

AGCL Tool Manual

Technical Report

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

1. Introduction

This manual briefly describes the implementation of AGCL in a software tool named jDomainApp [1] and how to set up and run the CourseMan software example using this tool.

2. Installation

Download the AGCL binary from GitHub [2] and place it in a directory on your 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 and the programs contained in these libraries that we will focus on in this guide. We will explain how to run the programs from the command line.

Table 1: AGCL library files

Library file	Description
domainapp.jar	The jDomainApp tool [1].
postgresql-9.2-jdbc4.jar	PostgreSQL JDBC driver (used for PostgreSQL DBMS) <ul style="list-style-type: none">• Used to run the CourseMan software examples
scrollabledesktop.jar	Third-party library for building the main GUI. <ul style="list-style-type: none">• Used to run the CourseMan software examples
agcl.jar	Implementation of the AGCL language.
courseman-act-example.jar	Contains the CourseMan-ACT model (see Section 2.1).
examples-data.zip	Contains the SQL script file for populating some test data for the CourseMan example.

2.1. CourseMan-ACT: CourseMan Example for Behavioural Modelling

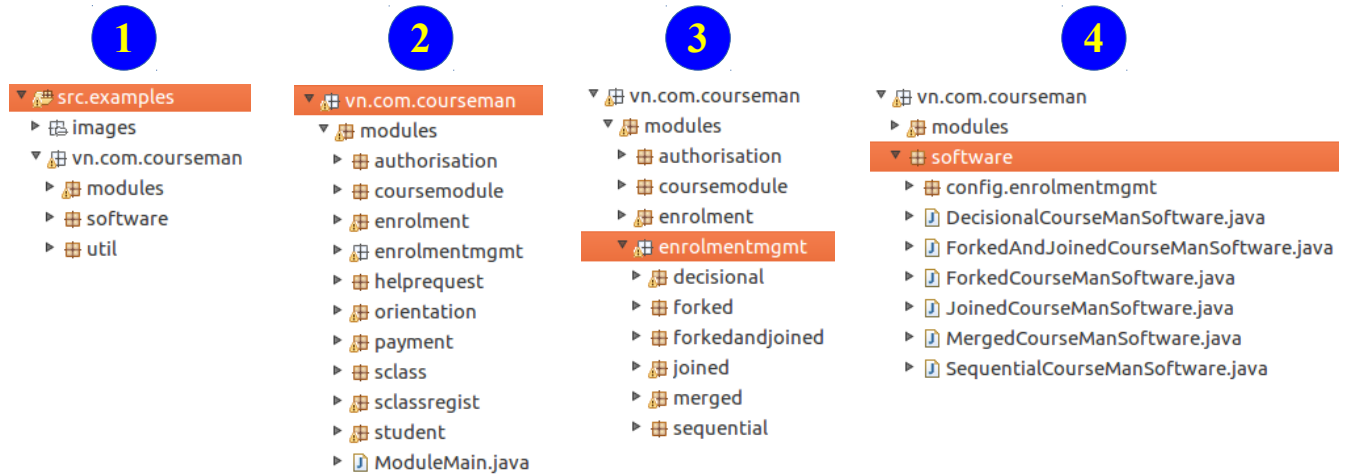


Figure 1: CourseMan-ACT model example (file: `courseman-act-example.jar`).

Figure 1 shows the package structure of the CourseMan-ACT model example. This example implements the software model that is referenced in paper [3]. It consists of the domain classes that make up the domain model, the activity classes of the behavioural modelling patterns discussed in the paper, the software modules and software that are created from these classes.

The entire binary of CourseMan-ACT is provided in the file `courseman-act-example.jar`.

Snapshot 1 in Figure 1 shows the top-level package structure. Snapshot 2 lists the content of the sub-package `vn.com.courseman.modules`. This package contains the software modules for the domain classes mentioned in the paper. For example, the module `vn.com.courseman.modules.authorisation` (first module in the list) contains the domain class `Authorisation` and the software module created from this class. The five versions of the enrolment management activity that are used in the paper are implemented as modules in the package `vn.com.courseman.modules.enrolmentmgmt`. This package is shown in snapshot 3. Note that package `forkedandjoined` is an extra package that implements another version of the activity not mentioned in the paper.

Finally, snapshot 4 shows the software generators for all the CourseMan software prototypes of the enrolment activity mentioned in paper [3]. These software are created from combinations of the software modules mentioned above. We briefly describe below the five CourseMan software that are of relevance to the paper:

- Sequential Software: CourseMan software that demonstrates the sequential behavioural pattern.
- Decisional Software: CourseMan software that demonstrates the decisional behavioural pattern.
- Forked Software: CourseMan software that demonstrates the forked behavioural pattern.
- Joined Software: CourseMan software that demonstrates the joined behavioural pattern.
- Merged Software: CourseMan software that demonstrates the merged behavioural pattern.

2.2. Installing PostgreSQL DBMS

The CourseMan software examples in this guide use the PostgreSQL DBMS to store objects. Please download and install this DBMS from the PostgreSQL web site (<http://www.postgresql.org>). The

PostgreSQL version that is used for the software examples is version 9.5.6. Later versions may be used but have not been tested.

Please ensure that PostgreSQL is configured to listen on the default port, which is 5432.

2.3. Creating a Database for CourseMan Software Examples

After installing PostgreSQL, use its database manager (pgAdmin) to:

1. create a user account: user = "admin", password = "password"
2. create a database named `domains`, which is owned by the user admin created in step 1 (see Figure 2). This database is used by the CourseMan software examples.

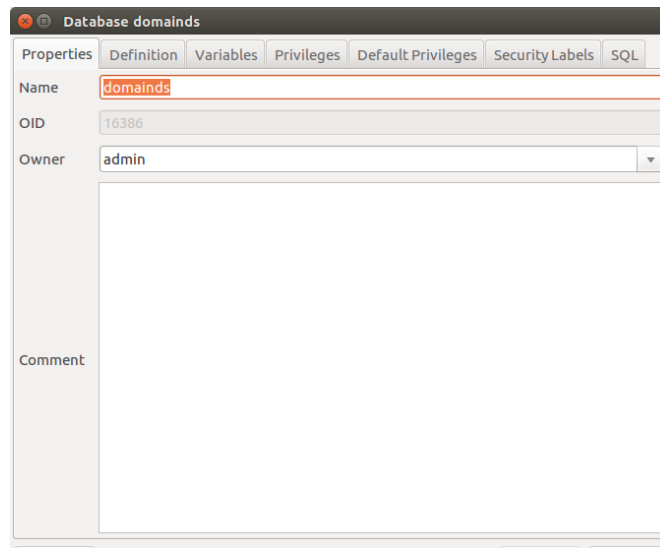


Figure 2: The properties dialog of the database `domains`.

2.4. Creating the CourseMan Test Data

Follow the steps below to import pre-defined test data for the three domain classes: `CourseModule`, `SClass` and `Orientation`.

1. Open the PostgreSQL database manager (pgAdmin)
2. Connect to the `domains` database
3. Open the SQL Editor and open the file `courseman-data.sql` (from the `examples/data` directory in the `examples-data.zip` file)
4. Execute the file to populate the tables of the three aforementioned domain classes with data

Note: this script also create three tables. Thus, if we run the script after the CourseMan software has been run then we need to comment out the CREATE statements from the file before running the script. This is because CourseMan software automatically create the underlying tables of the domain classes.

3. Running the CourseMan Example

Sections 3.1-3.5 the explain how to run five CourseMan software using the CourseMan-ACT model. As discussed earlier, these software demonstrate the five behavioural modelling patterns of paper [3].

3.1. Sequential Software

Command

```
java -Dlogging=true -cp lib/domainapp.jar:lib/postgresql-9.2-jdbc4.jar:lib/scrollabledesktop.jar:
lib/agcl.jar:lib/courseman-act-example.jar
vn.com.courseman.software.SequentialCourseManSoftware
```

Result

- (1) Figure 3 displays the main GUI and GUI action that starts off the activity
- (2) Figure 4 displays the subsequent actions that need to be taken on each GUI till finished

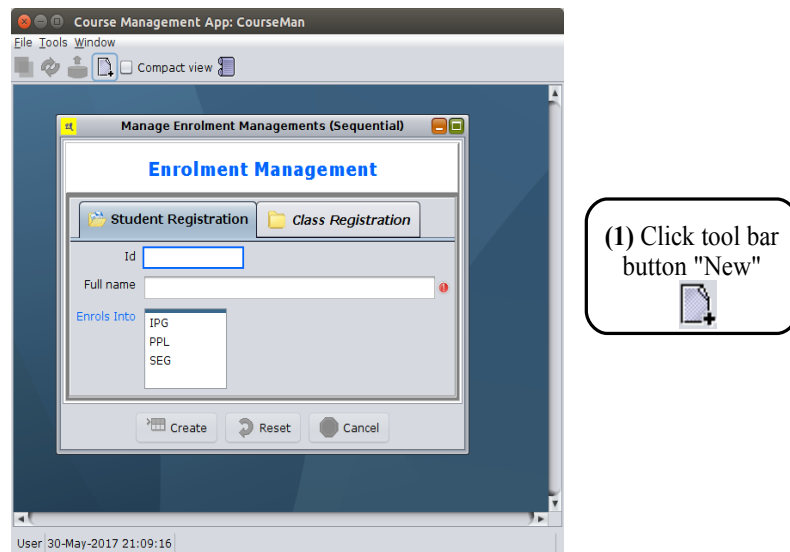


Figure 3: The main GUI and first action to be taken.

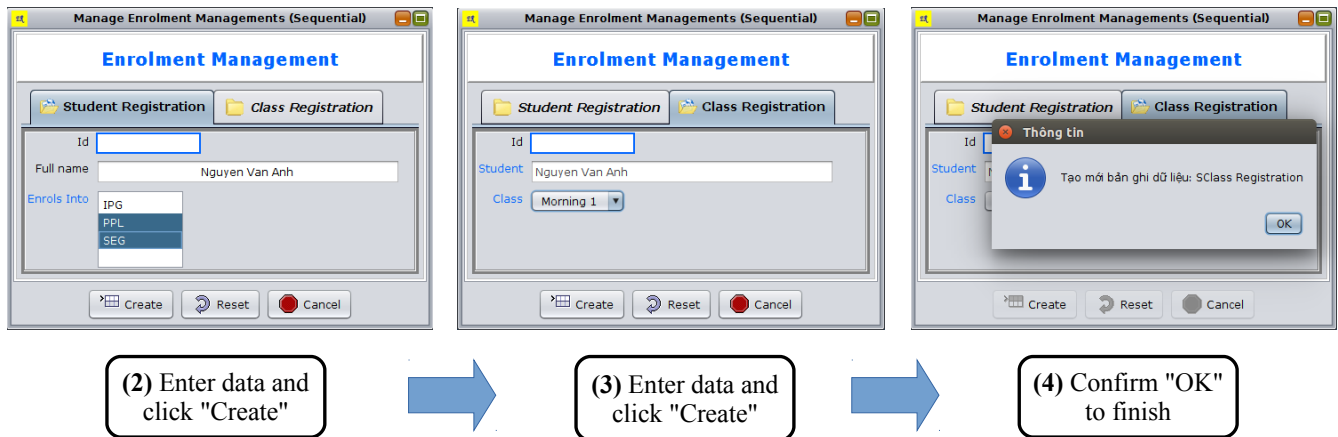


Figure 4: The subsequent actions to be taken on each GUI.

3.2. Decisional Software


Command

```
java -Dlogging=true -cp lib/domainapp.jar:lib/postgresql-9.2-jdbc4.jar:lib/scrollabledesktop.jar:
lib/agcl.jar:lib/courseman-act-example.jar
vn.com.courseman.software.DecisionalCourseManSoftware
```

Result

- (1) Figure 5 displays the actions taken for the first case of the activity (help is requested).
- (2) Figure 6 displays the actions taken for the second case of the activity (help is not requested).


The left screenshot shows the 'Enrolment Management' form with 'Student Registration' selected. The form contains fields for 'Id', 'Full name' (Nguyen Van Anh), 'Needs help?' (checked), and 'Enrols Into' (IPG, PPL, SEG). The right screenshot shows the 'Help Request' form with 'Class Registration' selected. The form contains fields for 'Id' (1), 'Student' (Nguyen Van Anh), and 'Content' (How many modules do I need to pass?).

(1) With Enrolment Management form selected, click the tool bar button , enter data on Student Registration s.t **Needs help?** = **true**, then click "Create"

(2) Enter data for Help Request and click "Create"

Figure 5: Decisional pattern example (2): help IS requested.

The left screenshot shows the 'Enrolment Management' form with 'Student Registration' selected. The form contains fields for 'Id', 'Full name' (Nguyen Van Anh), 'Needs help?' (unchecked), and 'Enrols Into' (IPG, PPL, SEG). The right screenshot shows the 'Class Registration' form with 'Student Registration' selected. The form contains fields for 'Id' (17), 'Student' (Nguyen Van Anh), and 'Class' (Morning 1).

(1) With Enrolment Management form selected, click the tool bar button , enter data on Student Registration s.t **Needs help?** = **false**, then click "Create"

(2) Enter data for Class Registration and click "Create"

Figure 6: Decisional pattern example: help is NOT requested.

3.3. Forked Software

Command

```
java -Dlogging=true -cp lib/domainapp.jar:lib/postgresql-9.2-jdbc4.jar:lib/scrollabledesktop.jar:lib/agcl.jar:lib/courseman-act-example.jar vn.com.courseman.software.ForkedCourseManSoftware
```

Result

- (1) Figure 7 displays the actions taken for the activity.
- (2) Figure 8 displays the Payment and Authorisation objects that are created by enrolment processing.

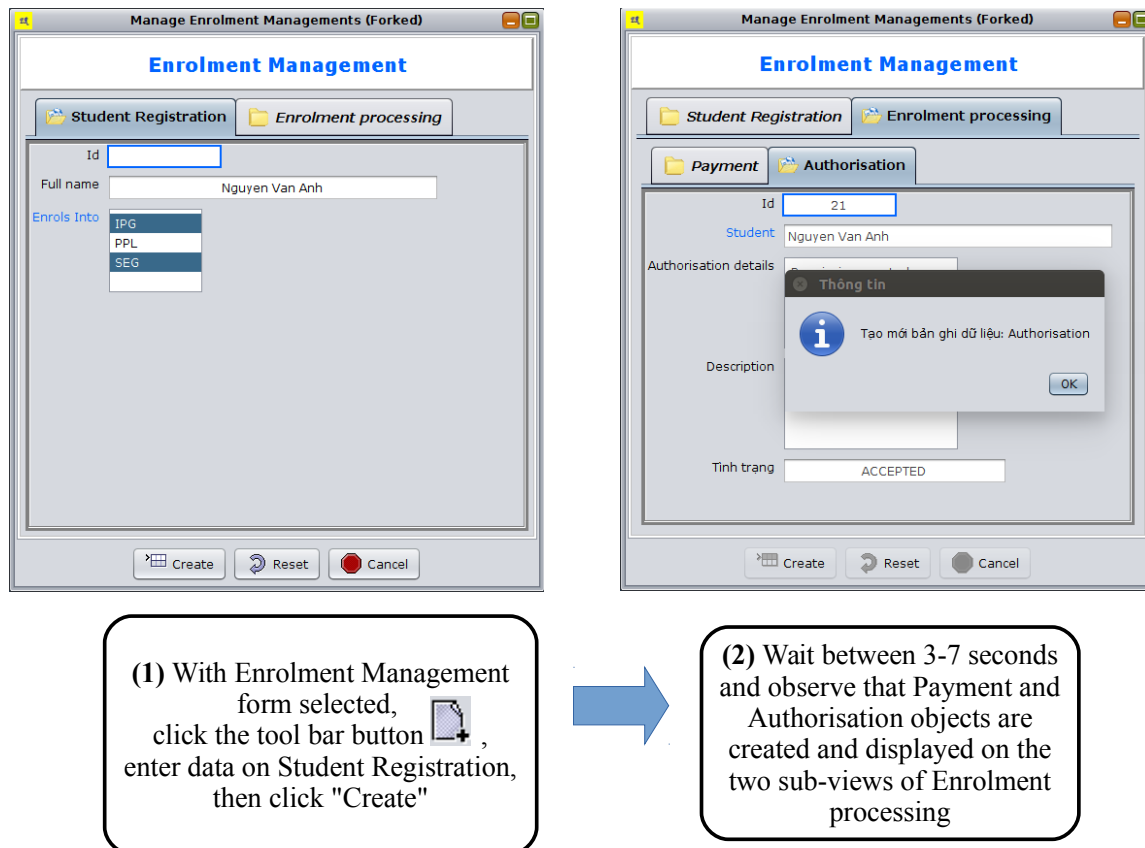


Figure 7: Forked pattern example: actions to be performed on the GUI.

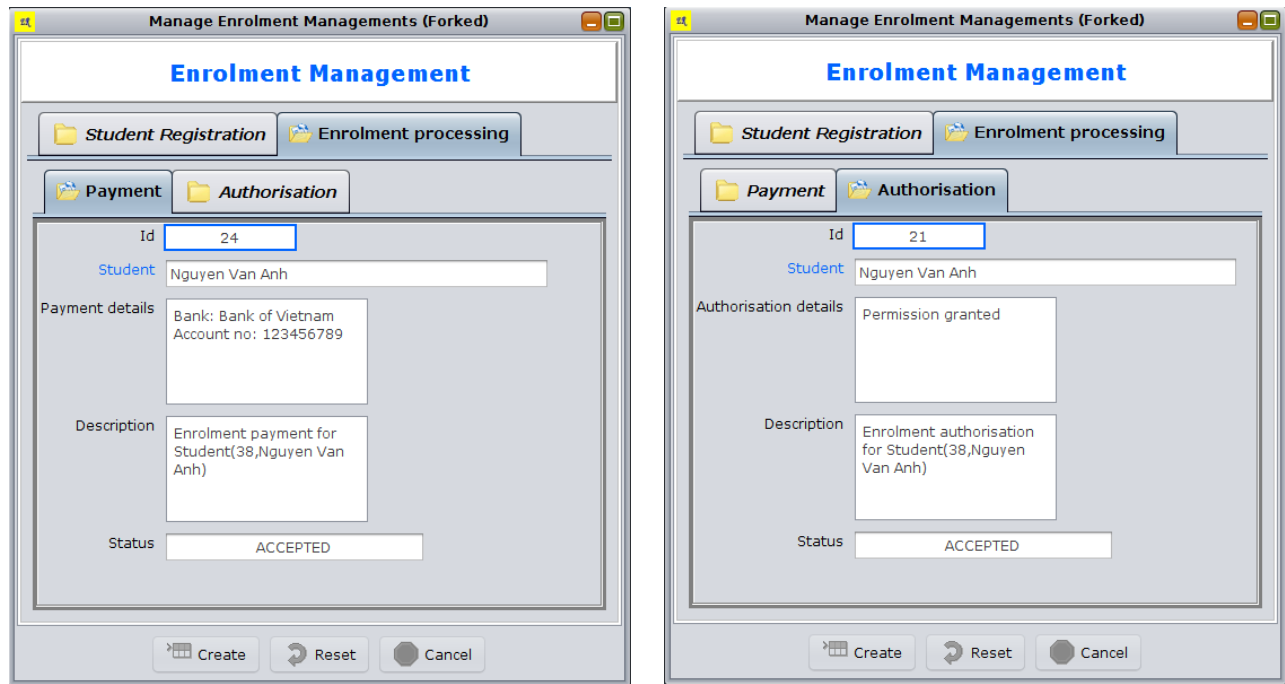


Figure 8: The two result objects of Enrolment processing: Payment and Authorisation.

3.4. Joined Software

Command

```
java -Dlogging=true -cp lib/domainapp.jar:lib/postgresql-9.2-jdbc4.jar:lib/scrollabledesktop.jar:
lib/agcl.jar:lib/courseman-act-example.jar
vn.com.courseman.software.JoinedCourseManSoftware
```

Result

- (1) Figure 9 displays the actions performed on the GUI for the activity.
- (2) Figure 10 displays the three objects created in the activity.

Enrolment Management

Student: 37| Nguyen Van Anh

Payment | Authorisation | Enrolment Approval

Id:

Student:

Payment details:

Description:

Tinh trang:

Create Reset Cancel

Enrolment Management

Student: 37| Nguyen Van Anh

Payment | Authorisation | Enrolment Approval

Id:

Payment status: ACCEPTED


Authorisation status: ACCEPTED

Student: Nguyen Van Anh

Approved? ☒

Note:

Create Reset Cancel

(1) With Enrolment Management form selected, click the tool bar button , select a Student from the drop down list, then click "Create"



(2) Wait between 3-7 seconds and observe that Enrolment Approval is activated with data automatically filled in. Optionally enter data for Note and click "Create"

Figure 9: Joined pattern example: actions to be performed on the GUI.

Enrolment Management

Student: 37| Nguyen Van Anh

Payment | Authorisation | Enrolment Approval

Id: 25

Student: Nguyen Van Anh

Payment details: Bank: Bank of Vietnam
Account no: 123456789

Description: Enrolment payment for Student(37|Nguyen Van Anh)

Status: ACCEPTED

Create Reset Cancel

Enrolment Management

Student: 37| Nguyen Van Anh

Payment | Authorisation | Enrolment Approval

Id: 22

Student: Nguyen Van Anh

Authorisation details: Permission granted

Description: Enrolment authorisation for Student(37|Nguyen Van Anh)

Status: ACCEPTED

Create Reset Cancel

Enrolment Management

Student: 37| Nguyen Van Anh

Payment | Authorisation | Enrolment Approval

Id: 1

Payment status: ACCEPTED

Authorisation status: ACCEPTED

Student: Nguyen Van Anh

Approved? ☒

Note:

Create Reset Cancel

Figure 10: The three result objects: Payment, Authorisation, Enrolment Approval.

3.5. Merged Software

Command

```
java -Dlogging=true -cp lib/domainapp.jar:lib/postgresql-9.2-jdbc4.jar:lib/scrollabledesktop.jar:
lib/agcl.jar:lib/courseman-act-example.jar
vn.com.courseman.software.MergedCourseManSoftware
```

Result

(1) Figure 11 displays the actions performed on the activity's GUI and the objects that are created.

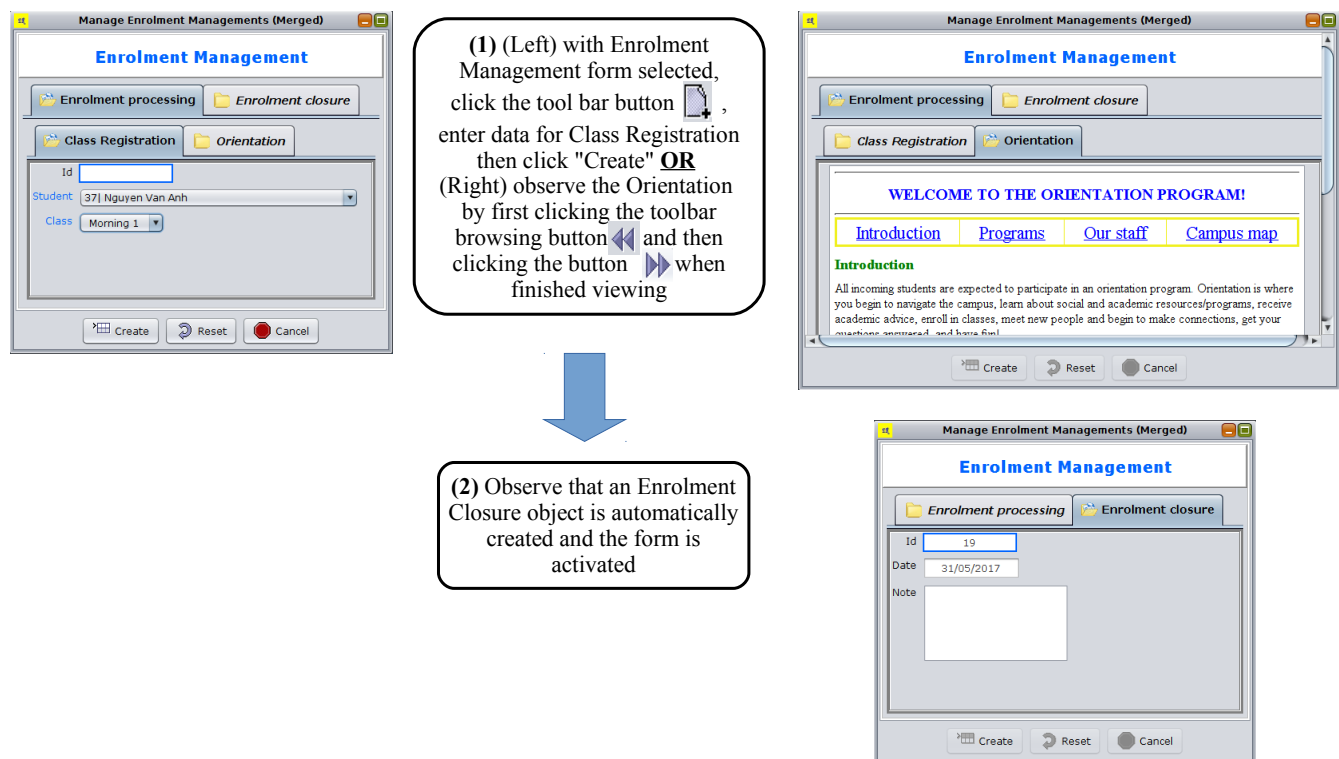


Figure 11: Merged pattern example: actions performed on the GUI.

References

- [1] D. M. Le, “jDomainApp: A Domain-Driven Application Development Framework in Java version 3.0,” Hanoi University, 2016.
- [2] D. M. Le, *AGCL implementation in jDomainApp*. vnu-dse, 2018.
- [3] D. M. Le, D.-H. Dang, and V.-H. Nguyen, “Activity-Based Modelling of Domain-Driven Software with Two Annotation-Based Domain Specific Languages [Extended],” VNU University of Engineering and Technology, Apr. 2018.