UML第一次作业接口使用说明

本次我们继续像是之前一样，提供封装好的jar包给大家。

这次的话，**我们已经将全部的主干业务逻辑进行了封装，只需要同学们实现一个核心交互类即可**。

除此之外，本次的**官方包还可以作为命令行工具使用，以便快速从`mdj`文件中导出并生成输入数据**。

功能实现

UmlInteraction接口

学生需要实现一个自己的`UmlInteraction`类，这个类必须继承接口`com.oocourse.uml1.interact.format.UmlInteraction`。

```
import com.oocourse.uml1.interact.format.UmlInteraction;

public class MyUmlInteraction implements UmlInteraction {
    // TODO : IMPLEMENT
}
```

接口源码设定：

```
package com.oocourse.uml1.interact.format;

import com.oocourse.uml1.interact.common.AttributeClassInformation;
import com.oocourse.uml1.interact.common.AttributeQueryType;
import com.oocourse.uml1.interact.common.OperationQueryType;
import com.oocourse.uml1.interact.exceptions.user.AttributeDuplicatedException;
import com.oocourse.uml1.interact.exceptions.user.AttributeNotFoundException;
import com.oocourse.uml1.interact.exceptions.user.ClassDuplicatedException;
import com.oocourse.uml1.interact.exceptions.user.ClassNotFoundException;
import com.oocourse.uml1.models.common.Visibility;

import java.util.List;
import java.util.Map;

/**
 * UML交互接口
 */
@SuppressWarnings("unused")
public interface UmlInteraction {
    /**
     * 获取类数量
     * 指令: CLASS_COUNT
     *
     * @return 类数量
     */
    int getClassCount();

    /**
     * 获取类操作数量
     * 指令: CLASS_OPERATION_COUNT
```

```
     *
     * @param className 类名
     * @return 类的操作数量
     * @throws ClassNotFoundException      类未找到异常
     * @throws ClassDuplicatedException    类重复异常
     */
    int getClassOperationCount(String className)
        throws ClassNotFoundException, ClassDuplicatedException;

    /**
     * 获取类属性数量
     * 指令: CLASS_ATTR_COUNT
     *
     * @param className 类名
     * @return 类属性操作数量
     * @throws ClassNotFoundException    类未找到异常
     * @throws ClassDuplicatedException 类重复异常
     */
    int getClassAttributeCount(String className)
        throws ClassNotFoundException, ClassDuplicatedException;

    /**
     * 统计类操作可见性
     * 指令: CLASS_OPERATION_VISIBILITY
     *
     * @param className     类名
     * @param operationName 操作名
     * @return 类操作可见性统计结果
     * @throws ClassNotFoundException    类未找到异常
     * @throws ClassDuplicatedException 类重复异常
     */
    Map<Visibility, Integer> getClassOperationVisibility(String className,
String operationName)
        throws ClassNotFoundException, ClassDuplicatedException;

    /**
     * 获取类属性可见性
     * 指令: CLASS_ATTR_VISIBILITY
     *
     * @param className     类名
     * @param attributeName 属性名
     * @return 属性可见性
     * @throws ClassNotFoundException       类未找到异常
     * @throws ClassDuplicatedException     类重复异常
     * @throws AttributeNotFoundException   属性未找到异常
     * @throws AttributeDuplicatedException 属性重复异常
     */
    Visibility getClassAttributeVisibility(String className, String
attributeName)
        throws ClassNotFoundException, ClassDuplicatedException,
        AttributeNotFoundException, AttributeDuplicatedException;

    /**
     * 获取类属性类型
     * 指令: CLASS_ATTR_TYPE
     *
     * @param className     类名
     * @param attributeName 属性名
```

```
     * @return 属性类型
     * @throws ClassNotFoundException        类未找到异常
     * @throws ClassDuplicatedException      类重复异常
     * @throws AttributeNotFoundException    属性未找到异常
     * @throws AttributeDuplicatedException 属性重复异常
     * @throws AttributeWrongTypeException   属性类型错误异常
     */
    String getClassAttributeType(String className, String attributeName)
        throws ClassNotFoundException, ClassDuplicatedException,
        AttributeNotFoundException, AttributeDuplicatedException,
AttributeWrongTypeException;

    /**
     * 统计类操作参数类型
     * 指令: CLASS_OPERATION_PARAM_TYPE
     *
     * @param className     类名
     * @param operationName 操作名
     * @return 类操作参数类型列表
     * @throws ClassNotFoundException     类未找到异常
     * @throws ClassDuplicatedException   类重复异常
     * @throws MethodWrongTypeException   方法参数类型错误异常
     * @throws MethodDuplicatedException 方法重复异常
     */
    List<OperationParamInformation> getClassOperationParamType(
        String className, String operationName
    ) throws ClassNotFoundException, ClassDuplicatedException,
        MethodWrongTypeException, MethodDuplicatedException;

    /**
     * 获取与类相关联的类列表
     * 指令: CLASS_ASSO_CLASS_LIST
     *
     * @param className 类名
     * @return 与类关联的类列表
     * @throws ClassNotFoundException    类未找到异常
     * @throws ClassDuplicatedException 类重复异常
     */
    List<String> getClassAssociatedClassList(String className)
        throws ClassNotFoundException, ClassDuplicatedException;

    /**
     * 获取顶级父类
     * 指令: CLASS_TOP_BASE
     *
     * @param className 类名
     * @return 顶级父类名
     * @throws ClassNotFoundException    类未找到异常
     * @throws ClassDuplicatedException 类重复异常
     */
    String getTopParentClass(String className)
        throws ClassNotFoundException, ClassDuplicatedException;

    /**
     * 获取实现的接口列表
     * 指令: CLASS_IMPLEMENT_INTERFACE_LIST
     *
     * @param className 类名
```

```
     * @return 实现的接口列表
     * @throws ClassNotFoundException    类未找到异常
     * @throws ClassDuplicatedException 类重复异常
     */
    List<String> getImplementInterfaceList(String className)
        throws ClassNotFoundException, ClassDuplicatedException;

    /**
     * 获取类中未隐藏的属性
     * 即违背了面向对象设计中的隐藏信息原则的属性
     * 指令: CLASS_INFO_HIDDEN
     *
     * @param className 类名
     * @return 未隐藏的属性信息列表
     * @throws ClassNotFoundException    类未找到异常
     * @throws ClassDuplicatedException 类重复异常
     */
    List<AttributeClassInformation> getInformationNotHidden(String className)
        throws ClassNotFoundException, ClassDuplicatedException;
}
```

除此之外，`UmlInteraction`类必须实现一个构造方法

```
public class MyUmlInteraction implements UmlInteraction {
    public MyUmlInteraction(UmlElement[] elements);
}
```

或者

```
public class MyUmlInteraction implements UmlInteraction {
    public MyUmlInteraction(UmlElement... elements);
}
```

构造函数的逻辑为将`elements`数组内的各个UML类图元素传入`UmlInteraction`类，以备后续解析。

交互模式

交互的模式为：

- 调用上述构造函数，生成一个实例，并将UML模型元素传入。
- 之后将调用此实例的各个接口方法，以实现基于之前传入的UML模型元素的各类查询操作。
- 官方接口通过调用方法的返回值，自动生成对应的输出文本。

开始运行

运行的模式和之前基本类似：

```
import com.oocourse.uml1.interact.AppRunner;

public class Main {
    public static void main(String[] args) throws Exception {
        AppRunner appRunner = AppRunner.newInstance(MyUmlInteraction.class);
        appRunner.run(args);
    }
}
```

将自己实现的类进行载入，即可运行。

数据生成

命令行工具

和之前不同的是，此次的官方jar包还可以作为命令行工具使用，简单的几个用法如下。

查看可导出的数据模型列表

用户可以通过这一功能查看支持导出的数据模型列表

```
java -jar U4T1.jar list -s "open-close.mdj"
```

输出结果

```
|   Name    |    Type     |
|-----------|-------------|
| Model     | UMLModel    |
| Model1    | UMLModel    |
```

导出指定的数据模型

用户可以通过这一功能对数据模型进行导出。

导出的数据格式可以直接作为数据模型的输入内容，在其后接上`END_OF_MODEL`和各类指令，即可构建为一个输入数据。

```
java -jar U4T1.jar dump -s "open-close.mdj" -n Model1
```

输出结果


```
{"_parent":"AAAAAAFq3tvYM76UevI=","visibility":"public","name":"Key","_type":"U
MLClass","_id":"AAAAAAFq7weIMSb5xqQ="}
{"_parent":"AAAAAAFq7weIMSb5xqQ=","visibility":"public","name":"equals","_type"
:"UMLOperation","_id":"AAAAAAFq7weIMSb8qxc="}
{"_parent":"AAAAAAFq7weIMSb8qxc=","name":"o","_type":"UMLParameter","_id":"AAAA
AAFq7weIMSb9G0k=","type":"Object","direction":"in"}
{"_parent":"AAAAAAFq7weIMSb8qxc=","name":null,"_type":"UMLParameter","_id":"AAA
AAAFq7weIMSb+Au4=","type":"boolean","direction":"return"}
{"_parent":"AAAAAAFq7weIMSb5xqQ=","visibility":"public","name":"getMatchedLockI
d","_type":"UMLOperation","_id":"AAAAAAFq7weIMSb\/6gM="}
{"_parent":"AAAAAAFq7weIMSb\/6gM=","name":null,"_type":"UMLParameter","_id":"AA
AAAAFq7weIMScAoOk=","type":"int","direction":"return"}
{"_parent":"AAAAAAFq7weIMSb5xqQ=","visibility":"public","name":"Operation1","_t
ype":"UMLOperation","_id":"AAAAAAFq7w1zLCePJrI="}
{"_parent":"AAAAAAFq7w1zLCePJrI=","name":"Parameter1","_type":"UMLParameter","_
id":"AAAAAAFq7w2dZCeV4K8=","type":
{"$ref":"AAAAAAFq7weIMSb5xqQ="},"direction":"return"}
{"_parent":"AAAAAAFq7weIMSb5xqQ=","name":null,"_type":"UMLGeneralization","_id"
:"AAAAAAFq7wfNvyd+GgY=","source":"AAAAAAFq7weIMSb5xqQ=","target":"AAAAAAFq7weqo
CcQE7I="}
{"_parent":"AAAAAAFq7weIMSb5xqQ=","visibility":"private","name":"keyID","_type"
:"UMLAttribute","_id":"AAAAAAFq7weIMSb6+v8=","type":"int"}
```

```
{"_parent":"AAAAAAFq7weIMSb5xqQ=","visibility":"private","name":"matchedLockID"
,"_type":"UMLAttribute","_id":"AAAAAAFq7weIMSb7oPM=","type":"int"}
{"_parent":"AAAAAAFq3tvYM76UevI=","visibility":"public","name":"ElcKey","_type"
:"UMLClass","_id":"AAAAAAFq7weqoCcQE7I="}
{"_parent":"AAAAAAFq7weqoCcQE7I=","visibility":"public","name":"equals","_type"
:"UMLOperation","_id":"AAAAAAFq7weqoCcTngY="}
{"_parent":"AAAAAAFq7weqoCcTngY=","name":"o","_type":"UMLParameter","_id":"AAAA
AAFq7weqoCcUI6g=","type":"Object","direction":"in"}
{"_parent":"AAAAAAFq7weqoCcTngY=","name":null,"_type":"UMLParameter","_id":"AAA
AAAFq7weqoCcVxI0=","type":"boolean","direction":"return"}
{"_parent":"AAAAAAFq7weqoCcQE7I=","name":"sdfsdf","_type":"UMLGeneralization","
_id":"AAAAAAFq7weqoCcRDg8=","source":"AAAAAAFq7weqoCcQE7I=","target":"AAAAAAFqp
yZaw1HqYaU="}
{"_parent":"AAAAAAFq7weqoCcQE7I=","visibility":"private","name":"sigCode","_typ
e":"UMLAttribute","_id":"AAAAAAFq7weqoCcSu1Q=","type":"long"}
```


其他

其他的一些操作在此不做过多描述，欢迎各位通过`-h`（或`--help`）参数查看帮助并探索。

注意事项

- **请确保构造函数正确实现，且类和构造函数均定义为`public`**，否则将无法进行实例化。
- **请保证传入的类继承了`UmlInteraction`接口**，否则将无法载入。
- 此外，对于`ClassNotFoundException`（全称`com.oocourse.uml1.interact.exceptions.user.ClassNotFoundException`）等几个异常类，在Java的标准库里面有与之同名的类（全称`java.lang.ClassNotFoundException`）。**请各位在使用的时候注意甄别，以免误用**。
-

其他

- 如果还有不清楚的地方，**建议去阅读相关部分的源代码**（大部分地方均配有javadoc注释）
- 本次作业在输出层面上分为加密版和非加密版
  - 非加密版完全公开。
  - 加密版只在评测机上使用且闭源，会对输出进行一定程度的加密处理。
  - 所以，请不要试图伪造输出，还请使用我们的接口。
  - 不仅如此，加密版本次编译时加入了源码混淆选项，所有非public的字段、方法、类以及方法实现都会被混淆。
  - 所以，请不要试图通过反射来破解接口。发现此类情况，也可以直接举报。