KIT - Karlsruhe Institute of Technology

Design and architecture report

Study Planning by Generating Workflows with Compliance Requirements: Plan Creation and Visualisation

January 11, 2017

IPD Böhm

Phase	**Supervisor**
Functional specification	Ali Bejhad
Design and architecture	Janek Westfechtel
Implementation	Clemens Naseband
Qualitycontrol	Robin Berger
Presentation	Jacques Huss

Contents

Part 1

General

Chapter 1

Introduction

This document will guide through our software design, using UML-diagrams. The main topic of this document will be the class diagram, inspired by the MVC-principle. The interaction of different objects will be presented in several sequence diagrams.
All desired criteria will be met, but not all interfaces have been added, to avoid empty methods in the implementation, but it is ensured that every missing part can be added easily.

Chapter 2

Design Overview

2.1 Architecture

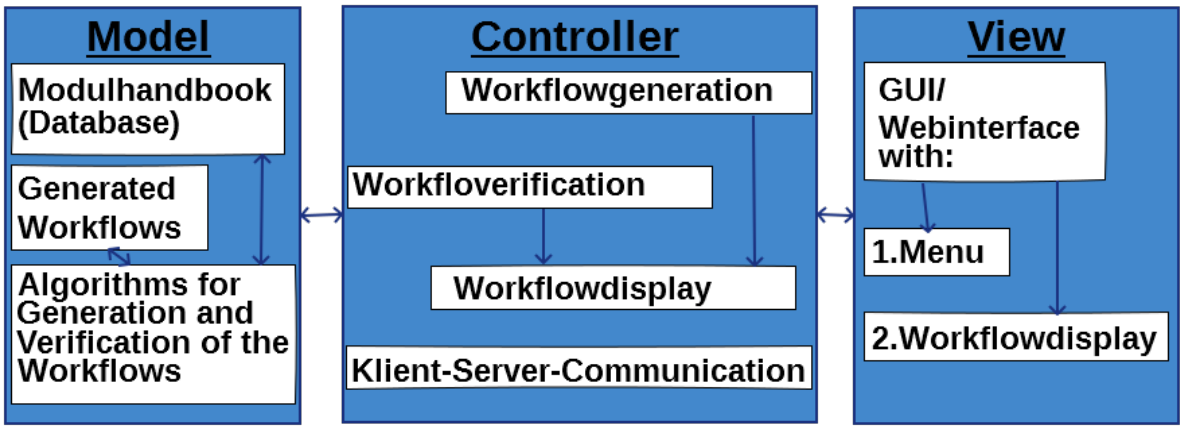


Figure 2.1: MVC-Architecture

The program is designed using the principles of the MVC-architecture. This divides it into three different components Model, View, and Controller, making it easier to modify certain aspects, change strategies, and porting it to other platforms.

Model:
The Model contains the user, system data and the workflow algorithms. A database is offering the module manual with all of it's constraints. The algorithms can verify and generate workflows.

View
The View has 2 tasks: On the one hand it provides the user with a GUI to create a workflow with the given preferences. On the other hand it presents the generated or verified workflow.

Controller
The Controller is the central unit of the system. It coordinates the generation and verification

of the workflows by translating the user input, received from the View, into the needed parameters from the Model. After that the new or verified workflow, received from the Model, is transferred to the View, so that it can display it.

2.2 Class diagram

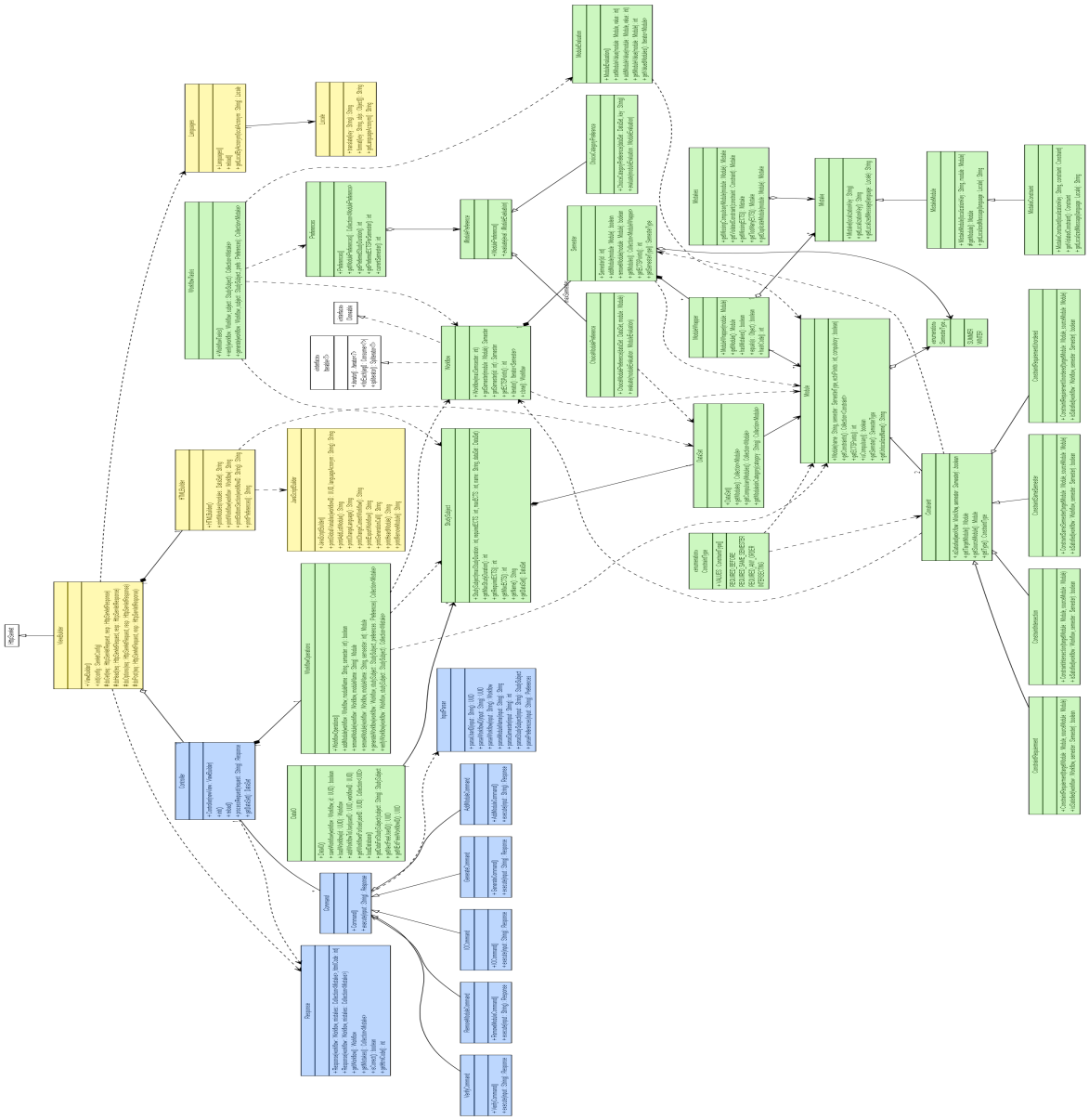


Figure 2.2: Class diagram

Model (Green), View (Yellow), Controller (Blue), Java libraries (White)

Part 2

Controller

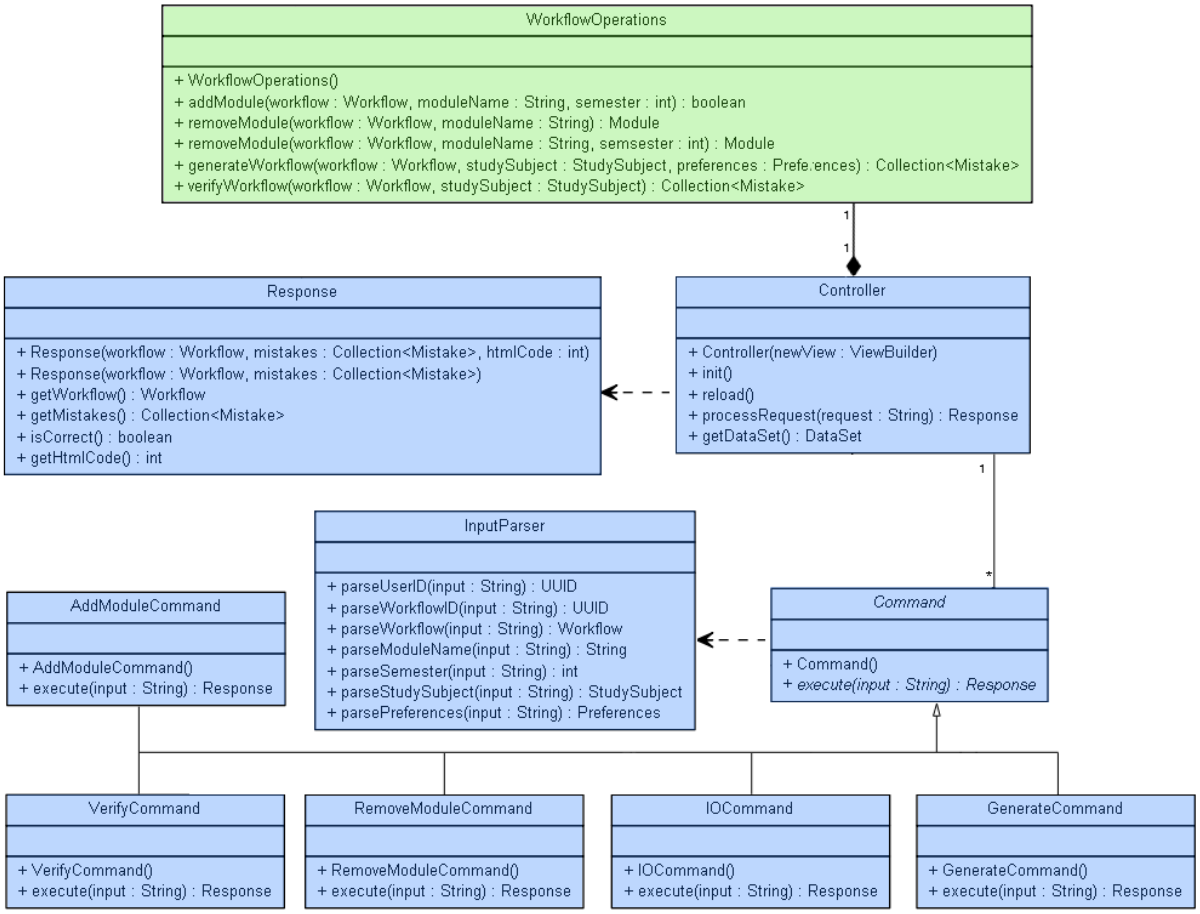


Figure 2.3: Controller

Chapter 3

Package studyplanning.controller.commands

Package Contents *Page*

Classes

3.1 Class AddModuleCommand

Adds a Module to an existing Workflow

3.1.1 Declaration

```
public class AddModuleCommand
  extends studyplanning.controller.commands.Command
```

3.1.2 Constructor summary

AddModuleCommand() Creates a new AddModuleCommand instance

3.1.3 Method summary

execute(String)

3.1.4 Constructors

- **AddModuleCommand**

```
public AddModuleCommand()
```

  - **Description** Creates a new AddModuleCommand instance

3.1.5 Methods

- **execute**

```
public abstract Response execute(String input)
```

  - **Description** Adds a Module to an existing Workflow
  - **Parameters**
    - `input` – A message generated by the View.
  - **Returns** – A Response containing the changed Workflow. If the addition violates any Contraints, some error messages will be contained in the Response.

3.1.6 Members inherited from class Command

`studyplanning.controller.commands.Command` (in 3.2, page 16)

- `public abstract Response execute(java.lang.String input)`

3.2 Class Command

With the abstract Command class commands can be created and added with the help of inheritance.

3.2.1 Declaration

```
public abstract class Command
  extends java.lang.Object
```

3.2.2 All known subclasses

VerifyCommand (in 3.6, page 20), RemoveModuleCommand (in 3.5, page 19), IOCommand (in 3.4, page 18), GenerateCommand (in 3.3, page 17), AddModuleCommand (in 3.1, page 15)

3.2.3 Constructor summary

Command()

3.2.4 Method summary

execute(String) This method will take all actions required to process a request. It will return a Response indicating the outcome of the Command.

3.2.5 Constructors

- **Command**

```
public Command()
```

3.2.6 Methods

- **execute**

```
public abstract Response execute(String input)
```

  - **Description** This method will take all actions required to process a request. It will return a Response indicating the outcome of the Command.
  - **Parameters**
    - `input` – A message generated by the View.
  - **Returns** – A Response the contents of which will be specified in the subclasses

3.3 Class GenerateCommand

Generates a completely new Workflow. The new Workflow will be created according to the constraints, and will be completed by the Model

3.3.1 Declaration

```
public class GenerateCommand
  extends studyplanning.controller.commands.Command
```

3.3.2 Constructor summary

GenerateCommand() Creates a new GenerateCommand instance

3.3.3 Method summary

execute(String) Creates a new Workflow, using the algorithms in the Model package.

3.3.4 Constructors

- **GenerateCommand**

```
public GenerateCommand()
```

  - **Description** Creates a new instance

3.3.5 Methods

- **execute**

```
public Response execute(String input)
```

  - **Description** Creates a new Workflow, using the algorithms in the Model package.
  - **Parameters**
    - `input` – The message generated by the View
  - **Returns** – A Response containing the newly generated Workflow.
    Errors during the creation of the Workflows or, if the constraints don't allow the creation of a correct Workflow, will cause the Response to contain the error messages

3.3.6 Members inherited from class Command

`studyplanning.controller.commands.Command` (in 3.2, page 16)

- `public abstract Response execute(java.lang.String input)`

3.4 Class IOCommand

The IOCommand manages user data, and is used to save and load Workflows.

3.4.1 Declaration

```
public class IOCommand
  extends studyplanning.controller.commands.Command
```

3.4.2 Constructor summary

IOCommand() Creates a new IOCommand instance

3.4.3 Method summary

execute(String)

3.4.4 Constructors

- **IOCommand**

  ```
  public IOCommand()
  ```

  - **Description** Creates a new IOCommand instance

3.4.5 Methods

- **execute**

  ```
  public abstract Response execute(String input)
  ```

  - **Description** Loads a Workflow or creates a new one, if there was no Workflow found with the given UUID.
  - **Parameters**
    - `input` – A message generated by the View.
  - **Returns** – A Response containing the Workflow with the given UUID or containing an empty Workflow, if the given UUID is not yet associated with a Workflow.

3.4.6 Members inherited from class Command

`studyplanning.controller.commands.Command` (in 3.2, page 16)

- `public abstract Response execute(java.lang.String input)`

3.5 Class RemoveModuleCommand

Removes Modules from a Workflow

3.5.1 Declaration

```
public class RemoveModuleCommand
  extends studyplanning.controller.commands.Command
```

3.5.2 Constructor summary

RemoveModuleCommand() Creates a new RemoveModuleCommand instance

3.5.3 Method summary

execute(String)

3.5.4 Constructors

- **RemoveModuleCommand**

```
public RemoveModuleCommand()
```

  - **Description** Creates a new instance of the RemoveModuleCommand.

3.5.5 Methods

- **execute**

```
public abstract Response execute(String input)
```

  - **Description** Removes a Module from a Workflow.
  - **Parameters**
    - `input` – A message generated by the View.
  - **Returns** – A Response that contains the changed Workflow, and information about the performed action

3.5.6 Members inherited from class Command

`studyplanning.controller.commands.Command` (in 3.2, page 16)

- `public abstract Response execute(java.lang.String input)`

3.6 Class VerifyCommand

Checks, whether the Workflow complies with the module manual.

3.6.1 Declaration

```
public class VerifyCommand
  extends studyplanning.controller.commands.Command
```

3.6.2 Constructor summary

VerifyCommand() Creates a new VerifyCommand instance

3.6.3 Method summary

execute(String) Verifies whether the Workflow is correct.

3.6.4 Constructors

- **VerifyCommand**

  ```
  public VerifyCommand()
  ```

  - **Description** Creates a new instance of the VerifyCommand.

3.6.5 Methods

- **execute**

  ```
  public Response execute(String input)
  ```

  - **Description** Verifies the Workflow.
  - **Parameters**
    - `input` – A message generated by the View.
  - **Returns** – A Response that shows if the Workflow is correct.
    The Response will contain error messages if the Workflow is not correct

3.6.6 Members inherited from class Command

`studyplanning.controller.commands.Command` (in 3.2, page 16)

- `public abstract Response` **`execute`**`(java.lang.String` **`input`**`)`

Chapter 4

Package studyplanning.controller

Package Contents *Page*

Classes

4.1 Class Controller

The Controller class exists to coordinate between Model and View.

4.1.1 Declaration

```
public class Controller
```

4.1.2 Constructor summary

Controller(ViewBuilder) Creates a new Controller instance

4.1.3 Method summary

getDataSet() Retrieves a DataSet containing all the Modules
init() Initiates the server program and loads the data base
processRequest(String) Processes a View request and returns a Response
reload() Reloads the database

4.1.4 Constructors

- **Controller**

```
public Controller(ViewBuilder newView)
```

- **Description** Creates a new Controller instance with a specified ViewBuilder.
- **Parameters**
  - `newView` – The Servlet to build an application runnable on a server

4.1.5 Methods

- **getDataSet**

```
public DataSet getDataSet()
```

  - **Description** Retrieves a DataSet containing all Modules
  - **Returns** – A DataSet containing every Module

- **init**

```
public void init()
```

  - **Description** Initiates the server program and loads the database

- **processRequest**

```
public Response processRequest(String request)
```

  - **Description** Processes a View request and returns a Response
  - **Parameters**
    - `request` – The String generated by the View
  - **Returns** – A Response giving the View its demanded information

- **reload**

```
public void reload()
```

  - **Description** Reloads the database

4.2 Class InputParser

The InputParser class is used to parse the messages generated by the View, and get the specified objects.
All of the View's messages are coded in a string, this class offers methods to decode these messages.

4.2.1 Declaration

```
public class InputParser
```

4.2.2 Method summary

parseModuleName(String) Parses the View's input to get the name of a Module
parsePreferences(String) Parses the Views input and retrieves the Preferences
parseSemester(String) Parses the View's input for the given Semester
parseStudySubject(String) Parses the View's input and the StudySubject from the Model
parseUserID(String) Parses a user UUID
parseWorkflow(String) Parses the View's input and parses a Workflow from the Model
parseWorkflowID(String) Parses a Workflow's UUID

4.2.3 Methods

- **parseModuleName**

  ```
  public static String parseModuleName(String input)
  ```

  - **Description** Parses the View's input to get the name of a Module
  - **Parameters**
    - `input` – The string message generated by the View
  - **Returns** – The Modules name as a string

- **parsePreferences**

  ```
  public static Preferences parsePreferences(jString input)
  ```

  - **Description** Parses the views input and retrieves the Preferences
  - **Parameters**
    - `input` – The message generated by the View
  - **Returns** – The parsed users Preferences

- **parseSemester**

  ```
  public static int parseSemester(String input)
  ```

  - **Description** Parses the View's input for the given Semester
  - **Parameters**
    - `input` – The message generated by the View
  - **Returns** – The parsed semester number

- **parseStudySubject**

  ```
  public static StudySubject parseStudySubject(String input)
  ```

  - **Description** Parses the View's input and the StudySubject from the Model
  - **Parameters**
    - `input` – The message generated by the View
  - **Returns** – The parsed StudySubject

- **parseUserID**

  ```
  public static UUID parseUserID(String input)
  ```

  - **Description** Parses a user id
  - **Parameters**
    - `input` The message generated by the View
  - **Returns** – The parsed id

- **parseWorkflow**

  ```
  public static Workflow parseWorkflow(String input)
  ```

  - **Description** Parses the View's input and parses a Workflow from the Model
  - **Parameters**
    - `input` – The message generated by the View
  - **Returns** – A parsed Workflow

- **parseWorkflowID**

  ```
  public static UUID parseWorkflowID(String input)
  ```

  - **Description** Parses a Workflow's id
  - **Parameters**
    - `input` – The message generated by the View
  - **Returns** – The parsed Workflow's id

4.3 Class Response

Passes generated and changed Workflows to the View.
Allows the Controller to communicate with the View.
Each Response consits of a Workflow, a Collection of Mistakes, and an HTML-statuscode

4.3.1 Declaration

```
public class Response
```

4.3.2 Constructor summary

Response(Workflow, Collection) Creates a new Response instance with the default "correct" an HTML-Code 200
Response(Workflow, Collection, int) Creates a new Response instance

4.3.3 Method summary

getHtmlCode() Getter method for the an HTML-Code.
getMistakes() Returns the Collection of Mistakes contained, if this is a Response for a failed generation or verification task, an empty Collection otherwise.
getWorkflow() Returns the current Workflow.
isCorrect() Checks whether there are any Mistakes contained in the Response.

4.3.4 Constructors

- **Response**

```
public Response(Workflow workflow, Collection mistakes)
```

  - **Description** Creates a new Response Instance with the default "correct" an HTML-Code 200
  - **Parameters**
    - `workflow` – A Workflow that needs to be passed on to the View
    - `mistakes` – A Collection containing information about relevant errors

- **Response**

```
public Response(Workflow workflow, Collection<Mistakes> mistakes
    , int htmlCode)
```

  - **Description** Creates a new Response instance
  - **Parameters**
    - `workflow` – A Workflow, that needs to be passed on to the View
    - `mistakes` – A Collection containing information about relevant errors
    - `htmlCode` – An HTML-Code that can be used to pass on different information to the View

4.3.5 Methods

- **getHtmlCode**

  ```
  public int getHtmlCode()
  ```

  - **Description** Getter method for the HTML-Code
  - **Returns** – The Response HTML-Code

- **getMistakes**

  ```
  public Collection<Mistake> getMistakes()
  ```

  - **Description** Returns the Collection of Mistakes contained in the Workflow, if this is a Response for a failed generation or verification task, an empty Collection otherwise. Empty if Workflow is correct.
  - **Returns** – A Collection of Mistakes.

- **getWorkflow**

  ```
  public Workflow getWorkflow()
  ```

  - **Description** Returns the current Workflow.
  - **Returns** – The current Workflow.

- **isCorrect**

  ```
  public boolean isCorrect()
  ```

  - **Description** Checks if there are any Mistakes contained in the Response.
  - **Returns** – True, if there are not any Mistakes

Part 3

Model

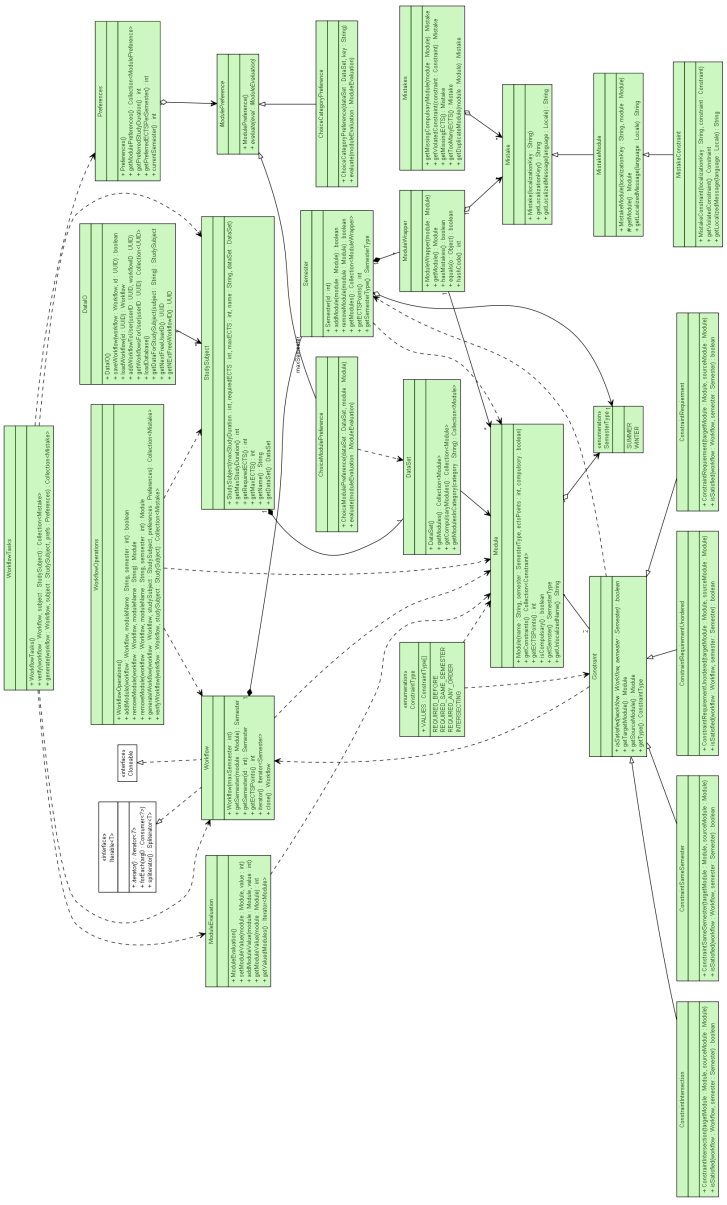


Figure 4.1: Model

Chapter 5

Package studyplanning.model

Package Contents *Page*

Classes

5.1 Class DataIO

Saves and loads user data. Loads the database.

5.1.1 Declaration

```
public final class DataIOt
```

5.1.2 Constructor summary

DataIO()

5.1.3 Method summary

addWorkflowToUser(UUID, UUID) Associates another Workflow ID with the given user.
getDataForStudySubject(String) Returns the StudySubject for the given name.
getNextFreeUserID() Generates and returns a new UUID for use with a user.
getNextFreeWorkflowID() Generates and returns a new UUID for use with a Workflow.
getWorkflowsForUser(UUID) Returns all Workflow IDs associated with the given user.
loadDatabase() Loads the database.
loadWorkflow(UUID) Loads the Workflow for the given UUID.
saveWorkflow(Workflow, UUID) Saves the Workflow with the given id, so it can be fetched with `loadWorkflow(UUID)`

5.1.4 Constructors

- **DataIO**

```
public DataIO()
```

  - **Description** Creates a new instance and loads the database with all StudySubjects.

5.1.5 Methods

- **addWorkflowToUser**

```
public static void addWorkflowToUser(UUID userID, UUID
    workflowID)
```

  - **Description** Associates another Workflow ID with the given user.
  - **Parameters**
    - `userID` – The user the Workflow ID should be added to.
    - `workflowID` – The Workflow ID to be added to the user.

- **getDataForStudySubject**

```
public StudySubject getDataForStudySubject(String subject)
```

  - **Description** Returns the StudySubject for the given name.
  - **Parameters**
    - `subject` – The name of the study subject.
  - **Returns** – The requested StudySubject.

- **getNextFreeUserID**

```
public UUID getNextFreeUserID()
```

  - **Description** Generates and returns a new UUID for use with a user.
  - **Returns** – A new user UUID.

- **getNextFreeWorkflowID**

```
public UUID getNextFreeWorkflowID()
```

  - **Description** Generates and returns a new UUID for use with a Workflow.
  - **Returns** – A new Workflow UUID.

- **getWorkflowsForUser**

  ```
  public Collection<UUID> getWorkflowsForUser(UUID userID)
  ```

  - **Description** Returns all Workflow IDs associated with the given user.
  - **Parameters**
    - `userID` – The user ID to get the Workflow ID's from.
  - **Returns** – A Collection of all Workflow IDs associated with the given user.

- **loadDatabase**

  ```
  public synchronized void loadDatabase()
  ```

  - **Description** Loads the database.

- **loadWorkflow**

  ```
  public Workflow loadWorkflow(UUID id)
  ```

  - **Description** Loads the Workflow for the given UUID.
  - **Parameters**
    - `id` – The UUID to load the Workflow from.
  - **Returns** – The Workflow previously saved for the given UUID.

- **saveWorkflow**

  ```
  public boolean saveWorkflow(Workflow workflow, UUID id)
  ```

  - **Description** Saves the Workflow with the given id, so it can be fetched with `loadWorkflow(UUID)`
  - **Parameters**
    - `workflow` – The Workflow to save.
    - `id` – The Workflow's identifier.
  - **Returns** – True, if saving succeeded

5.2 Class WorkflowOperations

Facade for all workflow operations.

5.2.1 Declaration

```
public class WorkflowOperations
```

5.2.2 Constructor summary

WorkflowOperations()

5.2.3 Method summary

addModule(Workflow, String, int) Adds a Module to the Workflow.
generateWorkflow(Workflow, StudySubject, Preferences) Use this method to generate a Workflow of the given preferences.
removeModule(Workflow, String) Removes a Module from the workflow.
removeModule(Workflow, String, int) Removes a Module from the given Semester of the Workflow.
verifyWorkflow(Workflow, StudySubject) Use this method to verify if the Workflow is correct.

5.2.4 Constructors

- **WorkflowOperations**

```
public WorkflowOperations()
```

  - **Description** This class works as a proxy object, making it easier to control access.

5.2.5 Methods

- **addModule**

```
public boolean addModule(Workflow workflow, String moduleName,
    int semester)
```

  - **Description** Adds a Module to the Workflow.
  - **Parameters**
    - `workflow` – Workflow to be edited
    - `moduleName` – Name and identifier of the Module
    - `semester` – The Semester the Module should be added to
  - **Returns** – True, if adding the Module succeeded

- **generateWorkflow**

```
public Collection<Mistake> generateWorkflow(Workflow workflow,
    StudySubject studySubject, Preferences preferences)
```

  - **Description** Use this method to generate a Workflow of the given Preferences. This changes the Workflow.
  - **Parameters**
    * `workflow` – The Workflow generated by the user or the system.
    * `studySubject` – The StudySubject related to the Workflow
    * `preferences` – All user preferences. Module related preferences have to be added in this object
  - **Returns** – A Collection of Mistakes occurred during generation.

- **removeModule**

```
public Module removeModule(Workflow workflow, String moduleName)
```

  - **Description** Removes a Module from the Workflow.
  - **Parameters**
    * `workflow` – Workflow to be edited
    * `moduleName` – Name and identifier of the Module
  - **Returns** – The Module which has been removed.

- **removeModule**

```
public Module removeModule(Workflow workflow, String moduleName,
    int semester)
```

  - **Description** Removes a Module from the given Semester of the Workflow.
  - **Parameters**
    * `workflow` – Workflow to be edited
    * `moduleName` –Identifier of the Module
    * `semester` – The Module's Semester
  - **Returns** – The removed Module, null if not found

- **verifyWorkflow**

```
public Collection<Mistake> verifyWorkflow(Workflow workflow,
   StudySubject studySubject)
```

  - **Description** Use this method to verify if the Workflow is correct.
  - **Parameters**
    * `workflow` – The Workflow generated by the user or the system.
    * `studySubject` – The related StudySubject the user studies
  - **Returns** – A Collection containing all module manual violations.

Chapter 6

Package studyplanning.model.mistake

Package Contents *Page*

Classes

6.1 Class Mistake

Errors made by the user are listed here. Objects of this class will be generated by the verify method of the Model class.

6.1.1 Declaration

```
public class Mistake
```

6.1.2 All known subclasses

MistakeModule (in 6.3, page 38), MistakeConstraint (in 6.2, page 36)

6.1.3 Constructor summary

Mistake(String) Creates a new Mistake object with a given message.

6.1.4 Method summary

getLocalizationKey() Returns the localization key for this Mistake.
getLocalizedMessage(Locale) Returns a readable String for a Mistake.

6.1.5 Constructors

- **Mistake**

```
public Mistake(String localizationKey)
```

  - **Description** Creates a new Mistake object with a given message.
  - **Parameters**
    - `localizationKey` – A string containing they key word for a Mistake

6.1.6 Methods

- **getLocalizationKey**

```
public String getLocalizationKey()
```

  - **Description** Returns the localization key for this Mistake. May not be null. Returns the localization key for this mistake.
  - **Returns** – The localization key for this Mistake.

- **getLocalizedMessage**

```
public String getLocalizedMessage(Locale language)
```

  - **Description** Returns a readable String for a Mistake.
  - **Parameters**
    - `language` – The target Language
  - **Returns** – The translated String

6.2 Class MistakeConstraint

A subclass of the `MistakeModule` (in 6.3, page 38) class. This class contains the affected Constraint and its Modules.

(in 6.3, page 38) class. This class contains the affected Constraint and its Modules.

6.2.1 Declaration

```
public class MistakeConstraint
 extends studyplanning.model.mistake.MistakeModule
```

6.2.2 Constructor summary

MistakeConstraint(String, Constraint) Creates a new Mistake with the given Mistake key word and the affected Constraint.

6.2.3 Method summary

getLocalizedMessage(Locale)
getViolatedConstraint() Returns the violated Constraint, this Mistake represents.

6.2.4 Constructors

- **MistakeConstraint**

  ```
  public MistakeConstraint(String localizationKey, Constraint
      constraint)
  ```

  - **Description** Creates a new Mistake with the given Mistake key-word and the affected Constraint.
  - **Parameters**
    - `localizationKey` – The key-word for the Mistake
    - `constraint` – The affected Constraint

6.2.5 Methods

- **getLocalizedMessage**

  ```
  public String getLocalizedMessage(Locale language)
  ```

  - **Description copied from Mistake** (**in 6.1, page 35**) Returns a readable String for a mistake.
  - **Parameters**
    - `language` – The target Language
  - **Returns** – The translated String

- **getViolatedConstraint**

  ```
  public Constraint getViolatedConstraint()
  ```

  - **Description** Returns the violated Constraint, this Mistake represents.
  - **Returns** – The violated Constraint.

6.2.6 Members inherited from class MistakeModule

`studyplanning.model.mistake.MistakeModule` (in 6.3, page 38)

- `public String` **getLocalizedMessage**`(Locale` **language**`)`
- `protected Module` **getModule**`()`

6.2.7 Members inherited from class Mistake

`studyplanning.model.mistake.Mistake` (in 6.1, page 35)

- `public String` **getLocalizationKey**`()`
- `public String` **getLocalizedMessage**`(studyplanning.view.Locale` **language**`)`

6.3 Class MistakeModule

A subclass of the `Mistake` (in 6.1, page 35) class, which is used for Mistakes connected to a certain Module. This class is used if a Mistake is strictly coherent with a Module.

6.3.1 Declaration

```
public class MistakeModule
 extends studyplanning.model.mistake.Mistake
```

6.3.2 All known subclasses

MistakeConstraint (in 6.2, page 36)

6.3.3 Constructor summary

MistakeModule(String, Module) Creates a new object with localization key-word and a Module.

6.3.4 Method summary

getLocalizedMessage(Locale)
getModule() Returns the Module causing the Mistake.

6.3.5 Constructors

- **MistakeModule**

  ```
  public MistakeModule(String localizationKey, Module module)
  ```

  - **Description** Creates a new object with localization key-word and a Module.
  - **Parameters**
    - `localizationKey` – The Mistake key word
    - `module` – The coherent Module

6.3.6 Methods

- **getLocalizedMessage**

  ```
  public String getLocalizedMessage(Locale language)
  ```

  - **Description copied from Mistake (in 6.1, page 35)** Returns a String, which can be understood by a human, for a Mistake.
  - **Parameters**
    - `language` – The target Language
  - **Returns** – The translated String

- **getModule**

  ```
  protected Module getModule()
  ```

  - **Description** Returns the Module causing the Mistake.
  - **Returns** – The Module affected by the Mistake

6.3.7 Members inherited from class Mistake

`studyplanning.model.mistake.Mistake` (in 6.1, page 35)

- `public String` **getLocalizationKey**`()`
- `public String` **getLocalizedMessage**`(studyplanning.view.Locale` **language**`)`

6.4 Class Mistakes

Class with factory methods for quick access to new `Mistake` (in 6.1, page 35)s.

6.4.1 Declaration

```
public class Mistakes
```

6.4.2 Method summary

getDuplicateModule(Module) Returns a new Mistake representing a Module being duplicated in a Workflow.
getMissingCompulsaryModule(Module) Returns a new Mistake representing a missing compulsory Module.
getMissingECTS() Returns a new Mistake, in case of a Workflow with not enough ECTS points.
getTooManyECTS() Returns a new Mistake, in case of a Workflow with too many ECTS points.
getViolatedConstraint(Constraint) Returns a new Mistake representing a violated Constraint in a Workflow.

6.4.3 Methods

- **getDuplicateModule**

  ```
  public static Mistake getDuplicateModule(Module module)
  ```

  - **Description** Returns a new Mistake representing a Module being duplicated in a Workflow.
  - **Parameters**
    - `module` – The duplicated Module
  - **Returns** – The constructed Mistake.

- **getMissingCompulsaryModule**

  ```
  public static Mistake getMissingCompulsaryModule(Module module)
  ```

  - **Description** Returns a new Mistake representing a missing compulsory Module.
  - **Parameters**
    - `module` – The Module, that is missing.
  - **Returns** – The constructed Mistake.

- **getMissingECTS**

  ```
  public static Mistake getMissingECTS()
  ```

  - **Description** Returns a new Mistake, in case of a Workflow with not enough ECTS points.
  - **Returns** – The constructed Mistake.

- **getTooManyECTS**

  ```
  public static Mistake getTooManyECTS()
  ```

  - **Description** Returns a new Mistake, in case of a Workflow with too many ECTS points.
  - **Returns** – The constructed Mistake.

- **getViolatedConstraint**

  ```
  public static Mistake getViolatedConstraint(Constraint
      constraint)
  ```

- **Description** Returns a new Mistake representing a violated Constraint in a Workflow.
- **Parameters**
  - `constraint` – The Constraint that was violated.
- **Returns** – The constructed Mistake.

Chapter 7

Package studyplanning.model.workflow.constraint

Package Contents *Page*

Classes

7.1 Class Constraint

This represents a constraint with a way to verify it.

7.1.1 Declaration

```
public abstract class Constraint
```

7.1.2 All known subclasses

ConstraintSameSemester (in 7.5, page 47), ConstraintRequirementUnordered (in 7.4, page 46), ConstraintRequirement (in 7.3, page 45), ConstraintIntersection (in 7.2, page 44)

7.1.3 Constructor summary

Constraint(Module, Module)

7.1.4 Method summary

getSourceModule() Returns the secondary from this constraint applies to.
getTargetModule() Returns the from this constraint applies to.
getType() Returns the constraint type
isSatisfied(Workflow, Semester) Returns whether the Workflow satisfies this constraint.

7.1.5 Constructors

- **Constraint**

```
public Constraint(Module targetModule, Module sourceModule)
```

  - **Description** Creates a new constraint between two modules. The specification of each specific Constraint are considered in sub classes.
  - **Parameters**
    - `targetModule` – The Module acting as a dependency.
    - `sourceModule` – The Module, this dependency applies to.

7.1.6 Methods

- **getSourceModule**

```
public Module getSourceModule()
```

  - **Description** Returns the secondary Module this constraint applies to. If Module A requires Module B, then this will return Module A.
  - **Returns** – The source Module

- **getTargetModule**

```
public Module getTargetModule()
```

  - **Description** Returns the Module this constraint applies to. If Module A requires Module B, then this will return Module B.
  - **Returns** – The target Module

- **getType**

```
public ConstraintType getType()
```

- **Description** Returns the constraint type. See `ConstraintType` (in 7.6, page 48)
- **Returns** – The type of Constraint.

- **isSatisfied**

```
public abstract boolean isSatisfied(Workflow workflow, Semester
    semester)
```

- **Description** Returns whether the Workflow satisfies this constraint..
- **Parameters**
  - `workflow` – The Workflow to verify with the current Constraint.
  - `semester` – The Semester, the current Module is in.
- **Returns** – True, if the constraint is satisfied

7.2 Class ConstraintIntersection

Objects of this class are checking if another Module is in the same Semester as this. They cannot be in the same Semester.

7.2.1 Declaration

```
public class ConstraintIntersection
 extends studyplanning.model.workflow.constraint.Constraint
```

7.2.2 Constructor summary

ConstraintIntersection(Module, Module)

7.2.3 Method summary

isSatisfied(Workflow, Semester)

7.2.4 Constructors

- **ConstraintIntersection**

```
public ConstraintIntersection(Module targetModule, Module
    otherModule)
```

- **Description** Instantiates a new Constraint between two Modules.
- **Parameters**
  - `targetModule` – The target Module
  - `sourceModule` – The source Module

7.2.5 Methods

- **isSatisfied**

```
public abstract boolean isSatisfied(Workflow workflow, Semester
    semester)
```

  - **Description copied from Constraint** (**in 7.1, page 42**) Returns whether the Workflow satisfies this constraint.
  - **Parameters**
    - `workflow` – The Workflow to verify with the current constraint.
    - `semester` – The Semester, the current Module is in.
  - **Returns** – true if the Constraint is satisfied

7.2.6 Members inherited from class Constraint

`studyplanning.model.workflow.constraint.Constraint` (in 7.1, page 42)

- `public Module` **getSourceModule**`()`
- `public Module` **getTargetModule**`()`
- `public ConstraintType` **getType**`()`
- `public abstract boolean` **isSatisfied**`(studyplanning.model.workflow.Workflow` **workflow**`, studyplanning.model.workflow.Semester` **semester**`)`

7.3 Class ConstraintRequirement

Objects of this class are checking if a required Module has been completed before.

7.3.1 Declaration

```
public class ConstraintRequirement
 extends studyplanning.model.workflow.constraint.Constraint
```

7.3.2 Constructor summary

ConstraintRequirement(Module, Module)

7.3.3 Method summary

isSatisfied(Workflow, Semester)

7.3.4 Constructors

- **ConstraintRequirement**

```
public ConstraintRequirement(Module targetModule, Module
    otherModule)
```

- **Description** Instantiates a new Constraint between two Modules.
- **Parameters**
  - `targetModule` – The target Module
  - `sourceModule` – The source Module

7.3.5 Methods

- **isSatisfied**

```
public abstract boolean isSatisfied(Workflow workflow, Semester
    semester)
```

- **Description copied from Constraint** (**in 7.1, page 42**) Returns whether the workflow satisfies this constraint.
- **Parameters**
  - `workflow` – The Workflow to verify with the current Constraint.
  - `semester` – The semester, the current Module is in.
- **Returns** – true if the constraint is satisfied

7.3.6 Members inherited from class Constraint

`studyplanning.model.workflow.constraint.Constraint` (in 7.1, page 42)

- `public Module` **getSourceModule**`()`
- `public Module` **getTargetModule**`()`
- `public ConstraintType` **getType**`()`
- `public abstract boolean` **isSatisfied**`(studyplanning.model.workflow.Workflow` **workflow**`, studyplanning.model.workflow.Semester` **semester**`)`

7.4 Class ConstraintRequirementUnordered

A Constraint for Modules which need other Modules in the same Workflow, but not in a specified order.

7.4.1 Declaration

```
public class ConstraintRequirementUnordered
 extends studyplanning.model.workflow.constraint.Constraint
```

7.4.2 Constructor summary

ConstraintRequirementUnordered(Module, Module)

7.4.3 Method summary

isSatisfied(Workflow, Semester)

7.4.4 Constructors

- **ConstraintRequirementUnordered**

```
public ConstraintRequirementUnordered(Module targetModule,
    Module otherModule)
```

  - **Description** Instantiates a new Constraint between two Modules.
  - **Parameters**
    - `targetModule` – The target Module
    - `sourceModule` – The source Module

7.4.5 Methods

- **isSatisfied**

```
public abstract boolean isSatisfied(Workflow workflow, Semester
    semester)
```

  - **Description copied from Constraint (in 7.1, page 42)** Returns whether the Workflow satisfies this constraint or not.
  - **Parameters**
    - `workflow` – The Workflow to verify with the current Constraint.
    - `semester` – The semester, the current Module is in.
  - **Returns** – True, if the Constraint is satisfied

7.4.6 Members inherited from class Constraint

`studyplanning.model.workflow.constraint.Constraint` (in 7.1, page 42)

- `public Module` **getSourceModule**`()`
- `public Module` **getTargetModule**`()`
- `public ConstraintType` **getType**`()`
- `public abstract boolean` **isSatisfied**`(studyplanning.model.workflow.Workflow` **workflow**`, studyplanning.model.workflow.Semester` **semester**`)`

7.5 Class ConstraintSameSemester

This represents a constraint between two Modules, which need to be in the same Semester.

7.5.1 Declaration

```
public class ConstraintSameSemester
 extends studyplanning.model.workflow.constraint.Constraint
```

7.5.2 Constructor summary

ConstraintSameSemester(Module, Module)

7.5.3 Method summary

isSatisfied(Workflow, Semester)

7.5.4 Constructors

- **ConstraintSameSemester**

  public ConstraintSameSemester(Module targetModule, Module otherModule)

  - **Description** Instantiates a new Constraint between two Modules.
  - **Parameters**
    - `targetModule` – The target Module
    - `sourceModule` – The source Module

7.5.5 Methods

- **isSatisfied**

  public abstract boolean isSatisfied(Workflow workflow, Semester semester)

  - **Description copied from Constraint** (**in 7.1, page 42**) Returns whether the Workflow satisfies this constraint.
  - **Parameters**
    - `workflow` – The Workflow to verify with the current Constraint.
    - `semester` – The semester, the current Module is in.
  - **Returns** – True if the constraint is satisfied

7.5.6 Members inherited from class Constraint

`studyplanning.model.workflow.constraint.Constraint` (in 7.1, page 42)

- `public Module` **getSourceModule**`()`
- `public Module` **getTargetModule**`()`
- `public ConstraintType` **getType**`()`
- `public abstract boolean` **isSatisfied**`(studyplanning.model.workflow.Workflow` **workflow**`, studyplanning.model.workflow.Semester` **semester**`)`

7.6 Class ConstraintType

Different types of Constraints. Can be accessed via `VALUES` (in 7.6.4, page 49)[id] `create(Module, Module)`

7.6.1 Declaration

```
public final class ConstraintType
```

7.6.2 Field summary

INTERSECTING
REQUIRED_ANY_ORDER
REQUIRED_BEFORE
REQUIRED_SAME_SEMESTER
VALUES Array containing all ConstraintTypes.

7.6.3 Method summary

create(Module, Module) Creates a new Constraint with the given target and source Module.
valueOf(String)
values()

7.6.4 Fields

- `public static final ConstraintType` **REQUIRED_BEFORE**
- `public static final ConstraintType` **REQUIRED_SAME_SEMESTER**
- `public static final ConstraintType` **REQUIRED_ANY_ORDER**
- `public static final ConstraintType` **INTERSECTING**
- `public static final ConstraintType[]` **VALUES**
  - Array containing all ConstraintTypes. Same as `values()` , but with greater performance.

7.6.5 Methods

- **create**

```
public Constraint create(Module targetModule, Module
    requiredModule)
```

  - **Description** Creates a new Constraint with the given target and source Module.
  - **Parameters**
    - `targetModule` – The target Module for this constraint. This corresponds to `getTargetModule()`
    - `sourceModule` – The source from for this constraint. This corresponds to `getSourceModule()`
  - **Returns** – A new Vonstraint of this type and the two Modules.

- **valueOf**

```
public static ConstraintType valueOf(String name)
```

- **values**

```
public static ConstraintType[] values()
```

7.6.6 Members inherited from class Enum

`java.lang.Enum`

- `protected final Object` **clone**`() throws CloneNotSupportedException`
- `public final int` **compareTo**`(Enum` **arg0**`)`
- `public final boolean` **equals**`(Object` **arg0**`)`
- `protected final void` **finalize**`()`
- `public final Class` **getDeclaringClass**`()`
- `public final int` **hashCode**`()`
- `public final String` **name**`()`
- `public final int` **ordinal**`()`
- `public String` **toString**`()`
- `public static Enum` **valueOf**`(Class` **arg0**`, String` **arg1**`)`

Chapter 8

Package studyplanning.model.workflow

Package Contents *Page*

Classes

8.1 Class DataSet

This represents the set of data we get from the database.

8.1.1 Declaration

```
public class DataSet
```

8.1.2 Constructor summary

DataSet() Creates a new DataSet and adds all Modules to the specific category.

8.1.3 Method summary

getCompulsaryModules() Returns all Modules a student of this study course needs to finish before graduating.
getModules() Returns all Modules contained in this DataSet.
getModulesInCategory(String) Returns all the Modules to a belonging category.

8.1.4 Constructors

- **DataSet**

```
public DataSet()
```

  - **Description** Creates a new DataSet and adds all Modules to the specific category.

8.1.5 Methods

- **getCompulsaryModules**

```
public Collection<Module> getCompulsaryModules()
```

  - **Description** Returns all Modules a student of this StudySubject needs to finish before graduating.
  - **Returns** – Returns all compulsory Modules.

- **getModules**

```
public Collection<Module> getModules()
```

  - **Description** Returns all Modules contained in this DataSet.
  - **Returns** – Returns all Modules in this instance

- **getModulesInCategory**

```
public Collection<Module> getModulesInCategory(String category)
```

  - **Description** Returns all the Modules that belong to the given category.
  - **Parameters**
    - `category` – The key-word for a categeroy
  - **Returns** – All Modules to the given key-word

8.2 Class Module

All Modules are objects of this class. These objects will be generated at the start of the system.

8.2.1 Declaration

```
public class Module
```

8.2.2 Constructor summary

Module(String, SemesterType, int, boolean) Creates a new Module with the given arguments.

8.2.3 Method summary

getConstraints() Returns a Collection of Constraints, all of which need to be met, when a workflow is considered as valid.
getEctsPoints() Returns the amount of ECTS points this Module rewards.
getSemester() Returns SemesterType.SUMMER, if this Module happens in the summer semester and SemesterType.WINTER, if this from happens in the winter semester.
getUnlocalizedName() Returns the unlocalized name of this Module.
isCompulsory() Whether this Module is compulsory or not.

8.2.4 Constructors

- **Module**

```
public Module(String name, SemesterType semester, int ectsPoints
    , boolean compulsory)
```

  - **Description** Creates a new Module with the given arguments.
  - **Parameters**
    - `name` – The name of the Module.
    - `semester` – The semester this event is offered.
    - `ectsPoints` – The amount of ECTS points granted, by completing this Module.
    - `compulsory` – Is the Module compulsory or not.

8.2.5 Methods

- **getConstraints**

```
public Collection<Constraint> getConstraints()
```

  - **Description** Returns a Collection of Constraints, all of which need to be met, when a Workflow is considered as valid.

- **Returns** – A Collection of Constraints.

- **getEctsPoints**

```
public int getEctsPoints()
```

  - **Description** Returns the amount of ECTS this Module rewards.
  - **Returns** – the amount of ECTS points of this Module.

- **getSemester**

```
public SemesterType getSemester()
```

  - **Description** Returns SemesterType.SUMMER, if this Module happens in the summer and SemesterType.WINTER, if this Module happens in the winter semester.
  - **Returns** – The SemesterType this Module happens in.

- **getUnlocalizedName**

```
public String getUnlocalizedName()
```

  - **Description** Returns the unlocalized name of this Module.
  - **Returns** – The unlocalized name of this Module.

- **isCompulsary**

```
public boolean isCompulsary()
```

  - **Description** Whether this Module is compulsory or not.
  - **Returns** – True, if the Module is compulsory, false otherwise.

8.3 Class ModuleWrapper

A class for wrapping around a Module to store workflow-specific info, like constraint violations.

8.3.1 Declaration

```
public final class ModuleWrapper
```

8.3.2 Constructor summary

ModuleWrapper(Module)

8.3.3 Method summary

getModule() Returns the currently wrapped Module.
hasMistakes() Returns whether the current Module in the current Workflow does not violate any of its Constraints.

8.3.4 Constructors

- **ModuleWrapper**

```
public ModuleWrapper(Module module)
```

  - **Description** Creates a new instance with the given Module. Instances are getting edited by the verification algorithm, making them visible for the View.
  - **Parameters**
    - `module` – The Module associated to this instance

8.3.5 Methods

- **getModule**

```
public Module getModule()
```

  - **Description** Returns the currently wrapped Module.
  - **Returns** – The current Module.

- **hasMistakes**

```
public boolean hasMistakes()
```

  - **Description** Returns whether the current Module in the current Workflow does not violate any of its Constraints. Used for drawing in the GUI.
  - **Returns** – Whether this Module is OK in the current Workflow.

8.4 Class Semester

This class is representing a semester of a Workflow.

8.4.1 Declaration

```
public class Semester
```

8.4.2 Constructor summary

Semester(int) Creates a new empty Semester

8.4.3 Method summary

addModule(Module) Adds the given Module to the Semester.
getEctsPoints() The sum of the ECTS points rewarded in the Semester.
getModules() Returns an immutable set of Modules in this Semester.
getSemesterType() Whether this Semester represents a Summer or Winter Semester.
removeModule(Module) Removes the Module of the Semester

8.4.4 Constructors

- **Semester**

```
public Semester(int id)
```

  - **Description** Creates a new empty Semester
  - **Parameters**
    - `id` – The semester number.

8.4.5 Methods

- **addModule**

```
public boolean addModule(Module module)
```

  - **Description** Adds the given Module to the Semester.
  - **Parameters**
    - `module` – The Module to be added to the Semester
  - **Returns** – True, if the from was successfully added to the Semester, false otherwise.

- **getEctsPoints**

```
public int getEctsPoints()
```

  - **Description** The sum of the ECTS points rewarded in the Semester.
  - **Returns** – The amount of ECTS points in this Semester.

- **getModules**

```
public java.util.Collection getModules()
```

  - **Description** Returns an immutable set of Modules in this Semester.
  - **Returns** – All Modules in this Semester.

- **getSemesterType**

  ```
  public SemesterType getSemesterType()
  ```

  - **Description** Whether this semester represents a Summer or Winter semester. `SemesterType` (in 8.5, page 57).
  - **Returns** – The type of this Semester.

- **removeModule**

  ```
  public boolean removeModule(Module module)
  ```

  - **Description** Removes the Module of the Semester
  - **Parameters**
    - `module` – The Module to remove from the Semester
  - **Returns** – True, if the Module was successfully removed from the Semester, false otherwise.

8.5 Class SemesterType

Enumeration for the semester type of modules. (Summer or Winter Semester)

8.5.1 Declaration

```
public final class SemesterType
```

8.5.2 Field summary

SUMMER
WINTER

8.5.3 Method summary

valueOf(String)
values()

8.5.4 Fields

- `public static final SemesterType` **SUMMER**
- `public static final SemesterType` **WINTER**

8.5.5 Methods

- **valueOf**

```
public static SemesterType valueOf(String name)
```

- **values**

```
public static SemesterType[] values()
```

8.5.6 Members inherited from class Enum

`java.lang.Enum`

- `protected final Object` **clone**`() throws CloneNotSupportedException`
- `public final int` **compareTo**`(Enum` **arg0**`)`
- `public final boolean` **equals**`(Object` **arg0**`)`
- `protected final void` **finalize**`()`
- `public final Class` **getDeclaringClass**`()`
- `public final int` **hashCode**`()`
- `public final String` **name**`()`
- `public final int` **ordinal**`()`
- `public String` **toString**`()`
- `public static Enum` **valueOf**`(Class` **arg0**`,String` **arg1**`)`

8.6 Class StudySubject

The subject of study. By default, the only existing subject is computer science.

8.6.1 Declaration

```
public class StudySubject
```

8.6.2 Constructor summary

StudySubject(int, int, int, String, DataSet) Creates a new StudySubject object with all needed informations.

8.6.3 Method summary

getDataSet()
getMaxECTS()
getMaxStudyDuration()
getName()
getRequiredECTS()

8.6.4 Constructors

- **StudySubject**

```
public StudySubject(int maxStudyDuration, int requiredECTS, int
    maxECTS, String name, DataSet dataSet)
```

  - **Description** Creates a new StudySubject object with all needed informations.
  - **Parameters**
    - `maxStudyDuration` – The max amount of semesters allowed to study
    - `requiredECTS` – The minimum amount of ECTS points which has to be reached
    - `maxECTS` – The maximum allowed amount of ECTS points
    - `name` – The name of this subject
    - `dataSet` – The belonging `DataSet` (in 8.1, page 51)

8.6.5 Methods

- **getDataSet**

```
public DataSet getDataSet()
```

  - **Returns** – The belonging `DataSet` (in 8.1, page 51)

- **getMaxECTS**

```
public int getMaxECTS()
```

  - **Returns** – The maximum allowed amount of ECTS points

- **getMaxStudyDuration**

```
public int getMaxStudyDuration()
```

  - **Returns** – The maximum amount of semesters allowed to be studied

- **getName**

```
public String getName()
```

  - **Returns** – The name of this subject

- **getRequiredECTS**

```
public int getRequiredECTS()
```

  - **Returns** – The minimum amount of ECTS point to be reached

8.7 Class Workflow

Representation of the study plan.

8.7.1 Declaration

```
public final class Workflow
 implements java.lang.Iterable, java.lang.Cloneable
```

8.7.2 Constructor summary

Workflow(int) Creates a new Workflow

8.7.3 Method summary

clone()
getEctsPoints() Returns the sum of all ECTS points of all Module in this workflow.
getSemester(int) Returns the semester object for the given semester index.
getSemester(Module) Returns the semester, the Module is in, in this specific Workflow or null, if a Module is not contained in this Workflow.
iterator() Allows to iterate over all semesters in this Workflow.

8.7.4 Constructors

- **Workflow**

  ```
  public Workflow(int maxSemester)
  ```

  - **Description** Creates a new Workflow
  - **Parameters**
    - `maxSemester` – The maximal amount of semesters allowed by the StudySubject.

8.7.5 Methods

- **clone**

  ```
  protected native Workflow clone() throws java.lang.
      CloneNotSupportedException
  ```

- **getEctsPoints**

  ```
  public int getEctsPoints()
  ```

  - **Description** Returns the sum of all ECTS points of all Modules in this Workflow.
  - **Returns** – The amount of ECTS points in this Workflow.

- **getSemester**

```
public Semester getSemester(int id)
```

  - **Description** Returns the semester object for the given semester index.
  - **Parameters**
    - `id` – The ordinal of the looked Semester
  - **Returns** – The requested Semester

- **getSemester**

```
public Semester getSemester(Module module)
```

  - **Description** Returns the semester the Module is in, in this specific Workflow, or null if a Module is not contained in this Workflow.
  - **Parameters**
    - `module` – The Module to be looked for
  - **Returns** – The Semenster which contains this Module

- **iterator**

```
public Iterator<Semester> iterator()
```

  - **Description** Allows to iterate over all Semesters in this Workflow.

8.8 Class WorkflowTasks

This class is dedicated to verifying and generating workflows.

8.8.1 Declaration

```
public class WorkflowTasks
```

8.8.2 Method summary

generate(Workflow, StudySubject, Preferences) Generates a new Workflow based on a given Workflow, filling in missing Modules following all Constraints.
verify(Workflow, StudySubject) Verifies a Workflow.

8.8.3 Constructors

- **WorkflowTasks**

```
public WorkflowTasks()
```

8.8.4 Methods

- **generate**

```
public static Collection<Mistake> generate(Workflow workflow,
    StudySubject subject, Preferences prefs)
```

  - **Description** Generates a new Workflow based on a given Workflow, filling in missing Modules following all Constraints.
    In case the given Workflow cannot be generated, because it already violates the module manual, all violations will be returned in a Collection.
  - **Parameters**
    - `workflow` – The Workflow to use as a base to generate.
    - `subject` – The StudySubject containing all demanded restrictions.
    - `prefs` – User preferences sent by the Controller.
  - **Returns** – A Collection of module manual violations.

- **verify**

```
public static Collection<Mistake> verify(Workflow workflow,
    StudySubject subject)
```

  - **Description** Verifies a Workflow.
  - **Parameters**
    - `workflow` – The Workflow to verify.
    - `subject` – The StudySubject to get the verification data from.
  - **Returns** – A Collection of module manual violations.

Chapter 9

Package studyplanning.model.workflow.generation

Package Contents *Page*

Classes

9.1 Class ModuleEvaluation

An individual evaluation of all Modules in the Workflow giving the generation algorithm easy access to the values of a Module.

9.1.1 Declaration

```
public class ModuleEvaluation
```

9.1.2 Constructor summary

ModuleEvaluation()

9.1.3 Method summary

addModuleValue(Module, int) Increases the value of a specified Module by the given parameter.
getModuleValue(Module) Returns the value of the given Module.
getValuedModules() Returns an iterator of all Modules evaluated here in their valued order (from highest value to lowest).
setModuleValue(Module, int) Evaluates the given Module to the specified value.

9.1.4 Constructors

- **ModuleEvaluation**

```
public ModuleEvaluation()
```

9.1.5 Methods

- **addModuleValue**

```
public void addModuleValue(Module module, int value)
```

  - **Description** Increases the value of a specified Module by the given parameter.
  - **Parameters**
    - `module` – The Module to evaluate.
    - `value` – The value to add

- **getModuleValue**

```
public int getModuleValue(Module module)
```

  - **Description** Returns the value of the given Module.
  - **Parameters**
    - `module` – The Module to get the value from.
  - **Returns** – The current value of the Module.

- **getValuedModules**

```
public Iterator<Module> getValuedModules()
```

  - **Description** Returns an iterator of the all Modules evaluated here in their valued order (from highest value to lowest).
  - **Returns** – An iterator of all evaluated modules.

- **setModuleValue**

```
public void setModuleValue(Module module, int value)
```

  - **Description** Evaluates the given Module to the specified value.
  - **Parameters**
    - `module` – The Module to evaluate
    - `value` – The value to set.

9.2 Class Preferences

Class representing all user input. Will be used for generating the Workflow.

9.2.1 Declaration

```
public class Preferences
```

9.2.2 Constructor summary

Preferences()

9.2.3 Method summary

currentSemester() Returns the semester the user is starting in.
getModulePreferences() Returns a Collection of `ModulePreference` (in 10.3, page 69), which can be weighted for Workflow generation.
getPreferredECTSPerSemester() Returns the preferred amount of ECTS points per semester.
getPreferredStudyDuration() Returns the preferred study duration.

9.2.4 Constructors

- **Preferences**

  ```
  public Preferences()
  ```

  - **Description** Creates a new empty instance. For a more dynamic dealing with user inputs, this class does not take any parameters in its constructor.

9.2.5 Methods

- **currentSemester**

  ```
  public int currentSemester()
  ```

  - **Description** Returns the semester the user is starting in. The generation won't add anything to Semesters with a lower index than this number.
  - **Returns** – The current semester the user is in.

- **getModulePreferences**

  ```
  public Collection<ModulePreference> getModulePreferences()
  ```

  - **Description** Returns a Collection of `ModulePreference` (in 10.3, page 69), which can be weighted for Workflow generation.

  - **Returns** – A Collection of individual preferences.

- **getPreferredECTSPerSemester**

  ```
  public int getPreferredECTSPerSemester()
  ```

  - **Description** Returns the preferred amount of ECTS per semester.
  - **Returns** – The preferred amount of ECTS per semester.

- **getPreferredStudyDuration**

  ```
  public int getPreferredStudyDuration()
  ```

  - **Description** Returns the preferred study duration.
  - **Returns** – The preferred study duration.

Chapter 10

Package studyplanning.model.workflow.generation.preference

Package Contents *Page*

Classes

10.1 Class ChoiceCategoryPreference

This class is valuing modules of a given category.

10.1.1 Declaration

```
public class ChoiceCategoryPreference
 extends studyplanning.model.workflow.generation.preference.
    ModulePreference
```

10.1.2 Constructor summary

ChoiceCategoryPreference(DataSet, String)

10.1.3 Method summary

evaluate(ModuleEvaluation)

10.1.4 Constructors

- **ChoiceCategoryPreference**

```
public ChoiceCategoryPreference(DataSet dataSet, String key)
```

  - **Description** Instantiates the object with all modules of a StudySubject and a key. The instance will only evaluate Modules matching the given key.
  - **Parameters**
    - `dataSet` – The container holding all modules of a StudySubject
    - `key` – The key to be considered in the evaluation of the DataSet

10.1.5 Methods

- **evaluate**

```
public abstract void evaluate(ModuleEvaluation eval)
```

  - **Description copied from ModulePreference** (**in 10.3, page 69**) Adds all Modules this preference affects to the given ModuleEvaluation.
  - **Parameters**
    - `eval` – The ModuleEvaluation to do the evaluation with.

10.1.6 Members inherited from class ModulePreference

`studyplanning.model.workflow.generation.preference.ModulePreference` (in 10.3, page 69)

- `public abstract void` **evaluate**`(studyplanning.model.workflow.generation.ModuleEvaluation` **eval**`)`

10.2 Class ChoiceModulePreference

This class is valuing a given Module.

10.2.1 Declaration

```
public class ChoiceModulePreference
 extends studyplanning.model.workflow.generation.preference.
    ModulePreference
```

10.2.2 Constructor summary

ChoiceModulePreference(DataSet, Module)

10.2.3 Method summary

evaluate(ModuleEvaluation)

10.2.4 Constructors

- **ChoiceModulePreference**

```
public ChoiceModulePreference(DataSet dataSet, Module module)
```

  - **Description** Instantiates an object with all Modules of a StudySubject and a key. The instance will only evaluate Modules matching the given key.
  - **Parameters**
    - `dataSet` – The container holding all Modules of a StudySubject
    - `module` – The Module to be considered in the evaluation

10.2.5 Methods

- **evaluate**

```
public abstract void evaluate(ModuleEvaluation eval)
```

  - **Description copied from ModulePreference (in 10.3, page 69)** Adds all Modules affected by this preference to the given ModuleEvaluation.
  - **Parameters**
    - `eval` – The ModuleEvaluation to do the evaluation with.

10.2.6 Members inherited from class ModulePreference

`studyplanning.model.workflow.generation.preference.ModulePreference` (in 10.3, page 69)

- `public abstract void` **evaluate**`(studyplanning.model.workflow.generation.ModuleEvaluation` **eval**`)`

10.3 Class ModulePreference

To enable a class to value the DataSet of the modules, it must inherit this class.

10.3.1 Declaration

```
public abstract class ModulePreference
```

10.3.2 All known subclasses

ChoiceModulePreference (in 10.2, page 68), ChoiceCategoryPreference (in 10.1, page 67)

10.3.3 Method summary

evaluate(ModuleEvaluation) Adds all modules affected by this preference to the given ModuleEvaluation.

10.3.4 Methods

- **evaluate**

```
public abstract void evaluate(ModuleEvaluation eval)
```

  - **Description** Adds all modules affected by this preference to the given ModuleEvaluation.
  - **Parameters**
    - `eval` – The ModuleEvaluation to do the evaluation with.

Part 4

View

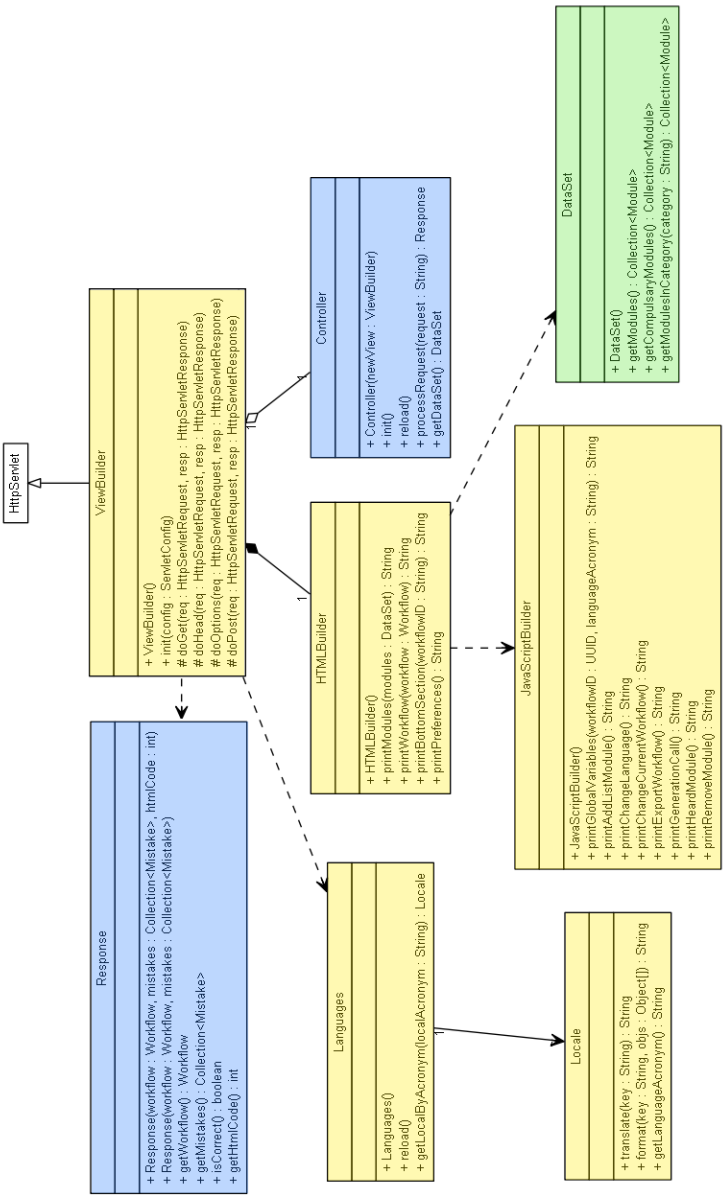


Figure 10.1: Model

Chapter 11

Package studyplanning.view

Package Contents *Page*

Classes

11.1 Class HTMLBuilder

Returns HTML coded objects

11.1.1 Declaration

public class HTMLBuilder

11.1.2 Constructor summary

HTMLBuilder()

11.1.3 Method summary

printBottomSection(String) Returns bottom section as HTML code string
printModules(DataSet) Returns left from section as HTML code string
printPreferences() Returns the default preferences as HTML code string
printWorkflow(Workflow) Returns right Workflow section as HTML code string

11.1.4 Constructors

- **HTMLBuilder**

  ```
  public HTMLBuilder()
  ```

11.1.5 Methods

- **printBottomSection**

  ```
  public String printBottomSection(String workflowID)
  ```

  - **Description** Returns bottom section as HTML code string
  - **Parameters**
    - `workflowID` – Workflow id to display
  - **Returns** – HTML result

- **printModules**

  ```
  public String printModules(DataSet modules)
  ```

  - **Description** Returns left from section as HTML code string
  - **Parameters**
    - `modules` – Modules to display
  - **Returns** – HTML result

- **printPreferences**

  ```
  public String printPreferences()
  ```

  - **Description** Returns the default preferences as HTML code string
  - **Returns** – HTML result

- **printWorkflow**

  ```
  public String printWorkflow(Workflow workflow)
  ```

  - **Description** Returns left from section as HTML code string
  - **Parameters**
    - `workflow` – Workflow to be displayed
  - **Returns** – HTML result

11.2 Class JavaScriptBuilder

Returns JavaScript functionality needed by HTMLBuilder.

11.2.1 Declaration

```
public class JavaScriptBuilder
```

11.2.2 Constructor summary

JavaScriptBuilder()

11.2.3 Method summary

printAddListModule() Returns a JavaScript method that calls 'ViewBuilder' that a Module is added.
printChangeCurrentWorkflow() Returns a JavaScript method that calls 'ViewBuilder' to change the current Workflow.
printChangeLanguage() Returns a JavaScript method that calls 'ViewBuilder' to change current language.
printExportWorkflow() Returns a JavaScript method that calls 'ViewBuilder' to request an export of the current Workflow. Please add 'onclick = "exportWorkflow()"' to your HTML to activate this generated method.
printGenerationCall() Returns a JavaScript method that calls 'ViewBuilder' to process a new Workflow with given preferences. Please add 'onclick = "generateWorkflow('preferences')"' to your HTML to activate this generated method.
printGlobalVariables(UUID, String) Returns JavaScript code containing global variables needed by other generated methods.
printHeardModule() Returns a JavaScript method that calls 'ViewBuilder' to mark a from as heard.
printRemoveModule() Returns a JavaScript method that calls 'ViewBuilder' to remove a from from the current Workflow.

11.2.4 Constructors

- **JavaScriptBuilder**

```
public JavaScriptBuilder()
```

11.2.5 Methods

- **printAddListModule**

```
public static String printAddListModule()
```

- **Description** Returns a JavaScript method that calls 'ViewBuilder' if a Module is added. Please add 'onclick = "addListModule('moduleAcronym')"' to your HTML to activate this generated method. Exp.: onclick = "addListModule("SWT1")"
- **Returns** – JavaScript event method

- **printChangeCurrentWorkflow**

```
public static String printChangeCurrentWorkflow()
```

  - **Description** Returns a JavaScript method that calls 'ViewBuilder' to change the current Workflow. Please add 'onclick = "changeWorkflowTo('workflowID')"' to your HTML to activate this generated method.
  - **Returns** – JavaScript event method

- **printChangeLanguage**

```
public static String printChangeLanguage()
```

  - **Description** Returns a JavaScript method that calls 'ViewBuilder' to change current language. Please add 'onclick = "changeLanguageTo('acronym')"' to your HTML to activate this generated method. Exp.: onclick = "changeLanguageTo("ger_ger")"
  - **Returns** – JavaScript event method

- **printExportWorkflow**

```
public static String printExportWorkflow()
```

  - **Description** Returns a JavaScript method that calls 'ViewBuilder' to request an export of the current Workflow. Please add 'onclick = "exportWorkflow()"' to your HTML to activate this generated method.
  - **Returns** JavaScript event method

- **printGenerationCall**

```
public static String printGenerationCall()
```

  - **Description** Returns a JavaScript method that calls 'ViewBuilder' to process a new Workflow with given preferences. Please add 'onclick = "generateWorkflow('preferences')"' to your HTML to activate this generated method.
  - **Returns** – JavaScript event method

- **printGlobalVariables**

```
public static String printGlobalVariables(UUID workflowID,
    String languageAcronym)
```

- **Description** Returns JavaScript code containing global variables needed by other generated methods. Always run this first before calling other 'JavaScriptBuilder' methods!
- **Parameters**
  - `workflowID` – Users current Workflow
  - `languageAcronym` – Users current locale
- **Returns** – JavaScript declarations

- **printHeardModule**

```
public static String printHeardModule()
```

- **Description** Returns a JavaScript method that calls 'ViewBuilder' to mark a Module as heard. Please add 'onclick = "heardModule('acronym')"' to your HTML to activate this generated method.
- **Returns** – JavaScript event method

- **printRemoveModule**

```
public static String printRemoveModule()
```

- **Description** Returns a JavaScript method that calls 'ViewBuilder' to remove a Module from the current Workflow. Please add 'onclick = "removeModule('acronym')"' to your HTML to activate this generated method.
- **Returns** – JavaScript event method

11.3 Class Languages

Class responsible for loading and retrieving `Locale` (in 11.4, page 78) objects.

11.3.1 Declaration

```
public class Languages
```

11.3.2 Constructor summary

Languages() Creates a new Language instance.

11.3.3 Method summary

getLocalByAcronym(String) Returns a localization set with given language acronym 'localAcronym' Returns null on unknown acronym input.
reload() Reloads all locales from the base locale directory

11.3.4 Constructors

- **Languages**

```
public Languages()
```

  - **Description** Creates a new Language instance.

11.3.5 Methods

- **getLocalByAcronym**

```
public Locale getLocalByAcronym(String localAcronym)
```

  - **Description** Returns a localization set with given language acronym 'localAcronym' Returns null on unknown acronym input.
  - **Parameters**
    - `localAcronym` – Language acronym
  - **Returns** – Resulting localization

- **reload**

```
public void reload()
```

  - **Description** Reloads all locales from the base locale directory

11.4 Class Locale

Class representing one language. It's used for translating strings to one specific language.

11.4.1 Declaration

```
public class Locale
```

11.4.2 Method summary

format(String, Object[]) This translates and formats the localized version of this key.
getLanguageAcronym() Returns the language acronym.
translate(String) This translates a key to its localized part.

11.4.3 Methods

- **format**

```
public String format(String key, Object[] objs)
```

  - **Description** This translates and formats the localized version of this key. `format(String, Object[])`
  - **Parameters**
    - `key` – The key to localize.
    - `objs` – The objects to use for formatting.
  - **Returns** – The formatted String.

- **getLanguageAcronym**

```
public final String getLanguageAcronym()
```

  - **Description** Returns the language acronym. English would be en_us, German would be de_de and so on.
  - **Returns** – The language acronym of the language.

- **translate**

```
public String translate(String key)
```

  - **Description** This translates a key to its localized part.
  - **Parameters**
    - `key` – The language key.
  - **Returns** – The localized value of this key or the key itself, if the mapping does not contain the key.

11.5 Class ViewBuilder

11.5.1 Declaration

```
public class ViewBuilder
  extends HttpServlet
```

11.5.2 Constructor summary

ViewBuilder() Creates a new View

11.5.3 Method summary

doGet(HttpServletRequest, HttpServletResponse)
doHead(HttpServletRequest, HttpServletResponse)
doOptions(HttpServletRequest, HttpServletResponse)
doPost(HttpServletRequest, HttpServletResponse)
init(ServletConfig) Init method called by apache tomcat, will initializes entire program

11.5.4 Constructors

- **ViewBuilder**

```
public ViewBuilder()
```

  - **Description** Creates a new View

11.5.5 Methods

- **doGet**

```
protected void doGet(HttpServletRequest req, HttpServletResponse
    resp)
```

  - **Parameters**
    - `req` – Request from website
    - `resp` – New website to be displayed
  - **See also**
    - HttpServlet.doGet(HttpServletRequest request, HttpServletResponse response)

- **doHead**

```
protected void doHead(HttpServletRequest req,
    HttpServletResponse resp)
```

  - **Parameters**
    - `req` – Request from website
    - `resp` – New website to be displayed
  - **See also**
    - HttpServlet.doHead(HttpServletRequest request, HttpServletResponse response)

- **doOptions**

```
protected void doOptions(HttpServletRequest req,
    HttpServletResponse resp)
```

- **Parameters**
  - `req` – Request from website
  - `resp` – New website to be displayed
- **See also**
  - HttpServlet.doOptions(HttpServletRequest request, HttpServletResponse response)

- **doPost**

```
protected void doPost(HttpServletRequest req,
    HttpServletResponse resp)
```

- **Parameters**
  - `req` – Request from website
  - `resp` – New website to be displayed
- **See also**
  - HttpServlet.doPost(HttpServletRequest request, HttpServletResponse response)

- **init**

```
public void init(ServletConfig config)
  throws ServletException
```

- **Description** Init method called by Apache Tomcat, will initialize the entire program
- **Parameters**
  - `config` – Apache tomcat configuration
- **Throws**
  - `ServletException` – Occurring error

Part 5

Other

Chapter 12

Procedures

12.1 Initialization

This sequence diagram shows how the system is initialized using **Tomcat**, which starts the ViewBuilder. After its initialization, the ViewBuilder continues to instantiate the Language object, which is responsible for getting the language specific strings of the website and the Controller-Class. The Controller Class is creating support objects to process commands and to communicate with the Model. One of this objects is the WorkflowOperation class, which during its initialization creates its own support objects.

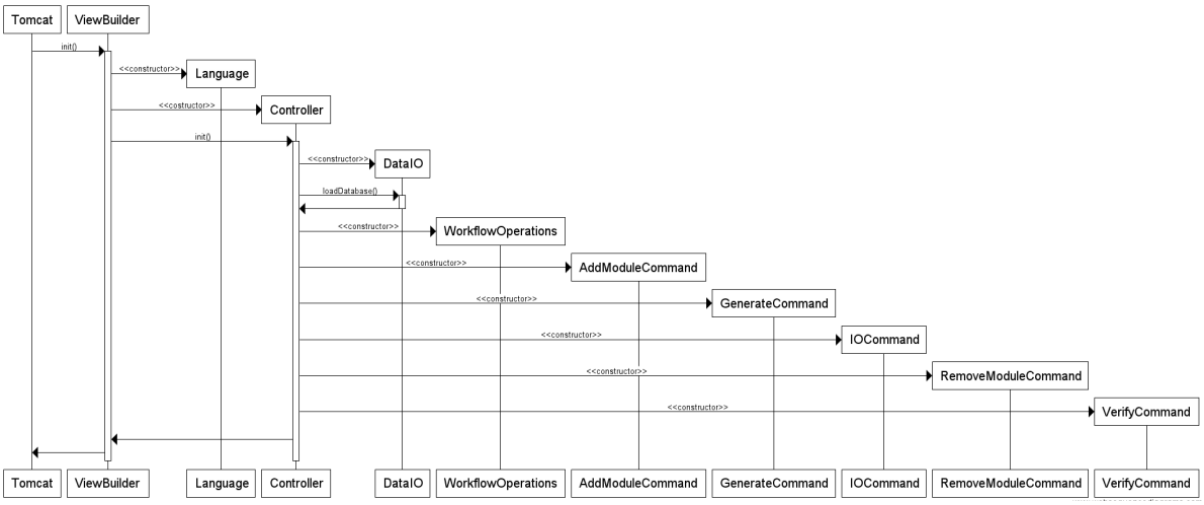


Figure 12.1: System initialization

12.2 Generic Interaction

This sequence diagram shows an generic interaction between the View and the Controller. It is exemplary for other interactions and possible outcomes.

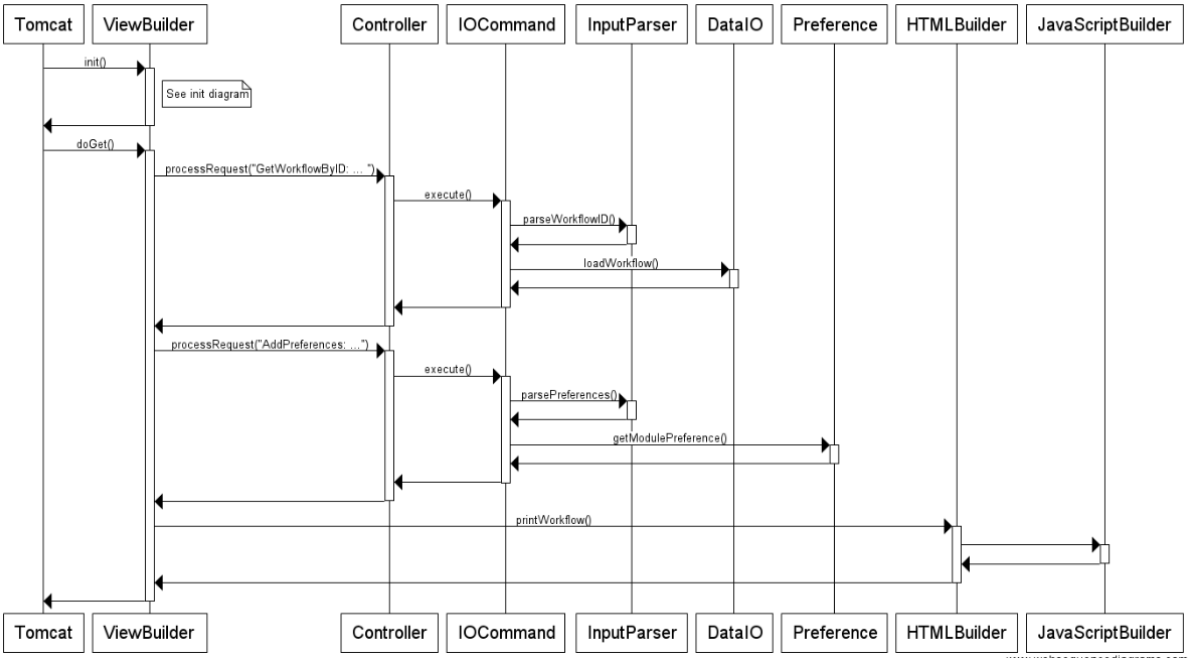


Figure 12.2: Example interaction

12.3 Generation

Generating requires parsing the workflow id, the study subject and the preferences. After loading the workflow, the Model uses the given informations to generate a workflow. The generation algorithm assigns each module a value. This value is calculated in accordance with the preferences and the module's own properties (e.g. compulsory module or desired module). The workflow is then assembled out of high-value modules, while making sure that it complies with the rules.

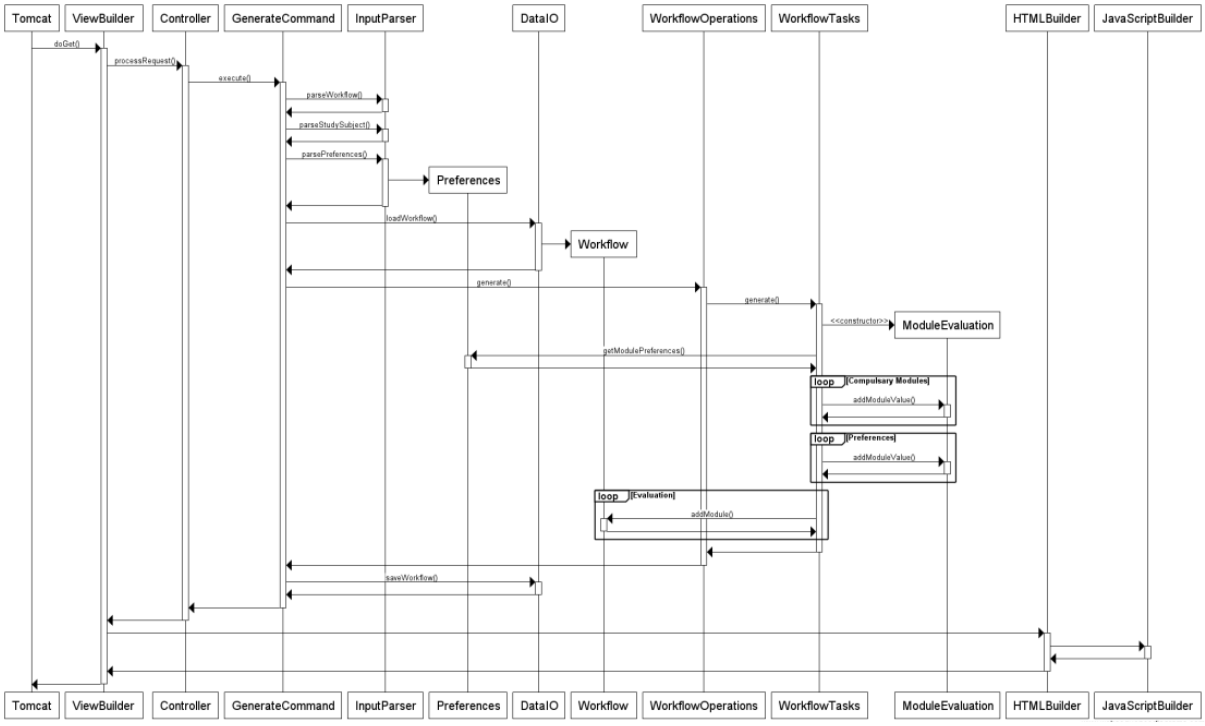


Figure 12.3: Generation

12.4 Verification

The Controller begins the verification process, by parsing the message generated by the View. It then loads the workflow and the workflow's study subject and hands both to the Model. The Model checks if every constraint is satisfied, and creates a Collection of Mistakes, leaving it empty if the workflow is valid. The Collection is handed back to the Controller, which then uses it to generate a Response for the View. In the end the Response is sent to the View, which uses it to build the new website.

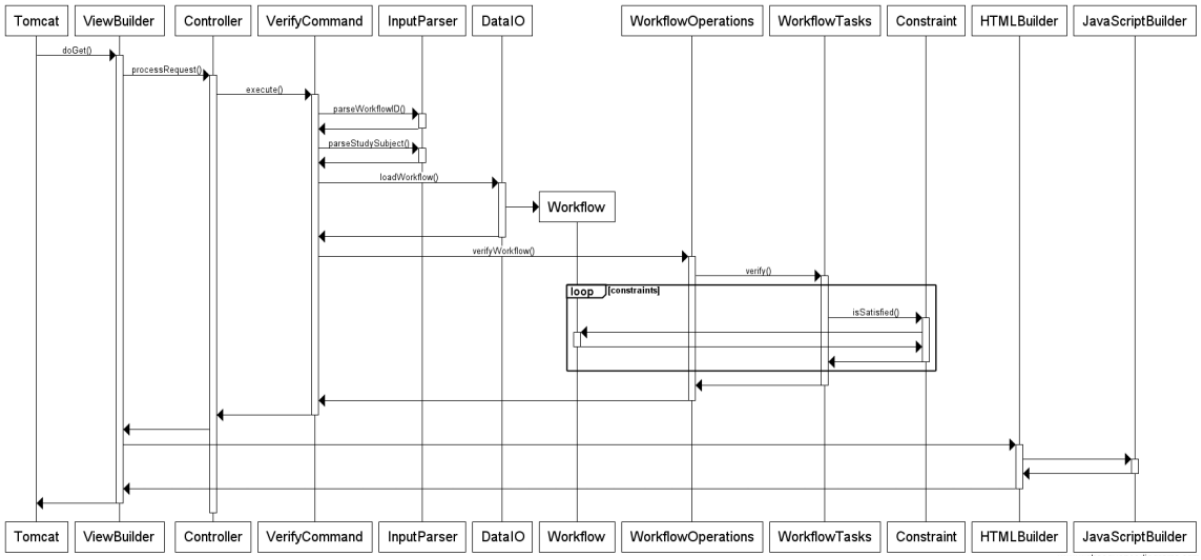


Figure 12.4: Verification

Chapter 13

Database

13.1 Given Database

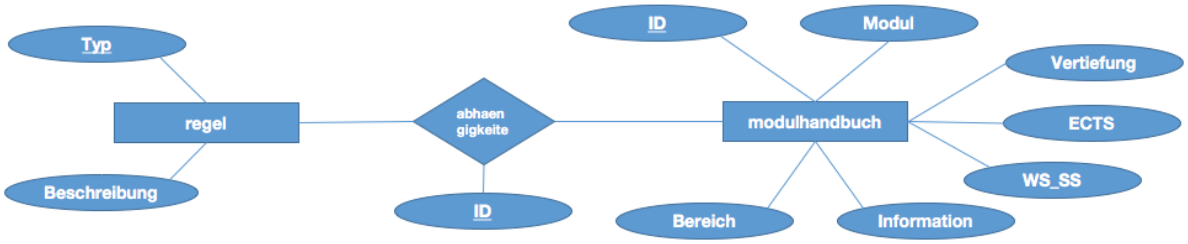


Figure 13.1: Given Database

- **modulhandbuch** - This table saves all modules with their specific data.

Column	Content
ID	Primary key for the modules. Used for referencing
Modul	Name of the module
Vertiefung	The specialization of the module
ECTS	The ECTS this module rewards
WS_SS	The semester this module is happening in
Information	Anything important like compulsory or core
Breich	The section of the module

- **abhaengigkeit** - This table shows the relation between two modules and their constraint.

Column	Content
ID	Primary key for the constraints
Modul1	The first module in this constraint
Modul2	The second module in this constraint
Typ	The ID of the constraint type

- **regel** - This table saves all constraints and their meaning/name.

Column	Content
Typ	Primary key of the constraint. Used for referencing
Beschreibung	The meaning/name of this constraint

The content of this table will be implemented as Enums (Constraint) in the program, making this table obsolete for the implementation.

13.2 Extended Database

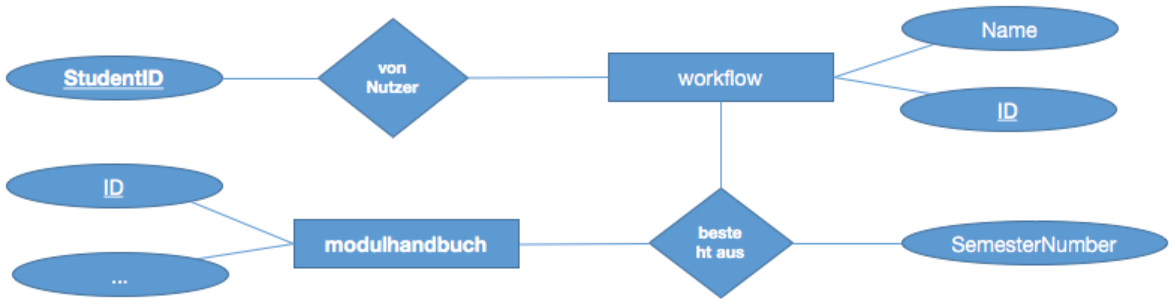


Figure 13.2: Extended Database

To fulfill all criteria, we need to add user data to the database. These are the tables we plan on adding.

- **von Nutzer** - This table is a relation between users and workflows.

Column	Content
StudnetID	The ID of the user and a primary key
WorkflowID	The ID of the workflow is a primary key

No user informations are stored in the database, but there can be multiple user entries for each new workflowID.

- **workflow** - This table stores all workflows and their specific informations.

Column	Content
ID	Primary key and identifier for references
Name	The user generated name of the workflow

- **besteht aus** - This table is a relation between the workflows and the modules in it. It shows in which semester a module takes place.

Column	Content
WorkflowID	The ID of the workflow, is a primary key
ModuleID	The ID of the module, is a primary key
SemesterNumber	The semester this module is part of

13.3 Combined Database

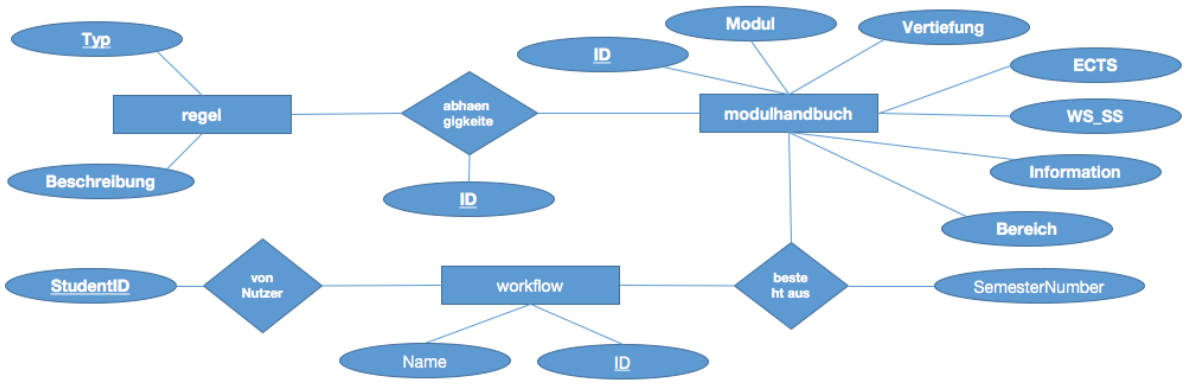
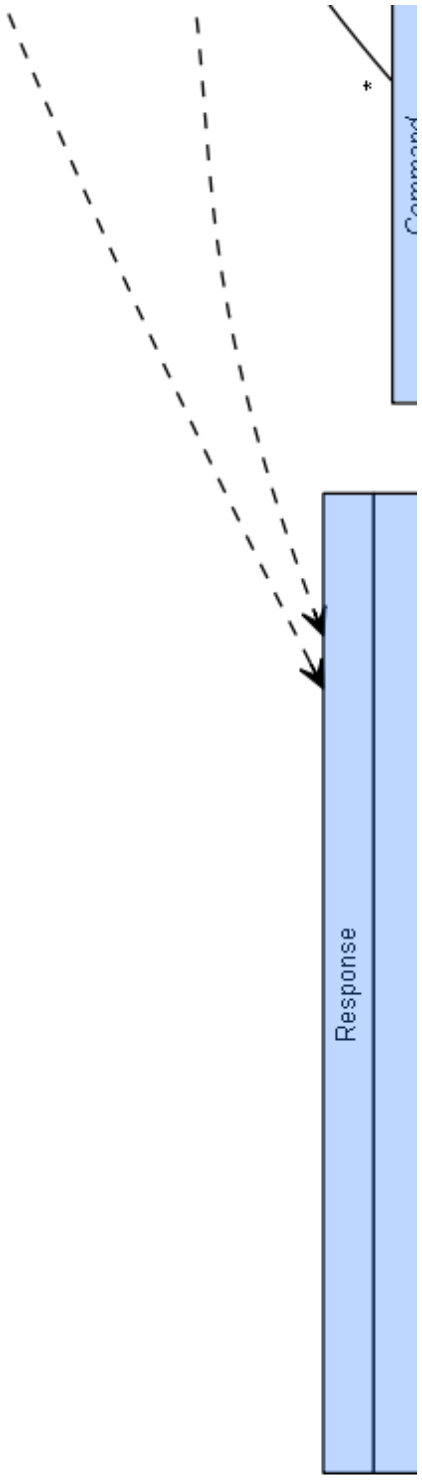
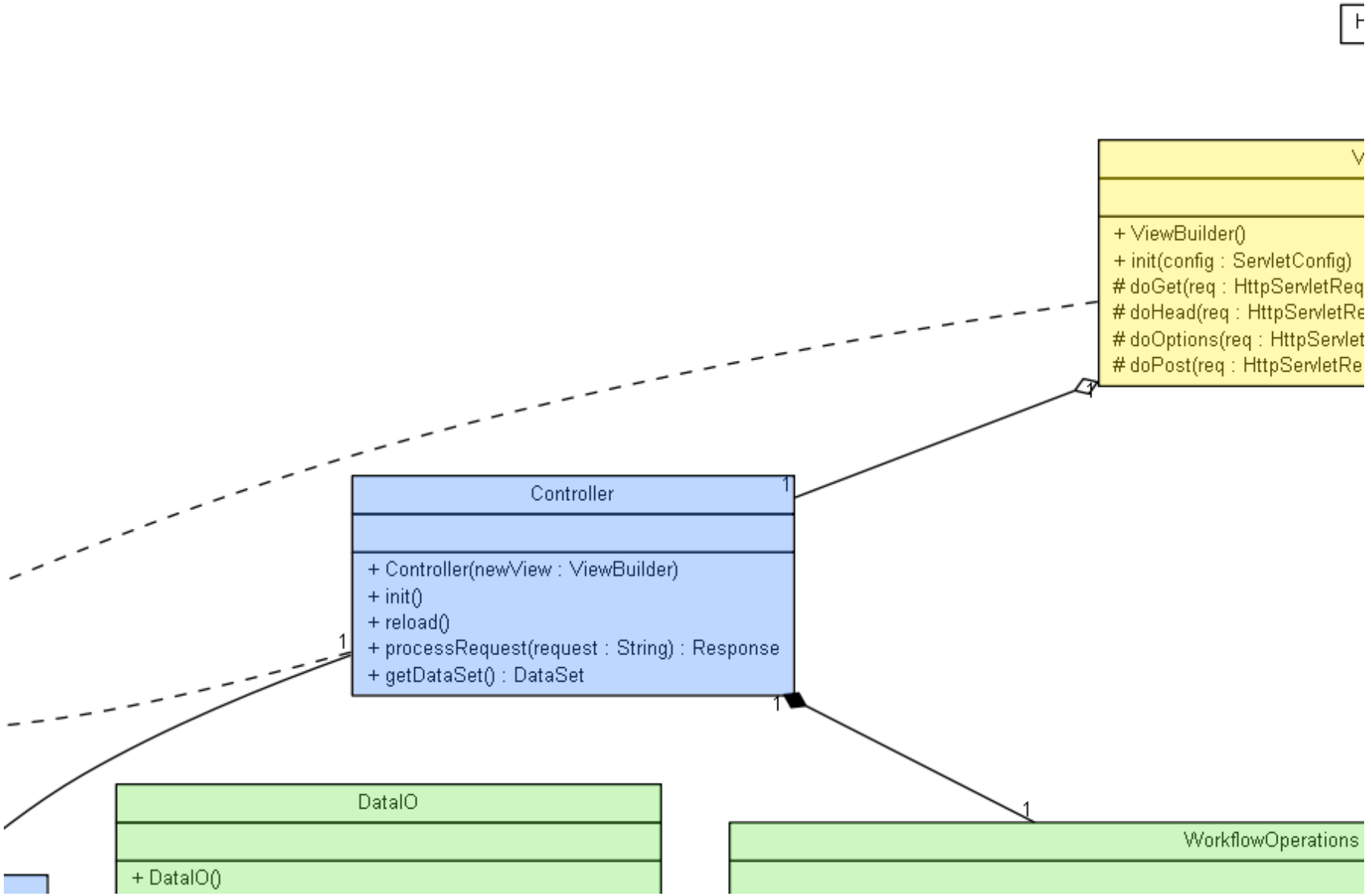
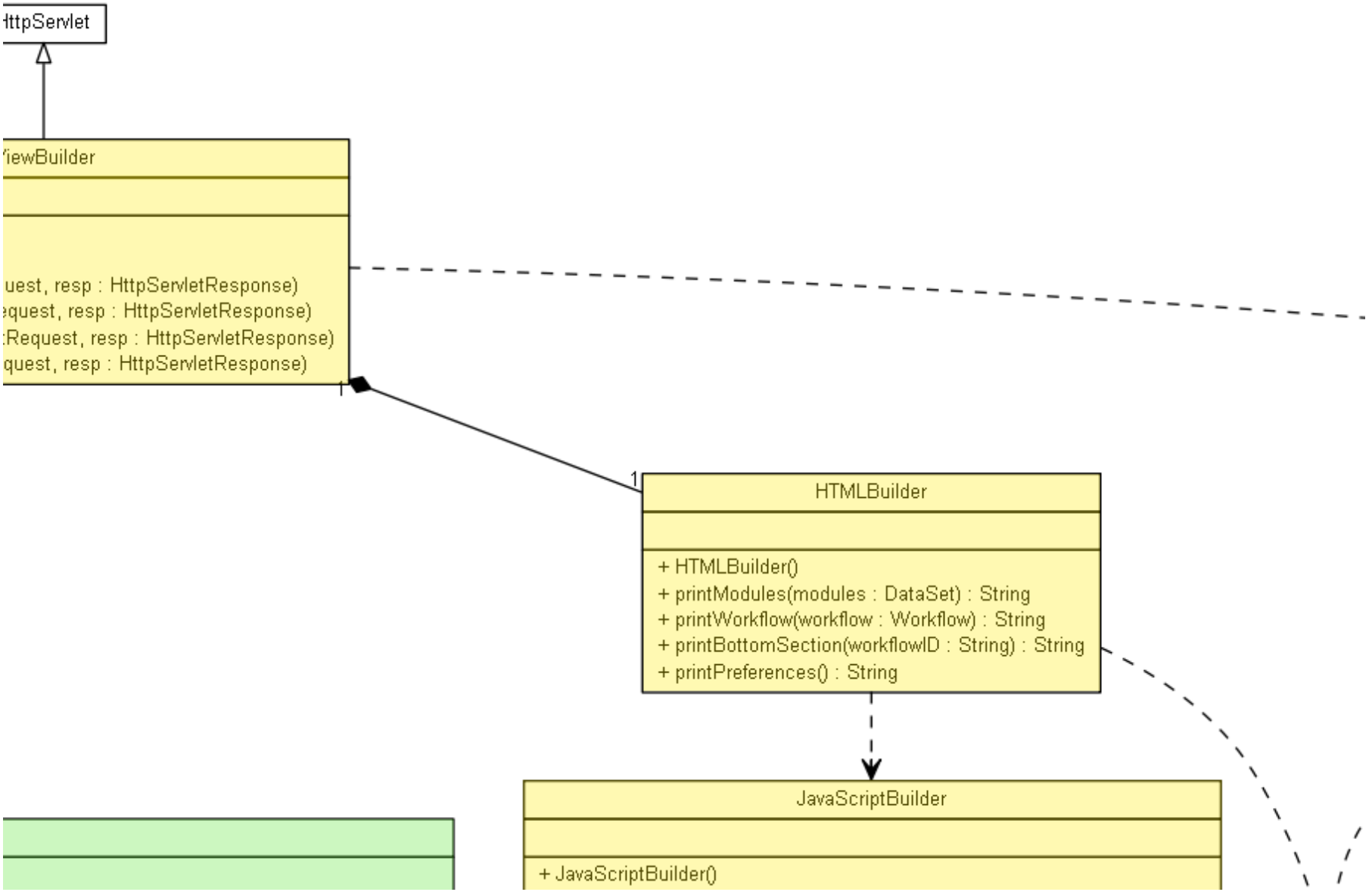


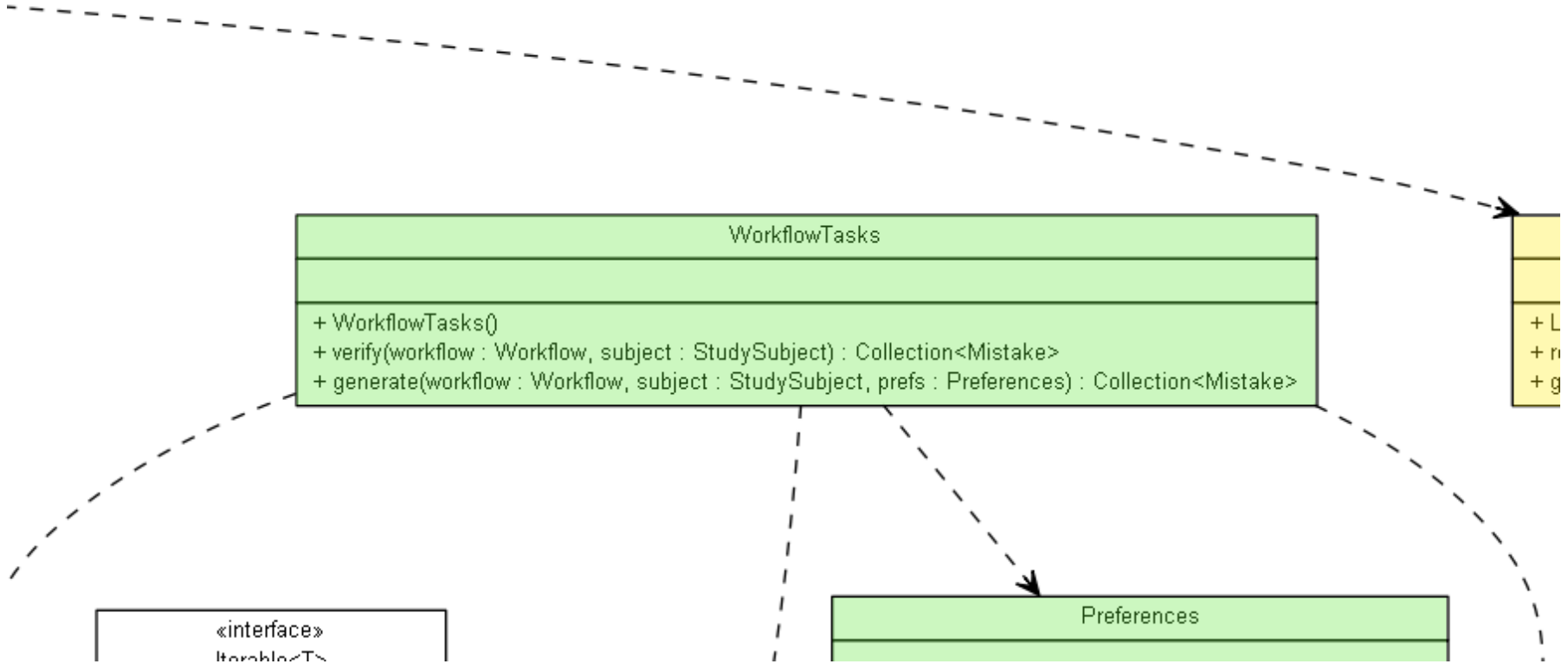
Figure 13.3: Combined Database

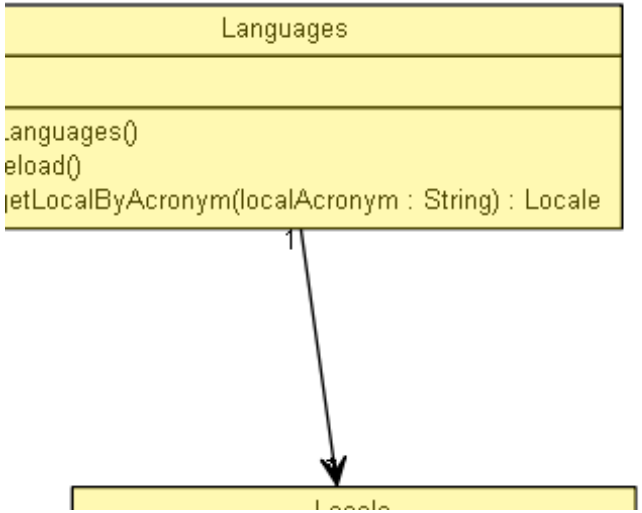
Appendices

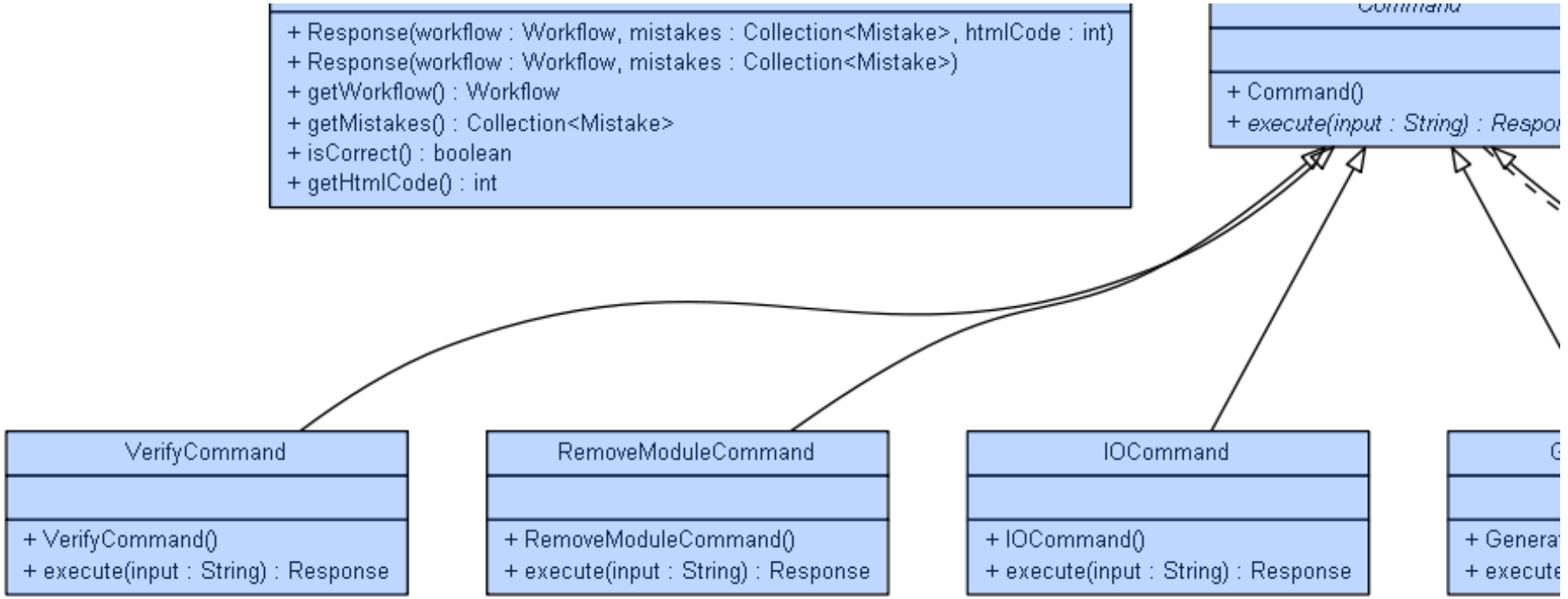

*
Response

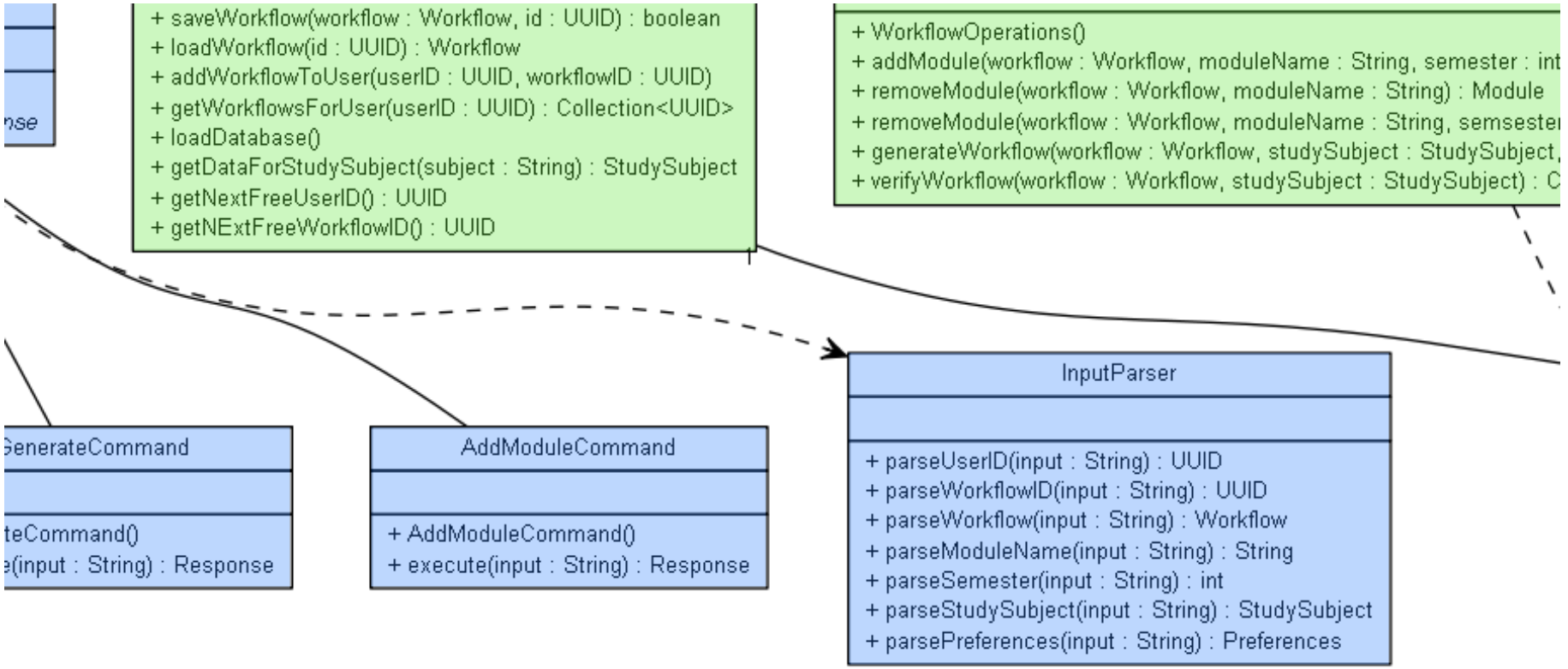

+ ViewBuilder()
+ init(config : ServletConfig)
# doGet(req : HttpServletReq
# doHead(req : HttpServletRe
# doOptions(req : HttpServlet
# doPost(req : HttpServletRe
Controller
+ Controller(newView : ViewBuilder)
+ init()
+ reload()
+ processRequest(request : String) : Response
+ getDataSet() : DataSet
DataIO
+ DataIO()
WorkflowOperations
1
1
1
1
1

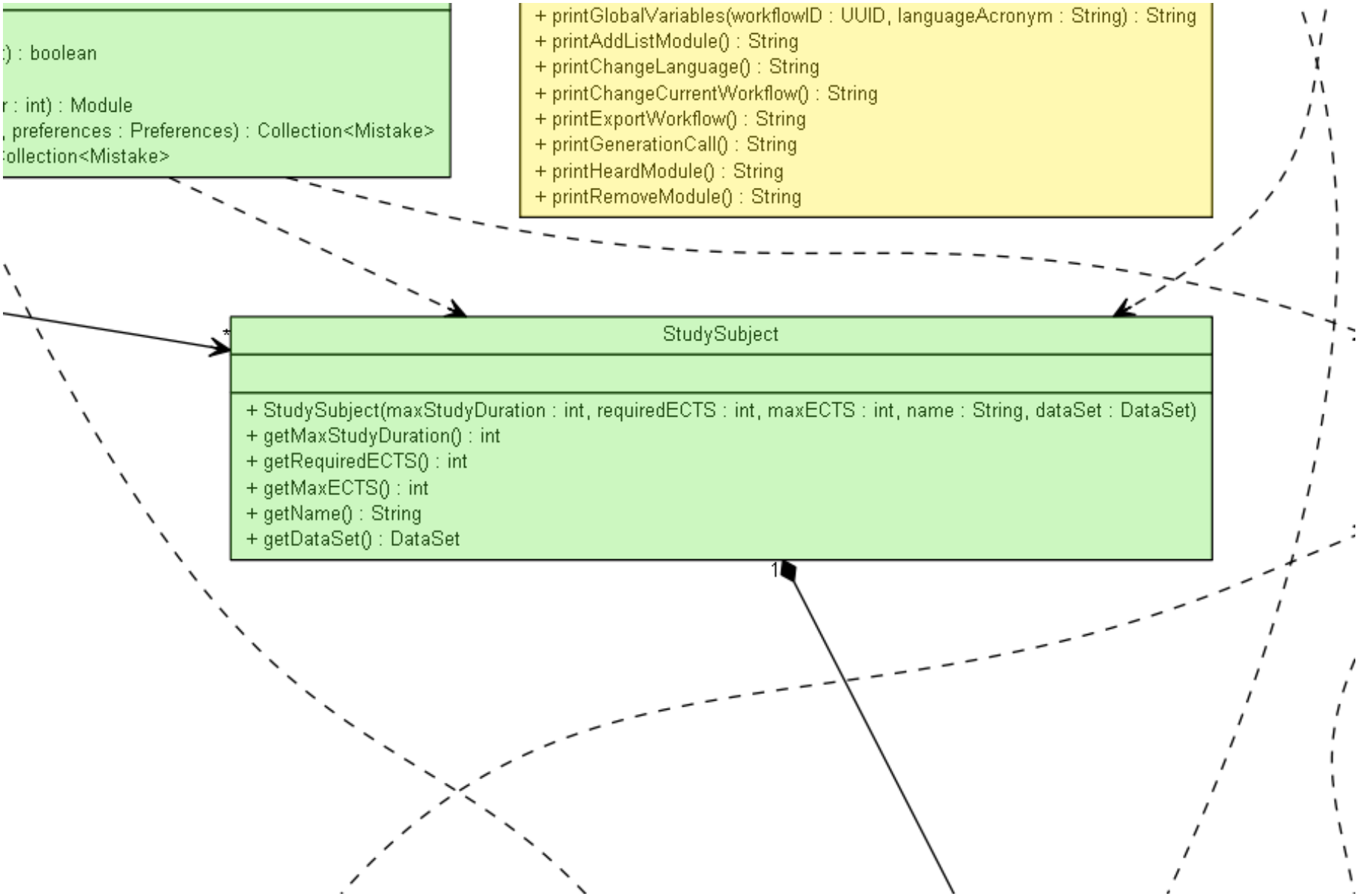

ttpServlet
iewBuilder
uest, resp : HttpServletResponse)
equest, resp : HttpServletResponse)
:Request, resp : HttpServletResponse)
quest, resp : HttpServletResponse)
1
1
HTMLBuilder
+ HTMLBuilder()
+ printModules(modules : DataSet) : String
+ printWorkflow(workflow : Workflow) : String
+ printBottomSection(workflowID : String) : String
+ printPreferences() : String
JavaScriptBuilder
+ JavaScriptBuilder()

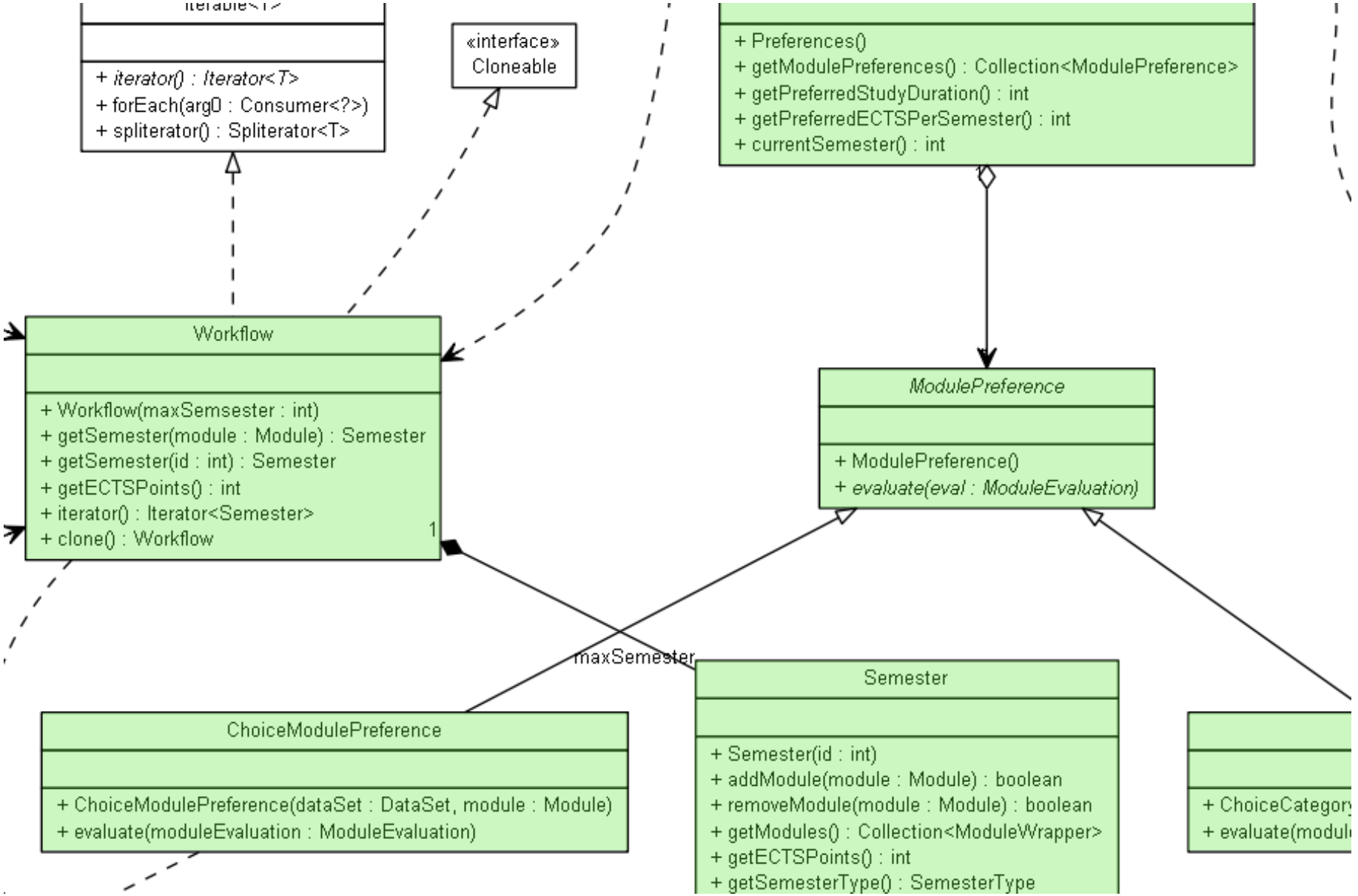

WorkflowTasks
+ WorkflowTasks()
+ verify(workflow : Workflow, subject : StudySubject) : Collection<Mistake>
+ generate(workflow : Workflow, subject : StudySubject, prefs : Preferences) : Collection<Mistake>
«interface»
Preferences


Languages
anguages()
eload()
etLocalByAcronym(localAcronym : String) : Locale
1

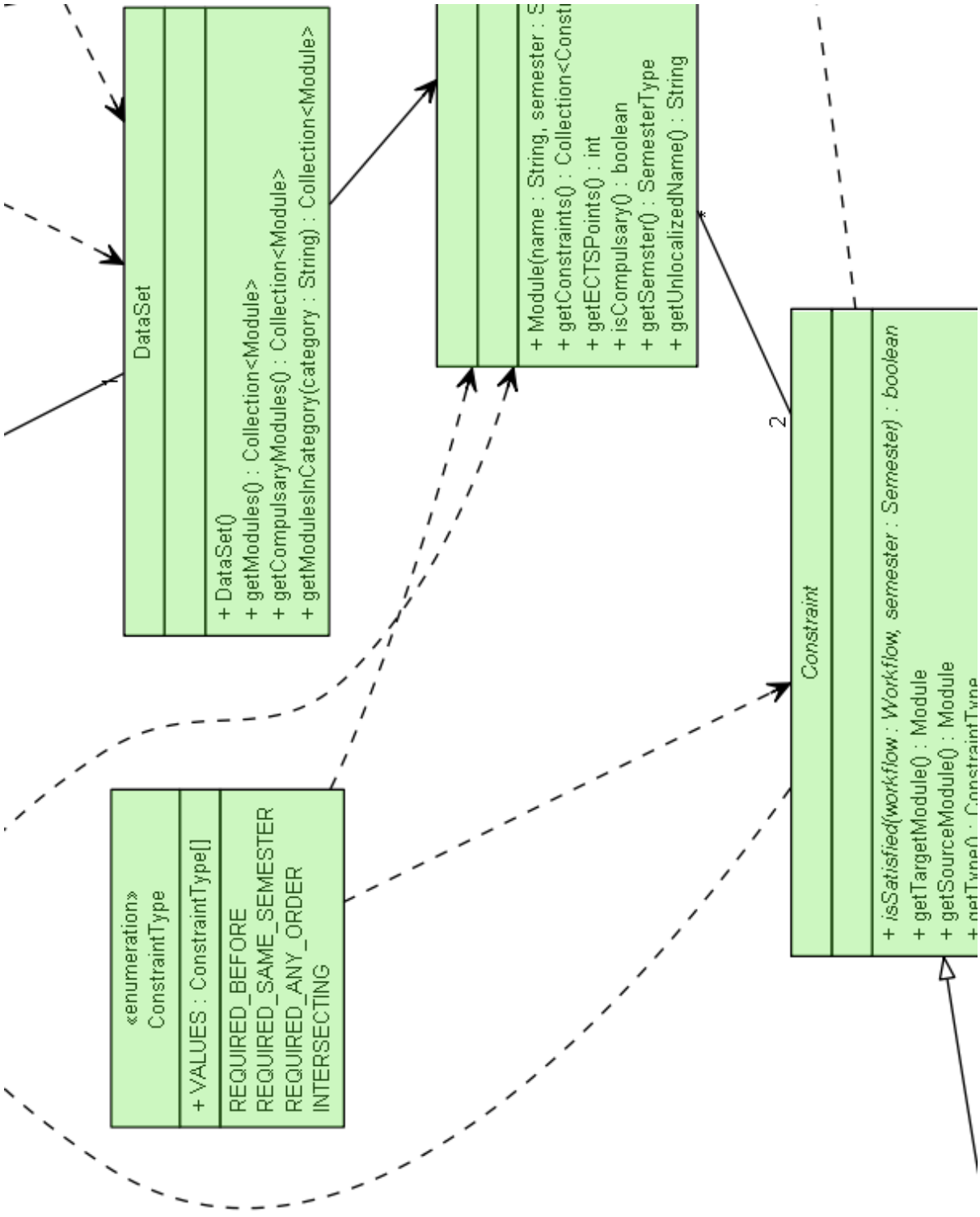

+ Response(workflow : Workflow, mistakes : Collection<Mistake>, htmlCode : int)
+ Response(workflow : Workflow, mistakes : Collection<Mistake>)
+ getWorkflow() : Workflow
+ getMistakes() : Collection<Mistake>
+ isCorrect() : boolean
+ getHtmlCode() : int
+ Command()
VerifyCommand
+ VerifyCommand()
+ execute(input : String) : Response
RemoveModuleCommand
+ RemoveModuleCommand()
+ execute(input : String) : Response
IOCommand
+ IOCommand()
+ execute(input : String) : Response

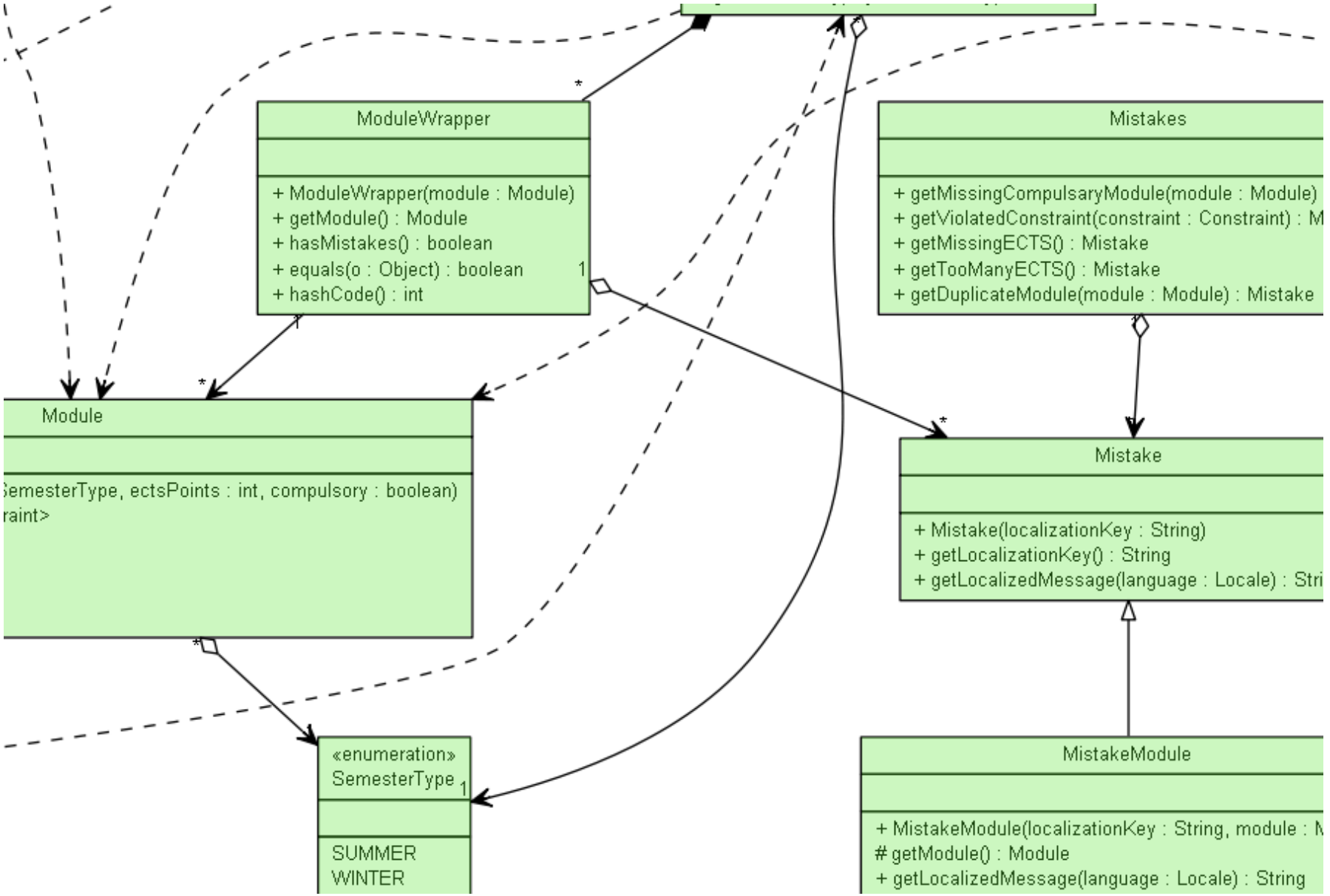

+ saveWorkflow(workflow : Workflow, id : UUID) : boolean
+ loadWorkflow(id : UUID) : Workflow
+ addWorkflowToUser(userID : UUID, workflowID : UUID)
+ getWorkflowsForUser(userID : UUID) : Collection<UUID>
+ loadDatabase()
+ getDataForStudySubject(subject : String) : StudySubject
+ getNextFreeUserID() : UUID
+ getNExtFreeWorkflowID() : UUID
+ WorkflowOperations()
+ addModule(workflow : Workflow, moduleName : String, semester : int
+ removeModule(workflow : Workflow, moduleName : String) : Module
+ removeModule(workflow : Workflow, moduleName : String, semseste
+ generateWorkflow(workflow : Workflow, studySubject : StudySubject,
+ verifyWorkflow(workflow : Workflow, studySubject : StudySubject) : C
InputParser
+ parseUserID(input : String) : UUID
+ parseWorkflowID(input : String) : UUID
+ parseWorkflow(input : String) : Workflow
+ parseModuleName(input : String) : String
+ parseSemester(input : String) : int
+ parseStudySubject(input : String) : StudySubject
+ parsePreferences(input : String) : Preferences
AddModuleCommand
+ AddModuleCommand()
+ execute(input : String) : Response


) : boolean
r : int) : Module
, preferences : Preferences) : Collection<Mistake>
ollection<Mistake>
+ printGlobalVariables(workflowID : UUID, languageAcronym : String) : String
+ printAddListModule() : String
+ printChangeLanguage() : String
+ printChangeCurrentWorkflow() : String
+ printExportWorkflow() : String
+ printGenerationCall() : String
+ printHeardModule() : String
+ printRemoveModule() : String
StudySubject
+ StudySubject(maxStudyDuration : int, requiredECTS : int, maxECTS : int, name : String, dataSet : DataSet)
+ getMaxStudyDuration() : int
+ getRequiredECTS() : int
+ getMaxECTS() : int
+ getName() : String
+ getDataSet() : DataSet
*
1

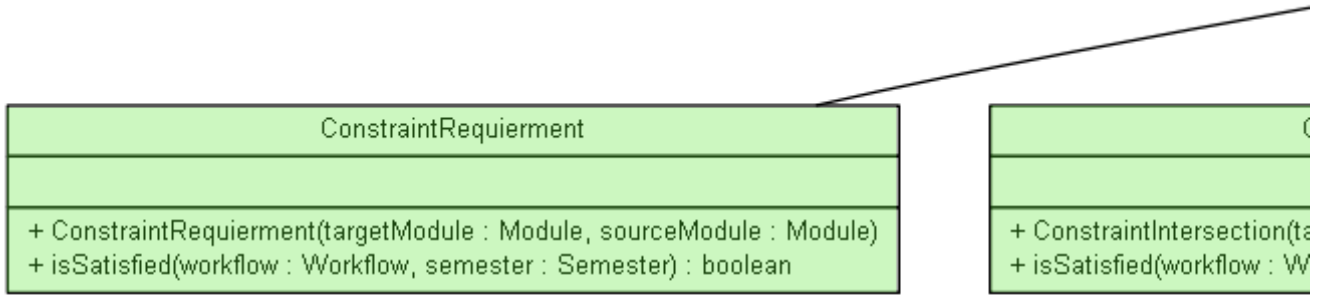

+ iterator() : Iterator<T>
+ forEach(arg0 : Consumer<?>)
+ spliterator() : Spliterator<T>
«interface»
Cloneable
+ Preferences()
+ getModulePreferences() : Collection<ModulePreference>
+ getPreferredStudyDuration() : int
+ getPreferredECTSPerSemester() : int
+ currentSemester() : int
Workflow
+ Workflow(maxSemsester : int)
+ getSemester(module : Module) : Semester
+ getSemester(id : int) : Semester
+ getECTSPoints() : int
+ iterator() : Iterator<Semester>
+ clone() : Workflow
1
ModulePreference
+ ModulePreference()
+ evaluate(eval : ModuleEvaluation)
maxSemester
ChoiceModulePreference
+ ChoiceModulePreference(dataSet : DataSet, module : Module)
+ evaluate(moduleEvaluation : ModuleEvaluation)
Semester
+ Semester(id : int)
+ addModule(module : Module) : boolean
+ removeModule(module : Module) : boolean
+ getModules() : Collection<ModuleWrapper>
+ getECTSPoints() : int
+ getSemesterType() : SemesterType

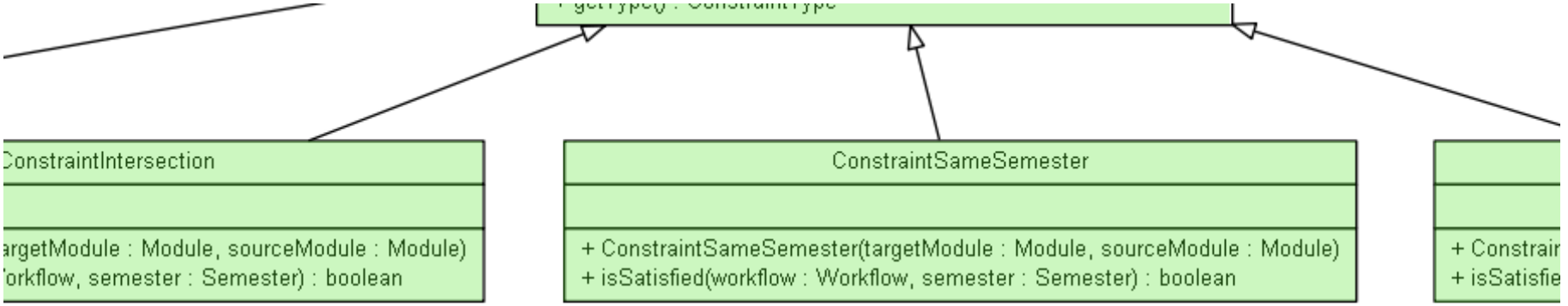
+ translate(key : String) : String
+ format(key : String, objs : Object[]) : String
+ getLanguageAcronym() : String
ModuleEvaluation
+ ModuleEvaluation()
+ setModuleValue(module : Module, value : int)
+ addModuleValue(module : Module, value : int)
+ getModuleValue(module : Module) : int
+ getValuedModules() : Iterator<Module>
ChoiceCategoryPreference
yPreference(dataSet : DataSet, key : String)
eEvaluation : ModuleEvaluation)

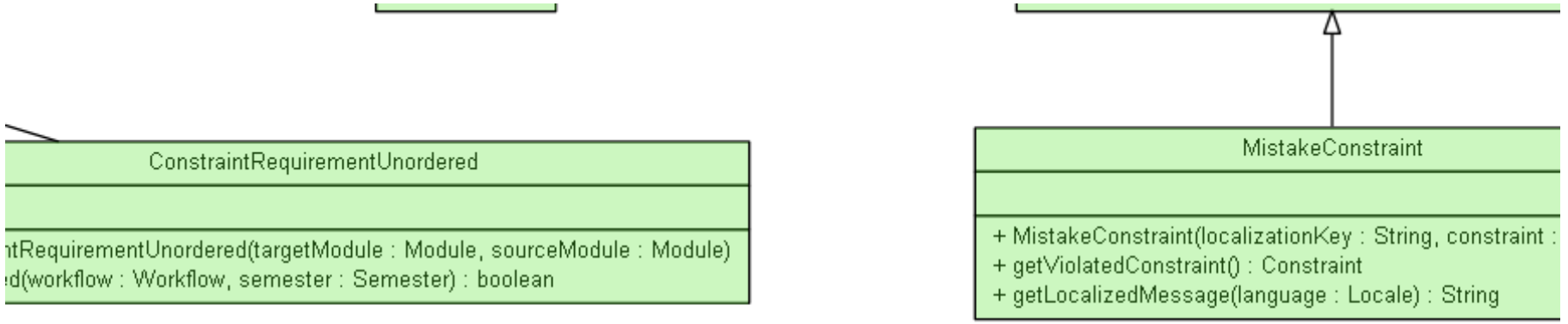

DataSet
+ DataSet()
+ getModules() : Collection<Module>
+ getCompulsaryModules() : Collection<Module>
+ getModulesInCategory(category : String) : Collection<Module>
+ Module(name : String, semester : S
+ getConstraints() : Collection<Constr
+ getECTSPoints() : int
+ isCompulsary() : boolean
+ getSemster() : SemesterType
+ getUnlocalizedName() : String
«enumeration»
ConstraintType
+ VALUES : ConstraintType[]
REQUIRED_BEFORE
REQUIRED_SAME_SEMESTER
REQUIRED_ANY_ORDER
INTERSECTING
Constraint
+ isSatisfied(workflow : Workflow, semester : Semester) : boolean
+ getTargetModule() : Module
+ getSourceModule() : Module
2
*


ModuleWrapper
+ ModuleWrapper(module : Module)
+ getModule() : Module
+ hasMistakes() : boolean
+ equals(o : Object) : boolean
+ hashCode() : int
Mistakes
+ getMissingCompulsaryModule(module : Module)
+ getViolatedConstraint(constraint : Constraint) : M
+ getMissingECTS() : Mistake
+ getTooManyECTS() : Mistake
+ getDuplicateModule(module : Module) : Mistake
Module
emesterType, ectsPoints : int, compulsory : boolean)
raint>
Mistake
+ Mistake(localizationKey : String)
+ getLocalizationKey() : String
+ getLocalizedMessage(language : Locale) : Stri
«enumeration»
SemesterType
SUMMER
WINTER
MistakeModule
+ MistakeModule(localizationKey : String, module : M
# getModule() : Module
+ getLocalizedMessage(language : Locale) : String

: Mistake
listake
ng
Module)


ConstraintRequierment
+ ConstraintRequierment(targetModule : Module, sourceModule : Module)
+ isSatisfied(workflow : Workflow, semester : Semester) : boolean
+ ConstraintIntersection(ta
+ isSatisfied(workflow : W


ConstraintIntersection
argetModule : Module, sourceModule : Module)
Workflow, semester : Semester) : boolean
ConstraintSameSemester
+ ConstraintSameSemester(targetModule : Module, sourceModule : Module)
+ isSatisfied(workflow : Workflow, semester : Semester) : boolean
+ Constrain
+ isSatisfie


ConstraintRequirementUnordered
ntRequirementUnordered(targetModule : Module, sourceModule : Module)
ed(workflow : Workflow, semester : Semester) : boolean
MistakeConstraint
+ MistakeConstraint(localizationKey : String, constraint :
+ getViolatedConstraint() : Constraint
+ getLocalizedMessage(language : Locale) : String


Constraint)