**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра САПР**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Объектно-ориентированное программирование»**

**Тема: Создание бота для Telegram**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3354 |  | Жуков К.В. |
| Преподаватель |  | Кулагин М.В. |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

|  |  |  |
| --- | --- | --- |
| Студент Жуков К.В. | | |
| Группа 3354 | | |
| Тема работы: Создание бота для Telegram | | |
| Исходные данные:  Язык Java версии 20.0.2, инструмент для сборки Maven, среда разработки IntelliJ IDEA. | | |
| Содержание пояснительной записки:  «Введение», «Внутреннее описание программы», «Интерфейс и выполнение программы», «Заключение», «Список использованных источников», «Приложение А» | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц | | |
| Дата выдачи задания: 09.10.2024 | | |
| Дата сдачи реферата: 26.12.2024 | | |
| Дата защиты реферата: 26.12.2024 | | |
| Студент |  | Жуков К.В. |
| Преподаватель |  | Кулагин М.В. |

**АННОТАЦИЯ**

В данной курсовой работе приводится создание бота для Telegram на языке Java, который выводит пользователю актуальные котировки валют и драгоценных металлов. Код пишется с помощью библиотеки TelegramBots, данные котировок взяты с сайта центрального банка РФ. В самом боте реализован удобный функционал с помощью кнопок, предоставляется выбор показа разных котировок.

**SUMMARY**

This course work describes the creation of a telegram bot in Java, which displays to the user current quotes for currencies and precious metals. The code is written using the TelegramBots library, quote data is taken from the website of the Central Bank of the Russian Federation. The bot itself implements convenient functionality using buttons, and provides a choice of displaying different quotes.

**СОДЕРЖАНИЕ**

|  |  |  |
| --- | --- | --- |
|  | Введение | 4 |
| 1. | Внутреннее описание программы | 6 |
| 1.1. | Диаграмма вариантов использования | 6 |
| 1.2. | Диаграмма классов | 7 |
| 2. | Интерфейс и выполнение программы | 9 |
| 2.1. | Описание интерфейса пользователя | 9 |
| 2.2. | Пример работы программы | 9 |
|  | Заключение | 12 |
|  | Список использованных источников | 13 |
|  | Приложение А. Весь код | 14 |

**ВВЕДЕНИЕ**

Целью работы является реализация бота для Telegram, который будет выводить котировки валют и драгоценных металлов, по выбору пользователя.

Основные задачи, это: создание самого бота, “привязка” его к сайту центрального банка РФ для получения котировок, а также реализация удобного интерфейса для пользователя.

Для решения этих задач используется помощник по созданию бота TelergamFather, стандартная библиотека для парсинга XML файлов JDK API и библиотека TelergamBots для организации работы самого бота.

**1. ВНУТРЕННЕЕ ОПИСАНИЕ ПРОГРАММЫ**

* 1. **Диаграмма вариантов использования**

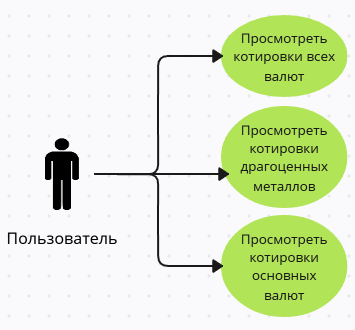


Рис. 1. Диаграмма вариантов использования программы

Пользователь, взаимодействуя с ботом, может получать информацию по котировкам всех валют, либо только основных (USD, EUR, GBR, CNY), либо драгоценных металлов (Золото, серебро, палладий, платина).

**1.2 Диаграмма классов объектной модели предметной области.**

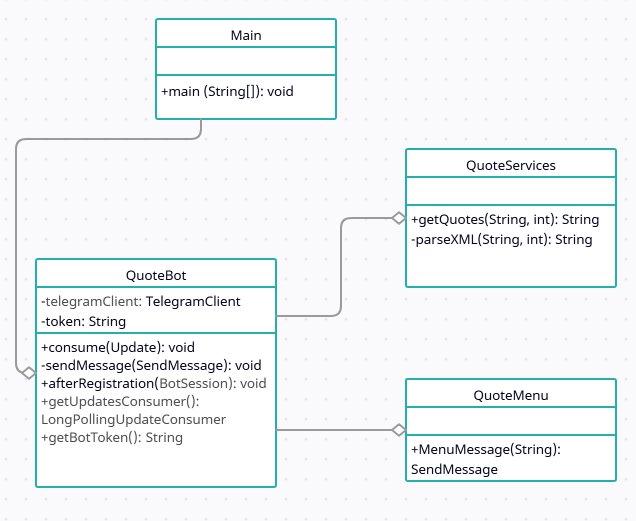


Рис.2 Диаграмма классов программы

Опишем спецификацию классов:

Класс Main

Методы: public static void main(String[] args) – запускает самого бота

Класс QuoteBot

Поля: telegramClient – Telergam клиент бота,

token – токен Telergam бота.

Методы: consume – основной метод бота, его управление,

sendMessage – для упрощения отправки сообщений,

afterRegistration – состояние бота,

getUpdatesConsumer – взаимодействия с ботом,

getBotToken – получение токена бота.

Класс QuoteServices

Методы: getQuotes – отправляет запрос на сайт банка и получает ответ,

parseXML – парсит ответ, полученный в формате XML.

Класс QuoteMenu

Методы: MenuMessage – создает основное меню для пользователя.

Коды всех классов приведены в Приложении А.

**2.** **ИНТЕРФЕЙС И ВЫПОЛНЕНИЕ ПРОГРАММЫ**

**2.1.Описание интерфейса пользователя**

При запуске бота пользователя встречает краткая информация и инструкция по работе с ботом. Для начала работы необходимо ввести в строку /start, после этого пользователю выводится основной интерфейс взаимодействия с ботом. Ему на выбор предоставляется 3 кнопки: котировки всех валют, котировки драгоценных металлов и котировки основных валют. При нажатии любой из кнопок выводится запрашиваемая информация в виде понятного текстового сообщения. Сразу после этого пользователю предлагается снова сделать выбор.

При отправлении боту любого сообщения, кроме /start, будет выводится надпись «Вы можете взаимодействовать только с главным меню», после чего появится само меню.

**2.2.Пример работы программы**

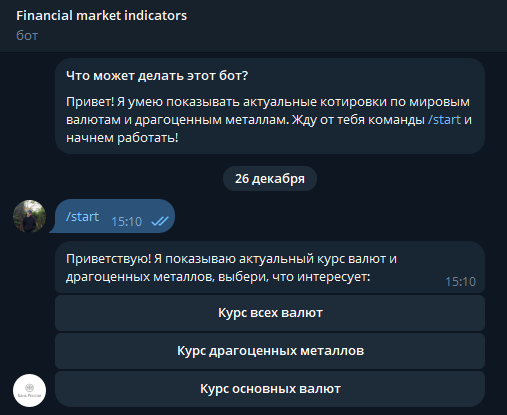
****

Рис.3 Начало работы с ботом

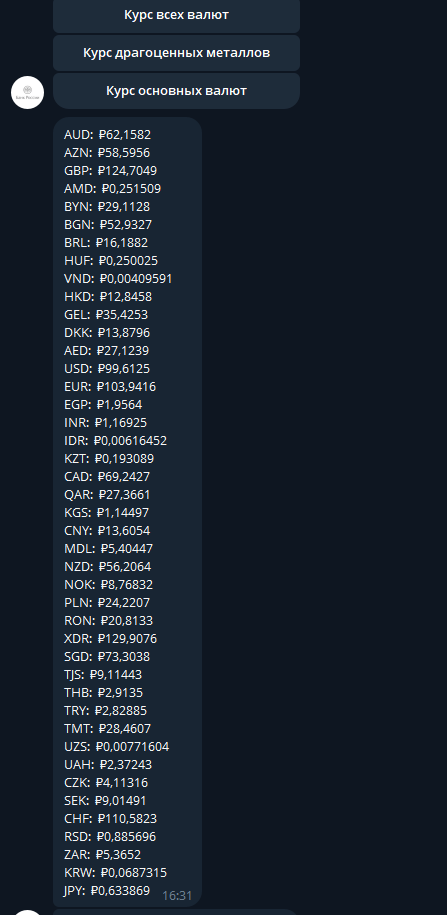


Рис.4 Вывод первого пункта

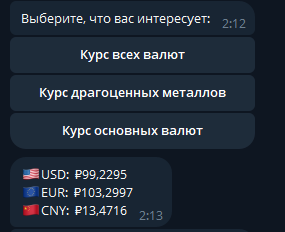


Рис.5 Вывод 3 пункта

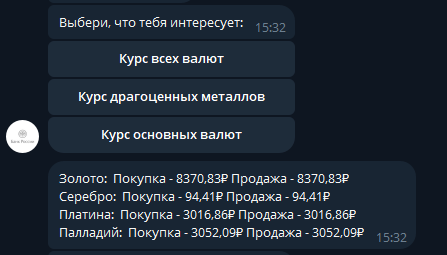


Рис.6 Вывод 2 пункта

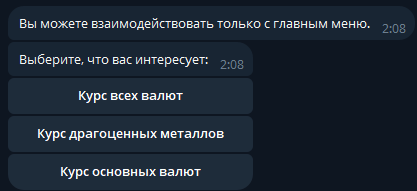


Рис.7 Обработка некорректного ввода

**ЗАКЛЮЧЕНИЕ**

В заключение работы можно сказать, что все поставленные задачи были выполнены, а цели достигнуты. Был написан телеграмм бот с требуемым функционалом, получены навыки работы с отправкой запросов на сайт, парсингом результатов и организацей самого бота.

Ссылка на гитхаб: https://github.com/Zhukov043/OOP\_BOT.git

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Статья о создании бота телеграмм // Telegram URL: <https://core.telegram.org/bots> (дата обращения 25.12.2024)
2. Статья о библиотеке по созданию бота // Telergam URL: <https://core.telegram.org/bots/api#replykeyboardmarkup> (дата обращения 25.12.2024)
3. Инструкция по созданию бота // SkillFactory Media URL: <https://blog.skillfactory.ru/kak-sozdat-bota-v-telegram/> (дата обращения 25.12.2024)
4. Артем Демиденко. Telegram Bot. Руководство по созданию бота в мессенджере Телеграм. 30с, 2023 год.

**ПРИЛОЖЕНИЕ А**

**КОД ПРОГРАММЫ**

MAIN

package org.example;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Main {  
 public static void main(String[] args) {  
 SpringApplication.*run*(Main.class, args);  
 }  
}

QuoteBot

@Component  
public class QuoteBot implements SpringLongPollingBot, LongPollingSingleThreadUpdateConsumer {  
  
 private final TelegramClient telegramClient;  
  
 public QuoteBot() {  
 telegramClient = new OkHttpTelegramClient(getBotToken());  
 }  
  
 @Override  
 public String getBotToken() {  
 return "7965420077:AAHbWeASdkmRtNm1C6E9BAwvIJGUmP77pe4";  
 }  
  
 @Override  
 public LongPollingUpdateConsumer getUpdatesConsumer() {  
 return this;  
 }  
  
 @Override  
 public void consume(Update update) {  
 if (update.hasMessage() && update.getMessage().hasText()) {  
 String chat\_id = update.getMessage().getChatId().toString();  
 String userMessage = update.getMessage().getText();  
 if (userMessage.equals("/start")){  
 sendMessage(QuoteMessage.*MenuMessage*(chat\_id));  
 }  
 else{  
 SendMessage message = new SendMessage(chat\_id, "Введите команду /start, чтобы начать работу с ботом");  
 sendMessage(message);  
 }  
 }  
 else if (update.hasCallbackQuery()) {  
 CallbackQuery callbackQuery = update.getCallbackQuery();  
 String callbackData = callbackQuery.getData();  
 String chat\_id = callbackQuery.getMessage().getChatId().toString();  
 switch (callbackData) {  
 case "allcurrency\_callback" -> {  
 String text = QuoteServices.*getQuotes*("http://www.cbr.ru/scripts/XML\_daily.asp?", 1);  
 SendMessage message = new SendMessage(chat\_id, text);  
 sendMessage(message);  
 }  
 case "metals\_callback" -> {  
 String text = QuoteServices.*getQuotes*("http://www.cbr.ru/scripts/xml\_metall.asp?date\_req1=26/12/2024&date\_req2=26/12/2024", 2);  
 SendMessage message = new SendMessage(chat\_id, text);  
 sendMessage(message);  
 }  
 case "currency\_callback" -> {  
 String text = QuoteServices.*getQuotes*("http://www.cbr.ru/scripts/XML\_daily.asp?", 3);  
 SendMessage message = new SendMessage(chat\_id, text);  
 sendMessage(message);  
 }  
 }  
 sendMessage(QuoteMessage.*MenuMessage*(chat\_id));  
 }  
 else if (update.hasMessage() && !update.getMessage().hasText()){  
 String chat\_id = update.getMessage().getChatId().toString();  
 String text = "Введите команду /start, чтобы начать работу с ботом";  
 SendMessage message = new SendMessage(chat\_id, text);  
 sendMessage(message);  
 }  
 }  
  
 private void sendMessage(SendMessage message) {  
 try {  
 telegramClient.execute(message); *// Sending our message object to user* } catch (TelegramApiException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @AfterBotRegistration  
 public void afterRegistration(BotSession botSession) {  
 System.*out*.println("Registered bot running state is: " + botSession.isRunning());  
 }  
}

QuoteServices

public class QuoteServices {  
 public static String getQuotes(String urlString, int code){  
 try {  
 URL url = new URL(urlString);  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
 connection.setRequestMethod("GET");  
 BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));  
 StringBuilder response = new StringBuilder();  
 String inputLine;  
 while ((inputLine = in.readLine()) != null) {  
 response.append(inputLine);  
 }  
 in.close();  
 return *parseXML*(response.toString(), code);  
 }  
 catch (IOException e){  
 System.*out*.println("Ошибка соединения");  
 return null;  
 }  
 }  
 private static String parseXML(String xmlData, int code){  
 StringBuilder rates = new StringBuilder();  
 try {  
 DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();  
 DocumentBuilder builder = factory.newDocumentBuilder();  
 Document doc = builder.parse(new InputSource(new StringReader(xmlData)));  
  
 if (code == 1) {  
 NodeList valuteList = doc.getElementsByTagName("Valute");  
 for (int i = 0; i < valuteList.getLength(); i++) {  
 Node valuteNode = valuteList.item(i);  
  
 String charCode = valuteNode.getChildNodes().item(1).getTextContent();  
 String value = valuteNode.getChildNodes().item(5).getTextContent();  
  
 rates.append(charCode).append(": ").append(" ₽").append(value).append("\n");  
 }  
 }  
 else if (code == 2){  
 String[] metals = new String[] {"Золото", "Серебро", "Платина", "Палладий"};  
 NodeList metalList = doc.getElementsByTagName("Record");  
 for (int i = 0; i < metalList.getLength(); i++) {  
 Node metalNode = metalList.item(i);  
  
 String buy = metalNode.getChildNodes().item(0).getTextContent();  
 String sell = metalNode.getChildNodes().item(1).getTextContent();  
  
 rates.append(metals[i]).append(": ").append(" Покупка - ").append(buy);  
 rates.append("₽ Продажа - ").append(sell).append("₽").append("\n");  
 }  
 }  
 else{  
 String[] currencyCode = new String[] {"GBR", "USD", "EUR", "CNY"};  
 NodeList valuteList = doc.getElementsByTagName("Valute");  
 for (int i = 0; i < valuteList.getLength(); i++) {  
 Node valuteNode = valuteList.item(i);  
 String charCode = valuteNode.getChildNodes().item(1).getTextContent();  
 if(Arrays.*stream*(currencyCode).anyMatch(s -> s.contains(charCode))) {  
 String value = valuteNode.getChildNodes().item(5).getTextContent();  
 rates.append(charCode).append(": ").append(" ₽").append(value).append("\n");  
 }  
 }  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 rates.append("Ошибка парсинга XML.");  
 }  
 return rates.toString();  
 }  
  
}

Quote message

public class QuoteMessage {  
 public static SendMessage MenuMessage(String chatId) {  
 String text = "Выбери, что тебя интересует: ";  
 SendMessage message = new SendMessage(chatId, text);  
  
 List<InlineKeyboardRow> keyboard = new ArrayList<>();  
  
 InlineKeyboardRow row = new InlineKeyboardRow();  
 InlineKeyboardButton button1 = new InlineKeyboardButton("Курс всех валют");  
 button1.setCallbackData("allcurrency\_callback");  
 row.add(button1);  
  
 InlineKeyboardRow row1 = new InlineKeyboardRow();  
 InlineKeyboardButton button2 = new InlineKeyboardButton("Курс драгоценных металлов");  
 button2.setCallbackData("metals\_callback");  
 row1.add(button2);  
  
 InlineKeyboardRow row2 = new InlineKeyboardRow();  
 InlineKeyboardButton button3 = new InlineKeyboardButton("Курс основных валют");  
 button3.setCallbackData("currency\_callback");  
 row2.add(button3);  
  
 keyboard.add(row);  
 keyboard.add(row1);  
 keyboard.add(row2);  
  
 InlineKeyboardMarkup keyboardMarkup = InlineKeyboardMarkup  
 .*builder*()  
 .keyboard(keyboard)  
 .build();  
  
 message.setReplyMarkup(keyboardMarkup);  
 return message;  
 }  
  
}

Все подключенные библиотеки

import java.io.\*;  
import java.net.HttpURLConnection; *//Отправка запроса*import java.net.URL; *//URL объекты*import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
  
import org.telegram.telegrambots.meta.api.objects.CallbackQuery;  
import org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.InlineKeyboardRow;  
import org.w3c.dom.Document;  
import org.w3c.dom.NodeList;  
import org.w3c.dom.Node;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
import org.springframework.stereotype.Component;  
import org.telegram.telegrambots.client.okhttp.OkHttpTelegramClient;  
import org.telegram.telegrambots.longpolling.BotSession;  
import org.telegram.telegrambots.longpolling.interfaces.LongPollingUpdateConsumer;  
import org.telegram.telegrambots.longpolling.starter.AfterBotRegistration;  
import org.telegram.telegrambots.longpolling.starter.SpringLongPollingBot;  
import org.telegram.telegrambots.longpolling.util.LongPollingSingleThreadUpdateConsumer;  
import org.telegram.telegrambots.meta.api.methods.send.SendMessage;  
import org.telegram.telegrambots.meta.api.objects.Update;  
import org.telegram.telegrambots.meta.api.objects.replykeyboard.InlineKeyboardMarkup;  
import org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.InlineKeyboardButton;  
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;  
import org.telegram.telegrambots.meta.generics.TelegramClient;  
import org.xml.sax.InputSource;