第一周作业

1. 使用 Wikipedia 查阅以下一些词条并翻译

   - **Programming paradigm**

     编程范型，是从事软件工程的一类典型的风格，一个编程语言可以支持多种范型。

   - **Structured Programming**

     结构化编程，是一种编程范型：它采用子程序、代码区块、for 循环以及 while 循环等结构，来取代传统的 goto 。希望借此来改善计算机程序的明晰性、品质以及开发时间，并且避免写出面条式代码。

   - **Object-Oriented Programming**

     面向对象编程，是种具有对象概念的程序编程泛型，同时也是一种程序开发的抽象方针。它包含数据、属性、代码与方法。

【答题内容】

1. Programming paradigm：编程范型，是软件工程中的一种风格，一个编程语言可以支持多种范型；
2. Structured Programming：结构化编程是一种编程范型，它使用子程序、for循环以及while循环等取代了传统的goto语句。希望用来改善计算机程序的质量和开发时间；
3. Object- Oriented Programming：面向对象编程是一种具有对象概念的编程范型，同时他也是一个抽象的概念，其中包含数据、属性及方法等。

第一周作业

2. 为什么面向对象方法在模块化和复用性方面优于传统方法

(1) **模块化**：面向对象方法的封装机制、通过接口访问（信息隐藏和高内聚低耦合）符合模块化的特点。
(2) **复用性**：封装、继承、多态机制，使面向对象方法有很好的复用性。

【答题内容】
因为面向对象方法在模块化中使用封装机制以及接口访问有着隐藏信息和内部高耦合度外部低耦合度的特点；在复用性上有着封装、继承、多态等机制，使其有着更好的复用性

【答题内容】

1. 面向对象方法将执行过程抽象成若干个对象之间的交互，对于对象外部对对象内部的操作，只需要调用一些接口函数即可。因此在模块化方面，面向对象可以基于对象（类）进行模块划分，相比于面向过程方法，模块化更为方便，抽象层次更高。
2. 面向对象方法提供了两个面向过程方法不具有的功能，即继承与多态。
    a. 继承方面，子类对象基于父类对象构建，实现对父类对象某些功能的复用，降低了代码的复杂度。同时通过重写父类函数，也可以实现子类之间的差异性。继承功能提高了代码的复用性。
    b. 多态方面，有些函数可能会对不同对象进行相同或相似的操作，此时利用多态，可以简化代码，提高复用性。

2.1给出队列（先进先出）的抽象数据类（代数方法）

Partial function

先进先出
分类讨论
Remove （A2）公理较为复杂，不作要求

ADT specification of queues

TYPES

QUEUE [G]

FUNCTIONS

- put: QUEUE [G] × G → QUEUE [G]
- remove: QUEUE [G] ↛ QUEUE [G]
- item: QUEUE [G] ↛ G
- empty: QUEUE [G] → BOOLEAN
- new: QUEUE [G]

AXIOMS: For any x: G, q: QUEUE [G]

A1 • item (put (q, x)) = empty (q) ? x : item (q)

A2 •A3 • empty (new) (or empty(new)=True)

A4 • not empty (put (q, x)) (or empty(put(s,x))=False)

PRECONDITIONS:

- remove (q: QUEUE [G]) requires not empty (q)
- item (q: QUEUE [G]) requires not empty (q)

2.2使用代数方法给出一个存储键值对的表（Map）的抽象数据类型定义（代数方法）

Map中存储的是一系列唯一的健（key）以及其对应的值（value）。

考虑：
①“创建新表(new)”
②“查询是否存在一个键(has) ”
③“取出一个键对应的值(item)”
④“插入一个键值对(put) ”
⑤“移除一个键值对(remove)”等操作

TYPES

MAP [K, V]

FUNCTIONS

- new: MAP[K, V]
- has(m, k): MAP[K, V]×K→BOOLEAN
- item(m, k): MAP[K, V]×K↛V
- put(m, k, x): MAP[K, V]×K×V↛MAP[K, V]
- remove(m, k): MAP[K, V]×K↛MAP[K, V]

PRECONDITIONS

P1 item(m, k) require has(m, k)
P2 put(m, k, x) require ¬has(m, k)
P3 remove(m, k) require has(m, k)

AXIOMS

A1 ¬has(new, k)
A2 has(put(m, k, x), k)
A3 has(put(m, k, x), l) = has(m, l), if l≠k
A4 ¬has(remove(m, k), k)
A5 has(remove(m, k), l) = has(m, l), if l≠k
A6 item(put(m, k, x), k) = x
A7 item(put(m, k, x), l) = item(m, l), if l≠k∧has(m, l)
A8 item(remove(m, k), l) = item(m, l), if l≠k∧has(m, l)

第三周作业

1. 解释应用 DbC 时子类断言与父类断言的关系

(1) 子类乃父类的特化，子类的实例也是父类的合法实例，因此子类将继承父类的断言
(2) 子类可以使用 require else 削弱父类的先验条件（前置断言），即子类的前置断言要弱于父类的前置断言；子类可以使用 ensure then 加强父类的后验条件（后置断言），即子类的后置断言要强于父类的后置断言；子类可以用 and 把不变式子句和子类所继承的父类的不变式子句结合起来，加强不变式断言，即子类的不变式要强于父类的不变式。

【答题内容】
子类中重新定义继承自父类的特性:
- 作为重定义的一部分，可以修改被继承的特性的契约
- 可以使用 require else 弱化先验条件
- 可以使用 ensure then 强化后验条件
- 子类继承它的父类的不定式
- 子类可以再不变式中增加子句，新增的子句and被继承的子句，从而强化了被继承的不变式

第三周作业

2. 解释 DbC 和防御性编程的异同

防御性编程的含义：
防御性编程是一种细致、谨慎的编程方法。为了开发可靠的软件，我们要设计系统中的每个组件，以使其尽可能地“保护”自己。我们通过明确地在代码中对设想进行检查，击碎了未记录下来的设想。这是一种努力，防止（或至少是观察）我们的代码以将会展现错误行为的方式被调用。
相同点：都可以提高软件的可靠性。
不同点：
(1) DbC 中先验条件是程序文档的组成部分，而产生异常的语句是程序体本身的组成部分。
(2) 采用注释来描述例程对参数的限制时，很难保证这个注释正确地描述了该限制。但可以相信具有显式先验条件检查的文档，因为断言在测试时得到了验证

【答题内容】
DbC和防御性编程的异同:
相同点:
- 可以提高软件的可靠性
不同点:
- DbC中先验条件由程序文档所组成，而产生异常的语句是由程序本身而组成
- 使用注释很可能没有达到说明的目的，而使用先验条件检查的文档，则经过测试更加可靠

第三周作业

3. 解释 Dbc 和断言的区别

① 断言不提供契约
② 客户不能将断言看做接口的一部分
③ 断言没有关联的语义规范
④ 不确定断言是表示前置条件，后置条件还是不变条件
⑤ 断言不支持继承
⑥ 断言不自动生成文档

【答题内容】
c++断言机制：
assert是一个宏，传给assert一个表达式后，如果得出条件为假，那么它先向stderr打印一条出错信息，然后通过调用abort来终止程序程序运行。他用于检测程序不应该发生的情况。

断言不提供契约；
断言不被视为接口的一部分提供给客户；
断言没有相关联的语义规范；
断言不显式指出是作为先验条件、后验条件还是不变式；
断言不支持继承；
断言不输出自动文档。

4.1从DbC的角度看，Java方法声明中的throws子句反映了类（supplier）和其使用者（client）之间的怎样的权利/义务？Java子类若重定义父类中的方法，其throws的异常有何限制？用DbC的Contract继承原则解释这个限制。

- **权利/义务关系：**
  - **义务：** client的义务是对抛出的异常做出相应处理，去catch或者继续throws；supplier的义务是必须捕获指定的异常并将其抛给client。
  - **权利：** client的权利是它能正常收到异常，且不需要考虑throws子句中没有声明的异常。supplier的权利即为它抛出的异常能够得到相应处理。

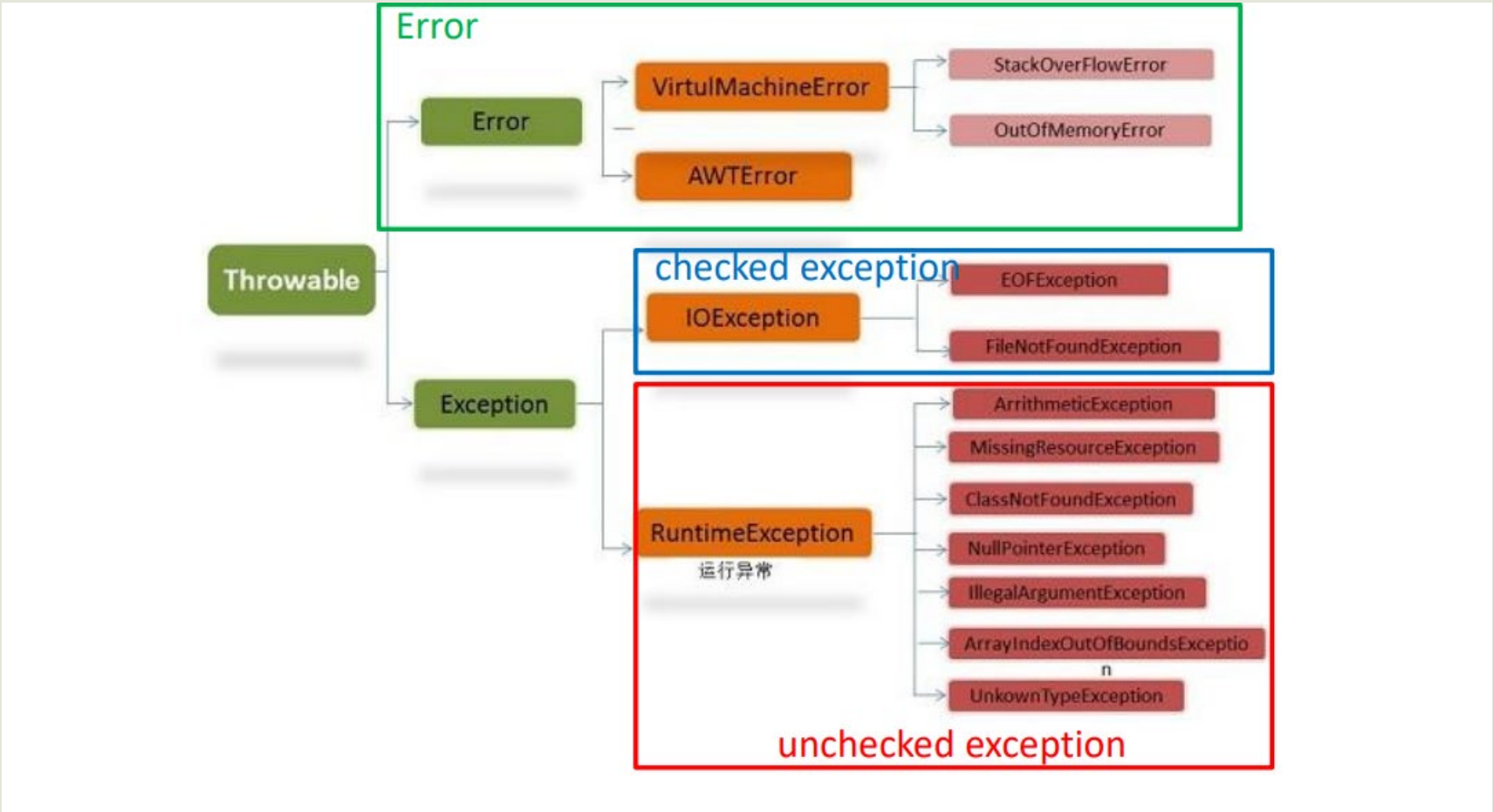
- **Dbc的Contract继承原则**
  1. 可以使用require else削弱先验条件
  2. 可以使用ensure then加强后验条件
  3. 用and把不变式子句和你所继承的不变式子句结合起来，就可以加强不变式

  从 DbC 的 Contract 继承原则的角度来看，子类不能抛出比父类更多的异常，子类方法可以 throws 的异常都是父类也可以throws的。符合Contract 继承原则中子类的postcondition不会比父类变弱。

4.2解释checked exception, unchecked exception和error三者的定义以及使用的区别

- **定义**
  - Checked exception：程序与外界交互所产生的异常，通常是由于外部环境出现未被考虑的情况而引起的。如找不到文件，错误的输入类型等等。
  - Unchecked exception：程序本身存在的问题，如数组越界，除以0，调用空对象等等。
  - Error：在控制范围之外的底层系统错误，并不是程序本身的错误，但程序会因此受到影响而终止，如内存不足等等。
- **使用区别**
  - Checked exception：在编译时被强制检查的异常，必须为所有的checked exception提供异常处理机制，否则编译不通过。
  - Unchecked exception：在编程过程中可人为避免，编译器不会强制检查的异常，它的产生往往意味着处理逻辑的错误，因此我们需要对程序进行修改。
  - Error：在控制范围之外，恢复较困难，由于 error 的产生往往导致程序本身运行所依赖的底层系统崩溃，程序本身没有办法对此进行处理，并且处理问题的责任往往不在程序自身，因此我们对于 error 通常不需要处理，就让程序中止即可。

4.2解释checked exception, unchecked exception和error三者的定义以及使用的区别



需要注意的问题

1. 一定要按时提交！！！

   如果截止时间过了无法提交，在QQ群里单独联系助教；

   就算系统中能提交，提交之后也要联系助教。

   逾期提交会酌情扣分

2. 以上答案仅供参考，大概意思相同言之有理即可