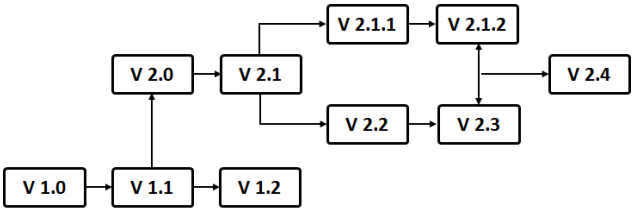
《软件工程》期终考试题

学号：　　姓名：

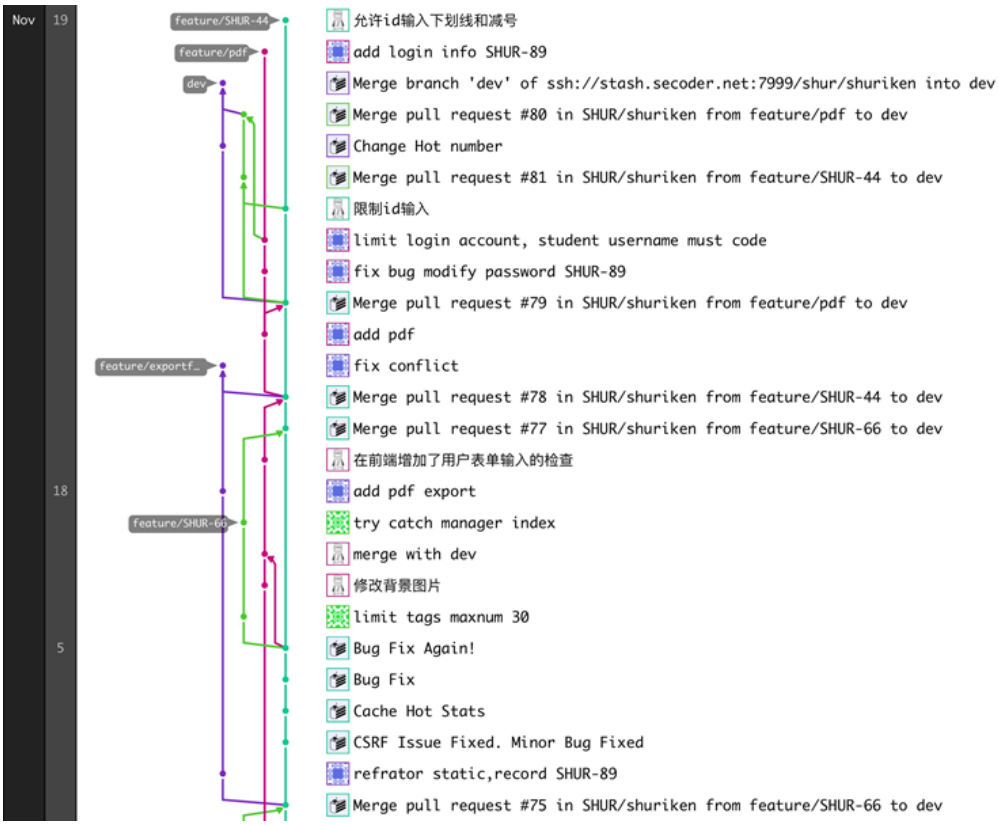
一、 简答题（20 分）

1. **版本管理**（10 分）。版本管理是跟踪软件组件不同版本的演化过程，有助于团队协同开发，通过分支、合并等操作，保持各个组件的变更互不影响，并同步修改结果。

a） 图 1 给出了一个系统的版本树样例，请分析其版本演化过程。（5 分）



b） 图 2 是本学期某大作业版本树片段。请结合版本管理的基本原则，分析其优点和不足之处。（5 分）



2. **软件过程**（10 分）。软件过程是交织着技术、协作、管理等内容的一个活动序列，软件过程模型刻画了软件开发活动的组织方式。本学期，我们以 Scrum 为例，学习实践了基于快速迭代的敏捷开发过程。

现假设你和同组的另外 2 位同学，要承担一个新的软件开发项目：为学校图书馆升级现有的图书管理系统，新增加跨馆借阅的功能模块。新增加的需求主要包括：

1） 管理员可以配置与其签订合作协议的图书馆信息（如服务访问的 URL 地址、用户名、密码等）；

2） 对于签订合作协议的图书馆：

a） 用户可以无需登录信息访问其开放的服务接口；

b） 用户可以查询其图书信息；

c） 用户可以申请跨馆借阅图书；

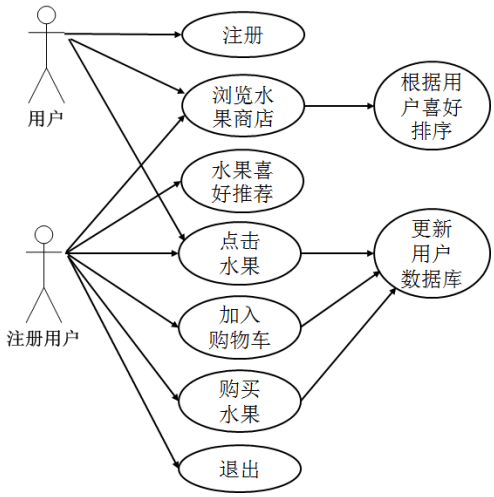
d） 管理员审批跨馆借阅申请，通过后统一办理借阅事宜，并在图书抵达后通知相关申请借阅的用户。

上述项目开发工作须在 1 个月之内完成，请设计你们组基于 Scrum 的迭代开发计划。

二、 软件需求（25 分）

假设要开发一个购买水果的 App，用户可以浏览、挑选水果，系统还可以根据用户的购买历史，分析用户喜好并为用户推荐。

a） 下图是系统需求分析的 UML 用例图，请分析该需求分析的完整性和一致性等质量问题。（10 分）



b） 请对该软件需求进行合理假设，给出“注册用户购买水果”这一功能的序列图。（10 分）

c） 请对该软件需求进行合理假设，给出 5 个非功能需求的实例。（5 分）

三、 代码质量（20 分）

1．**代码注释**（10 分）。代码块注释有助于提高程序的可理解性和可维护性。请评审下面两段代码块注释，比较其优劣并分析主要问题。

```
/ **
 * Method declaration
 *
 *
 * @param username
 * @param password
 *
 * @return
 *
 * @throws SQLExecution
 * /
synchronized Channel connect
            (String username, String password)
            throws SQLException {
```

A

```
/ **
 * Executes an SQL INSERT, UPDATE, or DELETE statement.
 * In addition, SQL statements that return nothing,
 * such as SQL DDL statements, can be executed.
 *
 * @param sql SQL INSERT, UPDATE, or DELETE statement
 * or a SQL statement that returns nothing @ return
 * either the row count for INSERT, UPDATE, or DELET
 * or 0 for SQL statements that return nothing
 * @exception SQLException if a database access
 * error occurs
 * /

public int executelpdate (String sql) throws SQLException {
```

B

2. **单元测试**（10 分）。下图中给出了 *find_object(name)* 的代码实现，请补充完成单元测试的代码。

```
1.  import sys


2.  def find_object(name):
3.      """Find object according to `name`.

4.      This method will try to import all the essential modules to find the
5.      requested object.

6.      Args:
7.          name (str): The name of the object, should contain full module
8.              names from the global scope.

9.      Returns:
10.         The object instance.

11.     Raises:
12.         ImportError: If object with `name` cannot be found.
13.         Exception: When importing external module, they may raise exceptions.

14.     Examples:
15.         >>> find_object('os')
16.         <module 'os' from '?'>

17.         >>> find_object('os.path')
18.         <module 'posixpath' from '?'>

19.         >>> find_object('os.remove')
20.         <function posix.remove>

21.         >>> find_object('os.path.split')
22.         <function posixpath.split>

23.         >>> find_object('os.path.split.__name__')
24.         'split'
25.     """
```

```
26.     # If name is empty, fail fast
27.     if not name:
28.         raise ImportError('Object name should not be empty.')

29.     # Try to import the closest module according to `name`.
30.     parts = name.split('.')

31.     # Try to treat some prefix of the name as module
32.     obj = None
33.     for i in xrange(len(parts), 0, -1):
34.         modname = '.'.join(parts[:i])
35.         try:
36.             obj = __import__(modname)
37.             obj = sys.modules[modname]
38.             parts = parts[i:]
39.             break
40.         except ImportError:
41.             # We can ignore the exception unless it is ImportError.
42.             # Otherwise there should be something wrong when importing
43.             # existing module.
44.             pass

45.     # If we failed to import the container module, then we raise an
46.     # ImportError
47.     if obj is None:
48.         raise ImportError("Couldn't find any module along `%s`." % name)

49.     # Try to get the object along attribute path
50.     for name in parts:
51.         if not hasattr(obj, name):
52.             raise ImportError(
53.                 "Object '%s' does not have attribute '%s'." % (obj, name))
54.         obj = getattr(obj, name)

55.     # Now we've got the object
56.     return obj
```

PyUnit 单元测试代码框架如下图所示，请补充其代码实现。

```
import unittest
from findobj import find_object


class FindObjectTestCase(unittest.TestCase):

    def test_find_module(self):
        import os
        self.assertEqual( 请补充参数 P1, os )

    def test_find_module_module(self):
        import os.path
        self.assertEqual( 请补充参数 P2, os.path )

    def test_find_module_module_object(self):
        import os.path
        self.assertEqual( 请补充参数 P3, os.path.split )

    def test_find_empty_name(self):
        self.assertRaises( ImportError, 请补充参数 P4, 请补充参数 P5 )
```

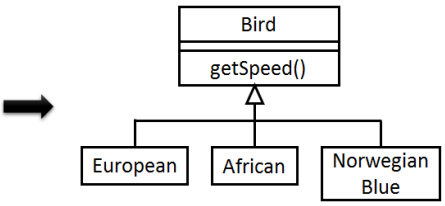
四、 软件设计（20 分）

1. **面向对象设计**（10 分）。下面有两组不同的面向对象设计，请分析比较设计的优劣。

a) 下图中，B 中的类结构是对 A 中代码段的重构，请比较重构前后的设计。

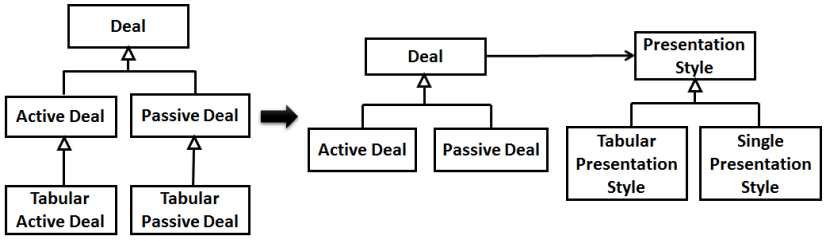
```
double getSpeed () {
  switch (_type) {
    case EUROPEAN:
          return getBaseSpeed ();
    case AFRICAN:
          return getBaseSpeed () –
          getLoadFactor () * _numOfCoconuts;
    case NORWEGIAN_BLUE:
          return (_isNailed) ?
          0: getBaseSpeed (_voltage);
    }
    throw new RuntimeException ();
}
```

A



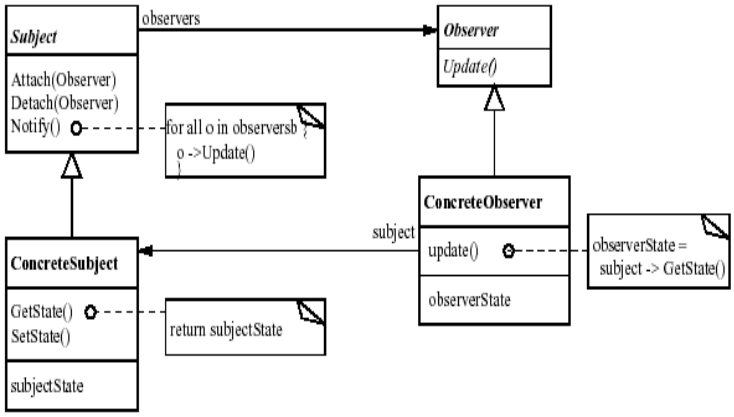
B

b) 下图中 B 是对 A 的重构，请比较重构前后的设计。



A B

2. **设计模式**（10 分）。“Design for Change”和“Design for Reuse”是软件设计的重要原则。在面向对象设计中，设计模式（Design Pattern）是一种有效的应对变更、软件复用的设计方法。下图描述了 Observer 设计模式的主要类结构图。



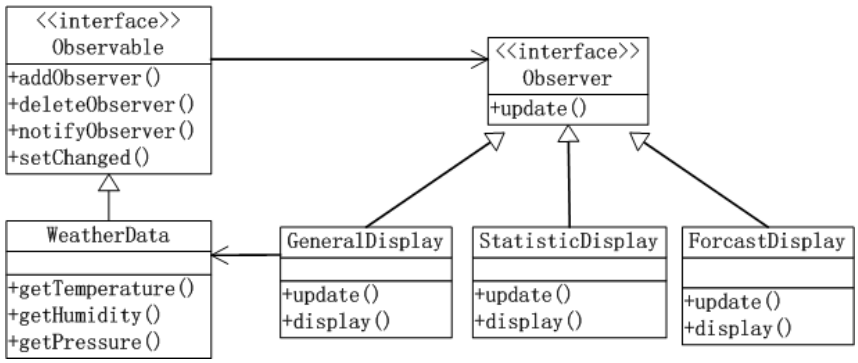
假设有一个天气预报系统，需要对所获得的气象数据 Weather Data 以三个布告板显示：目前状况（温度、湿度、气压等）General Display、气象统计 Statistical Display 和天气预报 Forecast Display，如下图所示。

第一号布告板　　第二号布告板　　第三号布告板

下面是 WeatherData 类的最初实现：

```
public class WeatherData {
   ……
   public void measurementsChanged () {
      float  temp = getTemperature ();
      float  humidity = getHumidity ();
      float  pressure = getPressure ();

      currentConditionsDisplay.update (temp, humidity, pressure);
      statisticsDisplay.update (temp, humidity, pressure);
      forecastDisplay.update (temp, humidity, pressure);
   }

   ……
}
```

我们基于 Java 内置的 Observable 和 Observer 接口，采用 Observer 模式重构代码实现，其类结构如下图所示：



请给出下面程序片段中指定部分的代码实现。

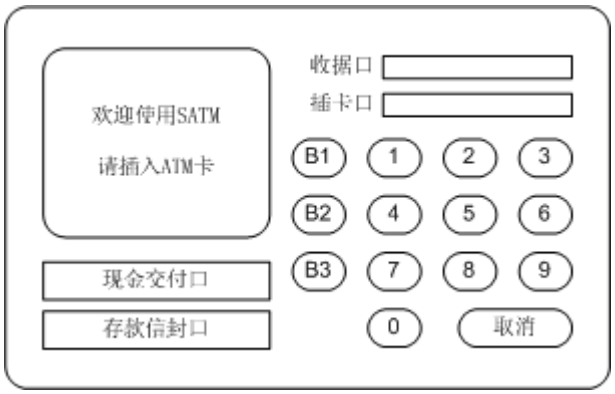
```
public class WeatherData extends Observable {
   private float temperature;
   private float humidity;
   private float pressure;
   ……
   public void setMeasurements (float temperature,
                        float humidity, float pressure) {
      this.temperature = temperature;
      this.humidity = humidity;
      this.pressure = pressure;
      measurementsChanged ();
   }

   public void measurementsChanged () {
      【请补充该方法的代码实现 1】
   }
   ……
}
```

```
public class GeneralDisplay implements Observer {
   Observable   observable;
   private float temperature;
   private float humidity;
   private float pressure;
   ……
   public GeneralDisplay (Observable observable) {
      【请补充该方法的代码实现 2】
   }

   public void update (Observable obs, Object arg) {
      【请补充该方法的代码实现 3】
   }

   public void display () {
      System.out.println ("Current conditions: "
            + temperature + "F degrees and " + humidity + "% humidity");
   }
   ……
}
```

五、 软件测试（15 分）

黑盒测试。黑盒测试根据需求，设计软件的输入，判定输出结果的正确性。请设计下面 SATM 系统的黑盒测试测试用例。

SATM 是一种自动取款机系统，采用下图所示的终端与用户交互。



SATM 的主要功能如下：

1). 用户通过带有个人帐户编码的银行卡访问 SATM 系统。
2). 系统提示用户输入密码。
3). 用户可以选择三种事务中的任意一种：存款(B1)、取款(B2)和余额查询(B3)。
4). 如果选择查询余额，系统通过与银行系统通信，获取用户帐户的余额信息，并显示。
5). 如果选择存款，则系统检测‘存款信封口’状态。如果状态正确，接受存款信封；如果状态错误，显示提示信息。
6). 如果选择取款，系统检测‘现金交付口’状态。如果状态正确，则提示输入取款信息，如果用户帐户上有足够的余额且系统有足够的现金，则付款并将该事务写入用户银行帐户；否则（状态不正确，或帐户余额不足，或系统现金不足），显示提示信息。
7). 当前交易结束后，用户可以选择是否进行另一个交易，或是退出系统。如果选择继续，则重新进入交易选择界面（第 3 步）；否则，退出系统。

请针对用户登录、存款、取款三项功能，设计测试用例，要求：

1） 每个功能设计 5 个测试用例；
2） 测试用例设计规范，需包括输入（数据及操作）以及预期输出；
3） 测试需要覆盖系统的正常输入和异常输入不同的情况；
4） 需采用分区测试和边界值测试方法，且说明测试用例如何覆盖所定义的分区和边界值。