面向对象UML系列第二次作业指导书

注意

请不要提交官方包代码！！！其代码仅供参考，请使用已经编译打包好的jar文件。

因提交官方包源代码导致的任何问题，请自行承担后果！！！

摘要

本次作业，在上次作业基础上，扩展解析器，使得能够**支持对UML顺序图和UML状态图的解析。**

问题

基本目标

扩展类图解析器，使得可以支持对UML状态图和顺序图的分析，可以通过输入相应的指令来进行相关查询。

基本任务

本次作业的程序主干逻辑（包括解析`mdj`格式的文件为关键数据）均已实现，只需要同学们完成剩下的部分，即：**通过实现官方提供的接口，来实现自己的UML分析器**

官方的**接口定义源代码**都已在接口源代码文件中给出，各位同学需要实现相应的官方接口，并保证**代码实现功能正确。**

具体来说，各位同学需要新建一个类，并实现相应的接口方法。

当然，还需要同学们**在主类中调用官方包的`AppRunner`类**，并载入自己实现的UML解析器类，来使得程序完整可运行，具体形式参考官方包目录下的README.md。

测试模式

本次作业继续**不设置互测环节。**针对本次作业提交的代码实现，课程将使用公测+bug修复的黑箱测试模式，具体测试规则参见下文。

输入输出

本次作业将会下发`mdj`文件解析工具、输入输出接口（实际上为二合一的工具，接口文档会详细说明）和全局测试调用程序

- 解析工具用于将`mdj`格式文件解析为包含了文件内模型中所有关键信息的元素字典表
- 输入输出接口用于对元素字典表的解析和处理、对查询指令的解析和处理以及对输出信息的处理
- 全局测试调用程序会实例化同学们实现的类，并根据输入接口解析内容进行测试，并把测试结果通过输出接口进行输出

输入输出接口的具体字符格式已在接口内部定义好，各位同学可以阅读相关代码，这里我们只给出程序黑箱的字符串输入输出。

规则

- 输入一律在标准输入中进行，输出一律向标准输出中输出
- 输入内容以指令的形式输入，一条指令占一行，输出以提示语句的形式输出，一句输出占一行
- 输入使用官方提供的输入接口，输出使用官方提供的输出接口
- 输入的整体格式如下：
  - 由`mdj`文件解析而来的关键元素表
  - `END_OF_MODEL`分隔开行
  - 指令序列，每条指令一行

关于类图的查询指令

（本段并无修改，和上次保持一致，仅有少量的补充说明以保证不存在歧义）

各个指令对应的方法名和参数的表示方法详见官方接口说明。

模型中一共有多少个类

输入指令格式：`CLASS_COUNT`

举例：`CLASS_COUNT`

输出：

- `Total class count is x.` x为模型中类的总数

类中的操作有多少个

输入指令格式：`CLASS_OPERATION_COUNT classname`

举例：`CLASS_OPERATION_COUNT Elevator`

输出：

- `Ok, operation count of class "classname" is x.` `x`为类中的操作个数
- `Failed, class "classname" not found.` 类不存在
- `Failed, duplicated class "classname".` 类存在多个

说明：

- 本指令中统计的一律为此类自己定义的操作，不包含继承自其各级父类所定义的操作
- 本指令不检查操作的合法性，所有操作均计入总数。如有重复操作分别计入总数。

类中的属性有多少个

输入指令格式：`CLASS_ATTR_COUNT classname`

举例：`CLASS_ATTR_COUNT Elevator`

输出：

- `Ok, attribute count of class "classname" is x.` `x`为类中属性的个数
- `Failed, class "classname" not found.` 类不存在
- `Failed, duplicated class "classname".` 类存在多个

说明：

- 本指令的查询均需要考虑属性的继承关系，即需包括各级父类定义的属性。

类的操作可见性

输入指令格式：`CLASS_OPERATION_VISIBILITY classname methodname`

举例：`CLASS_OPERATION_VISIBILITY Taxi setStatus`

输出：

- `Ok, operation visibility of method "methodname" in class "classname" is public: xxx, protected: xxx, private: xxx, package-private: xxx.` 该操作的实际可见性统计
- `Failed, class "classname" not found.` 类不存在
- `Failed, duplicated class "classname".` 类存在多个

说明：

- 本指令中统计的一律为此类自己定义的操作，不包含其各级父类所定义的操作
- 在上一条的前提下，需要统计出全部的名为methodname的方法的可见性信息
- 如果不存在对应的方法，则全部置0

类的属性可见性

输入指令格式：`CLASS_ATTR_VISIBILITY classname attrname`

举例：`CLASS_ATTR_VISIBILITY Taxi id`

输出：

- `Ok, attribute "attrname" in class "classname"'s visibility is public/protected/private/package-private.` 该属性的实际可见性
- `Failed, class "classname" not found.` 类不存在
- `Failed, duplicated class "classname".` 类存在多个
- `Failed, attribute "attrname" not found in class "classname".` 类中没有该属性
- `Failed, duplicated attribute "attrname" in class "classname".` 类中属性存在多个同名

说明：

- 本指令的查询均需要考虑属性的继承关系。
- 其中对于父类和子类均存在此名称的属性时，需要按照`duplicated attribute`处理。

类的属性类型

输入指令格式：`CLASS_ATTR_TYPE classname attrname`

举例：`CLASS_ATTR_TYPE Taxi id`

输出：

- `Ok, the type of attribute "attrname" in class "classname" is typeA.` 该属性的类型
- `Failed, class "classname" not found.` 类不存在
- `Failed, duplicated class "classname".` 类存在多个
- `Failed, attribute "attrname" not found in class "classname".` 类中没有该属性
- `Failed, duplicated attribute "attrname" in class "classname".` 类中属性存在多个同名
- `Failed, wrong type of attribute "attrname" in class "classname".` 属性类型错误

说明：

- 类型`type`的数据类型有两种，分别为`ReferenceType`和`NamedType`。对于这两种情形，

    - `NamedType`为命名型类别，只考虑JAVA 语言八大基本类型(`byte`，`short`，`int`，`long`，`float`，`double`，`char`，`boolean`)和`String`，其余一律视为错误类型。
    - `ReferenceType`为依赖型类别，其指向已定义的类或接口，类型名为对应的类名或接口名。
- 本指令的查询均需要考虑属性的继承关系。
- 其中对于父类和子类均存在此名称的属性时，需要按照`duplicated attribute`处理。

类的操作的参数类型

输入指令格式：`CLASS_OPERATION_PARAM_TYPE classname methodname`

举例：`CLASS_OPERATION_PARAM_TYPE Taxi setStatus`

输出：

- `Ok, method "methodname" in class "classname" has parameter tables and return value: (type1, return: type2), (type3, return: type4), (type5, type6, no return).` 该操作有三种参数表和返回值的搭配，其中
    - 传出列表时可以乱序，内部参数类型也可以乱序，官方接口会自动进行排序（但是**需要编写者自行保证不重不漏**）
    - 如果不存在该命名的操作，则不输出任何搭配。
- `Failed, class "classname" not found.` 类不存在
- `Failed, duplicated class "classname".` 类存在多个
- `Failed, wrong type of parameters or return value in method "methodname" of class "classname".` 存在错误类型
- `Failed, duplicated method "methodname" in class "classname".` 存在重复操作

说明：

- 对于参数类型
    - `NamedType`只考虑JAVA 语言八大基本类型(`byte`，`short`，`int`，`long`，`float`，`double`，`char`，`boolean`)和`String`，其余一律视为错误类型。
    - `ReferenceType`指向已定义的类或接口，类型名为对应的类名或接口名。
- 对于返回值类型
    - `NamedType`只考虑JAVA 语言八大基本类型(`byte`，`short`，`int`，`long`，`float`，`double`，`char`，`boolean`)和`String`，`void`，其余一律视为错误类型。（实际上，`void`也算是一种类型，C/C++/Java对于这件事也都是这样的定义）。`void`不等同于无返回值。
    - `ReferenceType`指向已定义的类或接口，类型名为对应的类名或接口名。
- 参数之间不分次序，即`op(int,int,double)`和`op(double,int,int)`视为具有相同参数类型，但参数和返回值之间是有区别的，且保证最多只有一个返回值，无返回值时相应位置返回**关键字**`null`。
- 如果两个操作的**操作名相同**，且**参数和返回值的类型也相同**，视为重复操作。
- 如果同时存在错误类型和重复操作两种异常，按错误类型异常处理。
- 本指令中统计的一律为此类自己定义的操作，不包含继承自其各级父类所定义的操作。

类的关联的对端是哪些类

输入指令格式：`CLASS_ASSO_CLASS_LIST classname`

举例：`CLASS_ASSO_CLASS_LIST Elevator`

输出：

- `Ok, associated classes of class "classname" are (A, B, C).` A、B、C为类所有关联的对端的类名，其中
  - 传出列表时可以乱序，官方接口会自动进行排序（但是**需要编写者自行保证不重不漏**；**特别的，对于同名但id不同的类，如果结果同时包含多个的话，需要在列表中返回对应数量个类名**）
  - 如果出现自关联的话，那么自身类也需要加入输出
- `Failed, class "classname" not found.` 类不存在
- `Failed, duplicated class "classname".` 类存在多个

注意：

- 同上一条，Association关系需要考虑父类的继承。即，假如X类继承了Y类，那么Y的Association对端节点也属于X。
- 不考虑由属性和操作参数类型引起的关联。

类实现的全部接口

输入指令格式：`CLASS_IMPLEMENT_INTERFACE_LIST classname`

举例：`CLASS_IMPLEMENT_INTERFACE_LIST Taxi`

输出：

- `Ok, implement interfaces of class "classname" are (A, B, C).` A、B、C为继承的各个接口
  - 传出列表时可以乱序，官方接口会自动进行排序（但是需要编写者自行保证不重不漏）
  - 特别值得注意的是，无论是直接实现还是通过父类或者接口继承等方式间接实现，都算做实现了接口
- `Failed, class "classname" not found.` 类不存在
- `Failed, duplicated class "classname".` 类存在多个

类的顶级父类

输入指令格式：`CLASS_TOP_BASE classname`

举例：`CLASS_TOP_BASE AdvancedTaxi`

输出：

- `Ok, top base class of class "classname" is top_classname.` `top_classname`为顶级父类
- `Failed, class "classname" not found.` 类不存在
- `Failed, duplicated class "classname".` 类存在多个

说明：

- 具体来说，对于类X，如果Y为其顶级父类的话，则满足
  - X是Y的子类（此处特别定义，X也是X的子类）
  - 不存在类Z，使得Y是Z的子类

类是否违背信息隐藏原则

输入指令格式：`CLASS_INFO_HIDDEN classname`

举例：`CLASS_INFO_HIDDEN Taxi`

输出：

- `Yes, information of class "classname" is hidden.` 满足信息隐藏原则。
- `No, attribute xxx in xxx, xxx in xxx, are not hidden.` 不满足信息隐藏原则。
- `Failed, class "classname" not found.` 类不存在
- `Failed, duplicated class "classname".` 类存在多个

说明：

- 信息隐藏原则，指的是**在类属性的定义中，不允许使用private以外的任何可见性修饰**
- 本指令中需要列出全部的非隐藏属性，同时也需要考虑继承自父类的非隐藏属性
- 值得注意的是，父类和子类中，是可以定义同名属性的（甚至还可以不同类型，不同可见性，感兴趣的话可以自己尝试尝试），然而**父类中定义的和子类中定义的实际上并不是同一个属性，需要在输出时进行分别处理**
- 同样的，返回的列表可以乱序，官方接口会进行自动排序（但是依然需要编写者保证不重不漏）

关于UML状态图的查询指令

给定状态机模型中一共有多少个状态

输入指令格式：`STATE_COUNT statemachine_name`

举例：`STATE_COUNT complex_sm`

输出：

- `Ok，state count of statemachine "complex_sm" is x.` x为应状态机模型complex_sm的状态总数.
- `Failed, statemachine "complex_sm" not found.` 未找到状态机模型complex_sm
- `Failed, duplicated statemachine "complex_sm".` 存在多个状态机模型complex_sm

说明：

- Initial State 和 Final State均算作状态。

给定状态机模型和其中的一个状态，有多少个不同的后继状态

输入指令格式：`SUBSEQUENT_STATE_COUNT statemachine_name statename`

举例：`SUBSEQUENT_STATE_COUNT complex_sm openned`

输出：

- `Ok, subsequent state count from state "openned" in statemachine "complex_sm" is x.` `x` 为状态机模型complex_sm中从openned状态可达的不同状态个数
- `Failed, statemachine "complex_sm" not found.` 未找到状态机模型complex_sm
- `Failed, duplicated statemachine "complex_sm".` 存在多个状态机模型complex_sm
- `Failed, state "openned" in statemachine "complex_sm" not found.` 在状态机模型complex_sm中未找到状态openned
- `Failed, duplicated state "openned" in statemachine "complex_sm".` 在状态机模型complex_sm中存在多个openned状态

说明：

- Initial State 和 Final State均算作状态。

给定状态机模型中和其中两个状态，引起状态迁移的所有触发事件

输入指令格式：`TRANSITION_TRIGGER statemachine_name statename1 statename2`

举例：`TRANSITION_TRIGGER door_sm open close`

输出：

- `Ok, triggers of transition from state "open" to state "close" in statemachine "door_sm" are (A, B, C).` A、B、C为应状态机模型door_sm中引起状态open迁移到状态close的触发事件，其中
  - 传出列表时可以乱序，官方接口会自动进行排序（但是需要编写者自行保证不重不漏）
  - 保证所有的触发事件名称都不相同，且不为空
- `Failed, statemachine "door_sm" not found.` 未找到状态机模型door_sm
- `Failed, duplicated statemachine "door_sm".` 存在多个状态机模型door_sm
- `Failed, state "open" in statemachine "door_sm" not found.` 在状态机模型door_sm中未找到状态open
- `Failed, duplicated state "open" in statemachine "door_sm".` 在状态机模型door_sm中存在多个open状态
- `Failed, transition from state "open" to state "close" in statemachine "door_sm" not found.` 在状态机模型door_sm中未找到从状态open到状态close的迁移

关于UML顺序图的查询指令

给定UML顺序图，一共有多少个参与对象

输入指令格式：`PTCP_OBJ_COUNT umlinteraction_name`

举例：`PTCP_OBJ_COUNT normal`

输出：

- `Ok, participant count of umlinteraction "normal" is x.` `x`为顺序图模型normal（UMLInteraction）中的参与对象个数（UMLLifeline）
- `Failed, umlinteraction "normal" not found.` 不存在normal这个顺序图模型
- `Failed, duplicated umlinteraction "normal".` 存在多个normal顺序图模型

给定UML顺序图和参与对象，有多少个incoming消息

输入指令格式：`INCOMING_MSG_COUNT umlinteraction_name lifeline_name`

举例：`INCOMING_MSG_COUNT normal door`

输出：

- `Ok, incoming message count of lifeline "door" in umlinteraction "normal" is x.` x为顺序图模型normal（UMLInteraction）中发送给door的消息个数
- `Failed, umlinteraction "normal" not found.` 不存在normal这个顺序图模型
- `Failed, duplicated umlinteraction "normal".` 存在多个normal顺序图模型
- `Failed, lifeline "door" in umlinteraction "normal" not found.`在顺序图模型normal中未找到参与对象door
- `Failed, duplicated lifeline "door" in umlinteraction "normal".`在顺序图模型normal中存在多个door参与对象

注意：

- 这里的UMLInteraction指UML所定义的一个类型

给定UML顺序图和参与对象，发出了多少个消息

输入指令格式：`SENT_MESSAGE_COUNT umlinteraction_name lifeline_name messagesort`

举例：`SENT_MESSAGE_COUNT normal door SYNCH_CALL`

输出：

- `Ok, sent message count of lifeline "door" of umlinteraction "normal" is x.` x为顺序图模型normal（UMLInteraction）中 door 发送的类别为 `SYNCH_CALL` 的消息个数
- `Failed, umlinteraction "normal" not found.` 不存在normal这个顺序图模型
- `Failed, duplicated umlinteraction "normal".` 存在多个normal顺序图模型
- `Failed, lifeline "door" in umlinteraction "normal" not found.`在顺序图模型normal中未找到参与对象door
- `Failed, duplicated lifeline "door" in umlinteraction "normal".`在顺序图模型normal中存在多个door参与对象

样例

模型部分导出自`mdj`文件：传送门，导出方法见官方包开源文档。

样例一

输入文本


```
{"_parent":"AAAAAAFF+qBWK6M3Z8Y=","visibility":"public","name":"Car","_type":"UM
LClass","_id":"AAAAAAF1SewAftFQQak="}
{"_parent":"AAAAAAF1SewAftFQQak=","visibility":"public","name":"fuel","_type":"U
MLOperation","_id":"AAAAAAF1Se7ei9PquT8="}
{"_parent":"AAAAAAF1Se7ei9PquT8=","name":"fuel","_type":"UMLParameter","_id":"AA
AAAAF1SfJIPdWQRq8=","type":"int","direction":"in"}
{"_parent":"AAAAAAF1SewAftFQQak=","visibility":"public","name":"move","_type":"U
MLOperation","_id":"AAAAAAF1SfM6ddXMG2g="}
{"_parent":"AAAAAAF1SfM6ddXMG2g=","name":"time","_type":"UMLParameter","_id":"AA
AAAAF1SfOEh9X3v2I=","type":"int","direction":"in"}
{"_parent":"AAAAAAF1SfM6ddXMG2g=","name":null,"_type":"UMLParameter","_id":"AAAA
AAF1SfON4NYM4oc=","type":"int","direction":"return"}
{"_parent":"AAAAAAF1SewAftFQQak=","name":null,"_type":"UMLGeneralization","_id":
"AAAAAAF1SffVQdb4hBU=","source":"AAAAAAF1SewAftFQQak=","target":"AAAAAAF1SeyladJ
Gxdo="}
{"_parent":"AAAAAAF1SewAftFQQak=","visibility":"private","name":"tyre","_type":"
UMLAttribute","_id":"AAAAAAF1SexBsdHNh3w=","type":"int"}
{"_parent":"AAAAAAF1SewAftFQQak=","visibility":"private","name":"fuel","_type":"
UMLAttribute","_id":"AAAAAAF1SfGLCdUvWpO=","type":"int"}
{"_parent":"AAAAAAFF+qBWK6M3Z8Y=","visibility":"public","name":"Ship","_type":"U
MLClass","_id":"AAAAAAF1Sewew9F5cgg="}
{"_parent":"AAAAAAF1Sewew9F5cgg=","visibility":"public","name":"move","_type":"U
MLOperation","_id":"AAAAAAF1SexK+dHTlvM="}
{"_parent":"AAAAAAF1SexK+dHTlvM=","name":"time","_type":"UMLParameter","_id":"AA
AAAAF1SfTvZdbYdII=","type":"int","direction":"in"}
{"_parent":"AAAAAAF1SexK+dHTlvM=","name":null,"_type":"UMLParameter","_id":"AAAA
AAF1SfTvZdbZrDA=","type":"int","direction":"return"}
{"_parent":"AAAAAAF1Sewew9F5cgg=","name":null,"_type":"UMLGeneralization","_id":
"AAAAAAF1Sff5gNcagM4=","source":"AAAAAAF1Sewew9F5cgg=","target":"AAAAAAF1SeyladJ
Gxdo="}
```

```
{"_parent":"AAAAAAF1Sewew9F5cgg=","visibility":"private","name":"displacement","
_type":"UMLAttribute","_id":"AAAAAAF1Sfbpj9buL6I=","type":"int"}
{"_parent":"AAAAAAFF+qBWK6M3Z8Y=","visibility":"public","name":"Plane","_type":"
UMLClass","_id":"AAAAAAF1Sewp6NGiMyM="}
{"_parent":"AAAAAAF1Sewp6NGiMyM=","visibility":"public","name":"move","_type":"U
MLOperation","_id":"AAAAAAF1SfVZqtbiHFQ="}
{"_parent":"AAAAAAF1SfVZqtbiHFQ=","name":"time","_type":"UMLParameter","_id":"AA
AAAAF1SfWNg9bpIpU=","type":"int","direction":"in"}
{"_parent":"AAAAAAF1SfVZqtbiHFQ=","name":null,"_type":"UMLParameter","_id":"AAAA
AAF1SfWNg9bqOqs=","type":"int","direction":"return"}
{"_parent":"AAAAAAF1Sewp6NGiMyM=","name":null,"_type":"UMLGeneralization","_id":
"AAAAAAF1SffpotcJOAw=","source":"AAAAAAF1Sewp6NGiMyM=","target":"AAAAAAF1SeyladJ
Gxdo="}
{"_parent":"AAAAAAFF+qBWK6M3Z8Y=","visibility":"public","name":"Vehicle","_type"
:"UMLClass","_id":"AAAAAAF1SeyladJGxdo="}
{"_parent":"AAAAAAF1SeyladJGxdo=","visibility":"public","name":"move","_type":"U
MLOperation","_id":"AAAAAAF1Se3ka9Nkpwc="}
{"_parent":"AAAAAAF1Se3ka9Nkpwc=","name":"time","_type":"UMLParameter","_id":"AA
AAAAF1SfASTdRtnh8=","type":"int","direction":"in"}
{"_parent":"AAAAAAF1Se3ka9Nkpwc=","name":null,"_type":"UMLParameter","_id":"AAAA
AAF1SfDxv9S94zk=","type":"int","direction":"return"}
{"_parent":"AAAAAAF1SeyladJGxdo=","name":null,"_type":"UMLAssociation","end2":"A
AAAAAF1Sgmletih6WY=","end1":"AAAAAAF1SgmletigGMU=","_id":"AAAAAAF1SgmletifmSI="}
{"reference":"AAAAAAF1SfuXJ9dbf2k=","multiplicity":"","_parent":"AAAAAAF1Sgmleti
fmSI=","visibility":"public","name":null,"_type":"UMLAssociationEnd","aggregatio
n":"none","_id":"AAAAAAF1Sgmletih6WY="}
{"reference":"AAAAAAF1SeyladJGxdo=","multiplicity":"","_parent":"AAAAAAF1Sgmleti
fmSI=","visibility":"public","name":null,"_type":"UMLAssociationEnd","aggregatio
n":"none","_id":"AAAAAAF1SgmletigGMU="}
{"_parent":"AAAAAAF1SeyladJGxdo=","visibility":"private","name":"speed","_type":
"UMLAttribute","_id":"AAAAAAF1Se8ZiNQacR4=","type":"int"}
{"_parent":"AAAAAAFF+qBWK6M3Z8Y=","visibility":"public","name":"Customer","_type
":"UMLClass","_id":"AAAAAAF1SfuXJ9dbf2k="}
{"_parent":"AAAAAAF1SfuXJ9dbf2k=","visibility":"public","name":"order","_type":"
UMLOperation","_id":"AAAAAAF1Sf0J69ebxDg="}
{"_parent":"AAAAAAF1Sf0J69ebxDg=","name":"from","_type":"UMLParameter","_id":"AA
AAAAF1Sf1R6Nei26k=","type":"int","direction":"in"}
{"_parent":"AAAAAAF1Sf0J69ebxDg=","name":"to","_type":"UMLParameter","_id":"AAAA
AAF1Sf2yldelIqc=","type":"int","direction":"in"}
{"_parent":"AAAAAAF1SfuXJ9dbf2k=","name":null,"_type":"UMLAssociation","end2":"A
AAAAAF1SgmC39gGGEo=","end1":"AAAAAAF1SgmC39gFR7s=","_id":"AAAAAAF1SgmC39gEcNc="}
{"reference":"AAAAAAF1SgUupNewiIs=","multiplicity":"","_parent":"AAAAAAF1SgmC39g
EcNc=","visibility":"public","name":null,"_type":"UMLAssociationEnd","aggregatio
n":"none","_id":"AAAAAAF1SgmC39gGGEo="}
{"reference":"AAAAAAF1SfuXJ9dbf2k=","multiplicity":"","_parent":"AAAAAAF1SgmC39g
EcNc=","visibility":"public","name":null,"_type":"UMLAssociationEnd","aggregatio
n":"none","_id":"AAAAAAF1SgmC39gFR7s="}
{"_parent":"AAAAAAFF+qBWK6M3Z8Y=","visibility":"public","name":"Platform","_type
":"UMLClass","_id":"AAAAAAF1SgUupNewiIs="}
{"_parent":"AAAAAAF1SgUupNewiIs=","visibility":"public","name":"handle","_type":
"UMLOperation","_id":"AAAAAAF1SgbuaNfllts="}
{"_parent":"AAAAAAF1SgbuaNfllts=","name":"from","_type":"UMLParameter","_id":"AA
AAAAF1Sge6PtfsbRU=","type":"int","direction":"in"}
{"_parent":"AAAAAAF1SgbuaNfllts=","name":"to","_type":"UMLParameter","_id":"AAAA
AAF1Sge6P9ftHNE=","type":"int","direction":"in"}
{"_parent":"AAAAAAF1SgbuaNfllts=","name":null,"_type":"UMLParameter","_id":"AAAA
AAF1Sge6P9fuZwg=","type":{"$ref":"AAAAAAF1SeyladJGxdo="},"direction":"return"}
```

{"_parent":"AAAAAAF1SgUupNewiIs=","name":null,"_type":"UMLAssociation","end2":"AAAAAAF1SgmZrNhPleA=","end1":"AAAAAAF1SgmZrNhO9XU=","_id":"AAAAAAF1SgmZq9hNwmU="}
{"reference":"AAAAAAF1SeyladJGxdo=","multiplicity":"","_parent":"AAAAAAF1SgmZq9hNwmU=","visibility":"public","name":null,"_type":"UMLAssociationEnd","aggregation":"none","_id":"AAAAAAF1SgmZrNhPleA="}
{"reference":"AAAAAAF1SgUupNewiIs=","multiplicity":"","_parent":"AAAAAAF1SgmZq9hNwmU=","visibility":"public","name":null,"_type":"UMLAssociationEnd","aggregation":"none","_id":"AAAAAAF1SgmZrNhO9XU="}
{"_parent":"AAAAAAF1SfhhWdcrp2I=","visibility":"public","name":"Interaction1","_type":"UMLInteraction","_id":"AAAAAAF1SfhhWdcs5+o="}
{"messageSort":"synchCall","_parent":"AAAAAAF1SfhhWdcs5+o=","visibility":"public","name":"handle","_type":"UMLMessage","_id":"AAAAAAF1SgyU59xUhN8=","source":"AAAAAAF1SgvKtdwNVp4=","target":"AAAAAAF1SgwAl9wuDMI="}
{"messageSort":"createMessage","_parent":"AAAAAAF1SfhhWdcs5+o=","visibility":"public","name":"create","_type":"UMLMessage","_id":"AAAAAAF1Sg2Dndxsrho=","source":"AAAAAAF1SgwAl9wuDMI=","target":"AAAAAAF1SfhOBNc7F4A="}
{"messageSort":"reply","_parent":"AAAAAAF1SfhhWdcs5+o=","visibility":"public","name":"returnVehicle","_type":"UMLMessage","_id":"AAAAAAF1Sg9L19ylGCg=","source":"AAAAAAF1SgwAl9wuDMI=","target":"AAAAAAF1SgvKtdwNVp4="}
{"messageSort":"synchCall","_parent":"AAAAAAF1SfhhWdcs5+o=","visibility":"public","name":"run","_type":"UMLMessage","_id":"AAAAAAF1Sg\/PTdy7EaU=","source":"AAAAAAF1SgvKtdwNVp4=","target":"AAAAAAF1SfhOBNc7F4A="}
{"messageSort":"reply","_parent":"AAAAAAF1SfhhWdcs5+o=","visibility":"public","name":"distance","_type":"UMLMessage","_id":"AAAAAAF1ShAR+9zRjZO=","source":"AAAAAAF1SfhOBNc7F4A=","target":"AAAAAAF1SgvKtdwNVp4="}
{"_parent":"AAAAAAF1SfhhWdcs5+o=","visibility":"public","name":"car","_type":"UMLLifeline","isMultiInstance":false,"_id":"AAAAAAF1SfhOBNc7F4A=","represent":"AAAAAAF1SfhOBNc6x94="}
{"_parent":"AAAAAAF1SfhhWdcs5+o=","visibility":"public","name":"customer","_type":"UMLLifeline","isMultiInstance":false,"_id":"AAAAAAF1SgvKtdwNVp4=","represent":"AAAAAAF1SgvKtdwMsZs="}
{"_parent":"AAAAAAF1SfhhWdcs5+o=","visibility":"public","name":"platform","_type":"UMLLifeline","isMultiInstance":false,"_id":"AAAAAAF1SgwAl9wuDMI=","represent":"AAAAAAF1SgwAl9wtEmA="}
END_OF_MODEL
INCOMING_MSG_COUNT Interaction1 customer
PTCP_OBJ_COUNT Interaction1
SENT_MESSAGE_COUNT Interaction1 customer SYNCH_CALL

输出文本

```
Ok, incoming message count of lifeline "customer" in umlinteraction
"Interaction1" is 2.
Ok, participant count of umlinteraction "Interaction1" is 3.
Ok, sent message count of lifeline "customer" of umlinteraction "Interaction1" is
2.
```

样例二

输入文本

{"_parent":"AAAAAAFF+h6SjaM2Hec=","name":"Parser","_type":"UMLStateMachine","_id":"AAAAAAFrXjtGRjOR9DA="}
{"_parent":"AAAAAAFrXjtGRjOR9DA=","visibility":"public","name":null,"_type":"UMLRegion","_id":"AAAAAAFrXjtGRjOS988="}

{"_parent":"AAAAAAFrXjtGRjOS988=","visibility":"public","name":null,"_type":"UML Pseudostate","_id":"AAAAAAFrXjtjdjOYbaE="}
{"_parent":"AAAAAAFrXjtGRjOS988=","visibility":"public","name":"Lex","_type":"UML State","_id":"AAAAAAFrXjtwqzOpfoU="}
{"_parent":"AAAAAAFrXjtGRjOS988=","visibility":"public","name":null,"_type":"UML FinalState","_id":"AAAAAAFrXju6jTPPllU="}
{"_parent":"AAAAAAFrXjtGRjOS988=","visibility":"public","name":"Parse","_type":"UMLState","_id":"AAAAAAFrXj5brDSMY1o="}
{"_parent":"AAAAAAFrXjtGRjOS988=","visibility":"public","name":"Optimize","_type":"UMLState","_id":"AAAAAAFrXkCzGTUIiOQ="}
{"_parent":"AAAAAAFrXjtGRjOS988=","visibility":"public","guard":null,"name":null,"_type":"UMLTransition","_id":"AAAAAAFrXjOePzQ6NBA=","source":"AAAAAAFrXjtjdjOYbaE=","target":"AAAAAAFrXjtwqzOpfoU="}
{"_parent":"AAAAAAFrXjOePzQ6NBA=","expression":null,"visibility":"public","name":"feedLex","_type":"UMLEvent","_id":"AAAAAAFrXj1OBTRMqDA=","value":null}
{"_parent":"AAAAAAFrXjtGRjOS988=","visibility":"public","guard":null,"name":null,"_type":"UMLTransition","_id":"AAAAAAFrXj4UnjR1h3o=","source":"AAAAAAFrXjtwqzOpfoU=","target":"AAAAAAFrXju6jTPPllU="}
{"_parent":"AAAAAAFrXj4UnjR1h3o=","expression":null,"visibility":"public","name":"lexFail","_type":"UMLEvent","_id":"AAAAAAFrXj4dJjSH2Do=","value":null}
{"_parent":"AAAAAAFrXjtGRjOS988=","visibility":"public","guard":null,"name":null,"_type":"UMLTransition","_id":"AAAAAAFrXj+1eDTXIog=","source":"AAAAAAFrXjtwqzOpfoU=","target":"AAAAAAFrXj5brDSMY1o="}
{"_parent":"AAAAAAFrXj+1eDTXIog=","expression":null,"visibility":"public","name":"feedParse","_type":"UMLEvent","_id":"AAAAAAFrXj\/G1TTpMJA=","value":null}
{"_parent":"AAAAAAFrXjtGRjOS988=","visibility":"public","guard":null,"name":null,"_type":"UMLTransition","_id":"AAAAAAFrXkAB8DTt8dU=","source":"AAAAAAFrXj5brDSMY1o=","target":"AAAAAAFrXju6jTPPllU="}
{"_parent":"AAAAAAFrXkAB8DTt8dU=","expression":null,"visibility":"public","name":"parseFail","_type":"UMLEvent","_id":"AAAAAAFrXkAHrDT\/Ojs=","value":null}
{"_parent":"AAAAAAFrXjtGRjOS988=","visibility":"public","guard":null,"name":null,"_type":"UMLTransition","_id":"AAAAAAFrXkDeGTUvgV8=","source":"AAAAAAFrXj5brDSMY1o=","target":"AAAAAAFrXkCzGTUIiOQ="}
{"_parent":"AAAAAAFrXkDeGTUvgV8=","expression":null,"visibility":"public","name":"feedOptimize","_type":"UMLEvent","_id":"AAAAAAFrXkDsnjVBOlQ=","value":null}
{"_parent":"AAAAAAFrXjtGRjOS988=","visibility":"public","guard":null,"name":null,"_type":"UMLTransition","_id":"AAAAAAFrXkGVRDVKmDc=","source":"AAAAAAFrXkCzGTUIiOQ=","target":"AAAAAAFrXju6jTPPllU="}
{"_parent":"AAAAAAFrXkGVRDVKmDc=","expression":null,"visibility":"public","name":"generate","_type":"UMLEvent","_id":"AAAAAAFrXkGh1TVctds=","value":null}
```
END_OF_MODEL
STATE_COUNT Parser
SUBSEQUENT_STATE_COUNT Parser Lex
TRANSITION_TRIGGER Parser Lex Parse
```

输出文本

```
Ok, state count of statemachine "Parser" is 5.
Ok, subsequent state count from state "Lex" in statemachine "Parser" is 3.
Ok, triggers of transition from state "Lex" to state "Parse" in statemachine
"Parser" are (feedParse).
```

判定

公测（包括弱测、中测与强测）数据基本限制

- `mdj`文件内容限制
  - 包含类图，类图在`UMLModel`内进行建模，且每个`UMLModel`内的元素不会引用当前`UMLModel`以外的元素（即关系是一个闭包）
  - 包含顺序图，与UMLModel平级，可能会引用到UMLModel中的模型元素
  - 包含状态图，一定处于UMLClass下面的层次，不会引用UMLModel中的其他模型元素
  - **原始mdj文件仅通过staruml工具建模生成**（不存在手改json等行为）
  - 原始mdj文件符合`starUML`规范，可在`starUML`中正常打开和显示
  - `mdj`文件中最多只包含`400`个元素
  - **此外为了方便本次的情况处理，保证所建模的模型均可以在不与前面所述的规定相矛盾的情况下，在Oracle Java 8中正常实现出来**
- 输入指令限制
  - 最多不超过300条指令
  - 输入指令满足标准格式
- 为了确保测试数据的合理性，测试数据有如下限制
  - 所有公测数据不会对
    - 接口中定义的属性
    - 类属性（static attribute）
    - 类方法（static method）
    - 做任何测试要求，本次作业不需要对这些情况进行考虑。
  - 类图相关
    - 确保数据中没有类的多继承，但可能出现接口的多继承（即与Java 8规范相同）。
  - 状态图相关
    - 确保每个State Machine中有且仅有一个Region；
    - 确保每个State Machine中最多只有一个Initial State，最多只有一个Final State；
    - 确保每个State Machine中所有状态均不重名；
    - 确保每个State Machine中Initial State和Final State的name均为null，查询指令中给出的状态也不会为Initial State或Final State；
    - 确保每个State Machine中，从某个状态到另一个状态的直接迁移均具有不同的Event或Guard，即从某个状态到另一个状态的直接迁移若有多个，则这些迁移一定互不相同；
    - 确保每个State Machine中，Initial State没有状态迁入，Final State没有状态迁出；
    - 确保每个State Machine中不包含复合状态。
  - 顺序图相关
    - 确保每个顺序图中，Lifeline和其Represent均一一对应。
- 我们保证，公测中的所有数据均满足以上基本限制。

测试模式

公测均通过标准输入输出进行。

指令将会通过查询UML各种信息的正确性，从而测试UML解析器各个接口的实现正确性。

对于任何满足基本数据限制的输入，程序都应该保证不会异常退出，如果出现问题则视为未通过该测试点。

程序运行的最大CPU时间为`10s`，保证强测数据有一定梯度。

提示&说明

- 本次作业中可以自行组织工程结构。任意新增`java`代码文件。只需要保证`UmlInteraction`类的继承与实现即可。
- **关于本次作业解析器类的设计具体细节，本指导书中均不会进行过多描述，请自行去官方包开源仓库中查看接口的规格，并依据规格进行功能的具体实现**，必要时也可以查看AppRunner的代码实现。关于官方包的使用方法，可以去查看开源库的`README.md`。
- 如果还有人不知道标准输入、标准输出是啥的话，那在这里解释一下
  - 标准输入，直观来说就是屏幕输入
  - 标准输出，直观来说就是屏幕输出
  - 标准异常，直观来说就是报错的时候那堆红字
  - 想更加详细的了解的话，请去百度
- 关于作业中的一些问题，在此进行统一的补充说明：
  - 对于基于类的查询，**除非明确表示查询类与接口，否则一律只针对类(UMLClass)进行查询。**
  - 对于所有的异常抛出，应该这样去思考：
    - 通过读代码，搞明白相关异常是什么样的意义
    - 通过读代码，搞明白抛出去的异常会被以怎么样的形式进行使用
    - 比如，十分显然的
      - **对于Duplicated一类的异常**，表示且仅表示当根据仅有的输入无法唯一确定需要查询的对象时（**即符合条件的超过1个**），所需要抛出的异常
      - **对于NotFound一类的异常**，表示且仅表示当根据仅有的输入无法找到需要查询的对象时（**即符合条件的为0个**），所需要抛出的异常
      - 以及，异常中所需要传入的类名之类的值，是用来输出的。**所以查询的输入值是什么，就传入什么，以保证和输入信息的对应性。**
- **如果同时满足多个异常，在查询上层模型发生“异常”后，我们自然不该再去查询这个“异常层次”的下层次模型。**
- 开源库地址
- 推荐各位同学在课下测试时使用Junit单元测试来对自己的程序进行测试
  - Junit是一个单元测试包，**可以通过编写单元测试类和方法，来实现对类和方法实现正确性的快速检查和测试。**还可以查看测试覆盖率以及具体覆盖范围（精确到语句级别），以帮助编程者全面无死角的进行程序功能测试。
  - Junit已在评测机中部署（版本为Junit4.12，一般情况下确保为Junit4即可），所以项目中可以直接包含单元测试类，在评测机上不会有编译问题。
  - 此外，Junit对主流Java IDE（Idea、eclipse等）均有较为完善的支持，可以自行安装相关插件。推荐两篇博客：
    - Idea下配置Junit
    - Idea下Junit的简单使用
  - 感兴趣的同学可以自行进行更深入的探索，百度关键字：`Java Junit`。
- 强烈推荐同学们
  - 去阅读本次的源代码
  - **去好好复习下本次和上次的ppt，并理清楚各个`UmlElement`数据模型的结构与关系。**
- **不要试图通过反射机制来对官方接口进行操作**，我们有办法进行筛查。此外，如果发现有人试图通过反射等手段hack输出接口的话，请加助教微信进行举报，**经核实后，将直接作为无效作业处理。**