ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ МОЛДОВЫ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАТИКИ

ДЕПАРТАМЕНТ ИНФОРМАТИКИ

**ГРИМАЛЮК ДЕНИС**

**Отчет по практике**

Руководитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Плешка Наталия, лектор

(подпись)

Автор: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Кишинев, 2019

**План проведения практики**

*03-06-2019 – 29-06-2019, ’*M-Testing’ SRL

|  |  |
| --- | --- |
| 25.06. – 25.06. | **Департамент/секция «’M-Testing’ SRL», организации «Allied Testing»** |
| 26.06 – 26.06 | **Формулировка задачи, требований и описание возможностей решения поставленной задачи. Проектирование решения поставленной задачи** |
| 27.06 – 27.06 | **Описание реализации представленной задачи** |
| 28.06 – 29.06 | **Тестирование и проверка на соответствие требований. Выводы** |

# **ДЕПАРТАМЕНТ/СЕКЦИЯ «’M-TESTING’ SRL», ОРГАНИЗАЦИИ «ALLIED TESTING»**

Организация, в которой была проведена практика называется «Allied Testing». Она является не только международной QA компанией, но и лабораторией качества.

Основанная в 2000 году, *Allied Testing* практически сразу закрепила за собой место одного из лидеров на рынке обеспечения качества финансовых и трейдинговых приложений.

При этом компания предоставляет самый широкий спектр услуг в области ручного и автоматизированного тестирования, программирования, *QA* консалтинга, разработки стратегий обеспечения качества. Кроме того, Allied Testing активно применяет методы *performance* тестирования и тестирования мобильных приложений.

В основе службы контроля качества и тестирования компании лежит конечное качество приложения. Работа заключается в создании экономичных и интеллектуальных решений для тестирования, позволяющих выявлять и устранять дефекты программного обеспечения, обеспечивая соответствие приложений их функциональным требованиям.

В дополнение к предоставлению широкого спектра услуг по тестированию, Allied Testing предлагает комплексные консалтинговые услуги по обеспечению качества. Основываясь на опыте в наиболее эффективных процессах и методологиях тестирования, а также на опыте предоставления услуг *QA* по всему миру ведущим мировым компаниям, работа с клиентами, чтобы обеспечить более эффективные процессы тестирования.

Чтобы не отставать на сегодняшнем все более конкурентном и фрагментированном рынке, организации постоянно совершенствуют свои продукты и системы, чтобы сократить время выхода на рынок и добиться более качественного выполнения для своих клиентов, тем самым улучшая удержание клиентов и увеличивая долю рынка. Благодаря единственному вниманию к области торговли и большому техническому опыту, кампания имеет уникальную возможность поддерживать разработку и эксплуатацию самых современных торговых систем.

Хорошо поддерживаемая тестовая среда с выделенной группой поддержки является критическим требованием для обеспечения эффективности *QA* сложных приложений при глобально распределенной разработке. Когда проблемы возникают, важно иметь специальную группу поддержки среды тестирования, которая может быстро и эффективно вмешиваться в их решение.

Портфель специализированных услуг *Allied* по тестированию для торговой индустрии помогает их клиентам решать вопросы качества различных критически важных для бизнеса компонентов, включая *DMA / OMS*, системы определения местоположения, связь с рынком, алгоритмы торговли, фронт, средний, бэк-офис и соответствие требованиям.

Офисы и представительства *Allied Testing* расположены в Москве, Кишиневе, Минске, Белграде, Лондоне и Нью-Йорке.

Кишиневское представительство *Allied Testing* сегодня – это более 300 сотрудников.

Компания работает с многочисленными клиентами, в том числе: с фондовыми биржами, инвестиционными банками, брокерскими конторами прямо или косвенно, тестируя соответствующие им механизмы, системы наблюдения, операции на рынке и отчетность, торговые шлюзы и шлюзы рыночных данных, почтовые торговые системы, связь с их клиентами, в результате.

*Allied Testing*. Компания, с которой, все обретает качество и вес!

# **II. ФОРМУЛИРОВКА ЗАДАЧИ, ТРЕБОВАНИЙ И ОПИСАНИЕ ВОЗМОЖНОСТЕЙ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ. ПРОЕКТИРОВАНИЕ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ**

## **2.1. Описание задачи**

Разработать приложение на *Java*, которое будет проводить валидацию данных о курсе валют на сайте www.point.md, используя официальную информацию с сервиса [www.bnm.md](http://www.bnm.md). Данное приложение имеет роль тестировать функционал сайта [www.point.md](http://www.point.md).

## **2.2. Формулировка требований к приложению: функциональные, требования к возможностям, скорости работы**

Важными требованиями для приложения являются следующие пункты:

* Приложение должно проводить тестирование сайта www.point.mdметодом *UI*, то есть взаимодействуя с интерфейсом страницы.
* Приложение должно получать информацию с www.bnm.md сервиса методом *API*, отправляй запрос на сервис, получая в ответ файл с официальными данными и десериализую получаемый файл в *Java* объекты.
* Приложение должно использовать *Firefox* браузер.
* Информация выведенные в режиме выполнении программы в консоль должна выводиться посредством Логов и сохраняться в файле, при этом максимальный размер файла должен составлять 5 мб.
* Приложение должно использовать *Test-Ng* Фреймворк.
* Десериализация должна проходить с использованием *XStream* библиотек.
* В качестве элементов на веб странице должны использоваться *Page Object Models*, все xpath-ы должны быть вынесены в этот класс, для читабельности основного кода.

## **2.3. Потенциальные пользователи**

Потенциальными пользователями данного приложения являются девелоперы/тестировщики сайта *point.md* у которых есть необходимость выполнять *regression* тестирования с целью исключить появление багов в релизе.

## **2.4. Проектирование решения поставленной задачи**

Необходимо выявить внутреннюю структуру класса функции которое понадобятся для выполнения основного тестового метода. В первую очередь понадобится метод *initialize()*, в котором будет запускать с браузер, будет инициализироваться Логер, драйвер, *Page Object Model* класс и wait, необходимый для ожидания загрузки страницы. Следующим методом понадобится *getBNMInfo()*, в котором будет реализована логика отправки запроса на bnm сервис и получение в результате файла с официальными данными. Для получения файла с данными необходимо указать дату в формате *dd.MM.yyyy*, для этого есть метод *getDateToString()*, который использует класс *SimpleDateFormat* для преобразования даты в строку. Одним из важнейших методов является *fromXml()*, в котором реализована логика десериализации *XML* строки в объекты *Java*. Для десериализации понадобятся объекты классов поля которых соответствуют элементам *XML* строки. Так же необходимо создать *POM* файл в котором будут находиться веб элементы point.md страницы.

# **ОПИСАНИЕ РЕАЛИЗАЦИИ ПРЕДСТАВЛЕННОЙ ЗАДАЧИ**

## **3.1. Описание технологий, языков программирования, которые были использованы для решения задачи**

### **3.1.1. Язык Java**

Со времени создания первых программируемых машин человечество придумало более восьми тысяч языков программирования (включая эзотерические, визуальные и игрушечные). Каждый год их число увеличивается. Некоторыми языками умеет пользоваться только небольшое число их собственных разработчиков, другие становятся известны миллионам людей. Профессиональные программисты могут владеть десятком и более разных языков программирования.

Язык программирования предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной вычислительный процесс, организовать управление различными объектами, и т. п. Язык программирования отличается от естественных языков тем, что предназначен для управления ЭВМ, в то время как естественные языки используются, прежде всего, для общения людей между собой. Большинство языков программирования использует специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений.

По разным оценкам, в настоящее время существует от двух с половиной до десяти тысяч различных языков программирования. Самыми популярными являются: ***Swift, Go, PHP, C++, Python, JavaScript, Java, C#, Kotlin*** и другие.

Для выполнения поставленной задачи был выбран такой язык программирования как *Java***,** так как это один из самых практичных языков программирования для изучения. Его популярность нельзя переоценить, так как большинство (90%) компаний из списка Fortune используют *Java* для разработки бэкэнд-систем и десктопных приложений. Кроссплатформенность достигнута благодаря *JVM (Java Virtual Machine).*

В *Java*, как и во многих современных языках, включая *C++, Python* etc. используется принцип объектно-ориентированного программирования (*ООП*). *Java* в основном используется для создания серверных приложений и мобильных.

Программы на *Java* транслируются в байт-код *Java*, выполняемый виртуальной машиной *Java (JVM)*— программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

### **Test – Ng Фреймворк**

*TestNG* - это среда тестирования для языка программирования *Java*, созданная Седриком Беустом и вдохновленная *JUnit* и *NUnit*. Целью проекта *TestNG* является охват более широкого диапазона категорий тестов: *unit, functional, end-to-end, integration* и т. д. С более мощными и простыми в использовании функциями.

Существенными преимуществами данного фреймворка являются:

* Большое количество аннотаций, которые упрощают написание кода;
* Параллельный запуск тестов;
* Использование *data provider*;
* Test Suit, в которых можно указывать какие тесты нужно запускать и в каком порядке, есть возможность указать какие тесты запустить параллельно;
* Удобный вывод в консоль и отчетность;
* Большое количество параметров у аннотации @Test.

### **Selenium tool**

***Selenium*** — это инструмент для автоматизации действий веб-браузера. В большинстве случаев используется для тестирования Web-приложений, но этим не ограничивается. В частности, реализация Selenium *WebDriver* для браузера *phantomjs* часто используется как веб-граббер.

*Selenium WebDriver* — это в первую очередь набор библиотек для различных языков программирования. Эти библиотеки используются для отправки *HTTP* запросов драйверу (отсюда и название *WebDriver*), с помощью протокола *JsonWireProtocol*, в которых указано действие, которое должен совершить браузер в рамках текущей сессии. Примерами таких команд могут быть команды нахождения элементов по локатору, переход по ссылкам, парсинг текста страницы/элемента, нажатие кнопок или переход по ссылкам на странице веб-сайта. Существуют как официальные привязки библиотеки к популярным языкам программирования, так и любительские. К примеру, библиотека для поддержки языка *PHP* не является официальной и разрабатывается *Facebook*.

Проектом *Selenium* и сообществом поддерживается работа с браузерами *Microsoft Internet Explorer, Google Chrome, Mozilla Suite и Mozilla Firefox* под управлением операционных систем *Microsoft Windows, Linux и Apple Macintosh.*

### **Log4J библиотека**

*Log4j*— библиотека журнализации *Java* программ, часть общего проекта «Apache Logging Project».

*Log4j* первоначально развивался в рамках зонтичного «*Apache Jakarta Project*», ответственного за все Java-проекты Apache, но впоследствии выделился в отдельный, очень популярный проект журналирования.

Данная библиотека позволяет выводить логи как в консоль, так и записывать их в файл. Существует возможность создать файл *logging.proprietes* где будет указан формат вывода логов.

*Log4j* позволяет использовать логи разного уровня*: trace, info, debug, fail, error* и др.

### **Maven**

***Apache Maven*** — фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке *POM* (*Project Object Model*), являющемся подмножеством *XML*. Проект *Maven* издаётся сообществом *Apache Software Foundation*, где формально является частью *Jakarta Project*.

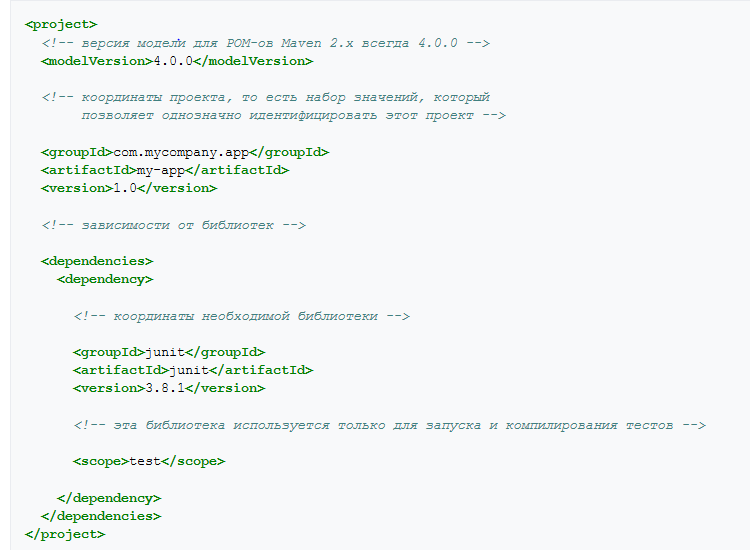
*Maven* обеспечивает декларативную, а не императивную сборку проекта. В файлах описания проекта содержится его спецификация, а не отдельные команды выполнения. Все задачи по обработке файлов, описанные в спецификации, *Maven* выполняет посредством их обработки последовательностью встроенных и внешних плагинов.

*Maven* используется для построения и управления проектами, написанными на *Java, C#, Ruby, Scala*, и других языках.

Среди примечательных альтернатив — система автоматической сборки Gradle, построенная на принципах *Apache Ant* и *Maven*, но использующая специализированный *DSL* на *Groovy* вместо *POM*-конфигурации.

Информация для сборки проекта, поддерживаемого *Apache Maven*, содержится в *XML*-файле с названием ***pom.xml***. При запуске *Maven* проверяет, содержит ли конфигурационный файл все необходимые данные и все ли данные синтаксически правильно записаны.

Пример файла pom.xml представлен на рис. 3.1:



**Рис. 3.1. Пример pom.xml файла**

Минимальная конфигурация включает версию конфигурационного файла, имя проекта, его автора и версию. С помощью pom.xml конфигурируются зависимости от других проектов, индивидуальные фазы процесса построения проекта (build process), список плагинов, реализующих порядок сборки.

Крупные проекты могут быть поделены на несколько модулей, или подпроектов, каждый со своим собственным *POM*. Операции над модулями могут выполняться через общий корневой *POM* единой командой.

### **XStream библиотека**

***XStream*** — это Java библиотека классов для сериализации объектов в *XML* (или *JSON*) формат. *XStream* использует рефлексию для исследования структуры объекта в процессе выполнения программы, и не требует внесения изменений в сам объект. Библиотека позволяет сериализовать приватные, а также, объявленные как *final*, поля. Кроме этого, поддерживаются не публичные и вложенные классы.

При сериализации объекта выполняется построение его полного графа. При этом сохраняются все кросс-ссылки объекта.

Для сериализации/десериализации используются аннотации библиотеки *XStream*: @XmlElement, @XStreamAsAttribute, @XstreamAlias, XStreamImplicit и другие.

## **Использование вышеперечисленных технологий в реализации поставленной задачи**

Для выполнения поставленной задачи мне понадобилось создать *Maven* проект так как задание требует импорта сторонних библиотек, в результате *pom.xml* выглядит как на рис. 3.2 и рис. 3.3:



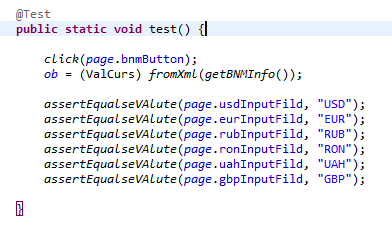
**Рис. 3.2. Содержимое файла pom.xml в проекте ч1**



**Рис. 3.3. Содержимое файла pom.xml в проекте ч2**

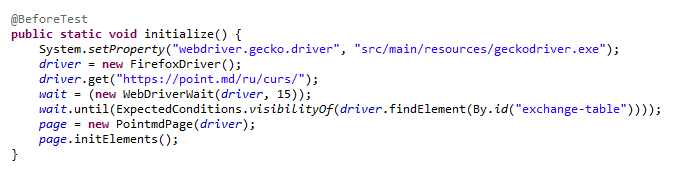
Между тегом *<dependencies> </dependencies>* расположены информация о библиотеках, которые необходимо скачать и локально сохранить в *Maven* репозитории. Это такие инструменты как: *Test-Ng, Selenium, XStream, Log4j* и другие.

Подгрузив все необходимые библиотеки, я создал главный метод, который обозначил аннотацией *@Test*, которая указывает что данный метод является тестовым (рис. 3.4).

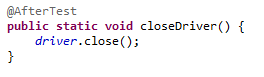


**Рис. 3.4. Метод test ()**

В качестве вспомогательных методов я так же создал два c аннотациями @*BeforeTest* (рис. 3.5) и @*AfterTest* (рис. 3.6), которые в свою очередь выполняет некоторые действия до и после теста.



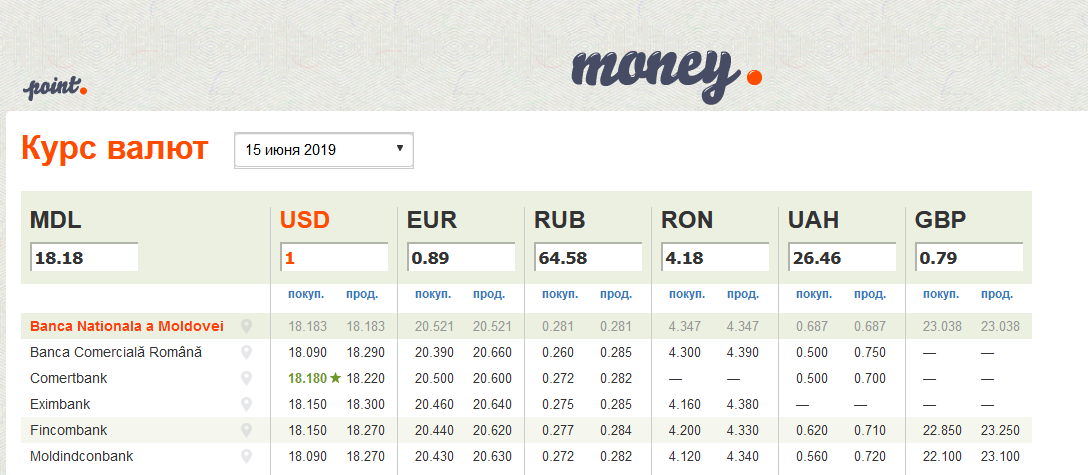
**Рис. 3.5, Метод с аннотацией @BeforeTest**



**Рис. 3.6. Метод с аннотацией @AfterTest**

Метод *initialize*() инициализирует все необходимые объекты и подготавливает среду для выполнения теста, а метод *closeDriver***()** после завершения главного тестового метода закрывает браузер.

Тестируемый сайт выглядит как это представлено в рис. 3.7:



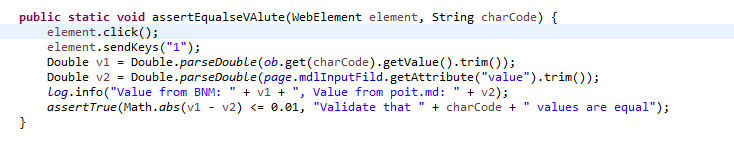
**Рис. 3.7. Интерфейс point.md страницы.**

Чтобы узнать курс одной из представленных валют, необходимо в её поле ввести «**1**» и получить значение из поля «***MDL*»,** что означает отношение данной валюты к молдавскому лею. Данную процедуру необходимо провести с каждым из 6 полей на странице, получить значение из поля «***MDL*»** и проверить совпадает ли значение с официальными данными.

Алгоритм выглядит следующим образом:

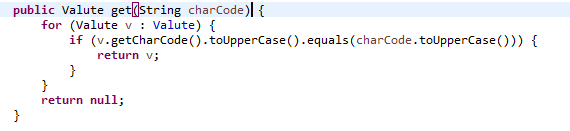
1. Открыть страницу;
2. Нажать на *Banca Nationala a Moldovei*;
3. Отправить запрос на *bnm* сервис на текущую дату;
4. Сделать десериализацию полученного ответа в Java объект;
5. В каждое поле справа от MDL по очереди ввести значение «**1**» и получить текст из MDL поля проверив на совпадение с ранее полученным объектом в результате десериализации.

Проверку интерпретирует следующий метод на рис. 3.8:



**Рис. 3.8. Метод *assertEqualsevalute ()***

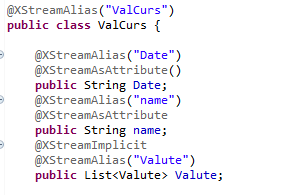
Метод *asssertEqualseValute()* получает два аргумента, *WebElement* в который будет отправлена «**1**» и название данной валюты. После чего метод *sendKeys()* отправляет в поле «**1**» и для читабельности кода я создал две переменные которые содержат значение из *XML* файла и из MDL поля. Переменная **v1** получает значение из xml объекта, если открыть метод *get()* (рис. 3.9), то можно заметить что в данном методе происходит поиск объекта:



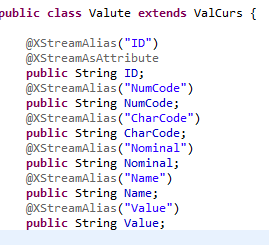
**Рис. 3.9. Метод get ().**

Чей *char code* совпадает с полученным, в случае если объект с таким названием валюты не будет найден вернется null. Переменная **v2** получает свое значение из MDL поля предварительно конвертированное в Double значение. Метод *assertTrue()* проверяет что разница этих двух значений не больше числа 0.01. В случае если разница v1 и v2 будет больше числа 0.01, то тест упадет и выявится баг.

Далее представлено как происходит десериализация объекта с использованием *XStream* библиотек. В моём проекте есть два объекта которые предназначены для хранения: *Valcurs* рис(3.10) и *Valute* рис(3.11).



**Рис. 3.10. Класс *ValCurs***



**Рис. 3.11. Класс *Valute***

Ответ с BNM сервиса выглядит как на рис 3.12:

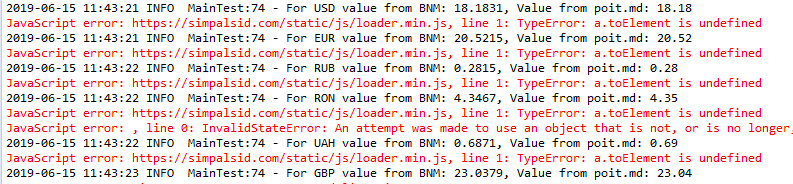


**Рис. 3.12. *XML* ответ с сервиса *BNM***

Аннотации помогают создать структуру XML файла, причем отделить XML элементы от их атрибутов, в случае, когда ожидается что встретиться один или более элементов с одним и тем же названием, используется аннотация @XStreamImplicit.

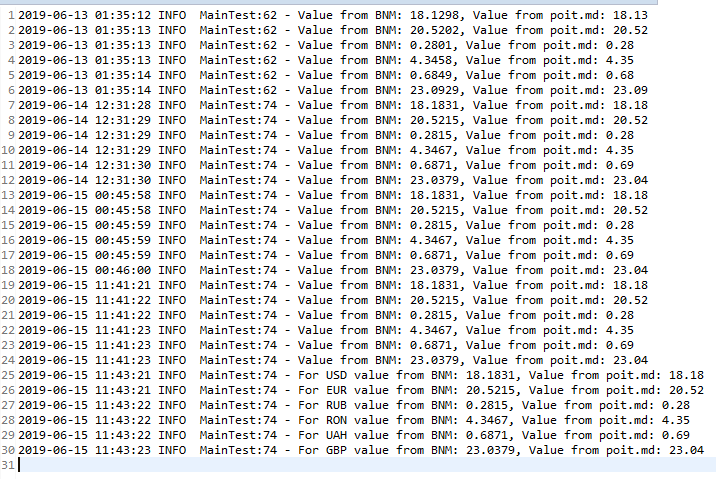
Стоит заметить, что в случае, когда один из XML элементов нет необходимости преобразовывать в Java объект, в необходимо так же создавать переменную для его хранения и использовать аннотации, так как если хоть одно поле не будет найдено в классе дессериализация не пройдёт успешно.

Далее представлено использование логов библиотеки *Log4j*. После выполнения теста в консоли наблюдается следующая информация (рис. 3.13):



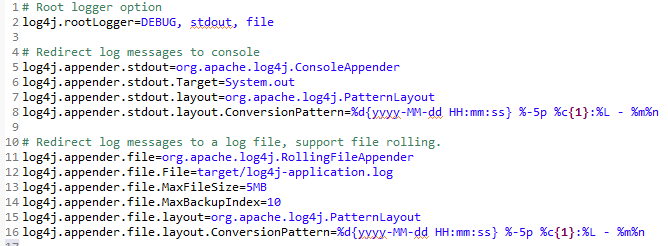
**Рис. 3.13. Логи запуска приложения.**

В случае если бы к примеру, на валюте RON тест упал, то это отличная возможность просмотреть причину падения теста. Так же, если есть необходимость сохранить историю логов, то для этого в моём проекте все логи заносятся в файл **log4j-application.log**(рис. 3.14)**.**



**Рис. 3.15. Файл log4j-application.log**

Предварительное создание файла *log4j.properties* позволяет настроить формат вывода логов в консоль и их запись в файл.



**Рис. 3.15. Файл log4j.properties**

Можно заметить, что в выше указанных методах не было явного вызова веб элементов по их *xpath*. Для начала необходимо объяснить что он из себя представляет.

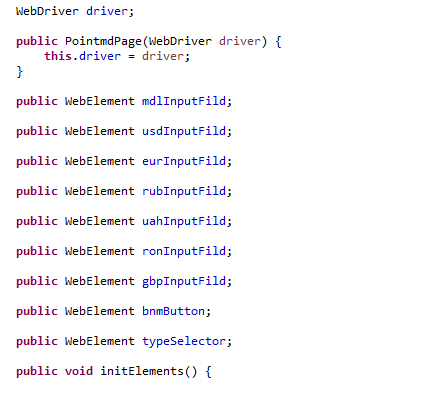
*XPath* (*XML Path Language*) — язык запросов к элементам *XML*-документа. Разработан для организации доступа к частям документа *XML* в файлах трансформации *XSLT. XPath* призван реализовать навигацию по *DOM (Document Object Model)* в *XML*. В *XPath* используется компактный синтаксис, отличный от принятого в *XML*.

К примеру, данный *xpath*:

"//tr[@class='exchangecalc']//th[count(//table[@id='exchangetable']//h3[text()='MDL']/preceding::h3)+1]/input"

находит MDL поле на point.md странице. Он считает количество предшествующих тегов полю MDL и выбирает поле номер которого в списке является этим числом +1, так как себя он не считает предшествующим самому себе.

В проекте был создан *POM (Page Object Model)* для тестируемой странице который выглядит следующим образом (рис. 3.16).



**Рис. 3.16. POM файл point.md страницы**

Метод *initialize()* (рис 3.17) присваивает переменным их веб элементы.



**Рис. 3.17. Метод *initialize () в POM файле***

# **ТЕСТИРОВАНИЕ И ПРОВЕРКА НА СООТВЕТСТВИЕ ТРЕБОВАНИЙ. ВЫВОДЫ**

## **Проблема с которыми пришлось столкнуться**

На этапе реализации пришлось столкнуться с большим количеством проблем, одними из них являются:

1. Не работал метод *clear()* чтобы очистить поле валюты перед тем как вставить туда «1», решилась проблема простым кликом по полю, тогда весь текст в нем выделяется и заменяется «1».
2. Проблемы с десериализацией, этот трудный процесс требовал использования аннотаций не только к переменным, но и к самому классу, мне пришлось подумать, прежде чем догадаться что это именно так.
3. Инициализация веб элементов в POM файле происходит вручную, в идеале должен использоваться метод *PageFactory.initElements (driver,this),* это получилось используя другой метод.
4. После импорта библиотеки *Log4J* логи не выводились в консоль, поискав в интернете стало понятно что необходимо создать *properties-файл* в котором указать куда должны выводиться логи и в каком формате.
5. Отсылая запрос на *bnm* сервис не приходил ответ, так как параметры запроса указывались не через **?** а через **&** символ. Вспомнив API проблему удалось исправить.

## **Высказывания руководителей**

Руководители оценили проект, так как в нем были представлены одни из лучших идей и четкая реализация алгоритма проверки получаемой информации. Плюс ко всему код получился наиболее читабельным без повторения одних и тех же строк, при этом главный тестовый метод содержит по минимуму кода для читабельности. Руководители хорошо высказались о том как выглядят *xpath*-ы (пути к веб элементу) в POM файле так как довольно редко используют и практикуют такие опции как *preceding-sibling*, *ancestor-sibling* и другие и понимание значения этих опций дает хороший опыт в *UI* тестировании.

## **Выводы, личное мнение о полученном опыте**

Все чему пришлось научиться оказался бесценный опыт так самому не получилось бы изучение таких вещей как десериализация, API, логи, *Maven* и другие незаменимые методы в программировании, потому что одному это довольно трудно изучить и тем более понять.

Кроме знаний появилось понимание как делать код читабельным, как предусматривать на перед возможные логические ошибки в коде и предугадывать наперед как поведет себя софт при той или иной ситуации.

Стоит заметить так же что помимо опыта в программировании я получил и опыт в собеседованиях на английском языке что поможет мне в будущем чувствовать себя увереннее в общении с иностранцами.