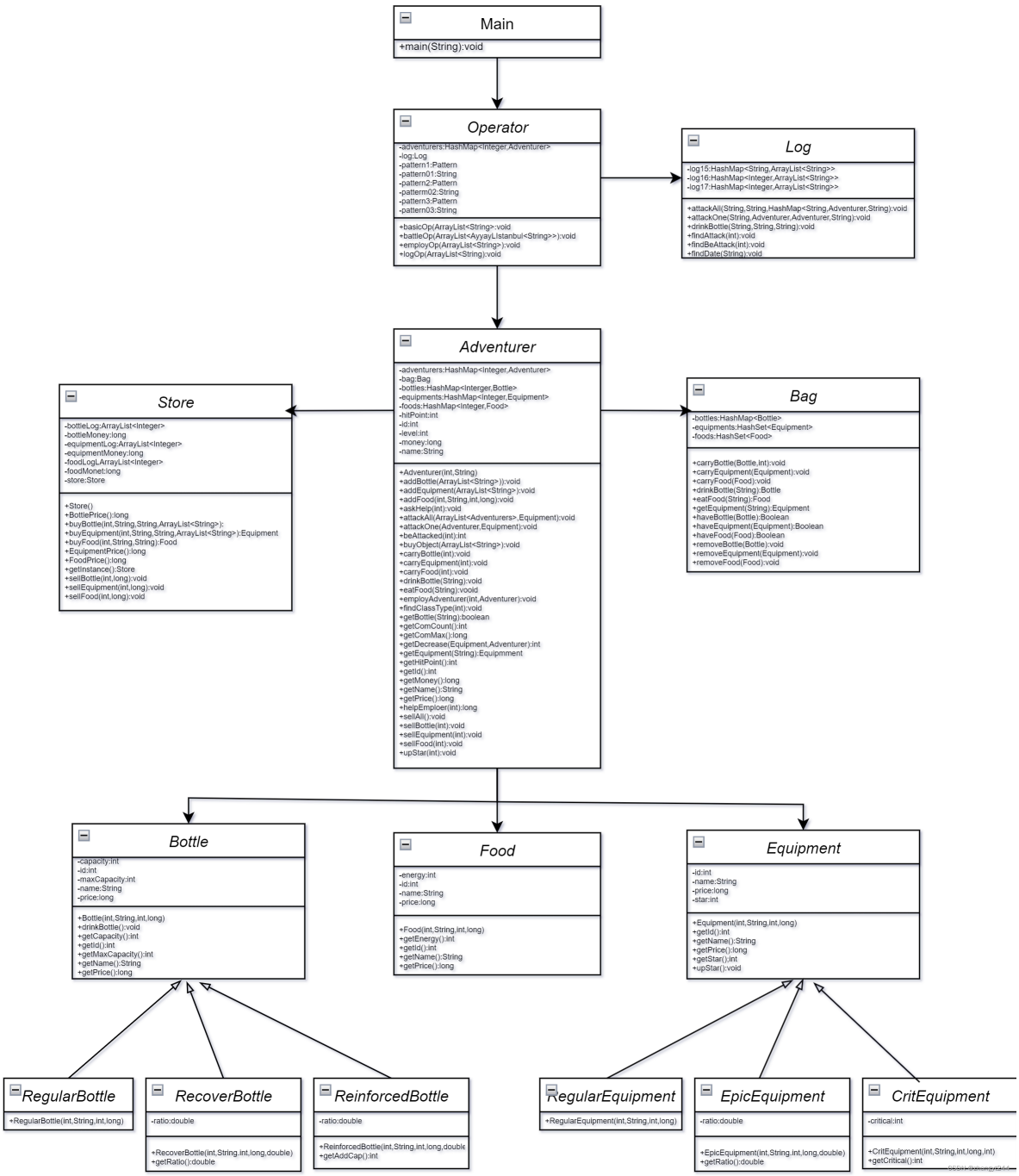
一、架构设计

设计要求

- 你将扮演一位穿越到魔法~~（提瓦特）~~大陆的冒险者，在旅途中，你需要收集、升级各种道具，在商店购买或销售各种物品，雇佣冒险者，不断提升自己的等级和配置，体验各种战斗……最终你将学会类这一新的概念、junit单元测试、对象管理、各种容器、正则表达式、简单的调试、继承以及接口、几种常见的设计模式，成为一名有能力踏入OO正课的大二学生……

UML类图



代码规模

Source File	Total Lines	Source Code Lines	Source Code Lines [%]	Comment Lines	Comment Lines [%]	Blank Lines	Blank Lines [%]
Adventurer.java	444	411	93%	0	0%	33	7%
AdventurerTest.java	94	82	87%	0	0%	12	13%
Bag.java	120	107	89%	0	0%	13	11%
BagTest.java	23	20	87%	0	0%	3	13%
Bottle.java	37	30	81%	0	0%	7	19%
CritEquipment.java	12	10	83%	0	0%	2	17%
CritEquipmentTest.java	13	9	69%	0	0%	4	31%
EpicEquipment.java	12	10	83%	0	0%	2	17%
EpicEquipmentTest.java	12	9	75%	0	0%	3	25%
Equipment.java	32	26	81%	0	0%	6	19%
Food.java	28	22	79%	0	0%	6	21%
Log.java	107	100	93%	0	0%	7	7%
Main.java	36	35	97%	0	0%	1	3%
Operator.java	177	171	97%	0	0%	6	3%
OperatorTest.java	263	223	85%	0	0%	40	15%
RecoverBottle.java	12	10	83%	0	0%	2	17%
RecoverBottleTest.java	12	9	75%	0	0%	3	25%
RegularBottle.java	5	5	100%	0	0%	0	0%
RegularEquipment.java	5	5	100%	0	0%	0	0%
ReinforcedBottle.java	18	16	89%	0	0%	2	11%
ReinforcedBottleTest.java	12	9	75%	0	0%	3	25%
Store.java	101	89	88%	0	0%	12	12%
StoreTest.java	32	30	94%	0	0%	2	6%
Total:	**1607**	**1438**	**89%**	**0**	**0%**	**169**	**11%**

迭代过程

1. 由于刚学习这门课的时候对面向对象编程十分陌生，所以导致第一次迭代作业很有大一程序设计课的风格，所以第二次迭代作业完全重新写了一遍。后来我~~被迫~~意识到了这个问题，在*github*以及*csdn*上观摩~~（模仿）~~了几份学长在OO正课上写的代码，逐渐熟悉了面向对象编程，使得自己写的迭代代码具有较强的可扩展性。
2. 在后来的几次课程中，我逐渐地将所学习到的正则表达式，输入解析，继承，各种模式加入到了自己迭代的代码中，形成了如今的架构设计~~（也侥幸没有体验到强测挂子后熬夜debug的痛苦）~~。
3. 存在的缺点：在倒数第二次迭代之前，老师讲了接口的概念，但是我考虑到只剩两次迭代了而且担心使用接口后可能会出现意外的bug，~~当然还有一部分原因是自己懒得改了hhh~~，总之没有在自己的迭代代码中用到接口，现在想想有点后悔当时没有重构qaq。

二、使用junit的心得体会

junit优势

1. 与以往在main里写个调用方法的语句来测试相比，junit测试更加便利；
2. 在后期的迭代中，junit测试代码不需要大改；
3. 可以精准地定位bug的位置；
4. 可以通过各种覆盖率量化自己的测试成果；
5. 将测试代码与源代码分离，这在实际工作中应该是一种非常有效且必要的要求。、

junit注意事项

1. 努力记下各种断言函数，如assertEquals,assertNull,assertTrue等；
2. 测试应该从最基本的单元开始，比如应首先测完ReinforcedBottle类后再测试Bottle类，或者方法A调用方法B，则应先测试完B再测试A；
3. 测试数据应该全面广泛，尤其应该尽可能地将各种分支都覆盖，做充分测试。

三、 学习oopre的心得体会

- 第一也是最基本的就是学习到了许多**面向对象编程**的基础知识、学会了java的使用，初步实现了从面向过程编程到面向对象编程的过渡；
- 虽然这门课是用java写作业，但是因为教的是面向对象的知识，也让我写的**其他语言**比如cpp以及python的代码更加规范科学；
- 这可能是我第一次在课内写这么长的代码，使我更加熟悉了如何很好地写一个**超长代码量**的程序；
- 以往自己写程序总是不注重代码的风格，就导致了写的代码的解释性非常差，由于这门课有对代码风格的强制检查，所以我的**代码风格**有了很大的改观。

四、建议

- 由于这是大多数人第一次接触面向对象语言，很多人在迭代的过程中总是会让自己的代码变得越来越乱。我想在明年的oopre课程中，可以给学弟学妹提供几份我们这一级的优秀代码，供他们参考~~（如果以后每年的oopre的作业是一模一样的当我没说这个建议）~~