MCCL Tool Manual

Technical Report

Duc Minh Le
duclm@hanu.edu.vn
Department Software Engineering, Hanoi University

1. Introduction

This manual describes how to run a tool that implements an annotation-based DSL named MCCL. MCCL is developed in a paper that we recently submitted to the KSE 2017 conference. It is implemented as part of a software tool named jDomainApp [1].

2. Installation

Download the MCCL binary from GitHub [2] and place it in a directory in the local hard drive. The binary basically includes a set of `jar` files that you can import and run directly in an IDE (e.g. Eclipse) or from the command line. Table 1 below describes the library files.

Note that the three programs that we will focus on in this guide are those that implement the MCC generator and the two MCC update functions that are discussed in the KSE 2017 paper. In this guide, we will explain how to run these programs from the command line.

Table 1: MCCL library files

Library file	Description
`domainapp.jar`	The `jDomainApp` tool [1]
`javaparser-core-3.2.5.jar`	The JavaParser tool [3]
`mccl.jar`	Implementation of MCCL and these three test programs for its three functions: • `MCCGen`: function MCCGen • `OnAddDomainFields`: function `OnAddDomainFields` • `OnUpdateDomainFields`: function `OnUpdateDomainFields`

3. Example: CourseMan

The source code folder of CourseMan is provided in the file named `courseman-examples.zip`. The content of this source code is shown in Figure 1. In particular, package `modulesgen` is used to run the program `MCCGen`; while package `modulesupdate` is used to run the two programs `OnAddDomainFields` and `OnUpdateDomainFields` (resp.)

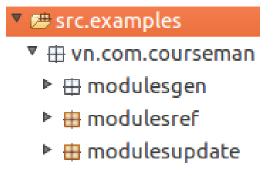


Figure 1: CourseMan source code example.

Denote `$MCCL_SRC_EXAMPLES` by the directory of this source code tree.

4. Running the tool

All the programs described in this section can optionally be executed with the system property: `-Ddebug=MCCModel`. This helps turn on additional output messages on the console.

4.1. MCCGen

Command

```
java -DrootSrcPath=$MCCL_SRC_EXAMPLES
  -cp lib/domainapp.jar:lib/javaparser-core-3.2.5.jar:lib/mccl.jar domainapp.modules.mccl.MCCGenTool
  <domain-class-FQN>
```

Where:

- `domain-class-FQN`: the FQN of domain class

Run output

In this example:

- `$MCCL_SRC_EXAMPLES = "/home/dmle/projects/domainapp/modules/mccl/src.examples"`
- `domain-class-FQN (1) = vn.com.courseman.modulesgen.student.model.Student`
- `domain-class-FQN (2) = vn.com.courseman.modulesgen.enrolmentmgmt.model.EnrolmentMgmt`

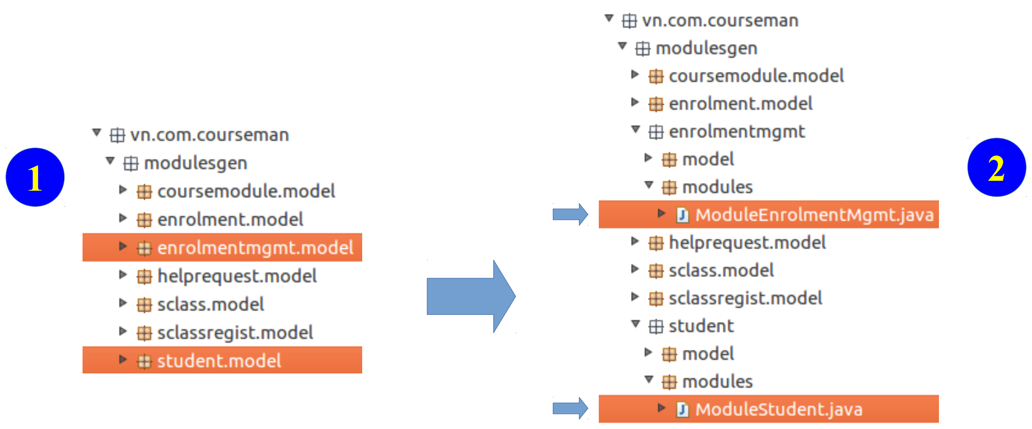
```
Running MCCGen...
   Domain class: vn.com.courseman.modulesgen.student.model.Student
   Source file:
/home/dmle/projects/domainapp/modules/mccl/src.examples/vn/com/courseman/modulesgen/student/model/Student.java
   Root class output dir: /home/dmle/projects/domainapp/modules/mccl/src.examples
...ok
```

```
Running MCCGen...
   Domain class: vn.com.courseman.modulesgen.enrolmentmgmt.model.EnrolmentMgmt
   Source file:
/home/dmle/projects/domainapp/modules/mccl/src.examples/vn/com/courseman/modulesgen/enrolmentmgmt/model/EnrolmentMgmt
.java
   Root class output dir: /home/dmle/projects/domainapp/modules/mccl/src.examples
...ok
```

Result

(1) Packages `vn.com.courseman.modulesgen.student` and `vn.com.courseman.modulesgen.enrolmentmgmt` ***before*** executing `MCCGen` on `Student` and `EnrolementMgmt` contain no MCCs.

(2) Packages `vn.com.courseman.modulesgen.student` and `vn.com.courseman.modulesgen.enrolmentmgmt` ***after*** executing `MCCGen` on `Student` and `EnrolementMgmt`: contain the MCCs `ModuleEnrolmentMgmt` and `ModuleStudent` (resp.)



4.2. OnAddDomainFields

Command

```
java -DrootSrcPath=$MCCL_SRC_EXAMPLES
  -cp lib/domainapp.jar:lib/javaparser-core-3.2.5.jar:lib/mccl.jar domainapp.modules.mccl.test.MCCUpdateTool
  add <domain-class-FQN> <update-spec>
```

Where:

- `domain-class-FQN`: the FQN of domain class
- `update-spec`: comma-separated list of new field names (e.g. "test1,test2")

Run output

In this example:

- `$MCCL_SRC_EXAMPLES = "/home/dmle/projects/domainapp/modules/mccl/src.examples"`
- `domain-class-FQN = vn.com.courseman.modulesupdate.student.model.Student`
- `update-spec = "test1,test2"`

Preparation

Add two test domain fields to class `vn.com.courseman.modulesupdate.student.model.Student` as shown below. We expect that after executing this program, class `vn.com.courseman.modulesupdate.student.model.ModuleStudent` will have two view fields that reflect these two new domain fields.

```
////// NEW FIELDS TO TEST ///////
@DAttr(name="test1",type=Type.String, serialisable=false)
private String test1;

@DAttr(name="test2",type=Type.String, serialisable=false)
private String test2;

////// END new fields ///////
```

Command output

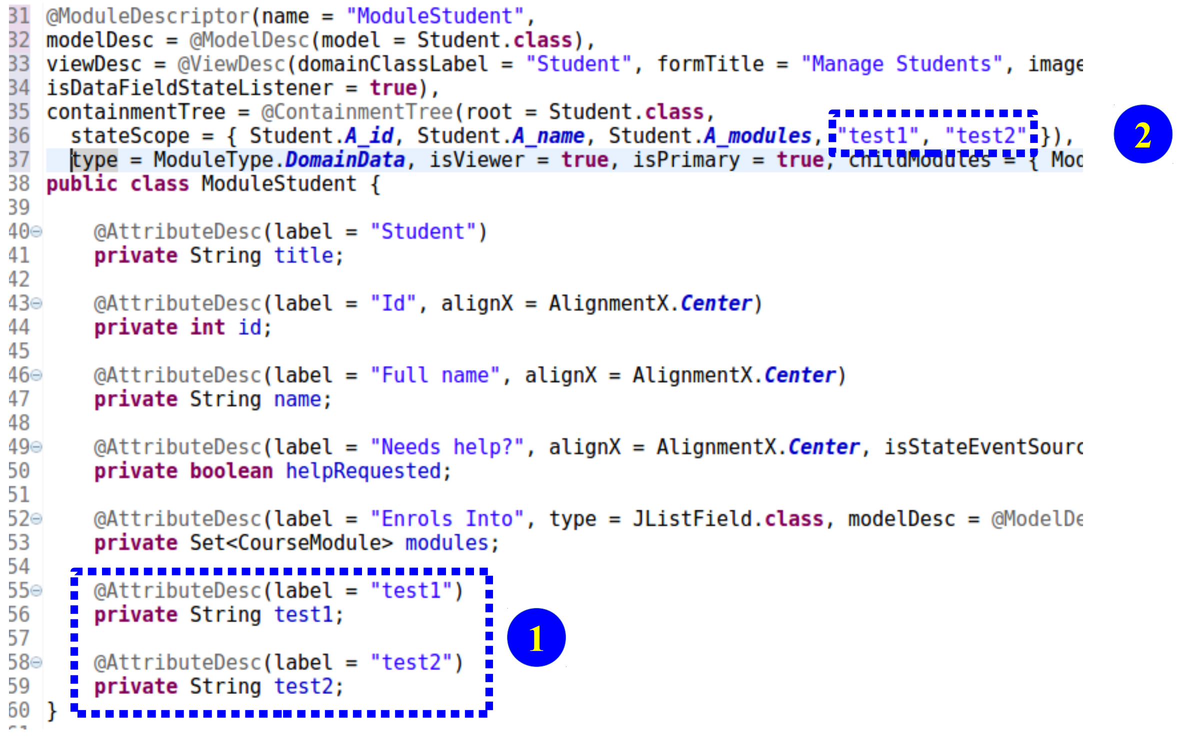
```
Creating KSE-2017 MCCModel...

--- MCCs ---

[vn.com.courseman.modulesupdate.student.model.Student, vn.com.courseman.modulesupdate.student.ModuleStudent]

[vn.com.courseman.modulesupdate.enrolmentmgmt.model.EnrolmentMgmt,
vn.com.courseman.modulesupdate.enrolmentmgmt.ModuleEnrolmentMgmt]


Running OnAddDomainFields...

   Domain class: vn.com.courseman.modulesupdate.student.model.Student

   New fields: [test1, test2]

...ok
```

Result

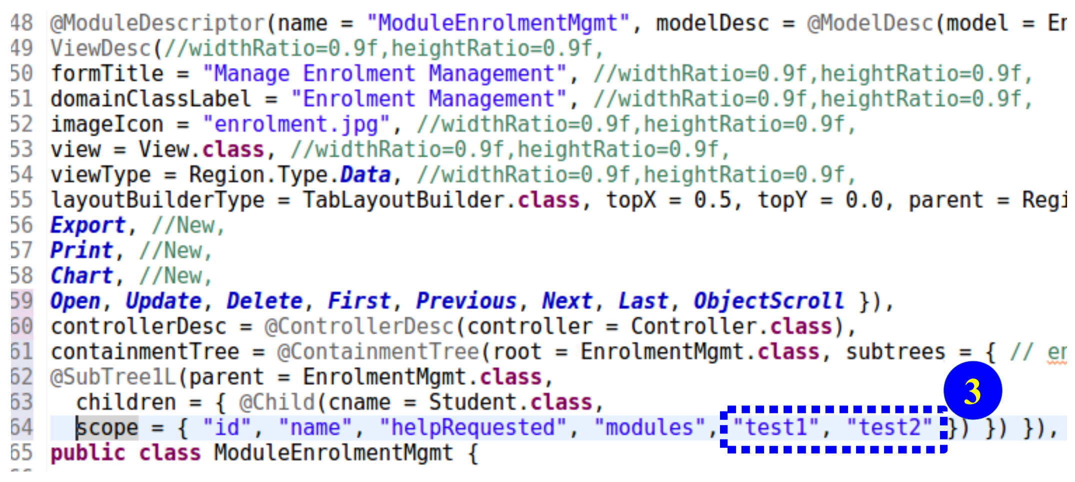
(1) Class `ModuleStudent` is updated with two new view fields (`test1`, `test2`).

(2) Property `ModuleDescriptor.containmentTree.stateScope` of `ModuleStudent` is also updated with names of these two view fields.

(3) Containment scope of `Student` in `ModuleEnrolmentMgmt` is updated with names of two new view fields



```
@ModuleDescriptor(name = "ModuleStudent",
modelDesc = @ModelDesc(model = Student.class),
viewDesc = @ViewDesc(domainClassLabel = "Student", formTitle = "Manage Students", image
isDataFieldStateListener = true),
containmentTree = @ContainmentTree(root = Student.class,
  stateScope = { Student.A_id, Student.A_name, Student.A_modules, "test1", "test2"}),
  type = ModuleType.DomainData, isViewer = true, isPrimary = true, childModules = { Moc
public class ModuleStudent {

    @AttributeDesc(label = "Student")
    private String title;

    @AttributeDesc(label = "Id", alignX = AlignmentX.Center)
    private int id;

    @AttributeDesc(label = "Full name", alignX = AlignmentX.Center)
    private String name;

    @AttributeDesc(label = "Needs help?", alignX = AlignmentX.Center, isStateEventSourc
    private boolean helpRequested;

    @AttributeDesc(label = "Enrols Into", type = JListField.class, modelDesc = @ModelDe
    private Set<CourseModule> modules;

    @AttributeDesc(label = "test1")
    private String test1;

    @AttributeDesc(label = "test2")
    private String test2;
}
```



```
@ModuleDescriptor(name = "ModuleEnrolmentMgmt", modelDesc = @ModelDesc(model = Er
ViewDesc(//widthRatio=0.9f,heightRatio=0.9f,
formTitle = "Manage Enrolment Management", //widthRatio=0.9f,heightRatio=0.9f,
domainClassLabel = "Enrolment Management", //widthRatio=0.9f,heightRatio=0.9f,
imageIcon = "enrolment.jpg", //widthRatio=0.9f,heightRatio=0.9f,
view = View.class, //widthRatio=0.9f,heightRatio=0.9f,
viewType = Region.Type.Data, //widthRatio=0.9f,heightRatio=0.9f,
layoutBuilderType = TabLayoutBuilder.class, topX = 0.5, topY = 0.0, parent = Regi
Export, //New,
Print, //New,
Chart, //New,
Open, Update, Delete, First, Previous, Next, Last, ObjectScroll }),
controllerDesc = @ControllerDesc(controller = Controller.class),
containmentTree = @ContainmentTree(root = EnrolmentMgmt.class, subtrees = { // er
@SubTree1L(parent = EnrolmentMgmt.class,
  children = { @Child(cname = Student.class,
  scope = { "id", "name", "helpRequested", "modules", "test1", "test2" }) }) }),
public class ModuleEnrolmentMgmt {
```

4.3. OnUpdateDomainFields

Command

```
java -DrootSrcPath=$MCCL_SRC_EXAMPLES
  -cp lib/domainapp.jar:lib/javaparser-core-3.2.5.jar:lib/mccl.jar domainapp.modules.mccl.test.MCCUpdateTool
  update <domain-class-FQN> <update-spec>
```

Where:

- `domain-class-FQN`: the FQN of domain class
- `update-spec`: semi-colon-separated list of (new-field-name,old-field-name) pairs (e.g. "(testA,test1);(testB,test2)")

Run output

In this example:

- `$MCCL_SRC_EXAMPLES = "/home/dmle/projects/domainapp/modules/mccl/src.examples"`
- `domain-class-FQN = vn.com.courseman.modulesupdate.student.model.Student`
- `update-spec = "(testA,test1);(testB,test2)"`

Preparation

Change the names of the newly added domain fields of class `vn.com.courseman.modulesupdate.student.model.Student` from `test1` and `test2` to `testA` and `testB` (respectively) as shown below. We expect that after executing this program, the two view fields `test1` and `test2` of class `vn.com.courseman.modulesupdate.student.model.ModuleStudent` will have their names and labels updated accordingly. Further, the state scope of `ModuleStudent` and the containment scope of class `Student` in `ModuleEnrolmentMgmt` will also be updated accordingly.

```
////// UPDATE FIELDS TO TEST ///////
@DAttr(name="testA",type=Type.String, serialisable=false)
private String testA;

@DAttr(name="testB",type=Type.String, serialisable=false)
```

```
  private String testB;

  ////// END fields ///////
```

Command output

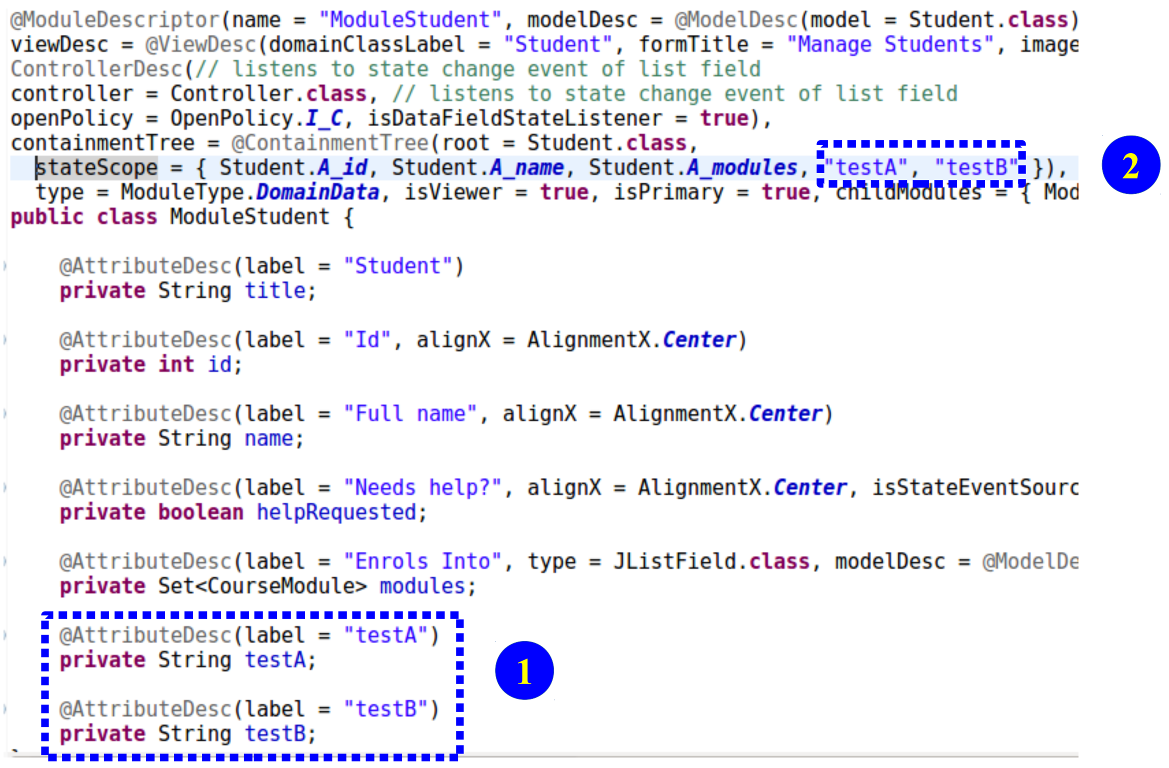
```
Creating KSE-2017 MCCModel...
--- MCCs ---
[vn.com.courseman.modulesupdate.student.model.Student, vn.com.courseman.modulesupdate.student.ModuleStudent]
[vn.com.courseman.modulesupdate.enrolmentmgmt.model.EnrolmentMgmt,
vn.com.courseman.modulesupdate.enrolmentmgmt.ModuleEnrolmentMgmt]

Running OnUpdateDomainFields...
   Domain class: vn.com.courseman.modulesupdate.student.model.Student
   Updated fields (new-name -> old-name): {testB=test2, testA=test1}
...ok
```

Result

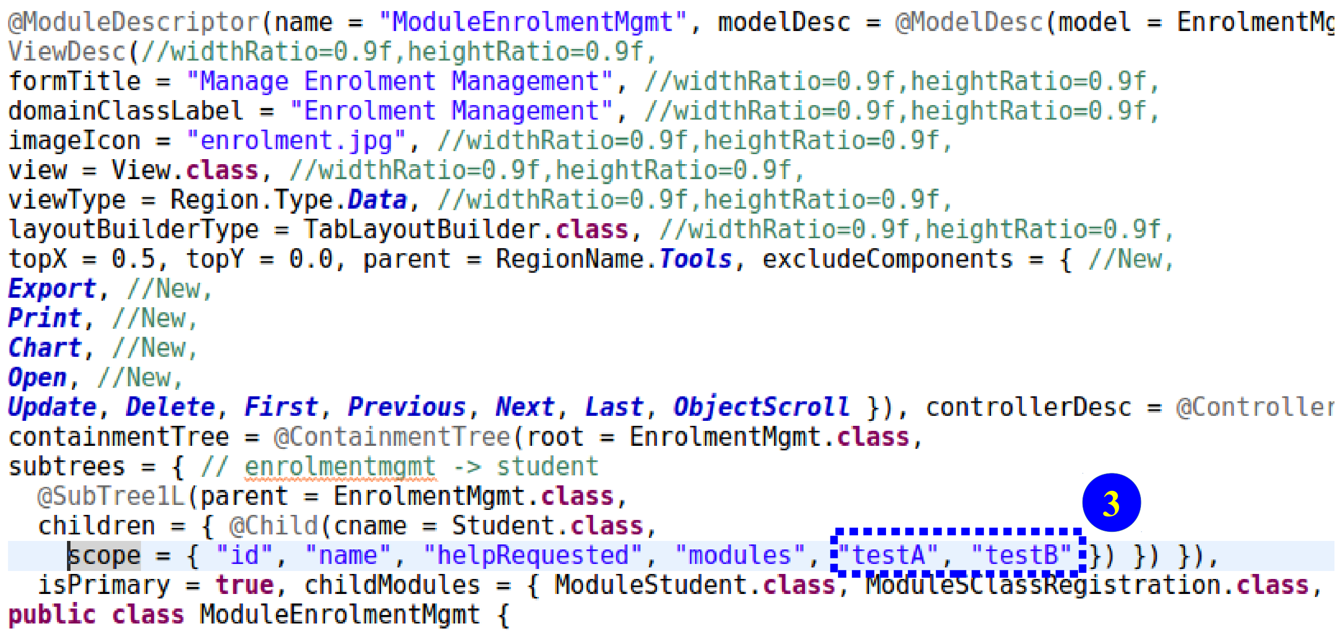
(1) Names and labels of two view fields `test1`, `test2` of `ModuleStudent` are changed to `testA`, `testB` (resp.).

(2) Property `ModuleDescriptor.containmentTree.stateScope` of `ModuleStudent` is also updated with the new names of the two view fields.

(3) Containment scope of `Student` in `ModuleEnrolmentMgmt` is updated with the new names of the two view fields.



```
@ModuleDescriptor(name = "ModuleStudent", modelDesc = @ModelDesc(model = Student.class)
viewDesc = @ViewDesc(domainClassLabel = "Student", formTitle = "Manage Students", image
ControllerDesc(// listens to state change event of list field
controller = Controller.class, // listens to state change event of list field
openPolicy = OpenPolicy.I_C, isDataFieldStateListener = true),
containmentTree = @ContainmentTree(root = Student.class,
  stateScope = { Student.A_id, Student.A_name, Student.A_modules, "testA", "testB" }),
  type = ModuleType.DomainData, isViewer = true, isPrimary = true, childModules = { Mod
public class ModuleStudent {

    @AttributeDesc(label = "Student")
    private String title;

    @AttributeDesc(label = "Id", alignX = AlignmentX.Center)
    private int id;

    @AttributeDesc(label = "Full name", alignX = AlignmentX.Center)
    private String name;

    @AttributeDesc(label = "Needs help?", alignX = AlignmentX.Center, isStateEventSourc
    private boolean helpRequested;

    @AttributeDesc(label = "Enrols Into", type = JListField.class, modelDesc = @ModelDe
    private Set<CourseModule> modules;

    @AttributeDesc(label = "testA")
    private String testA;

    @AttributeDesc(label = "testB")
    private String testB;
```



```
@ModuleDescriptor(name = "ModuleEnrolmentMgmt", modelDesc = @ModelDesc(model = EnrolmentMg
ViewDesc(//widthRatio=0.9f,heightRatio=0.9f,
formTitle = "Manage Enrolment Management", //widthRatio=0.9f,heightRatio=0.9f,
domainClassLabel = "Enrolment Management", //widthRatio=0.9f,heightRatio=0.9f,
imageIcon = "enrolment.jpg", //widthRatio=0.9f,heightRatio=0.9f,
view = View.class, //widthRatio=0.9f,heightRatio=0.9f,
viewType = Region.Type.Data, //widthRatio=0.9f,heightRatio=0.9f,
layoutBuilderType = TabLayoutBuilder.class, //widthRatio=0.9f,heightRatio=0.9f,
topX = 0.5, topY = 0.0, parent = RegionName.Tools, excludeComponents = { //New,
Export, //New,
Print, //New,
Chart, //New,
Open, //New,
Update, Delete, First, Previous, Next, Last, ObjectScroll }), controllerDesc = @Controller
containmentTree = @ContainmentTree(root = EnrolmentMgmt.class,
subtrees = { // enrolmentmgmt -> student
  @SubTree1L(parent = EnrolmentMgmt.class,
  children = { @Child(cname = Student.class,
    scope = { "id", "name", "helpRequested", "modules", "testA", "testB" }) }) }),
  isPrimary = true, childModules = { ModuleStudent.class, ModuleClassRegistration.class,
public class ModuleEnrolmentMgmt {
```

5. References

[1] D. M. Le, "jDomainApp: A Domain-Driven Application Development Framework in Java," Hanoi University, 2016.

[2] Duc Minh Le, *MCCL implementation in jDomainApp*. vnu-dse, 2017. [Online]. Available: https://github.com/vnu-dse/mccl

[3] F. Tomassetti, D. van Bruggen, and N. Smith, "JavaParser," 2016. [Online]. Available: http://javaparser.org/. [Accessed: 31-Oct-2016].