Master Schedulizer

Developer's Manual

Created by:

Anh Huynh

Alex Marion

Mahmuda Liza

Purav Barot

Ryan Efendy

Version 1.0

March 3rd, 2016

ABOUT THIS MANUAL

Audience

This manual is intended for programmers wishing to customize, extend or interact with MasterSchedulizer. Programmers wishing to extend or customize MasterSchedulizer are expected to have intermediate experience in the Java programming language, as well as object oriented design principles.

What's in the manual

- General description and specifications of the program
- Detailed description about each module of the program

TABLE OF CONTENTS

1. **About the MasterSchedulizer**
2. **Modules**
   - 2.1. DriverPackage
     - 2.1.1. Main
     - 2.1.2. POSManager
   - 2.2. IOPackage
     - 2.2.1. IO
     - 2.2.2. Terminal IO
   - 2.3. MajorPackage
     - 2.3.1. Major
     - 2.3.2. BaseMajor
     - 2.3.3. MajorDecorator
     - 2.3.4. CSMajor
   - 2.4. MenuPackage
     - 2.4.1. MenuManager
     - 2.4.2. MenuChoice
     - 2.4.3. Menu
     - 2.4.4. StartMenu
     - 2.4.5. BackMenu
     - 2.4.6. QuitMenu
     - 2.4.7. ViewSamplePOSMenu
     - 2.4.8. LoadPOSMenu
     - 2.4.9. CreatePOSMenu
     - 2.4.10. MoveCourseMenu
     - 2.4.11. RemoveCourseMenu
     - 2.4.12. AddCourseMenu
     - 2.4.13. SaveMenu
     - 2.4.14. ViewPOSMenu
     - 2.4.15. SelectCourseMenu
     - 2.4.16. SelectQuarterMenu
     - 2.4.17. SelectYearMenu
   - 2.5. MinorPackage
     - 2.5.1. Minor
     - 2.5.2. BaseMinor
     - 2.5.3. MinorDecorator
   - 2.6. ParserPackage
     - 2.6.1. CSVParser
     - 2.6.2. CourseLoader
     - 2.6.3. CSVWriter
     - 2.6.4. TMSMasterList
   - 2.7. POSPackage

2.7.1. Course
2.7.2. GeneratePOS
2.7.3. POS
2.7.4. Quarter
2.7.5. Year
2.8. TestingPackage
2.9. ValidPOSGenerator
2.9.1. ValidPOSGenerator

1. About MasterSchedulizer

1.1. Goal

MasterSchedulizer serves to streamline the process of creating a plan of study, as well as to allow users to keep track of requirements and electives. Given a user's major(s) and minor(s), MasterSchedulizer creates a generic template which outlines a full college plan. The user can then add tracks, minors, electives, and other classes of interest. MasterSchedulizer will also track requirements and make sure that the plan of study doesn't violate university policy and is possible in the given timeframe (all courses must be available in the chosen term, etc.).

If integrated into Drexel's systems MasterSchedulizer would allow advisors to track advisees with greater ease. Heads of departments could also track the number of students who are taking a specific course in any given quarter giving a better estimate of the needed sections.

1.2. Specifications

1.2.1. Design Method

The design of this product utilizes an object-oriented approach.

1.2.2. User Interface

The user will be interfacing with the Master Schedulizer System that will allow user to select tracks and help generate Plan of Study. Given the user's major and tracks, it will generate a sufficient Plan of Study for graduation.

Master Schedulizer will be designed such that the user can accurately self-report their own behavior. Overall interface will be intuitive to use and easy to remember.

1.2.3. Hardware Interface

Master Schedulizer will perform on any machines with the JVM (Java Virtual Machine) installed.

1.2.4. Software Interface

Master Schedulizer will be platform independent as a Java program (preferably running in Eclipse IDE).

2. Modules

2.1. DriverPackage

2.1.1. Main

The Main class is the main driver of the code, it instantiates the Menu and allows for user input to the program. It also instantiates the TMSMasterList which is a list of all classes offered by drexel.

2.1.2. POSManager

The POSManager (plan of study manager) class is the main part of the program which the user interacts with. When the user gives input via the menu, the input is given to the POSManager which can perform actions on a user's plan of study.

2.2. IOPackage

2.2.1. IO

The IO class is an abstract class which was created to separate input and output from the functionality of the program (separating the implementation from the abstraction).

2.2.2. Terminal IO

The TerminalIO class is a subclass of the IO class which prints output to the terminal.

2.3. MajorPackage

2.3.1. Major

The Major class is an abstract class which represents the requirements for a college major. The Major classes follow the decorator pattern in order to stack multiple majors on top of one another.

2.3.2. BaseMajor

The BaseMajor is a sublcass of Major and contains attributes shared by all majors in the university. The BaseMajor is wrapped in the center of all MajorDecorators.

2.3.3. MajorDecorator

The MajorDecorator is abstract subclass of the Major class designed so that it can contain another instance of type Major. This allows for the creation of double majors without having to re-write concrete majors.

2.3.4. CSMajor

The CSMajor is a subclass of MajorDecorator and is a concrete decorator. CSMajor contains lists of all of the requirements for the computer science major.

2.4. MenuPackage

2.4.1. MenuManager

The MenuManager class contains a list of the menus and runs them in a loop. It handles operations between menus and handles the menu flow.

2.4.2. MenuChoice

The MenuChoice class is contained by each menu and contains a string which represents the choice text and an index which represents its position in the choice list.

2.4.3. Menu

The Menu class simulates the command prompt where the user will type in the command they wish to execute.

2.4.4. StartMenu

The StartMenu class simulates the welcoming screen when the user first starts the program, allowing the user to either load an existing POS, create a new POS or view a sample POS.

2.4.5. BackMenu

The BackMenu class allows going back to the previous menu and prints a message saying so.

2.4.6. QuitMenu

The QuitMenu class simulates

2.4.7. LoadPOSMenu

The LoadPOSMenu simulates the screen after the user chooses to load a POS from the local machine.

2.4.8. CreatePOSMenu

The CreatePOSMenu simulates the screen that prompts user to create a new POS from scratch.

2.4.9. MoveCourseMenu

The MoveCourseMenu class moves course throughout the POS and simulates the screen that shows whether or not that course has been successfully moved.

2.4.10. RemoveCourseMenu

The RemoveCourseMenu class removes course from the POS and simulates the screen that shows whether or not that course has been successfully removed from the POS.

2.4.11. AddCourseMenu

The AddCourseMenu class adds course to the POS and simulates the

screen that shows whether or not that course has been successfully added to the POS.

2.4.12. SaveMenu

The SaveMenu class saves current POS to a local machine.

2.4.13. PrintSamplePOSMenu

The PrintSamplePOSMenu class simulates the screen that shows user the sample POS.

2.4.14. ViewPOSMenu

The ViewPOSMenu class simulates the screen that allows user to view the current POS.

2.4.15. SelectCourseMenu

The SelectCourseMenu class simulates the screen that allows user to choose an action to do with a course after entering the course's name.

2.4.16. SelectQuarterMenu

The SelectQuarterMenu class simulates the screen that allows user to choose an action to do with this quarter after a quarter has been chosen in the previous screen.

2.4.17. SelectYearMenu

The SelectYearMenu class simulates the screen that allows user to choose an action to do with a year after this year has been chosen in the previous screen.

2.5. MinorPackage

2.5.1. Minor

The Major class is an abstract class which represents the requirements for a college minor. The Minorclasses follow the decorator pattern in order to stack multiple minors on top of one another.

2.5.2. BaseMinor

The BaseMinor is a sublcass of Minor and contains attributes shared by all minors in the university. The BaseMinor is wrapped in the center of all MinorDecorators.

2.5.3. MinorDecorator

The MinorDecorator is abstract subclass of the Minor class designed so that it can contain another instance of type Minor. This allows for the creation of double majors without having to re-write concrete majors.

2.6. ParserPackage

2.6.1. CSVParser

The CSVParser class takes in the filepath of a csv file and parse the fields of that csv into an array of Strings to be used later on in the CourseLoader class.

2.6.2. CourseLoader

The CourseLoader class populates Course objects with the information from the csv file. This is achieved by having an instance of the CSVParser class described above.

2.6.3. CSVWriter

The CSVWriter class writes an ArrayList of POS objects to a csv file. This supports the saving and storing POS requirement of the program.

2.6.4. TMSMasterList

The TMSMasterList class constructs an ArrayList of all the courses that are offered at Drexel University. Other classes can use this class to check whether or not a class given by the user is a valid one that Drexel offers.

2.7. POSPackage

2.7.1. Course

The Course class is the implementation of a typical Drexel class, with the following properties:

- Subject Code
- Course Num
- Course Title
- Credits
- Prerequisites
- Corequisites
- Terms Offered
- Term Taken by the student

2.7.2. Quarter

The Quarter class is the implementation of a typical Drexel quarter, with the following properties:

- Quarter ID
- Credits
- Term
- Courses

2.7.3. Year

The Year class is the implementation of a typical Drexel school year, with the following properties:

- Year Name
- Quarter (an ArrayList of Quarter Objects)

2.7.4. POS

The Course class is the implementation of a Plan of Study for a particular student, with the following properties:

- Student Name
- Student ID
- Major
- Minor
- Years (an ArrayList of Year Objects)

2.7.5. GeneratePOS

The GeneratePOS class constructs the table template for a typical POS.

2.8. TestingPackage

This Package contains unit test cases for multiple classes in other modules. The test cases are implemented using JUnit.

2.9. ValidPOSGenerator

2.9.1. ValidPOSGenerator

The ValidPOSGenerator class generates all valid plans of study in order to check that a user generated plan of study is valid.