|  |
| --- |
| **UNIVERSITY “POLITEHNICA” OF BUCHAREST** |
| **SOFTWARE DESIGN DESCRIPTION**  **“MusicDepot” Java Based application**  *Student:*  *Cristina GEORGESCU- MSE1*  *Coordinator: Date created:*  *Prof. Andrei VASILATEANU Monday, December 3rd, 2018* |

***Table of contents***

[***1.******Introduction*** *- 3 -*](#_Toc534561088)

[*1.1. Purpose - 3 -*](#_Toc534561089)

[*1.2. Scope - 3 -*](#_Toc534561090)

[*1.3.Overview of Document - 3 -*](#_Toc534561091)

[***2.******Deployment Diagram*** *- 4 -*](#_Toc534561092)

[***3.******Architectural Design*** *- 4 -*](#_Toc534561093)

[*3.1. Flow of events - 6 -*](#_Toc534561094)

[*3.2. pom.xml - 7 -*](#_Toc534561095)

[*3.3.DroolRules.drl - 7 -*](#_Toc534561096)

[*3.4. Main.java - 8 -*](#_Toc534561097)

[*3.5. Cart.java - 8 -*](#_Toc534561098)

[*3.6. CartStatus.java - 8 -*](#_Toc534561099)

[*3.7. Products.java - 8 -*](#_Toc534561100)

[*3.8. ProductsInCart.java - 9 -*](#_Toc534561101)

[*3.9. ProductsInPending.java - 9 -*](#_Toc534561102)

[*3.10. User.java - 9 -*](#_Toc534561103)

#### Fig 1. Deployment Diagram……………………………………………………………….……………*...-4-*

Fig 2. Top-Level Architectural Diagram………………………………………………………….……………...-6-

# ***Introduction***

## 1.1. Purpose

This document contains the complete design description of the “*MusicDepot*” Java-based application. This includes the architectural features of the system down through details of what operations each code module will perform and the database layout. It also shows how the use cases detailed in the SRS will be implemented in the system using this design.

The primary audiences of this document are the software developers.

## 1.2. Scope

There are 2 parts of this system:

* The imperative/functional part, completed with Java in Eclipse;
* The declarative part, i.e. the rules that the structure of the program should follow, is done with Drools.

For this application, we have used the following frameworks:

* Maven 3.23.1
  + The primary goal of Maven is to allow any developer to comprehend the complete state of a development effort in the shortest period of time
  + We use Maven in this application in order to create a special file for the rule dependencies done in the declarative part
* Java 11
* Eclipse as the IDE, version 2018 12.
* Drools 7.15.0
  + Drools is a business rule management system (BRMS) with a forward and backward chaining inference-based rules engine, more correctly known as a production rule system,
  + KIE (Knowledge Is Everything) is the new umbrella name to Drools,
  + Drools supports the Java Rules Engine API (Java Specification Request 94) standard for its business rule engine and enterprise framework for the construction, maintenance, and enforcement of business policies in an organization, application, or service.

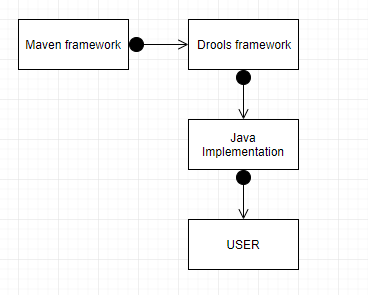
## Overview of Document

*Chapter 2* is a *Deployment Diagram* that shows the physical nodes on which the system resides. This allows a clear explanation of where each design entity will reside. No design unit may straddle two nodes but must have components on each, which collaborate to accomplish the service.

*Chapter 3* is the *Architectural Design*. This is the heart of the document. It specifies the design entities that collaborate to perform the functionality of the system. Each of these entities has an Abstract Specification and an Interface that expresses the services that it provides to the rest of the system. In turn each design entity is expanded into a set of lower-level design units that collaborate to perform its services.

# ***Deployment Diagram***

Deployment diagrams show how elements of software and hardware are connected to one another. Since they can describe hardware, deployment diagrams are unique in the UML world.



#### Fig 1. Deployment Diagram

The products for the User to choose from are implemented directly in the Java code. Every element by itself has its own Java file, so it’s easier to read the code and modify when necessary i.e. the user is computed separately, the cart, the products put in the cart etc.

The Maven framework influences the business rules. The rules are influencing the code, to verify if the following are true:

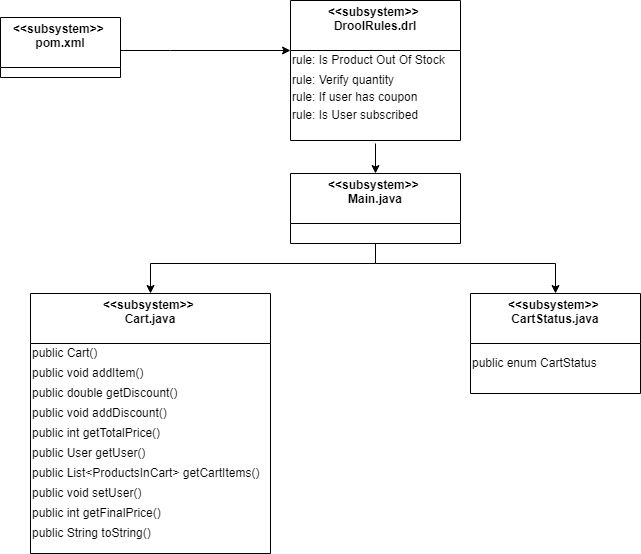
* If a product is out of stock
* If the quantity that the user desires is greater than the available stock for that product
* If the user has a discount coupon of type ‘GMC10’, then a 10% discount will be applied to the final price of the order
* If the user is subscribed, there will be no shipping fee (this fee is of 10€ in general)

The user that interacts with the application will see only the display of the final result.

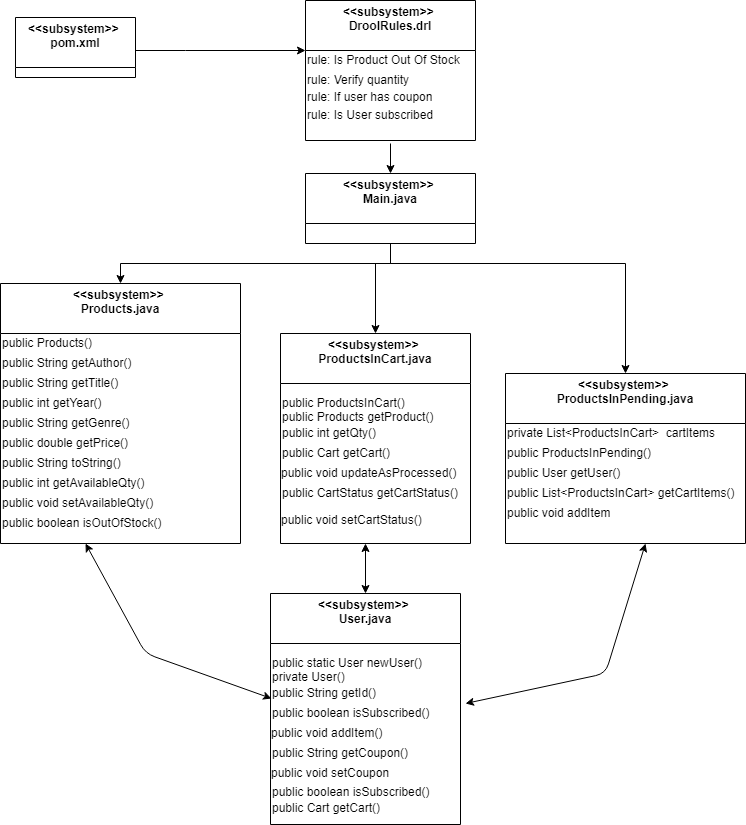
# ***Architectural Design***

Software architecture involves the high-level structure of software system abstraction, by using decomposition and composition, with architectural style and quality attributes. A software architecture design must conform to the major functionality and performance requirements of the system, as well as satisfy the non-functional requirements such as reliability, scalability, portability, and availability.

A software architecture must describe its group of components, their connections, interactions among them and deployment configuration of all components.



*(continuation of the previous schema)*

Fig 2. Top-Level Architectural Diagram

## 3.1. Flow of events

* There are 3 main parts to the concept of this application:
  + The user part
  + The cart part
  + The product part
* The user and the products that the user has chosen is implemented directly in main.java, this is done to see how the Drool rules work on a given test set
* The properties of a user are preset: if the user has a coupon and if the user is subscribed
* The list of products is predefined in the Main.java
* Every user is linked to his cart
* Every cart has one of the following 3 status, depending on the operations done so far:
  + NEW – when there were no operations done
  + PROCESSED – product is in the user’s cart
  + PENDING – awaiting verification for available stock
* Firstly, we define the products.
* Each product is defined throughout the application by the following attributes:
  + Author of the music album
  + Title
  + Year of appearance on the market
  + Price in € like 20.99€ for example
* The available quantity for each product is set
* Second, we define the user, by giving him/her a name, that will act as the unique identifier over the whole application of that particular user
* The Drool rules are fired, verifying everything and displaying the corresponding messages and the program is done when the final price is displayed, after the discounts when the case is so.

## 3.2. pom.xml

This file is necessary for the proper functioning of the Drool rules.

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects.

## DroolRules.drl

In this file we have set all the rules discussed previously, that will apply to the Java code itself.

The operations are:

* rule "Is Product Out Of Stock"
  + applies to the cart and it does the verification on each cart item, one at a time
  + as long as the status of the cart is different than “Processed”, the rule checks if the available stock is 0
  + in case that the available stock is indeed 0, the rule displays an according message, which contains the full description of the product in case
* rule "Verify quantity"
  + applies to the cart and it does the verification on each cart item, one at a time
  + as long as the status of the cart is different than “Processed”, and the product is not out of stock, the rule will check the quantity that the user wants with the available stock
  + the rule displays a message to let the user know the real available quantity
* rule "If user has coupon = 10% discount from Grand Total"
  + apples to the user
  + check is the user has a coupon of type “GMC10”
  + if the condition is true, then a 10% discount is applied to the Grand Total of the cart
* rule "If subscribed= -10 euros"
  + applies to the user
  + checks if the user is subscribed or not
  + if the condition is true, the shipping fee of 10€ is subtracted from the Grand Total price on that cart.

## 3.4. Main.java

This file is basically connecting all the other files, to make the final output. This is the file to be run in order to make the application work.

The operations are:

* creating a session to call in the rules detailed previously
* creating a new user
* making the list of products

E.G: Products p1 = **new** Products("LinkinPark", "A Thousand Suns", 2010 ,"Rock", 20.99);

* set the available quantity for each product

E.G: p1.setAvailableQty(250);

* for each user, add items in the cart and the desired quantity of that product

E.G: u1.addItem(p1, 5);

* set the coupon for each user
* set subscription
* display all the cart items

## 3.5. Cart.java

This file contains all the information to build the cart.

The operations are:

* creating the user as an object
* each product is considered an object
* the products are added to the cart
* the discount is added
* the Grand Total is calculated
* the Final Price is calculated as Grand Total - discount

## 3.6. CartStatus.java

Any cart has 3 possible statuses, that help in upcoming operations:

* + NEW – when there were no operations done
  + PROCESSED – product is in the user’s cart
  + PENDING – awaiting verification for available stock

## 3.7. Products.java

Any product is built by getting all it’s attributes from the main file where they are populated.

Each product is defined throughout the application by the following attributes:

* + Author of the music album
  + Title
  + Year of appearance on the market
  + Price in €, like 20.99€ for example

The functions from this file are mostly used in the Main.

## 3.8. ProductsInCart.java

When a user first receives an item, the corresponding cart gets the status NEW.

Every cart is characterized by the product put in it and the quantity of that product.

## 3.9. ProductsInPending.java

A cart has the status NEW when the first item gets assigned. While there are operations done on the items in the cart, it has the status PENDING, in wait for the Drool rules to be checked.

## 3.10. User.java

The user is associated with a cart, a name that acts like an id, a possible coupon and a possible subscription. The last 2 are not mandatory, so they are Boolean functions in the code.