

Paper	Class	Author
Currency manager architecture	Software engineering	Nikola Stjelja, nstjelja@unipu.hr

Currency manager HLA

Software Engineering

School of Informatics
Faculty of Economics
Pula

Last change date	Version	Status
	X.00.00.00	Draft

Project overview

The Currency Management application is a simple Java desktop client for setting and calculating exchanges for a fixed set of Currencies.

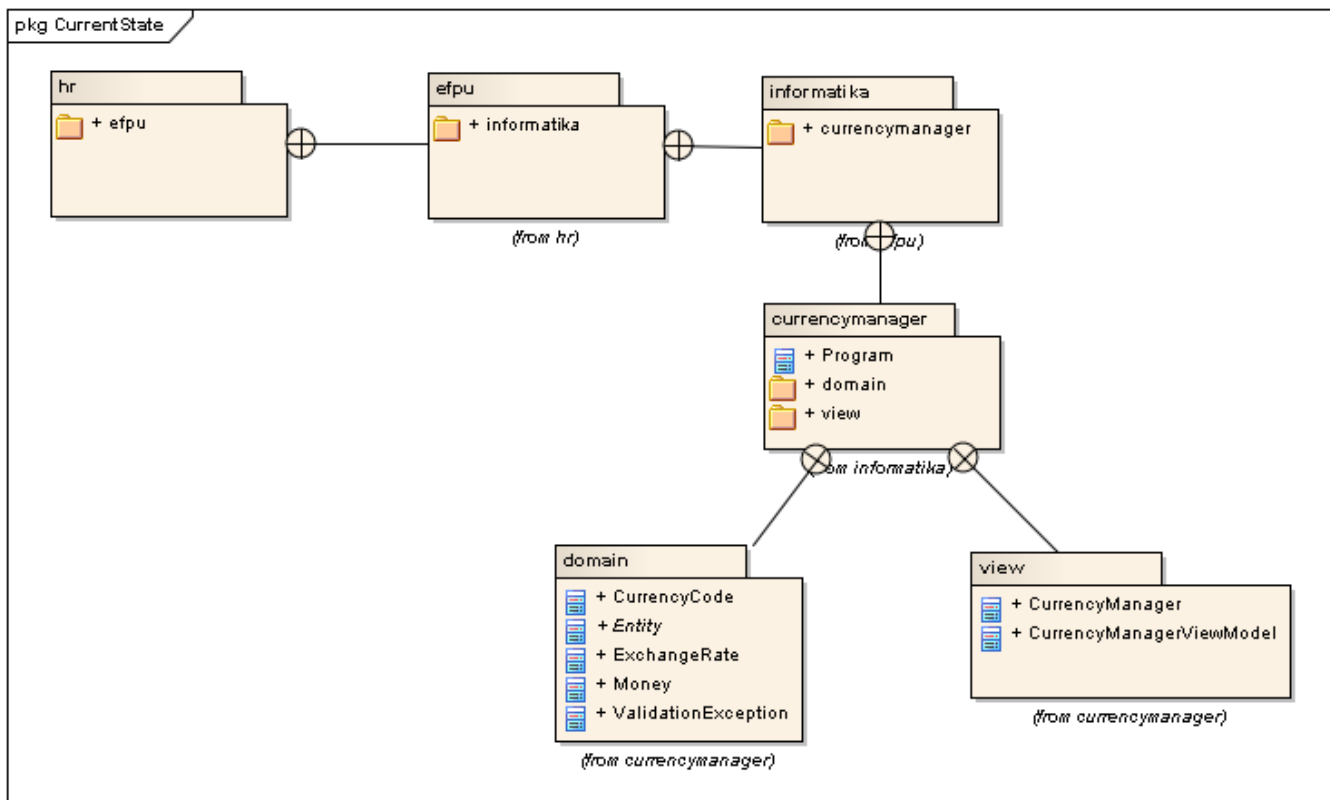
The application is developed with the following technologies:

1. OpenJDK 1.7 on Linux
2. SWING

The source code of the application and the entire project is located here :

[git://github.com/nstjelja/efpu-informatika.git](https://github.com/nstjelja/efpu-informatika.git)

High Level Architecture



The Currency Management application is built around a 2-layer model (since it doesn't require persistence of its entities the usual third layer is missing).

The application layers are:

1. `hr.efpu.informatika.currencymanager.view`
2. `hr.efpu.informatika.currencymanager.domain`

The Class program is a thin element used to run the main form of the application located in the **view** package.

The **`hr.efpu.informatika.currencymanager.view`** where all the SWING, e.g. UI components are located and is tasked only with the presentation and orchestration of the business logic executed and contained within the domain layer. The MVVM pattern is used to provide a testable environment for testing UI related logic without the use of complicated Functional testing tools.

The **`hr.efpu.informatika.currencymanager.domain`** is the main business logic layer where the entire application logic is organized and executed. Here are defined all the application entities, their interactions and all the business actions that can be executed on them.

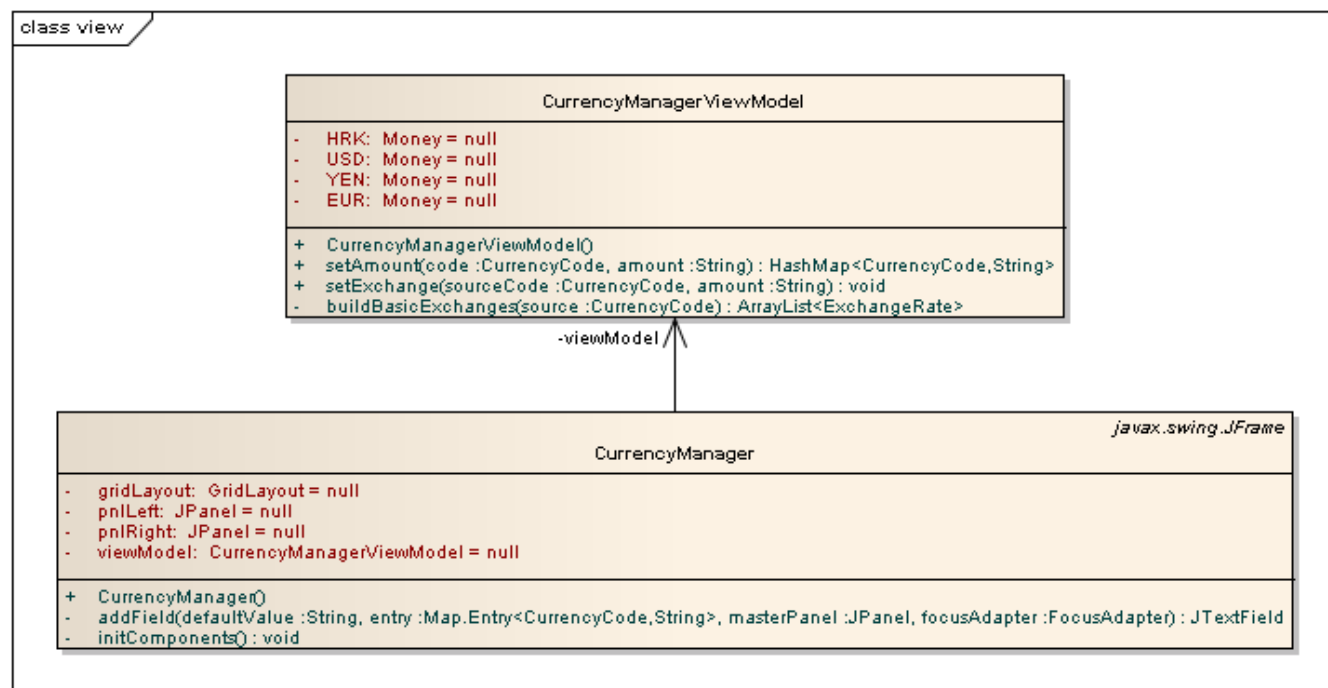
The principal goal is to model the entire application domain inside the domain layer and use

the UI only as an orchestration and presentation mechanism thus enabling domain logic testability and multiple UI reuse (REST and SOAP web Services, JSP web pages or JavaFX UI etc) .

Detailed Design

The API specification can be found in the **javadoc** folder.

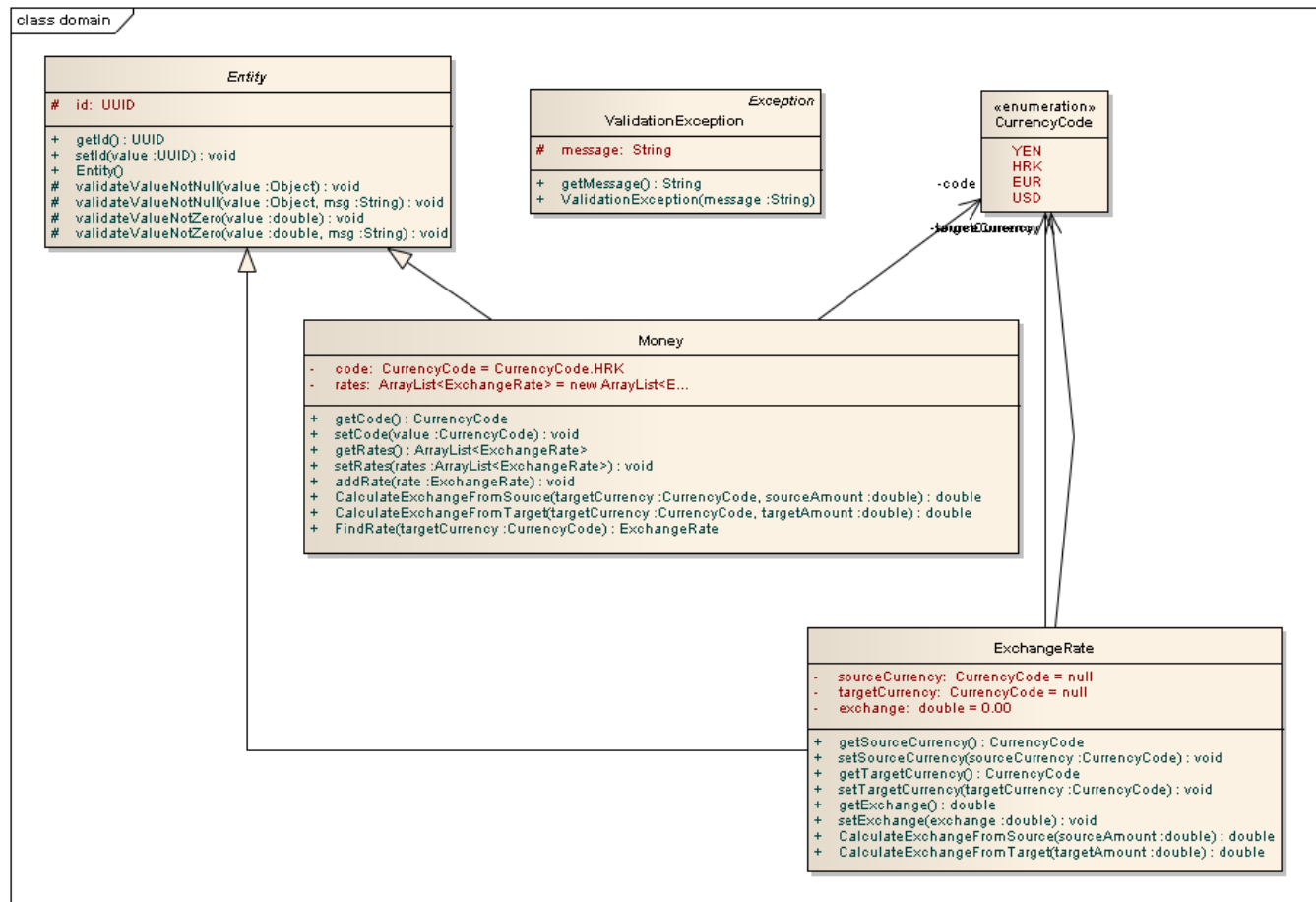
hr.efpu.informatika.currencymanager.view



There are two components in the View package:

Name	Description
CurrencyManager	Main window of the application. Its only goal is to present and gather data from and to the user. The rest of the orchestration logic is handled by its associated view model
CurrencyManagerViewModel	Main orchestration component of the view. It handles input from the form class, translates it into domain objects and orchestrates their usages.

hr.efpu.informatika.currencymanager.domain



The following is a list of domain components and their goals and responsibilities:

Name	Description
Entity	Root abstract class inherited by all entities. It provides a system unique identifier shared by all its children. It also provides common validation methods.
ValidationException	Exception raised whenever a value is entered into an entity which would set it in an invalid state.
Money	Domain principal entity. It tracks the association between a currency code and its exchange rates toward other currency codes.
CurrencyCode	Enumeration listing basic currency codes.
ExchangeRate	Entity offering services and states tasked with the goal to convert a source currency code to a target currency code and vice versa.

Patterns and Practices

Layered Architecture

All application and business concerns will be separated into appropriate layers each feeding the layer above with functionality needed for that layer to operate and getting functionality from the layer directly below.

The main reason is a clear separation of concerns and a simple flow of responsibility and data across the application.

Domain Driven Design

The entire business domain is represented as application objects with the names and interactions with closely (as possible) represent the domain they mimic. The domain is the most important part of the application and only it can execute and contain business logic.

Model View View Model


The MVVM design pattern will be used to extract the UI logic in a easily testable class. Thus the UI system can be easily unit tested without worrying about UI components and their interactions.

Unit testing






Using the Junit unit testing framework all the Domain and View classes (except those inheriting from the SWING package) will be tested in order to ascertain their validity as software components.




Configuration Management

The entire application source and all its documents are located inside GitHub a GIT distributed version control system and all the tasks assigned to the project are managed through its issue tracker.

← → ↻  GitHub, Inc. [US] <https://github.com/nstjelja/efpu-informatika/tree/master/CurrencyManager> ☆

For quick access, place your bookmarks here on the bookmarks bar. [Import bookmarks now...](#)






github [Explore](#) [Gist](#) [Blog](#) [Help](#)  **nstjelja**    

 **nstjelja / efpu-informatika** [Admin](#) [Unwatch](#) [Fork](#) [Pull Request](#)  1  1

Code Network Pull Requests 0 Issues 6 Wiki 0 Stats & Graphs

branch: master ▾ Files Commits Branches 1 Tags Downloads

efpu-informatika / CurrencyManager

name	age	message	history
..			
 documents	2 hours ago	Added intial class model [nikola]	
 nbproject	2 hours ago	Added intial class model [nikola]	
 src	2 hours ago	Added intial class model [nikola]	
 build.xml	2 hours ago	Added intial class model [nikola]	
 manifest.mf	2 hours ago	Added intial class model [nikola]	

The above image is a representation of the project document tree pushed to the main branch of the GIT server.

GitHub, Inc. [US] <https://github.com/nstjelja/efpu-informatika/issues>

For quick access, place your bookmarks here on the bookmarks bar. [Import bookmarks now...](#)

Code Network Pull Requests 0 Issues 6 Wiki 0 Stats & Graphs

Browse Issues Milestones Search: Issues & Milestones... [New Issue](#)

Everyone's Issues 6
Assigned to you 1
Mentioning you 0

No milestone selected

Labels
bug 0
duplicate 0
enhancement 0
invalid 0
question 0
wontfix 0

[Manage Labels](#)

New label name

No active filters. Use the sidebar to filter issues. Keyboard shortcuts available

6 open issues 0 closed issues Submitted Updated Comments

Close Label Assignee Milestone

- #6 [Currency] Create the UI of the Application
by nstjelja 2 days ago
- #5 [Currency] Create the Repository Layer of the application
by nstjelja 2 days ago
- #4 [Currency] Create the Repository Layer of the application
by nstjelja 2 days ago
- #3 [Currency] Extract from the javadoc the API
by nstjelja 2 days ago
- #2 [Currency] Create detailed class design and document it
by nstjelja 2 days ago
- #1 [Currency] Create Architecture diagram for the project and document it
by nstjelja 2 days ago 2 comments

6 open issues in this view

The above image is view on the currently open issue/tasks on the project which need to be completed before end. The goal is to document all the tasks, problems and defects which need to be completed before project end.

The screenshot displays a GitHub issue page. At the top, the issue title is "[Currency]Create Architecture diagram for the project and document it", marked as "Closed" in a red box. It was opened by user "nstjelja" 2 days ago. The issue is assigned to "nstjelja" with no milestone. The description reads: "Create an HLA diagram and document it". On the right, it shows "3 comments" and a "Labels" section. Below the issue header, it indicates "1 participant" and an "Add a comment" button. The comment history includes: 1) A comment from "nstjelja" 2 days ago: "Starting on the design. Will use Apache derby as an embedded database". 2) A comment from "nstjelja" 3 hours ago: "Created models into EA, will write architecture and design spec now." 3) A commit reference from "nstjelja" 2 hours ago (commit ac2b86a) with the message "Added document with structure and basic template". 4) A commit reference from "nstjelja" 6 minutes ago (commit 224bb08) with the message "Added all the documentation and diagrams, missing the API spec". 5) A commit reference from "nstjelja" 3 minutes ago (commit 96f9497) with the message "Added some text to the Configuration management section".

The above image shows the complete activity taken around an issue:

- Issue description
- Issue workflow and assignee
- Pushes linked with the issue

The principal points to take from this screen are as follows:


- The task and the steps taken to solve it are document
- All the parties which worked on an issue are known and trackable

- All pushes linked with the issue are visible and we can see all the changes made in the system for that issue.

Added document with structure and basic template

[Browse code](#)

#1

 **nikola** authored 2 hours ago 1 parent [1b68a1136e](#) commit [ac2b86a7992695dee7fed7c011f79dff97252509](#)

Showing 3 changed files with 0 additions and 0 deletions.

+

CurrencyManager/documents/document_template.odt

BIN


+

CurrencyManager/documents/hla.odt


BIN

CurrencyManager/documents/models.eap


BIN

 CurrencyManager/documents/document_template.odt [View file @ ac2b86a](#)

Binary file not shown


 CurrencyManager/documents/hla.odt [View file @ ac2b86a](#)

Binary file not shown

 CurrencyManager/documents/models.eap [View file @ ac2b86a](#)

Binary file not shown

0 notes on commit [ac2b86a](#) [Show line notes below](#)



Write Preview

Comments are parsed with [GitHub Flavored Markdown](#)

The above image is a view on the exact push tied to the previously shown issue. If plain text files were pushed a diff would also be shown.