用户加入，从而扩大了社交的范围。若用户已在某个群里，可以选择退出该群，退出群之后，该群从用户的通讯录中去除，用户不再出现在此群的用户列表中，避免了再收到不感兴趣的消息。但该用户之前在群里所发的消息仍然保留在群的会话中。

为快速查找历史消息，用户可以输入关键词来查询自己加入的所有群的会话，得到所有包含该关键词的所有消息。

注：虽然以上文字描述以现实中的“微信 App”为背景，但与微信并非完全一致，在做题时请以上述文字描述为基础，不要借鉴真实的微信，也无需考虑未在上文提及的任何内容。

一　需求分析（10 分）

从上述需求描述中识别出至少 5 个用户故事，简明扼要填写下表。

用户故事编号	用户故事描述（句式：作为 XX，我希望做 XX，以便 XX）
1	
2	

用户故事编号	用户故事描述（作为 XX，我希望做 XX，以便 XX）
3	
4	
5	
	（如果愿意，可以在该行内填写更多的用户故事）

二 软件测试（30 分）

假设微信的源代码中包含如下代码片段：

```
1   public class SocialNetwork {

2       private ArrayList<String> nodes = new ArrayList<String>();
3       //such as {"a","b","c","d"} 表示微信社交网络中的用户集合
4       private ArrayList<String> edges = new ArrayList<String>();
5       //such as {"a-b", "a-d", "c-d"} 表示社交网络中社交关系的集合

        //检查用户uID1和uID2之间是否存在社交关系
6       public boolean checkExist(String uID1, String uID2) {
7           if (edges.contains(uID1 + "-" + uID2)
                || edges.contains(uID2 + "-" + uID1)) return true;
8           return false;
9       }
        //在用户uID和用户friendID之间添加一条代表社交关系的边
10      public void addRelation(String uID, String friendID) {
11          edges.add(uID + "-" + friendID);
12      }
        //向社交网络中增加一个用户
13      public void addUser(String id) {
14          nodes.add(id);
15      }
        //检查社交网络中是否已包含某个用户
16      public boolean findUser(String id) {
17          if (nodes.contains(id)) return true;
18          return false;
19      }
```

```
        //将社交网络转化为字符串形式
20      @Override
21      public String toString() {
22          return edges.toString(); // such as "[a-b, a-d, c-d]"
23      }

        //根据用户列表userList生成他们之间的社交网络并输出为字符串
24      public static String generateSocialNetwork(List<User> userList) {
25          SocialNetwork sn = new SocialNetwork();
26          for (int i = 0; i < userList.size(); i++) {
27              User u = userList.get(i);
28              if (!sn.findUser(u.ID))
29                  sn.addUser(u.ID);
30              List<User> friends = u.friends;
31              for (int j = 0; j < friends.size(); j++) {
32                  User friend = friends.get(j);
33                  if (!sn.findUser(friend.ID))
34                      sn.addUser(friend.ID);
35                  if (sn.checkExist(friend.ID, u.ID))
36                      continue;
37                  sn.addRelation(u.ID, friend.ID);
38              }
39          }
40          return sn.toString();
41      }
42  }

    //用户类
43  class User {
44      public String ID;//用户ID
45      public ArrayList<User> friends;//与用户有社交关系的其他用户构成的集合

46      public User(String id) { this.ID = id;}
47      public void addFriends(ArrayList list) {
48          friends.addAll(list);
49      }
50      public void addFriends(User friend) {
51          if (friends.contains(friend))   return;
52          friends.add(friend);
53      }
54  }
```

请使用基本路径法为函数 generateSocialNetwork()设计白盒测试用例：首先给出控制流图（其中的节点采用代码行号进行标识），然后从中识别出所有基本路径，最后给出测试用例。注：如果遇到调用该函数之外的其他函数的情况，被调用函数可作为黑盒处理，无需考虑其内部细节。

(1) （5 分）控制流图

(2) （9 分）基本路径（按需填写，未必要全部填满）

路径编号	路径的具体构成（使用代码行号表示，例如 1-2-3-4-5）
1	
2	
3	
4	
5	
6	
7	

院系_____ 学号_____ 姓名_____ 授课教师_____

密 封 线

(3) （6 分）测试用例（按需填写，覆盖所有基本路径即可，未必要全部填满）

编号	输入数据（List<User> userList） 由于这是一个列表，并且列表中的成员是 User 对象，请采用以下示例展示的简化形式： 用户 a，通讯录{b,c} 用户 b，通讯录{a,c} 用户 c，通讯录{b} 用户 d，通讯录{}	期望输出（String） 例如：[a-b, a-c, b-c]	覆盖路径编号
1			
2			
3			
4			
5			
6			

(4) （10 分）任选上表中的某个期望输出不为空字符串的测试用例，使用 JUnit 编写 generateSocialNetwork()的测试程序。以下四个区域请按需填写，你认为不需要的区域可以留空白。

```
public class SocialNetworkTest {
    [                                                    ]

    @Before
    public void setUp() throws Exception {
    [                                                    ]
    }

    @Test
    public void generateSocialNetworkTest() {
    [                                                    ]
    }
```

```
    @After
    public void tearDown() throws Exception {
        
    }
}
```

三　代码评审（10 分）

对题目二中的代码进行人工评审（review），发现其中不符合代码规范或者可能造成潜在 bug 的地方，在下表做简要说明。请给出 5 项问题及其修改建议。同类型问题只需填写一次，重复填写不计分。

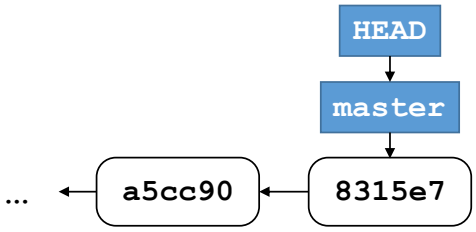
代码行号（第 X 行）或代码行范围（第 X-Y 行）	存在的问题	建议如何修改

密 封 线

院系＿＿＿＿＿＿ 学号＿＿＿＿＿＿ 姓名＿＿＿＿＿＿ 授课教师＿＿＿＿＿＿

四　配置管理（10 分，每题 5 分）

有三个开发者参与一个项目，A 负责开发初始代码，B 负责修复 bug 和优化代码，C 负责测试并报告 bug。项目的 Git 服务器为 S，三人的本地 Git 仓库已经配置好远程服务器（名字均为 origin）。项目的 Git 版本状态如图所示，三人的本地 Git 仓库的状态也是如此，其中包含主分支 master，当前工作分支是 master。



此时他们三人开展了以下工作：

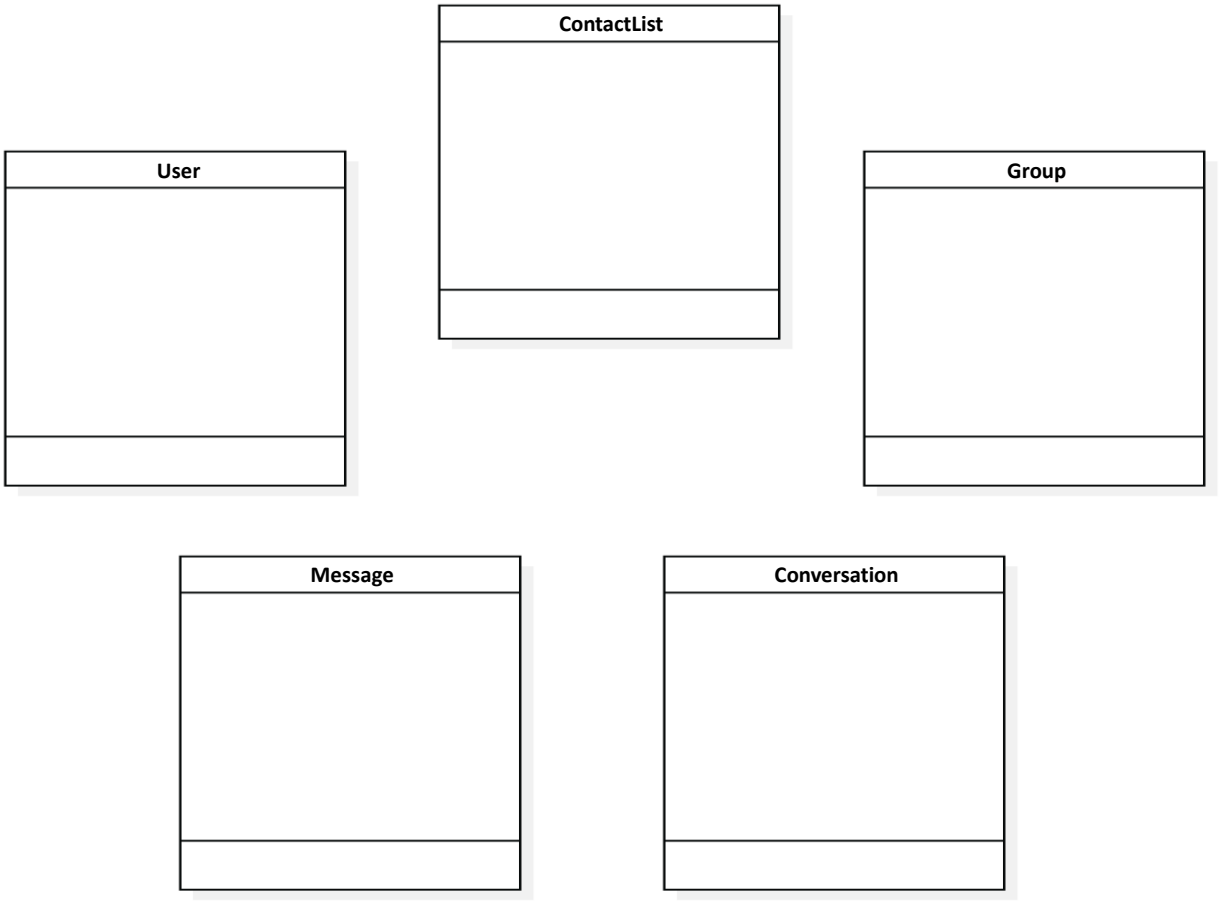
a) A 开发了某项新功能，他创建了分支 b1 并在该分支上提交新代码，推送至服务器 S；
b) C 获取了 A 的提交，并在其上开辟了新分支 b2，在 b2 上撰写测试程序并提交和推送至服务器 S；
c) C 在执行测试程序过程中发现了 A 的代码存在问题，C 将 bug 信息报告给 B；
d) B 获取了 C 推送的包含测试程序的版本，在其基础上开辟了一个新分支 b3 用于 bug 修复，当 B 确认修改后的代码可通过所有测试用例之后，向 Git 做了一次提交，将 b3 合并到 b2 上并推送至服务器 S；
e) C 获取 B 的修复代码并重新执行其中包含的测试程序，确认 bug 已被修复，故将其合并到主分支 master 上，推送至服务器 S，对外发布。

题目：

(1) 在图上补全上述活动结束后服务器 S 上的版本状态图（需注明各分支的名字与位置）；
(2) 写出 B 为完成步骤 d 所需的全部 Git 指令，指令需包含完整的参数。

五　面向对象分析（25 分）

(1) （9 分）第 1 页的需求描述中提及了以下实体类：用户（`User`）、用户的通讯录（`ContactList`）、群（`Group`）、消息（`Message`）、群会话（`Conversation`）。在下图中的相应区域填入实体类的属性（格式：数据类型 属性名，每行一个）。请务必包含各关联属性。属性名请使用中文。



(2) （8 分）识别这些实体类之间的静态结构关系，在上图中补充绘制这些关系（至少包含：线或箭头、关系的多重性）使该图成为完整的领域类图。由于空间狭小，请确保补充后仍清晰可读。

(3) （8 分）“退群”这个动作最适合作为哪个实体类的操作？思考一下该操作的内部实现逻辑，它本质上对该实体类的哪个（些）属性做了何种更新？除了该实体类，还会改变其他哪个（些）实体类的什么属性？具体做了何种更新？

六　简答题（15 分，任选两题作答，若回答全部问题，按前两题评分）

(1) （9 分）本学期 Lab1 要求开发一个 Java 程序，从文本文件中读取数据并生成有向图，进而在图上进行若干操作，其中有一个操作是“随机游走”，其函数规约为 `String randomWalk(type G)`。如果要对该函数进行黑盒测试，设计测试用例并用 JUnit 编写测试程序，但由于该函数的内部实现需要多次执行随机函数，这导致其输出结果不固定，多次执行所产生的路径可能均不同，故 JUnit 测试函数中无法使用 `assertEquals()` 来判定程序是否通过测试。如何解决？

(2) （6 分）动态代码评审工具采集被测程序运行时的数据，进而评估程序的时空性能。目前的动态代码评审工具有两种实现策略：采样、代码注入。请结合你在本学期 Lab4 中使用 VisualVM 进行 Java 程序动态评审的体验，简要阐述二者在分析代码性能的能力与性能上有何差异。

(3) （6 分）“缓存（Cache）”和“分布式集群”均可用来改善 Web 系统在大规模并发情况下的系统响应时间（即用户发出请求到得到系统反馈所需要的时间）。请简要分析这两种架构设计策略分别以什么原理来改善“响应时间”？有什么差异？