DCSL Tool Manual

Technical Report

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

1. Introduction

This manual brielfy describes the implementation of DCSL in a software tool named jDomainApp [1] and how to set up and run the CourseMan software example using this tool. Technical details about DCSL and the CourseMan example are described in paper [2], which we recently submitted to the Journal of Systems and Software.

2. Installation

Download the DCSL binary from GitHub [3] 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: DCSL library files

Library file	Description
`domainapp.jar`	The `jDomainApp` tool [1]. • the base library which contains DCSL and is used by both BSpaceGen and the CourseMan software examples
`postgresql-9.2-jdbc4.jar`	PostgreSQL JDBC driver (used for PostgreSQL DBMS) • Used to run the CourseMan software examples
`scrollabledesktop.jar`	Third-party library for building the main GUI. • Used to run the CourseMan software examples
`javaparser-core-3.0.1-alpha.6.jar`	Third-party library named JavaParser version 3.0.1 [4]. • Used by BSpaceGen to manipulate the Java source code tree
`bspacegen.jar`	Implementation of the BSpaceGen algorithm: • `BSpaceGenTool`: the tool used to execute the algorithm.
`activitymodel.jar`	Implementation of the UML activity model. • Used to run the CourseMan software examples

Library file	Description
`courseman-bsg-base-example.jar`	Contains the base classes of the CourseMan-BSG model example (see Section 2.1). These base classes contain only the state space specification (no behaviour space specification).
`courseman-act-example.jar`	Contains the CourseMan-ACT model (see Section 2.2).

2.1. CourseMan-BSG: CourseMan Example for BSpaceGen

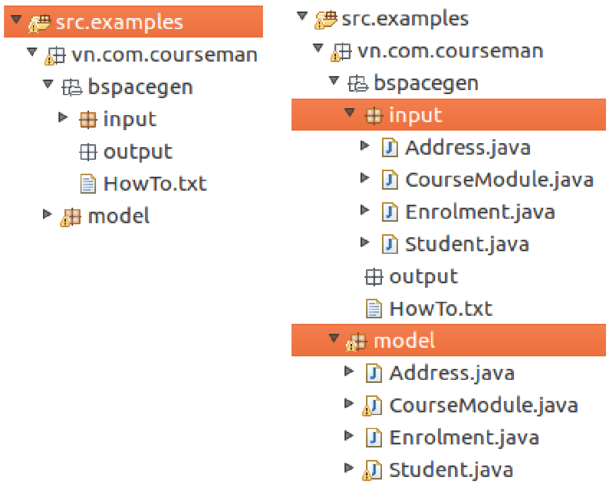


Figure 1: CourseMan-BSG model example (file: `courseman-bsg-example.zip`).

Figure 1 shows the the package structure of the CourseMan-BSG model example. This model implements a small part of the CourseMan domain model that is referenced in paper [2]. It includes four domain classes that are used as test subjects for algorithm BSpaceGen. Package `vn.com.courseman.bspacegen.input` contains the test classes with just the base state space specification (no behavioural specification). Before running this algorithm (explained in Section 5.1), we need to copy these classes to the package `vn.com.courseman.bspacegen.output`. This package is then used as input for the algorithm. Package `vn.com.courseman.model`, on the other hand, contain the same four classes but with the complete specification. They were written by hand and checked by the developer and thus can be used as the expected output to validate the output generated by the algorithm.

The entire source code of CourseMan-BSG is provided in the file `examples/courseman-bsg-example.zip`. In this guide, we will denote `$BSG_SRC_EXAMPLE` by the directory containing the source code of CourseMan-BSG. The binaries of the base classes in the package `vn.com.courseman.bspacegen.output` (before running BSpaceGen) are stored in the file `courseman-bsg-base-example.jar`. This file is used in the class path of BSpaceGen to present the classes as data types.

2.2. CourseMan-ACT: CourseMan **Example for Behavioural Modelling**

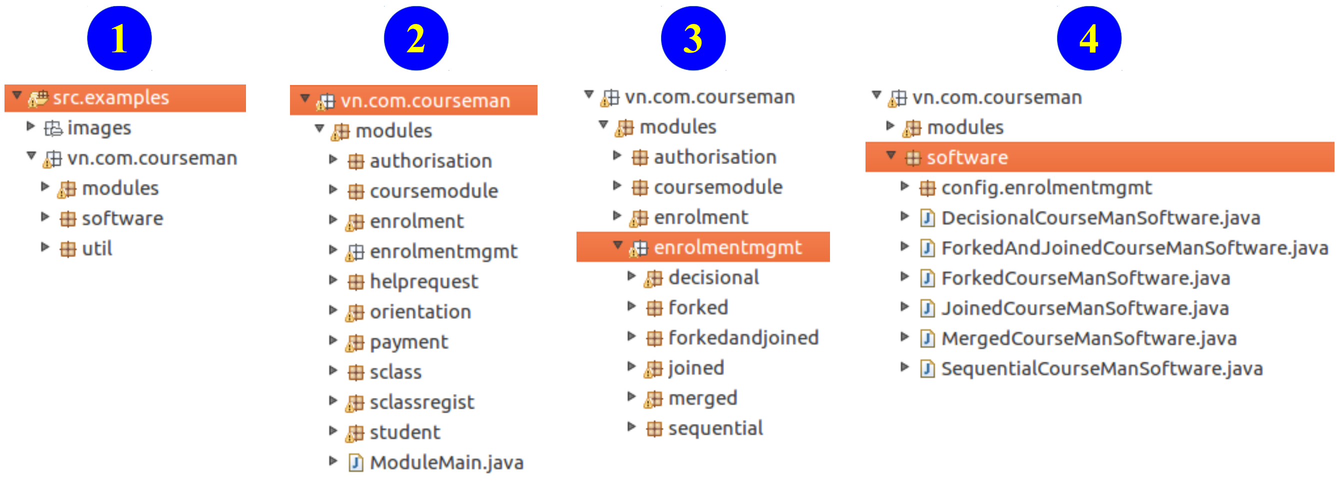


Figure 2: CourseMan-ACT model example (file: `courseman-act-example.jar`).

Figure 2 shows the package structure of the CourseMan-ACT model example. This example implements the domain model that is referenced in paper [2]. It is basically a Java model that implements the domain classes, 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 2 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 show 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 all versions of the CourseMan software that demonstrate the enrolment enrolment activity mentioned in paper [2]. 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.

2.3. Installing PostgreSQL DBMS

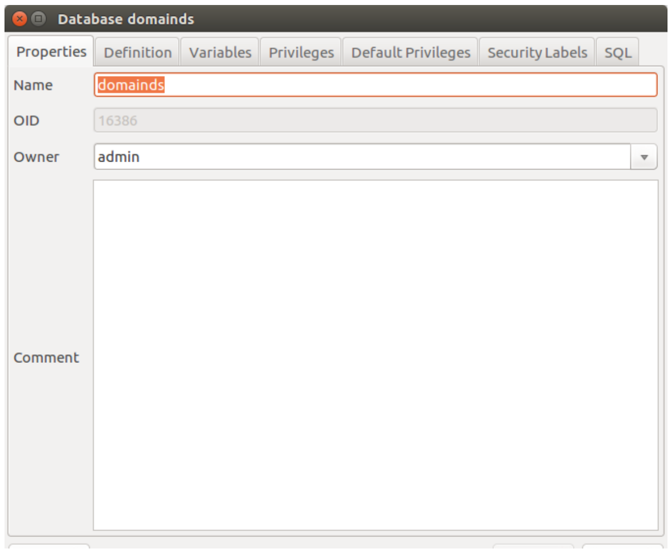
The CourseMan software examples in this guide use the PostgreSQL DBMS to store objects. Please download and install this DBMS from the PostgreSQL web site (http://www.postgresql.org). The PostgreSQL version that is used for the software examples is version 9.5.6. Later versions may be used but have not been tested.

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

2.4. 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 `domainds`, which is owned by the user admin created in step 1 (see Figure 3). This database is used by the CourseMan software examples.



Name: domainds
OID: 16386
Owner: admin

Figure 3: The properties dialog of the database `domainds`.

2.5. 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 `domainds` database
3. Open the SQL Editor and open the file `courseman-data.sql` (from the `examples/data` directory)
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. DCSL

DCSL is developed from a set of Java annotations that were historically written as part of jDomainApp. Thus, unlike BSpaceGen and activity modelling DCSL is not defined as a separate module. Figure 4 shows a partial package structure of jDomainApp, which highlights the package where DCSL is located. This package is named `domainapp.basics.model.meta`.

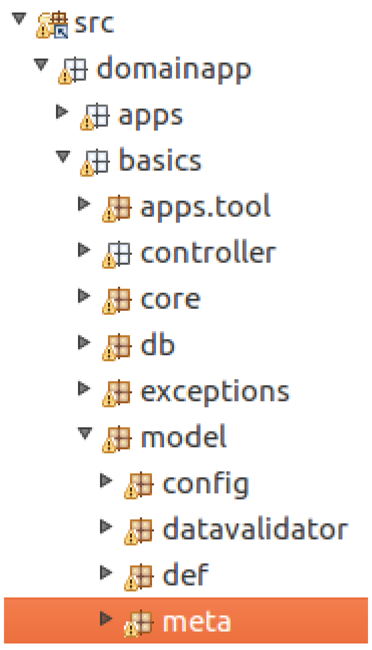


Figure 4: DCSL as a package in jDomainApp.

4. BSpaceGen Tool

This tool provides a developer-friendly API for executing the BSpaceGen algorithm of the paper [2].

5. Running the CourseMan Example

Section 5.1 explains how to run BSpaceGenTool using the CourseMan-BSG model. Sections 5.2-5.6 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 [2].

5.1. BSpaceGenTool

Command

```
java -Dlogging=true -DrootSrcPath=$BSG_SRC_EXAMPLE
   -cp lib/domainapp.jar:lib/javaparser-core-3.0.1-alpha.6.jar:lib/bspacegen.jar:lib/courseman-bsg-base-example.jar
   domainapp.modules.bspacegen.BSpaceGenTool
   <package-FQN> <domain-class-simple-names>
```

Where:

- `$BSG_SRC_EXAMPLE` is the directory where the source code of the CourseMan-BSG model is located (see Section2.1)

- `package-FQN`: the FQN of the package containing the base domain classes
- `domain-class-simple-names`: a comma-separated list of the simple names of the base domain classes that are contained in the specified package (e.g. "Student, PostGrad, UnderGrad")

Run output

In this example:

- `$BSG_SRC_EXAMPLE = /home/dmle/projects/domainapp/modules/bspacegen/src.examples`
- `package-FQN = vn.com.courseman.bspacegen.output`
- `domain-class-simple-names = "Student,CourseModule,Enrolment,Address"`

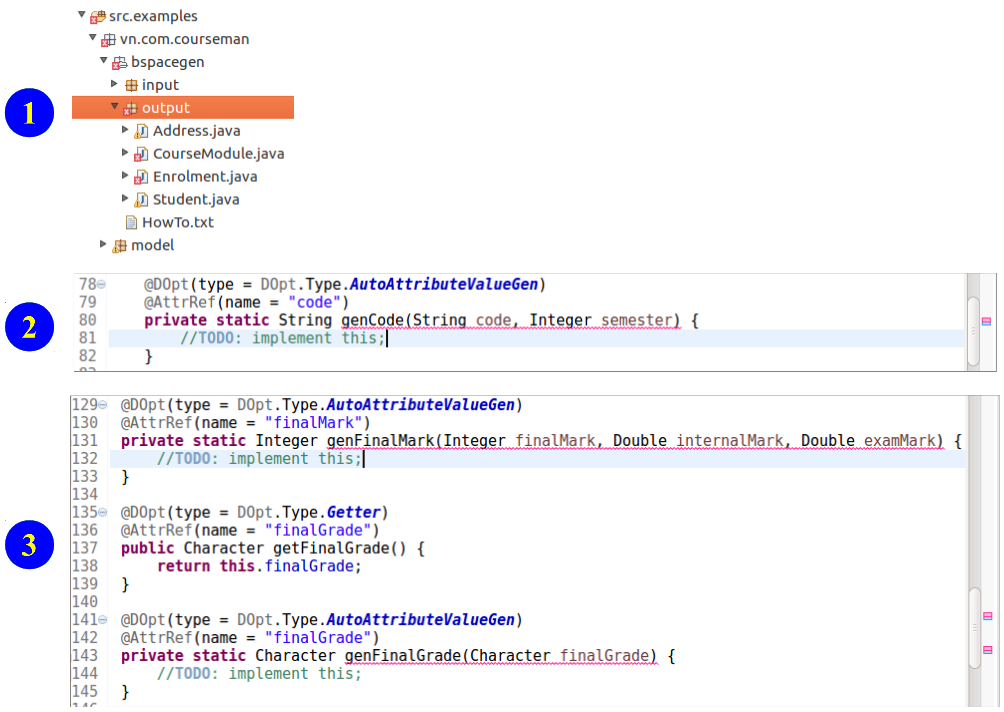
```
Running BSpaceGen...
   Root source path: /home/dmle/projects/domainapp/modules/bspacegen/src.examples
   Package name: vn.com.courseman.bspacegen.output (path:
/home/dmle/projects/domainapp/modules/bspacegen/src.examples/vn/com/courseman/bspacegen/output)
   Domain classes: [Student, CourseModule, Enrolment, Address]

   Domain class: vn.com.courseman.bspacegen.output.Student
   ...to be overwritten
...ok
   Domain class: vn.com.courseman.bspacegen.output.CourseModule
   ...to be overwritten
...ok
   Domain class: vn.com.courseman.bspacegen.output.Enrolment
   ...to be overwritten
...ok
   Domain class: vn.com.courseman.bspacegen.output.Address
   ...to be overwritten
...ok
```

Result

(1) Four base domain classes are filled with the behavioural specification. Two classes, CourseModule and Enrolment, are reported to have some compilation errors. These errors are to do with certain code segments (explained next) that need to filled in by the developer.

(2) CourseModule: one method named `genCode` whose body needs to be filled in by the developer.

(3) Enrolment: two methods (`genFinalMark` and `genFinalGrade`) whose bodies need to be filled in by the developer.



```
src.examples
  vn.com.courseman
    bspacegen
      input
      output
        Address.java
        CourseModule.java
        Enrolment.java
        Student.java
        HowTo.txt
      model
```

```
    @DOpt(type = DOpt.Type.AutoAttributeValueGen)
    @AttrRef(name = "code")
    private static String genCode(String code, Integer semester) {
        //TODO: implement this;
    }
```

```
@DOpt(type = DOpt.Type.AutoAttributeValueGen)
@AttrRef(name = "finalMark")
private static Integer genFinalMark(Integer finalMark, Double internalMark, Double examMark) {
    //TODO: implement this;
}

@DOpt(type = DOpt.Type.Getter)
@AttrRef(name = "finalGrade")
public Character getFinalGrade() {
    return this.finalGrade;
}

@DOpt(type = DOpt.Type.AutoAttributeValueGen)
@AttrRef(name = "finalGrade")
private static Character genFinalGrade(Character finalGrade) {
    //TODO: implement this;
}
```

Figure 5: Result of running BSpaceGen on the four base domain classes.

5.2. Sequential Software

Command

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

Result

(1) Figure 6 displays the main GUI and GUI action that starts off the activity

(2) Figure 7 displays the subsequent actions that need to be taken on each GUI till finished

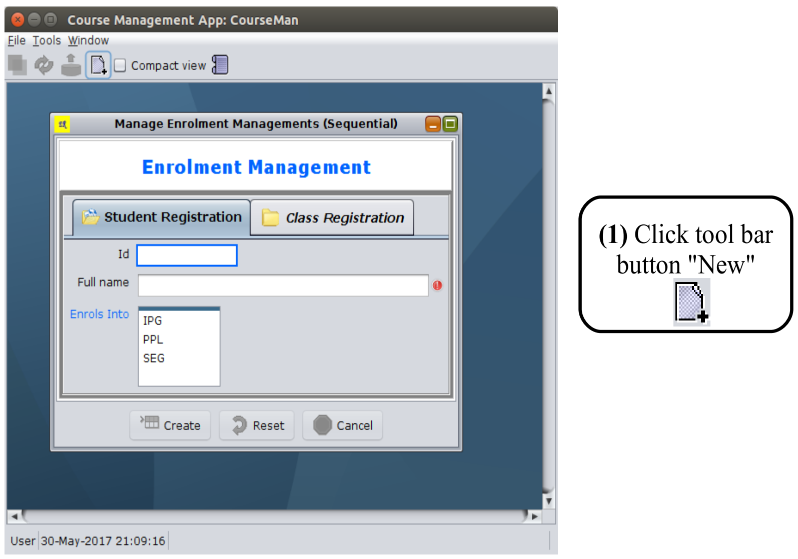
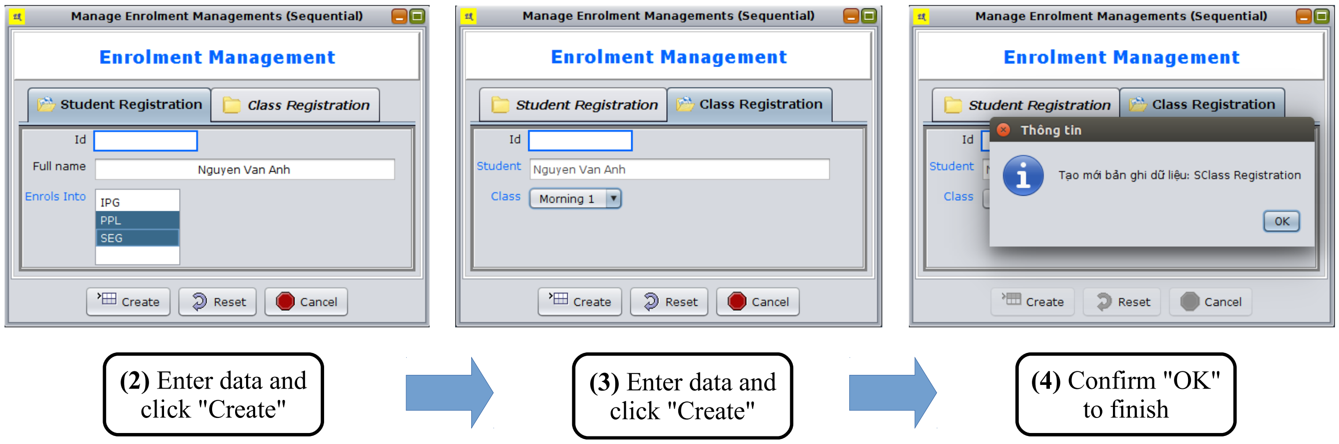


Figure 6: The main GUI and first action to be taken.


Manage Enrolment Managements (Sequential)
Enrolment Management
Student Registration
Class Registration
Id
Full name
Enrols Into
IPG
PPL
SEG
Create
Reset
Cancel
Student
Class
Thông tin
OK
(2) Enter data and click "Create"
(3) Enter data and click "Create"
(4) Confirm "OK" to finish

Full name: Nguyen Van Anh

Student: Nguyen Van Anh

Class: Morning 1

Tạo mới bản ghi dữ liệu: SClass Registration

Figure 7: The subsequent actions to be taken on each GUI.

5.3. Decisional Software

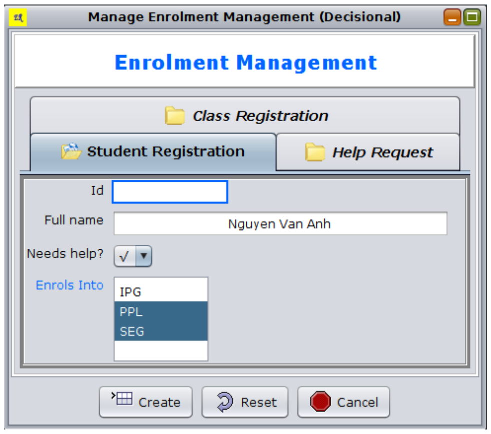
Command

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

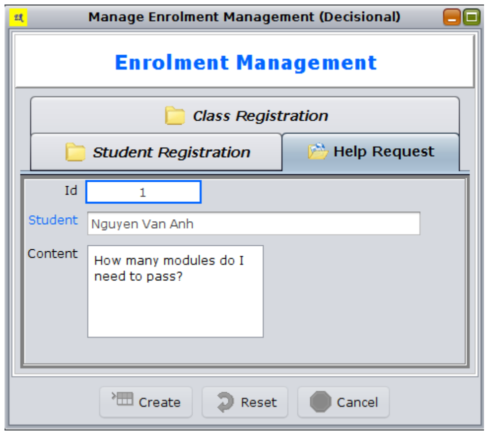
Result

(1) Figure 8 displays the actions taken for the first case of the activity (help is requested).

(2) Figure 9 displays the actions taken for the second case of the activity (help is not requested).



Id	
Full name	Nguyen Van Anh
Needs help?	√
Enrols Into	IPG, PPL, SEG

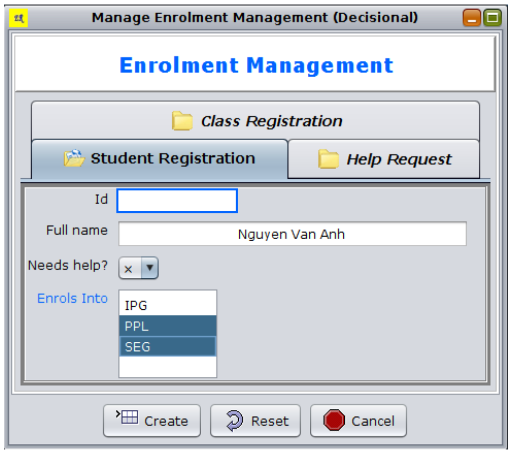


Id	1
Student	Nguyen Van Anh
Content	How many modules do I need to pass?

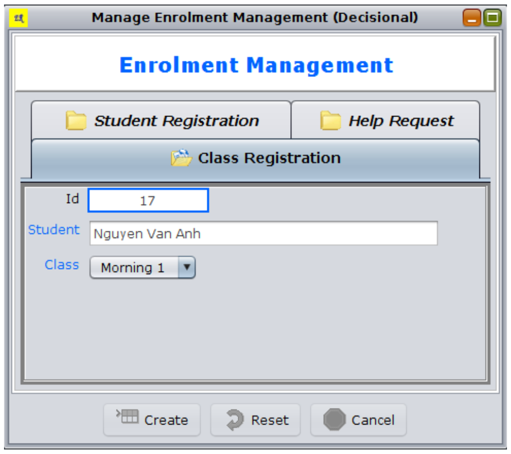
(1) With Enrolment Management form selected, click the tool bar button , enter data on Student Registration s.t **Needs help? = true**, then click "Create"

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

Figure 8: Decisional pattern example (2): help IS requested.



Id	
Full name	Nguyen Van Anh
Needs help?	×
Enrols Into	IPG, PPL, SEG



Id	17
Student	Nguyen Van Anh
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 9: Decisional pattern example: help is NOT requested.

5.4. Forked Software

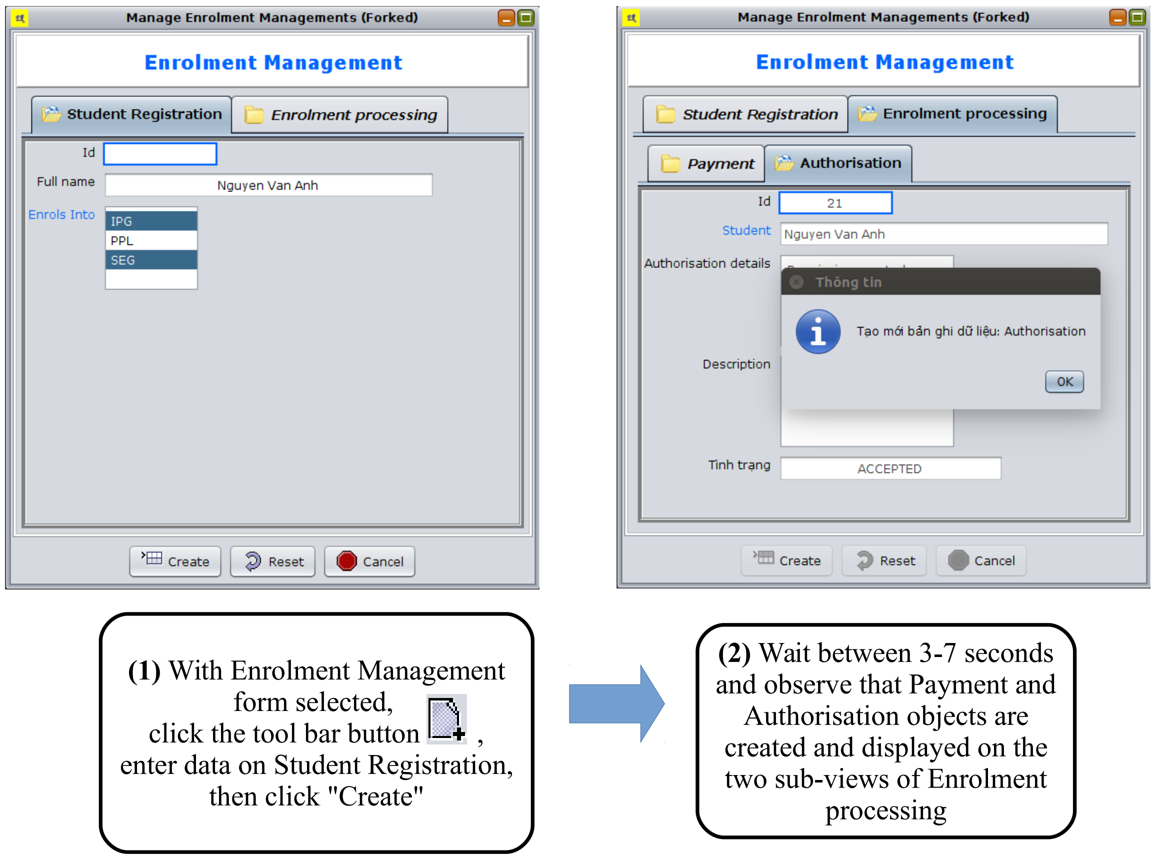
Command

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

Result

(1) Figure 10 displays the actions taken for the activity.

(2) Figure 11 displays the Payment and Authorisation objects that are created by enrolment processing.



Id

Full name: Nguyen Van Anh

Enrols Into: IPG, PPL, SEG

Id: 21

Student: Nguyen Van Anh

Authorisation details

Description

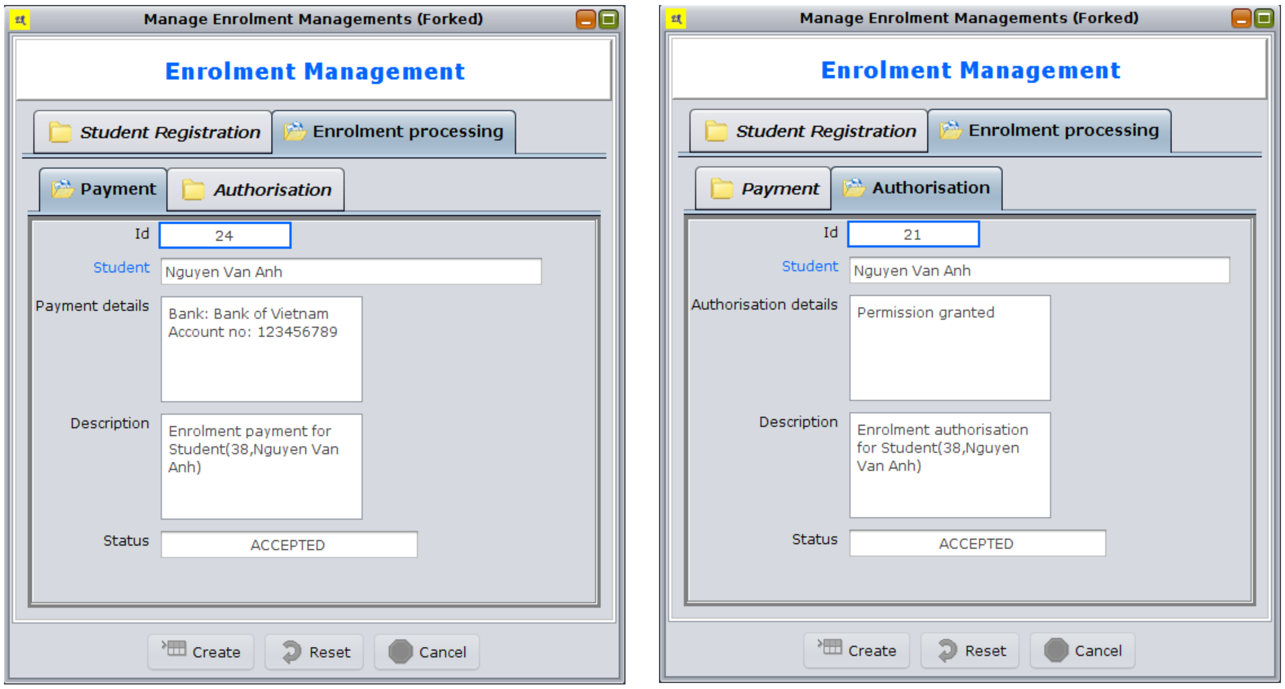
Tinh trang: ACCEPTED

Tạo mới bản ghi dữ liệu: Authorisation

(1) With Enrolment Management form selected, click the tool bar button , enter data on Student Registration, then click "Create"

(2) Wait between 3-7 seconds and observe that Payment and Authorisation objects are created and displayed on the two sub-views of Enrolment processing

Figure 10: Forked pattern example: actions to be performed on the GUI.



Id	24
Student	Nguyen Van Anh
Payment details	Bank: Bank of Vietnam Account no: 123456789
Description	Enrolment payment for Student(38,Nguyen Van Anh)
Status	ACCEPTED

Id	21
Student	Nguyen Van Anh
Authorisation details	Permission granted
Description	Enrolment authorisation for Student(38,Nguyen Van Anh)
Status	ACCEPTED

Figure 11: The two result objects of Enrolment processing: Payment and Authorisation.

5.5. Joined Software

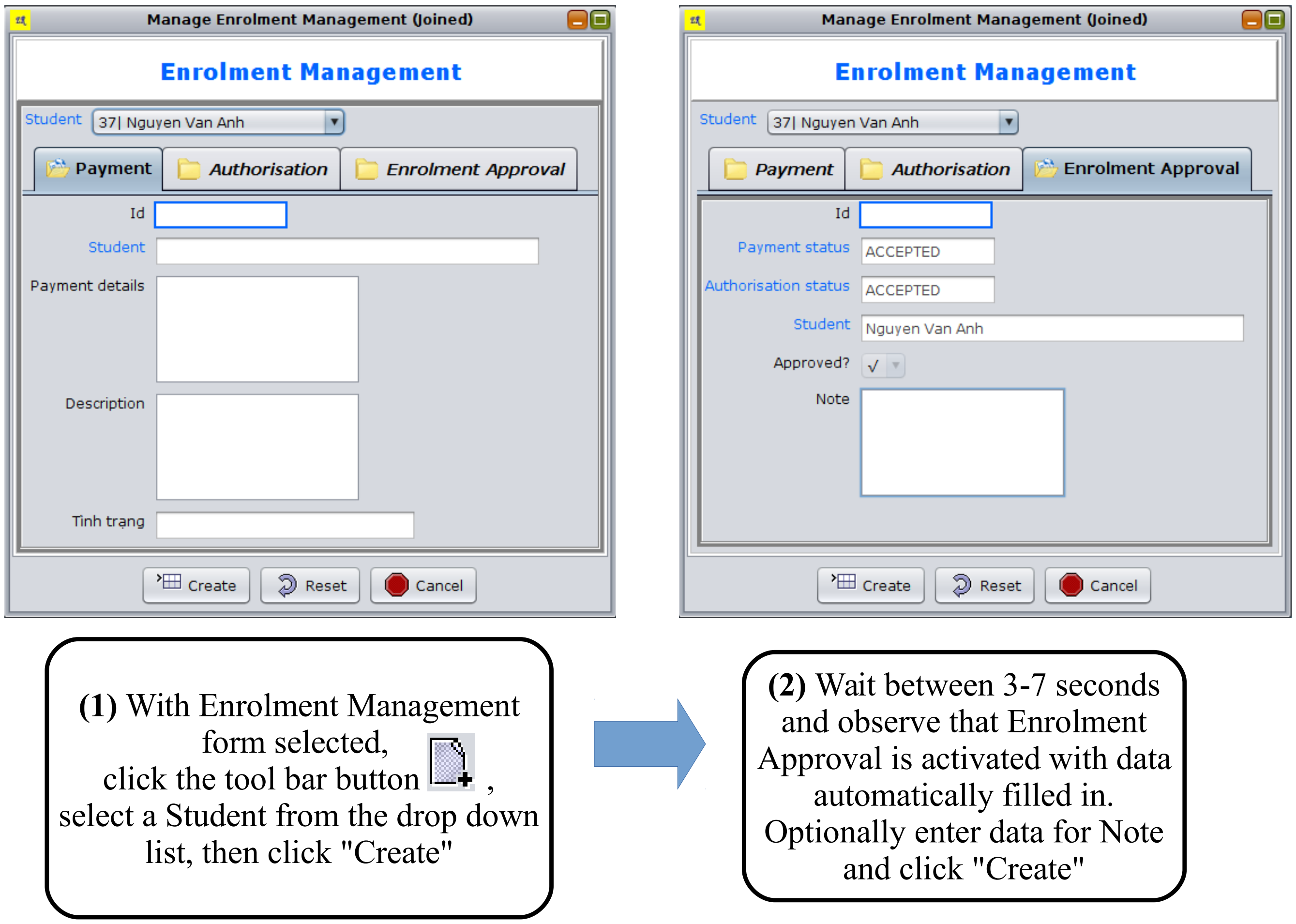
Command

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

Result

(1) Figure 12 displays the actions performed on the GUI for the activity.

(2) Figure 13 displays the three objects created in the activity.



Student: 37| Nguyen Van Anh

Id:
Student:
Payment details:
Description:
Tinh trạng:

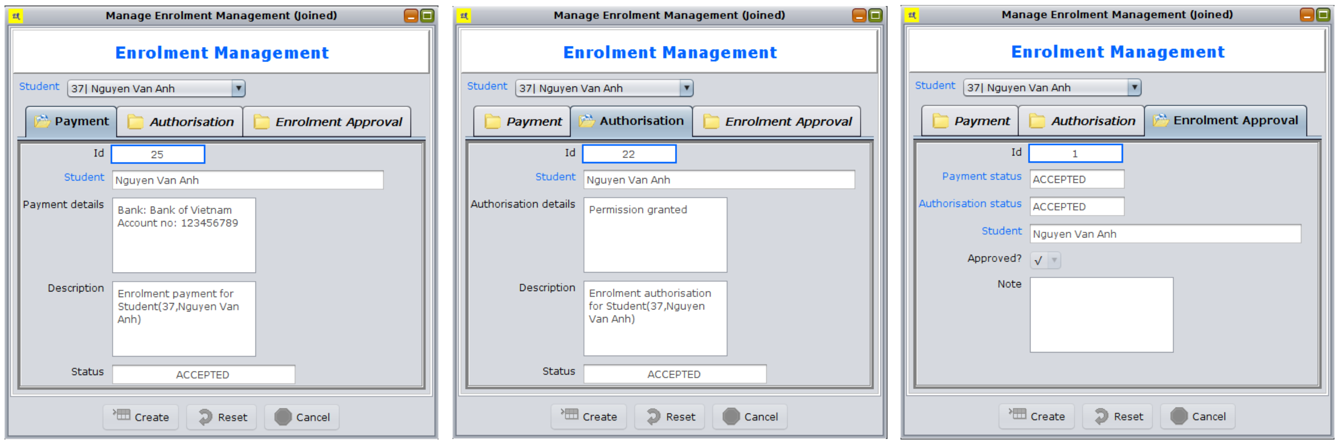
Student: 37| Nguyen Van Anh

Id:
Payment status: ACCEPTED
Authorisation status: ACCEPTED
Student: Nguyen Van Anh
Approved? √
Note:

(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 12: Joined pattern example: actions to be performed on the GUI.



Student: 37| Nguyen Van Anh

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

Student: 37| Nguyen Van Anh

Id: 22
Student: Nguyen Van Anh
Authorisation details: Permission granted
Description: Enrolment authorisation for Student(37,Nguyen Van Anh)
Status: ACCEPTED

Student: 37| Nguyen Van Anh

Id: 1
Payment status: ACCEPTED
Authorisation status: ACCEPTED
Student: Nguyen Van Anh
Approved? √
Note:

Figure 13: The three result objects: Payment, Authorisation, Enrolment Approval.

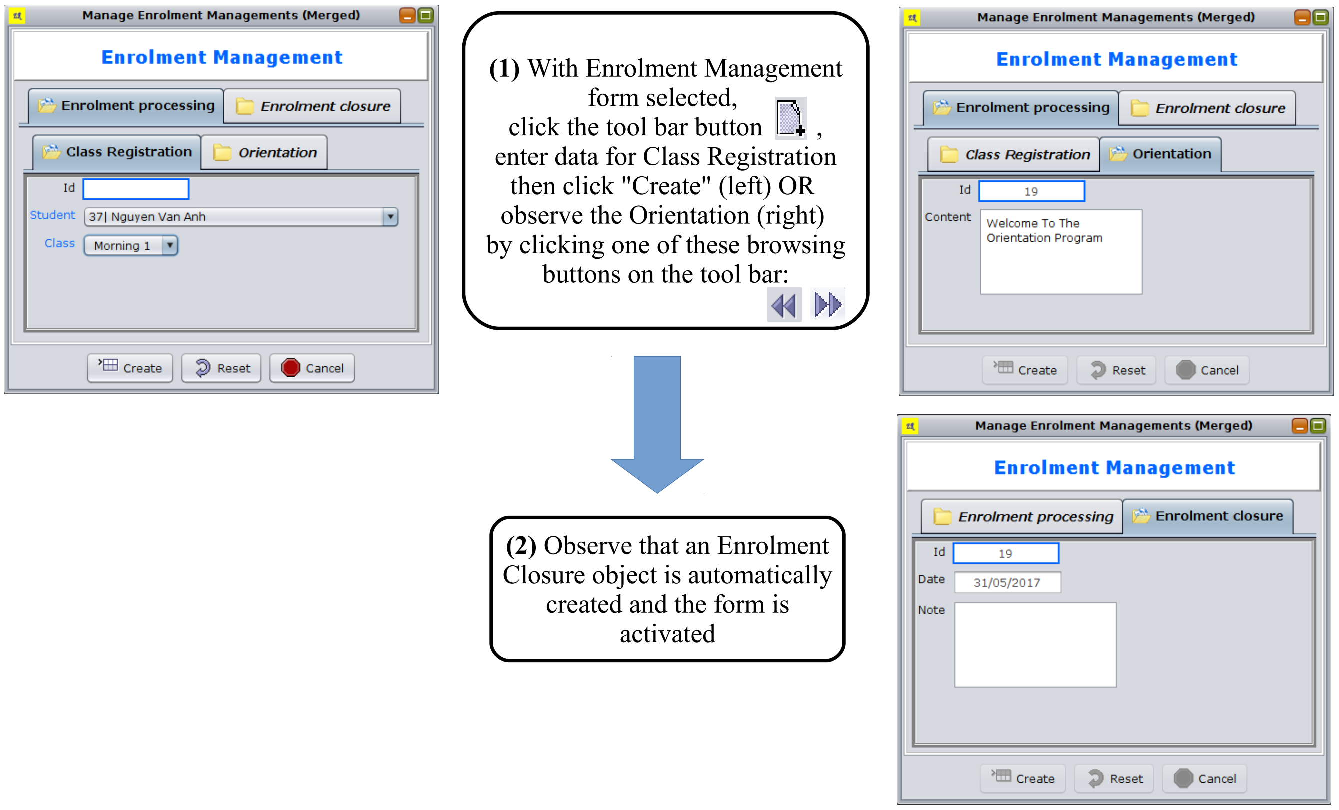
5.6. Merged Software

Command

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

Result

(1) Figure 14 displays the actions performed on the activity's GUI and the objects that are created.



Id

Student: 37| Nguyen Van Anh

Class: Morning 1

(1) With Enrolment Management form selected, click the tool bar button , enter data for Class Registration then click "Create" (left) OR observe the Orientation (right) by clicking one of these browsing buttons on the tool bar:

Id: 19

Content: Welcome To The Orientation Program

(2) Observe that an Enrolment Closure object is automatically created and the form is activated

Id: 19

Date: 31/05/2017

Note

Figure 14: Merged pattern example: actions performed on the GUI.

References

[1] D. M. Le, “jDomainApp: A Domain-Driven Application Development Framework in Java,” Hanoi University, 2016.
[2] D. M. Le, D.-H. Dang, and V.-H. Nguyen, “Domain Driven Design Using Annotation-Based DSL: Bridging the Modelling Gaps and Beyond [Extended],” VNU University of Engineering and Technology, 2017.
[3] D. M. Le, *DCSL implementation in jDomainApp*. vnu-dse, 2017. [Online]. Available: https://github.com/vnu-dse/dcsl
[4] F. Tomassetti, D. van Bruggen, and N. Smith, “JavaParser,” 2016. [Online]. Available: http://javaparser.org/. [Accessed: 31-Oct-2016].